DESIGN OF DIRECT DIGITAL FREQUENCY SYNTHESIZER FOR WIRELESS APPLICATIONS

Lakshmi Sri Jyothi Chimakurthy

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama

August 8, 2005

DESIGN OF DIRECT DIGITAL FREQUENCY SYNTHESIZER FOR WIRELESS APPLICATIONS

Lakshmi Sri Jyothi Chimakurthy

_____

Signature of Author

_____

Date

Copy sent to:

Name Date

VITA

Lakshmi Sri Jyothi Chimakurthy, eldest daughter of Sri Rama Murthy and Sudha Rani Chimakurthy, was born in Chirala, India. Her father is one of the leading entrepreneurs in Hyderabad, India whose company manufactures Industrial transformers, Inductors, UPS and a wide variety of electrical equipment.

Jyothi has received her Bachelor's degree in Electronics and Communications Engineering from Jawaharlal Nehru Technological University, Hyderabad, India in July, 2002. During her Bachelors she had done an internship with the Defense Research and Development Laboratories, Hyderabad, India. She had joined Auburn University to pursue her Masters degree in the area of RF Analog /Digital mixed signal circuit design for wireless applications. She worked at AMSTC as a graduate research assistant under the guidance of Dr. Foster Dai and Dr. Richard C. Jaeger.

THESIS ABSTRACT

DESIGN OF DIRECT DIGITAL FREQUENCY SYNTHESIZER FOR WIRELESS APPLICATIONS

Lakshmi Sri Jyothi Chimakurthy
Master of Science, August 8, 2005
(B-TECH, Jawaharlal Nehru Technological University, 2002)

94 Typed Pages

Directed by Foster Dai

Direct Digital Synthesis can be practically defined as a means of generating highly accurate and harmonically pure digital representations of signals. High speed DDS presents an attractive alternative to use of Phase locked loop approach in the design of high bandwidth frequency synthesizers because of its features like sub-hertz frequency resolution, fast settling time, continuous phase switching response and low phase noise. Although the principle of the DDS has been known for many years, the DDS did not play a dominant role until recent years. Earlier DDSs were limited to produce narrow bands of closely spaced frequencies, due to limitations of digital logic and D/A converter technologies. Recent advances in integrated circuit technologies have brought about remarkable progress in this area. By programming the DDS, adaptive channel bandwidths modulation formats, frequency hopping, and data rates are easily achieved. This is an important step towards applications like software radio which can be used in various systems. The DDS could be applied in the modulator or demodulator, in the

communication systems. The applications of DDS are restricted to the modulator in the base station.

One of the important factors determining the spectral purity of DDS is the resolution of the values stored in the sine look up table (ROM). Increasing the size of the ROM for better spectral purity is not a good approach because of the higher power consumption, lower speed and increased cost of a larger ROM. The first part of this thesis discusses the design and implementation of phase accumulator in the high speed ROM less DDS with sine-weighted DAC. The basic logic blocks are implemented in SiGe technology. The second part of the thesis proposes a novel DDS architecture with a compressed ROM without degradation of quantization noise. The ROM compression algorithm proposed achieved a better compression ratio of 94.3:1 with little increase in hardware when compared to many popular compression algorithms, giving a worst case spur of -90.3 dBc.

ACKNOWLEDGEMENTS

I would like to begin by thanking my parents Sri Rama Murthy and Sudharani, my siblings Sri bala and Uday kiran. Over the years my family made numerous sacrifices so that I could pursue my career goals. I would like to thank them for their support, blessings and well wishes. Secondly, I would like to thank my close friends Malinky Ghosh and Blessil George for their strong companionship and emotional support which was very important for my fruitful research. I would also thank my cousins Subba Rao and Mohan for their encouragement and confidence in me.

I am greatly indebted to Dr. Foster Dai, for being my advisor and teacher for the last two years. He had constantly motivated me to work towards excellence. Despite my repeated failures, his enthusiasm and strong support motivated me to finally succeed in this project. I would like to express my deepest gratitude to Prof. Richard C. Jaeger for all his support, valuable suggestions and making my graduate school life, a rich and rewarding experience. He has constantly amazed me with his profound knowledge and ability to simplify complicated research problems. I thank Dr. Niu for his complete cooperation. I would like to thank my colleagues Dayu yang and Qi Xiu for their ideas and helping me to successfully take the FPGA measurements. Life was never dull over the last two years, thanks to the exciting and entertaining moments created by all the friends I met at Auburn. The memorable times spent with Anand, Sailaja and Prasanthi will be fondly remembered for years to come.

Style manual or journal used IEEE Transactions on Circuits and Systems

Computer software used Microsoft Word 2000

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Modern communication systems, especially spread spectrum systems, are placing increasing demands on the resolution and bandwidth requirements of frequency synthesizer sub systems in order to gain improved performance. Today's spread spectrum applications require a frequency synthesizer that is capable of tuning to different output frequencies with extremely fine frequency resolution with a switching speed of the order of nanoseconds. The resolution requirements of many systems are so severe that they are surpassing the performance capabilities of conventional analog phase locked loop. Although limited by Nyquist criteria, DDS allows frequency resolution control on the order of milli-hertz or even nano-hertz of phase resolution control. The increasing availability of high speed DACs make a direct digital approach to frequency synthesis an enticing alternative to conventional analog synthesizers.

1.1 Direct Digital Frequency Synthesizer

The simplest architecture of DDFS uses an N-bit accumulator, a read only memory (ROM) and a digital to analog converter, implemented using the same reference clock f_{clk} as is presented by Tierney Rader, Gold et al [1]. A conventional ROM-based DDFS is show in Figure 1.1. The output frequency of the DDFS depends on the W bit input word called Frequency control word (FCW). At each positive edge of the reference

clock cycle the FCW is added to the value in the W-bit accumulator. At any instant the value in the accumulator represents the phase of the sinusoid, so it is referred as phase accumulator register. The value in the phase accumulator register is used to address the ROM storing the values of the sinusoid. And its resolution is shown in Figure 1.1. The output frequency of the DDS depends on f_{clk} as

$$f_{out} = f_{clk} \frac{FCW}{2^N} \tag{1.1}$$

The finest resolution possible is

$$f_{res} = \frac{f_{clk}}{2^N} \tag{1.2}$$

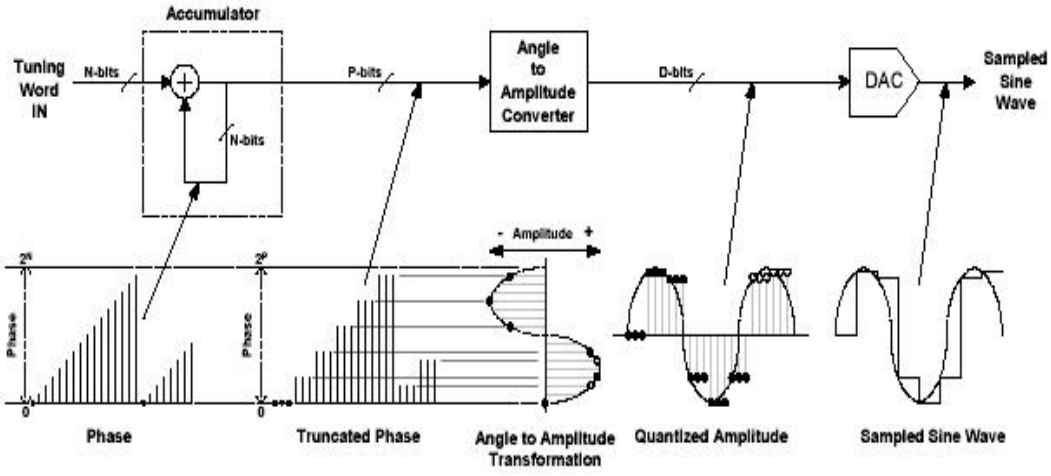


Figure 1.1 - A Conventional ROM Based DDFS Architecture

DDS is a digital technique for generating an output waveform (sine, square or triangular) or clocking signal from a fixed-frequency clock source.

Some DDS advantages:

- Micro Hertz frequency resolution and sub-degree phase offset resolution.

- Extremely fast frequency transition or “hopping speed” : <5 ns to change frequency.
- All digital control eliminates need for manual system tweaking and allows output frequency to be conveniently adjusted.
- Easily synchronized allowing quadrature and other exact signal phase relationships to be obtained.
- Unparalleled matching of I and Q outputs.
- Provides accurate, high speed PSK (phase shift keying) and FSK (frequency shift keying).

DDSs are found in a wide range of applications:

-waveform generation

-impedance measurement

- sensor excitation

- digital modulation/demodulation

- test and measurement equipment

- clock recovery

- programmable clock generator

- liquid and gas flow measurement

- sensory applications

- medical equipment

- HFC data, telephony, and video modem

- FM chirp source for radar and scanning systems

- agile, LO frequency synthesis

- commercial and amateur RF exciter

- wireless and satellite communications
- cellular base station hopping synthesizer
- broadband communications
- VHF/UHF-LO synthesis
- tuners
- military radar
- automotive radar
- SONET/SDH clock synthesis
- acousto-optic device driver
- PSK/FSK/Ramped FSK modulation

Many companies like Analog devices, Qualcomm etc. have integrated Complete-DDS products [2] which present an attractive alternative to analog PLLs for agile frequency synthesis applications. Direct digital synthesis (DDS) has long been recognized as a superior technology for generating highly accurate, and frequency-agile (rapidly changeable frequency over a wide range), low-distortion output waveforms.

A major advantage of a DDS system is that its output frequency and phase can be precisely and rapidly manipulated under digital processor control. Other inherent DDS attributes include the ability to tune with extremely fine frequency- and phase resolution (frequency control in the millihertz (mHz) range and phase control < 0.09°, and to rapidly "hop" in frequency (up to 23 million output frequency changes per second). These characteristics have combined to make the technology extremely popular in military radar and communications systems. In fact, DDS technology was previously relegated almost exclusively to high-end and military applications: it was costly, power-hungry

(dissipations specified in *watts*), difficult to implement, required a discrete high-speed DAC, and had a set of user-hostile system interface requirements.

DDS should be uniquely attractive for local oscillator (LO) and up/down frequency conversion stages-which were until now the exclusive domain of PLL-based analog synthesizers. The Complete-DDS architecture by Analog devices holds distinct advantages over an equivalent PLL-based agile analog synthesizer. For example:

- Output frequency resolution: The AD98x0 C-DDS products have 32-bit phase accumulators, which enable output frequency tuning resolutions much finer than a PLL-based synthesizer can enjoy. The AD9850 has a tunable output resolution of 0.06 Hz, with a clock frequency of 125 MHz; the AD9830 has a tuning resolution of 0.012 Hz, with a reference clock of 50 MHz. Furthermore, the output of these devices is phase-continuous during the transition to the new frequency. In contrast, the basic PLL-based analog synthesizer typically has an output tuning resolution of 1 kilohertz; it lacks the inherent resolution afforded by the digital signal processing.
- Output-frequency switching time: The analog PLL frequency switching time is a function of its feedback loop settling time and VCO response time, typically > 1 ms. C-DDS-based synthesizer switching time is limited only by DDS digital processing delay; the AD9850's minimum output frequency switching time is 43 ns.
- Tuning range: A critical feedback loop bandwidth and input reference frequency relationship determines the stable (usable) frequency range of the typical analog PLL circuit. C-DDS-based synthesizers are immune to such loop filter stability issues and are tunable over the full Nyquist range (< 1/2 the clock rate).

- Phase noise: Because of the frequency division, C-DDS-based solutions have a clear advantage over analog PLL synthesizers in output phase-noise. The output phase noise of a C-DDS synthesizer is actually better than that of its reference clock source, while analog PLL-based synthesizers have the disadvantage of actually multiplying the phase noise present in their frequency reference.
- Implementation complexity: DDS which include the signal DAC, translate to ease of system design. There is no longer an element of RF design expertise required to implement a DDS solution. A simple digital instruction set for control minimizes the complexity of support hardware. Digital system design replaces the analog-intensive system design required for PLL-based analog synthesizer solutions to similar problems.

This basic DDFS architecture however incorporates a huge ROM, which restricts high speed applications and requires huge power consumption. To reduce the power dissipation and the die area there are two possible options:

- Implement ROM compression techniques.
- Replace the ROM and linear DAC with non linear DAC.

This work suggests a new algorithm for ROM compression with an improved compression ratio at the same quantization noise levels. The second part of the work involves study of ROM-less DDS with nonlinear DAC, design and implementation of high-speed accumulator in SiGe technology for High speed DDS.

1.2 Overview of the work

In chapter 1 the operation of conventional DDS as presented by Tierney Rader, Gold et al [1] has been discussed. DDS has been described as an attractive alternative to analog PLL for agile frequency synthesis. Its applications and advantages and

comparison to PLL have been provided. Two alternatives have been suggested to improve the speed and reduce the power consumption of the ROM based DDS. The first one is the design of ROM-less high speed DDS with non-linear sine weighted DAC and a novel ROM compression technique without degradation in spurious response has been presented as a second alternative.

In chapter 2 the architecture for the ROM-less DDS has been discussed. CML logic has been used to implement the basic logic blocks of the high speed DDS. Two types of logic gates with levels of three transistors stacked vertically i.e. two input gates and with levels of four transistors stacked vertically i.e. three input gates have been proposed. A novel resettable flip-flop architecture has also been discussed. Merging logic has been used to combine the gates in a full adder. Finally a full adder circuit with a single current source to implement the carry logic and a single current source to implement the sum logic has been presented. Discussion on different accumulator architectures like the pipelined architecture and the carry look ahead adder accumulator architecture has been made. These architectures have been implemented in CADENCE and simulation results have been presented. An 8 bit fully pipelined phase accumulator at 6.5 GHz has been presented.

Chapter 3 discusses the noise analysis of the output of DDS, which is very important for the frequency planning. Brief discussion about the effect of reference clock spurs, phase truncation, angle to amplitude mapping (quantization noise due to finite resolution of values stored in ROM), DAC's quantization noise, and nonlinearities on the output spectrum of DDFS has been presented.

In chapter 4, a novel ROM compression algorithm with improved compression ratio, without degradation of quantization noise at the output of the DDFS has been presented. The code for the proposed DDS architecture has been written in VERILOG and the results are verified on Xilinx Spartan II FPGA at 25 MHz. The results of this proposed architecture have been compared to the architecture proposed by Nicholas which is the most popular DDS architecture with compressed ROM. Analog devices manufactured a DDS chip based on Nicholas's architecture.

CHAPTER 2

ROM-LESS HIGH SPEED DDS DESIGN

Direct digital synthesis (DDS) provides precise frequency resolution and direct modulation capability. However, the majority of the DDS designed so far is limited to low frequency applications with clock frequencies less than a few hundred MHz. Digitally generating highly complex wide bandwidth waveforms at the highest possible frequency instead of down near base band would considerably reduce the transmitter architecture in terms of size, weight and power requirements as well as cost. This chapter presents a low power, high-speed direct digital synthesizer (DDS) designed at Auburn University in a 47GHz SiGe technology. The ROM-less DDS includes an 8-bit accumulator, column/row decoders and an 8 bit cosine-weighted digital-to-analog converter (DAC) operating at maximum 6GHz clock frequency to synthesize and modulate the inter-median frequency (IF) of up to L-band. The DDS could achieve better than -45dB spectral purity. The DDS core occupies an area of 2 mm^2 and consumes less than 2W power with a 3.3V supply voltage. The 5GHz MMIC provides a frequency synthesis and modulation means for L-band applications. The cosine weighted DAC, eliminate the sine look up table, which is the bottleneck of speed and area for high-speed DDS implementations. The low power consumption we achieved using SiGe technology is much lower than that of an equivalent DDS implemented in a 137GHz InP technology [3], where the DDS consumes about 15 W, operating at 9.2GHz clock frequency with an

8-bit accumulator and an 8-bit nonlinear DAC.

2.1 ROM-Less DDS Architecture

The conventional DDS architecture utilizes a ROM look-up table to convert the accumulated phase word into sine/cosine words that are further used to drive a linear binary weighted DAC. The speed bottleneck of a conventional DDS architecture lies upon the large look-up table with multi-level decoders. The huge look-up table not only restricts high speed operation, but also occupies large area and consumes large amount of power. To reduce the power dissipation and the die size, a nonlinear DAC with cosine-weighted current cells is employed in this design as shown in Figure 2.1. The nonlinear DAC converts the digital phase information directly into an analog output. The proposed architecture exploits the quadrant symmetry property of the sine function around $\pi/2$ and π. Thus the 2 MSB's of the accumulator output signify the different quadrants of the sinusoid. The remaining w-2 bits from the complementor are used to generate the waveform of sinusoid in the first quadrant. The complete sinusoid can be generated with the 2 MSB's.

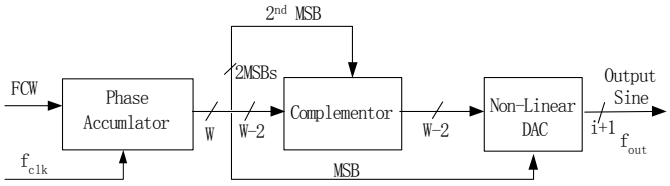


Figure 2.1 - High-speed DDS with a nonlinear cosine-weighted DAC.

2.2 Algorithm used for DAC Implementation [4]

The phase output ϕ of the accumulator is divided into three parts α (most significant part), β (middle part), γ (least significant part) such that number of bits in

α , β and γ are a, b and c respectively.

Based on trigonometric identities, the first quadrant of the sinusoid

$$\sin\frac{\pi(\alpha+\beta+\gamma)}{2(2^{a+b+c}-1)}=[\sin\frac{\pi(\alpha+\beta)}{2(2^{a+b+c}-1)}\cos\frac{\pi\gamma}{2(2^{a+b+c}-1)}+\cos\frac{\pi(\alpha+\beta)}{2(2^{a+b+c}-1)}\sin\frac{\pi\gamma}{2(2^{a+b+c}-1)}] \quad (2.1)$$

$$\sin\frac{\pi(\alpha+\beta+\gamma)}{2(2^{a+b+c}-1)}=[\sin\frac{\pi(\alpha+\beta)}{2(2^{a+b+c}-1)}+\cos\frac{\pi(\alpha+\beta)}{2(2^{a+b+c}-1)}\sin\frac{\pi\gamma}{2(2^{a+b+c}-1)}] \quad (2.2)$$

The Equation (2.2) is implemented using a nonlinear DAC divided into coarse DAC and fine DAC. The first term on the right hand side is realized using a coarse DAC and the second term is realized using a fine DAC. The first term is monotonic and can be realized as a coarse nonlinear DAC using the full thermometer code non linear DAC technique. If the nonlinear DAC has i+1 bits of amplitude resolution, each coarse DAC cell output (o_k) is

$$O_{k=}\begin{cases}\text{int}\left[(2^{i}-1)\sin\frac{\pi(0.5)}{2(2^{a+b}-1)}\right] & \text{for } k=0\\ \text{int}\left[(2^{i}-1)\sin\frac{\pi(k+0.5)}{2(2^{a+b}-1)}-\sum_{n=0}^{k-1}O_n\right] & \text{for } 1\le k\le 2^{a+b}-1\end{cases} \quad (2.3)$$

where k is $\frac{\alpha+\beta}{2^c}$, a and b are the number of bits in segments α and β respectively. As the number of phase bits increase the number of DAC cells increase and therefore the power requirements, die area also increase. Since the number of bits in α and β are less than the total number of input phase bits without segmentation the number of coarse DAC cells are much less than that required without segmentation. The second term in Equation (2.2) forms the fine nonlinear DAC to interpolate the amplitudes between adjacent coarse DAC outputs. In Equation 2.2 the value of β has been approximated

using the average value, β_{avg} so that the fine DAC term depends only on α and γ to reduce the number of fine non linear DAC cells. The fine DAC output is not monotonic in α and γ. So a fine DAC is constructed using $2^a - 1$ nonlinear sub DACs, where a is the number of bits in segment α. The m[th] DAC cell output in the αth sub DAC, $o_{\alpha,m}$, can be approximated as

$$o_{\alpha,k=}\begin{cases} \text{int}\left[(2^i-1)\cos\dfrac{\pi(\alpha+\beta_{avg})}{2(2^{a+b+c}-1)}.\sin\dfrac{(0.5)\pi}{2\left(2^{a+b+c}-1\right)}\right] & \text{for m=0} \\ \text{int}\left[(2^i-1)\cos\dfrac{\pi(\alpha+\beta_{avg})}{2(2^{a+b+c}-1)}.\sin\dfrac{(m+0.5)\pi}{2\left(2^{a+b+c}-1\right)}-\sum_{n=0}^{m-1}o_{\alpha,n}\right] & \text{for } 1\leq m\leq 2^c\text{-}1 \end{cases} \quad (2.4)$$

The total output is the sum of the current outputs of the coarse DAC and the fine sub DACs if current steering technique is used. Different segmentations give different performance due to the different amplitude errors in the approximation. An architecture based on this algorithm has been discussed in [5].

2.3 Phase accumulator

Phase accumulator is the first module in the DDS design. The phase accumulator adds the N-bit input frequency control word to itself once every clock cycle. The speed of the phase accumulator depends upon the N bit adder and the flip-flop design. In the proposed architecture, even if the nonlinear DAC operates at 10 GHz, the speed of the DDS is limited by the speed of accumulator. This thesis discusses two different architectures for the 8-bit phase accumulator, the pipelined accumulator and the pipelined accumulator with Carry Look Ahead (CLA) adders.

2.3.1 SiGe Technology

Silicon-germanium (SiGe) semiconductor technology has been used to implement the phase accumulator. SiGe technology has long held the promise of high-frequency operation with high levels of integration. This technology claims device transition frequencies that could surpass expensive gallium-arsenide (GaAs) technology while using standard silicon wafers. While the technology is now making its way into a variety of integrated circuits (ICs), mainly for cellular handsets and wireless-local-area-network (WLAN) cards, its integration potential has been largely untapped. Companies like Centellax, Broadcom have worked on high speed fractional N-synthesizers that take advantage of the excellent high-frequency performance of SiGe as well as the potential for integration when using silicon CMOS processing.

Advantages

- high levels of integration
- excellent high-frequency performance
- potential for integration when using silicon CMOS processing.
- Traditionally, HBT devices have been fabricated on GaAs or InP substrates, requiring complex and expensive process technologies. One of the benefits of SiGe HBT devices is that they can be fabricated with conventional silicon CMOS technology with only a few additional process steps.

SiGe made it possible for integration of VCO, synthesizer, and digital signal processing on a single chip offering a significant savings in size, cost, and power consumption compared to existing solutions for a wide range of broadband applications in commercial, industrial, and military systems. CML logic with SiGe Transistors has

been used for implementation of all the basic logic blocks in the accumulator.

2.3.2 Background on CML and ECL circuit design

2.3.2.1 DC Operation

ECL and CML circuits are based on the non saturated emitter-coupled pair. The CML circuit (Figure 2.3, [6]) is a simplified version of an ECL circuit (Figure 2.2, [6]) in which the emitter follower is omitted for a higher integration density. This emitter follower acts as a level shifter and as an impedance adapter (it supplies current to the output and as a result the circuit speed and fan-out are higher). A direct consequence of using the emitter follower is that the Q1 collector voltage is isolated against any output load. Taking the proper values for R_L and Io, the circuit can be designed so that the emitter-coupled pair transistors will never operate in the saturation region. This is the main reason for the low propagation delay time of the ECL and CML circuits. In most cases, both, fan-out and fan-in are not determinant factors in this kind of circuits. However, this is true only in DC design. In transient design, if a high circuit operation speed is desired, the propagation

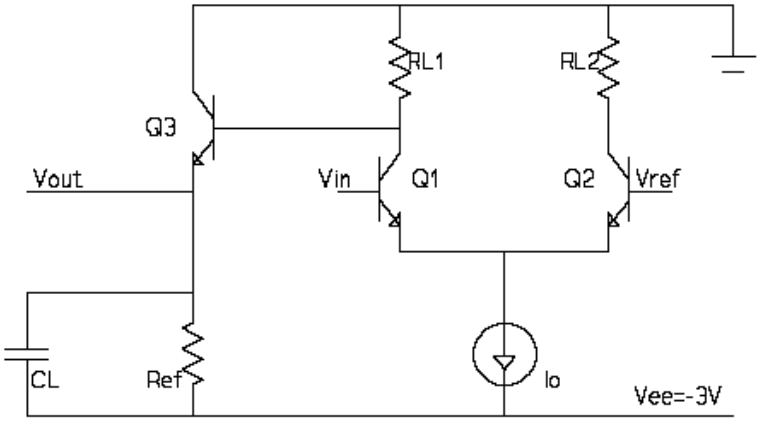


Figure 2.2 – ECL circuit diagram

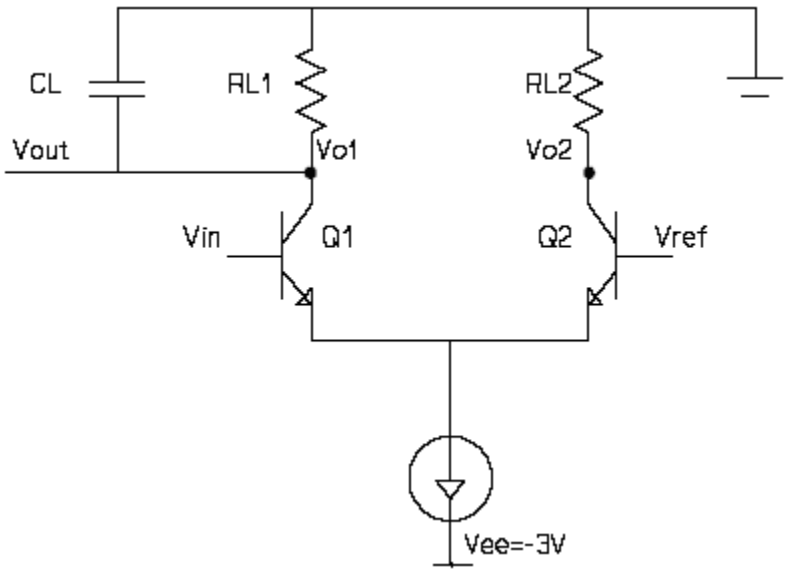


Figure 2.3 – CML Circuit diagram

delay has to be taken into account and the number of gates loading a given gate has to be limited. The study of the problems related to transient design is the object of the next section.

2.3.2.2 Transient Analysis [6]

There are several factors that contribute to a logic circuit propagation delay. One of them is the wiring capacitance. This capacitance appears mainly due to the coupling between the interconnections and the surface material. At this point, it should be stated that, GaAs and its alloys show smaller capacitances than Si for the same structure. This leads to a higher operation speed in III-V HBTs as compared with silicon BJTs. Another source of capacitance is fan-out. Gates inputs are outputs capacitances to the preceding gates associated with the active components. This capacitance varies with the voltage, making difficult the estimation of the propagation delay. Another effect is the intrinsic

delay associated with the electron transit time across the transistor. This transit time is characterized by the intrinsic frequency fT, $\tau=1/2\pi fT$. The transit time can be minimized using materials or structures with high electron mobilities. The propagation delay t_p and rising and falling times (t_{pLH} and t_{pHL}) degrade with fan-out. For a given fan-out, ECL gates demonstrate a faster response than CML ones. Besides, ECL gates have a higher fan-out. However, as it will be discussed later, the power consumption of an ECL gate is greater than that of a CML circuit. Furthermore a CML circuit has fewer components, hence a higher integration density for a given package. The linear wiring capacitance has a decisive influence on the transient response. In logic circuitry this capacitance ranges over a wide range depending on the interconnection length. The circuit propagation delay can be estimated by the well known expression, $i = C\left(\frac{dv}{dt}\right)$. It is necessary to increase the available current for charging and discharging this load capacitance when the gate is strongly loaded. The propagation delay variation with the load capacitance is smaller than with the fan-out. Besides, CML gates are more sensitive to a load capacitance variation than ECL gates.

The average power consumption of a logic circuit can be expressed as a linear combination of the average static power consumption and the average dynamic power consumption. We calculate the static power consumption, i.e. the power consumption when there is a high level at the output (P_{OH}) and the power consumption when there is a low level at the output (P_{OL}). For the ECL circuit P_{OH} can be calculated from,

$$P_{OH} = I_O(V_{ee}) + \frac{(V_{ee}+V_1)^2}{R_{ef}} + V_{ref} \cdot I_{ref} \qquad (2.5)$$

where V1 is the high level voltage. Similarly, P_{OL} can be expressed as,

$$P_{OL} = I_O(\ V_{ee}) + \frac{(\ V_{ee} + V_o)^2}{R_{ef}} + V_{ref} \cdot I_{ref} \qquad (2.6)$$

where V_0 is the low level voltage. These expressions can also be used for the CML circuit shown in Figure 2.3 removing the term related to the emitter follower. Because of this, the power consumption of CML gates is smaller than that of ECL gates. Furthermore, if a little current I_{ref} is supposed for both gates (this is valid because it is a base current), it can be seen that, for the ECL gate, the power consumption varies with the input (Vin), while for the CML gate the power consumption does not depend on the input. As a consequence, the P_{OH} and P_{OL} difference for the ECL circuit is greater than that of a CML circuit.

The static power values have to be added to the dynamic power consumption. The dynamic power consumption is a direct consequence of the required energy to charge and discharge the circuit capacitances in a unit time. For ECL and CML circuits, these capacitances are concentrated in a unique load capacitance C_L. This capacitor models the wiring capacitance between two consecutive stages. The required energy to charge and discharge this capacitor is $\frac{C_L \Delta V^2}{2}$, ΔV being the voltage swing. If the input signal is a periodic signal and its frequency is f, the capacitor charges and discharges once a cycle. Consequently, the dynamic power consumption can be expressed as, $P_D = f\, C_L \Delta V^2$. In typical designs, the contribution of P_D to the overall power consumption at the working frequency can be neglected. For example, for a C_L of 100 fF and a frequency of 1GHz the dynamic power consumption is 16μW. This dynamic power consumption is three orders of magnitude less than the static power. As a result, the total average power consumption can be expressed approximately as the average static power consumption. The

propagation delay of ECL and CML circuits are mainly affected by two elements: the wiring capacitance (C_L), and the fan-out (FO). The behavior of the propagation delay is quite linear over a wide range of C_L values. Propagation delay dependence with fan-out is not linear. tp is more sensitive to fan-out variations than to C_L variations. For this reason, in a design using ECL or CML circuits, the designer has to put more attention to fan-out requirements than to the wiring capacitances requirements. ECL and CML power consumption is more than the power consumed by other circuits like MESFETs. For this reason, ECL and CML gates are only useful for high performance circuits where the power consumption is not so critical or specific applications where the logic gates are strongly loaded. An example of this kind of application is a circuit with high wiring capacitance or with high fan-out. A figure of merit used in logic circuits evaluation is the power-delay product. MESFETS circuits are better in this regard than the other circuits, even HBTs ones. Another figure where the fan-out is included is the power-delay product divided by the maximum fan-out of the logic family. This figure provides evidence for using HBTs in digital circuits with high loads in its nodes.

2.4 Analysis of two input and three input NAND gate

A NAND gate has been implemented in differential mode. The bias current is set at 400uA. The resistor values have been taken as 530 ohms which gives an output swing of about 430 mV. The peak ft current for 1um npn transistor is about 600um. It is biased at 400 um for optimal speed and minimum current consumption. The bottom transistor when biased properly acts as a current source. The current source should be always out of saturation.

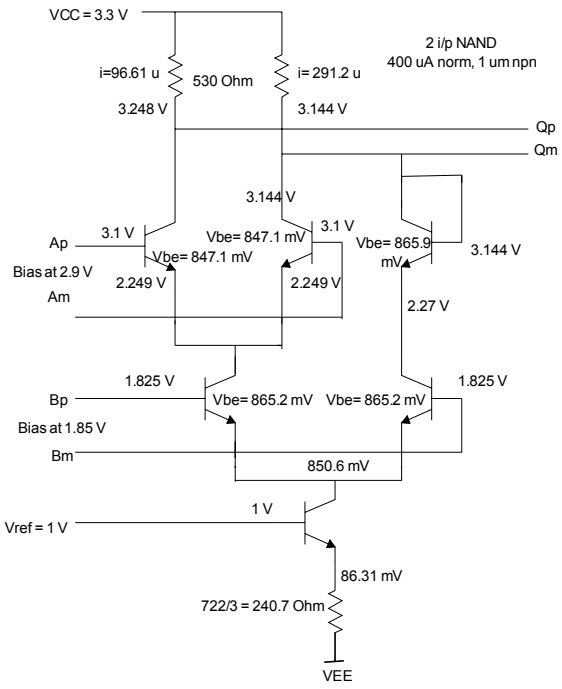


Figure 2.4 – Schematic showing DC analysis of NAND gate

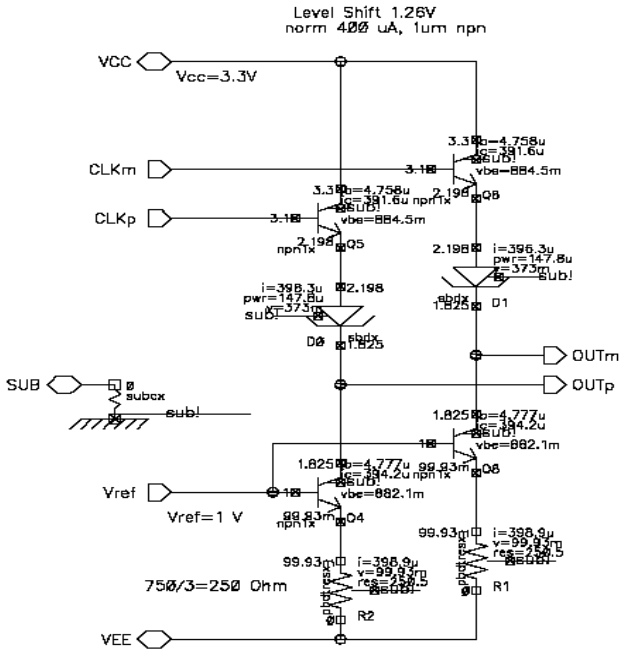


Figure 2.5 – Schematic showing DC analysis of Level shifter.

The B pair of transistors in the NAND gate has to be biased as at 1.85 V. The level shifter used is shown in the Figure 2.5. Schottky diodes provide an extra voltage drop of about 0.3-0.4 V. The worst case output delay of this NAND gate is about 26 ps.

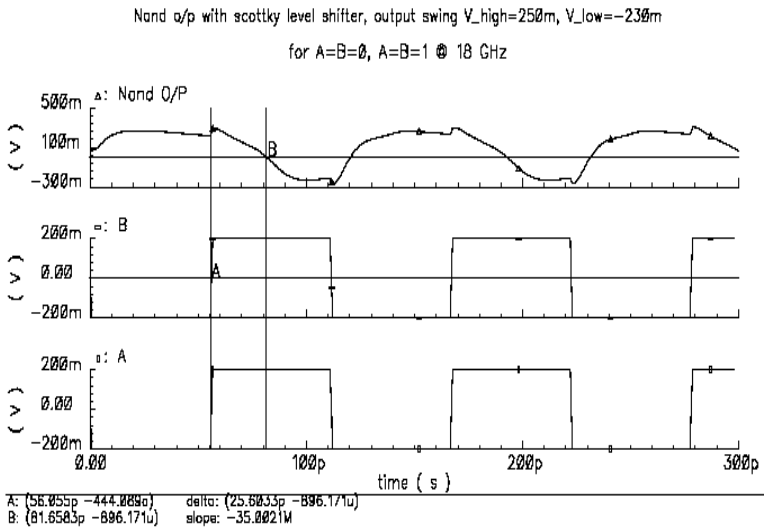


Figure 2.6 – The output of NAND gate showing a worst case propagation delay of 26ps.

A three input NAND gate has been implemented to reduce the power consumption of the adders. The current source is implemented using an nFET and is biased in saturation. The current source is always in saturation as $V_{ds} > V_{gs} - V_t$ over the entire range of the input swing. If a bipolar transistor had been used in the place of nFET the current source could be in deep saturation with the voltage levels handled.

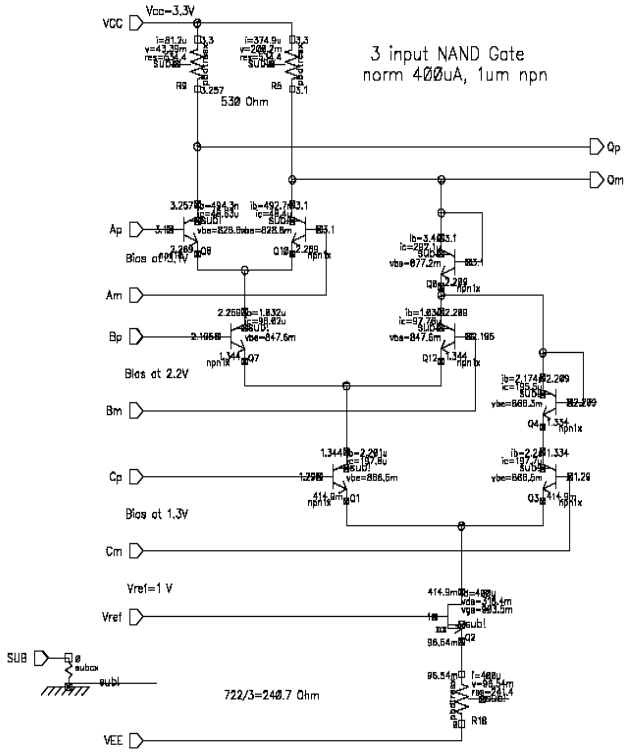


Figure 2.7 – Schematic showing DC analysis of 3 input NAND gate.

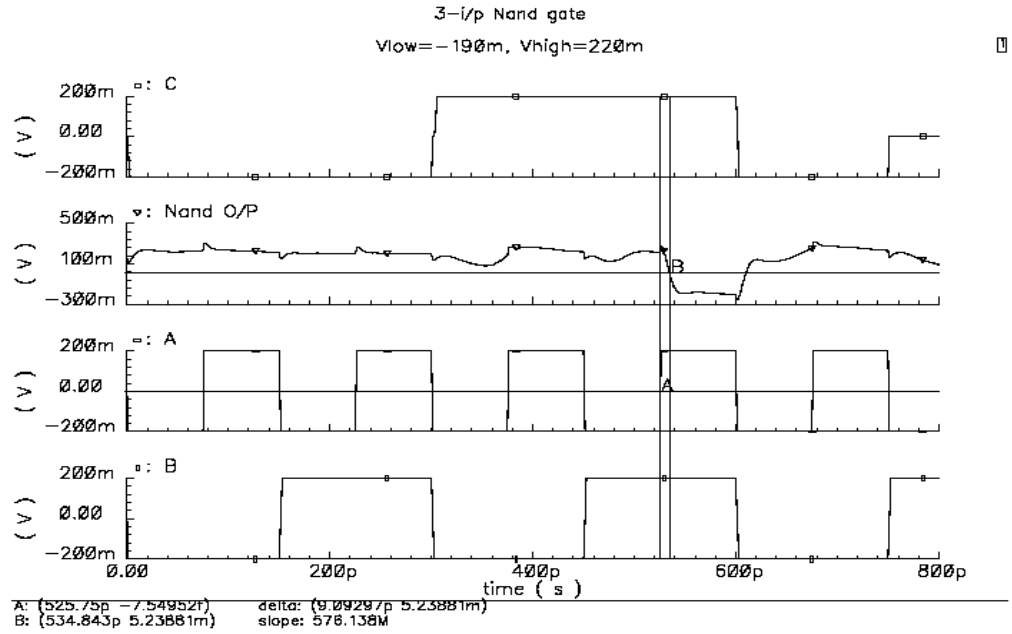


Figure 2.8 – The output of 3 input NAND gate showing a propagation delay of 9ps.

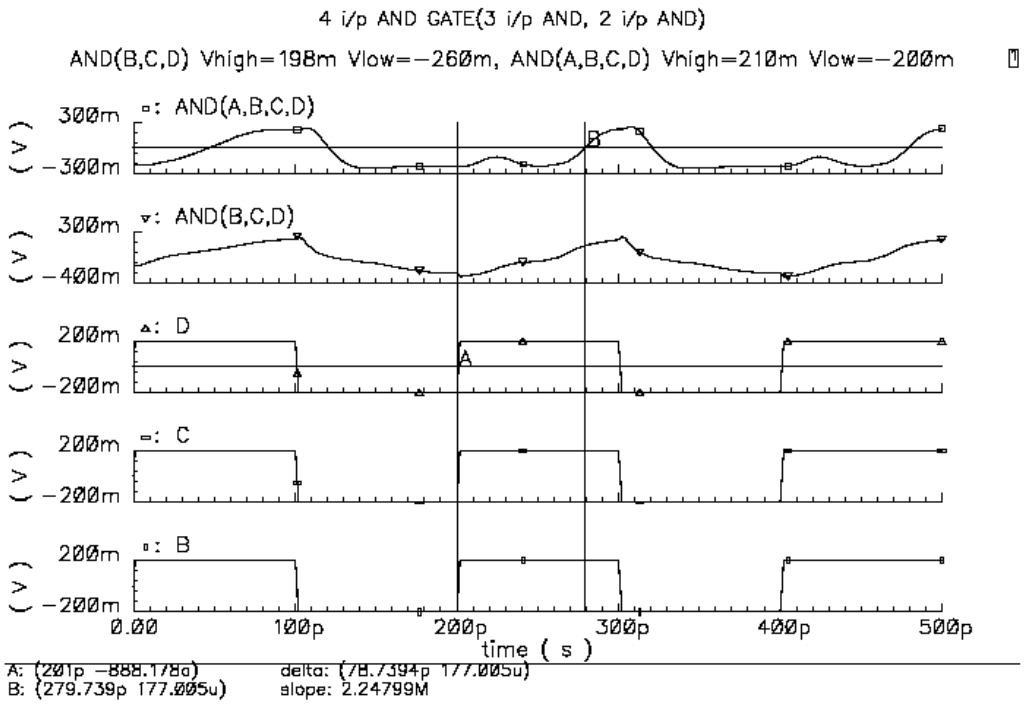


Figure 2.9 – Delay of 3 input AND gate is 50ps and 4 input AND gate is 79ps.

2.5 Pipelined accumulator

The proposed architecture uses an 8 bit pipelined accumulator. Figure 2.10 shows generic architecture of an NxM pipelined accumulator. To realize an 8 bit accumulator we have taken N=1 and M=8. Pipelining enables us to run the NxM bit accumulator at the speed of N bit accumulator. Therefore the 8 bit accumulator runs at the speed of a 1 bit accumulator consisting of a full-adder and a reset flip-flop. Maximum frequency attained by a 1-bit accumulator implemented using CML logic with 400uA tail bias current is about 6.5 GHz. The gate delay to obtain the output includes a delay of the flip-flop and 3-gate delay of the full adder circuit. The accumulator flip-flops are rising-edge triggered master slave flip-flops with reset. The reset is active low asynchronous reset.

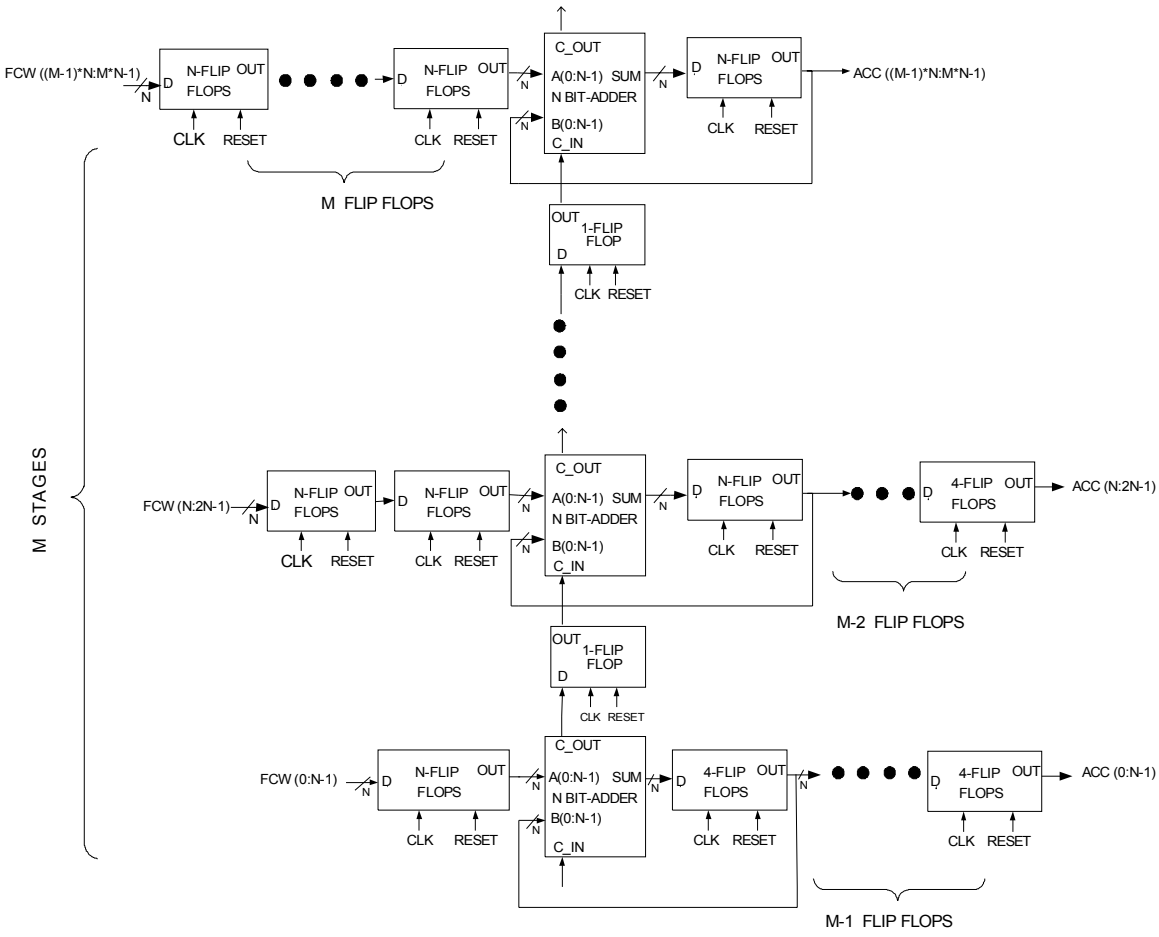


Figure 2.10 - Generic Architecture of N x M Pipelined Accumulator.

The accumulator flip-flops are rising edge triggered master slave flip-flops with active low asynchronous reset. For low supply voltage, the reset transistors cannot be vertically stacked at the bottom of the CML circuitry without saturating the latch transistors. It is rather critical for high-speed circuit to keep all the transistors from saturation. We have used a novel reset CML latch circuitry that implements the reset in a parallel structure with only three levels of transistors including the current source transistor. The proposed reset CML flip-flop circuitry is very suitable for low supply

voltage applications although the delay is little higher. This architecture has a latency period of about 7 clock cycles.

The full adder used in the accumulator architecture is realized using the following equations:

$$CARRY = A \cdot B + B \cdot C + C \cdot A$$

$$SUM = A \oplus B \oplus C \tag{2.7}$$

Maximum frequency attained by a 1-bit full adder is about 10GHz. All the gates used to realize the full adder are two input gates. Hence the maximum delay to obtain the carry in a full adder is 3 gate delays. A fully pipelined 8-bit accumulator contains about 210 gates and consumes about 150mA. The power consumption is about 0.5W and the area occupied is approximately 210 x $(2025um)^2$= 425250 um^2. The simulated output of the 8-bit accumulator operating at 6.5GHz clock frequency is shown in Figure 2.11. The number of clock cycles till the vertical line in the Figure 2.11 indicates the latency period.

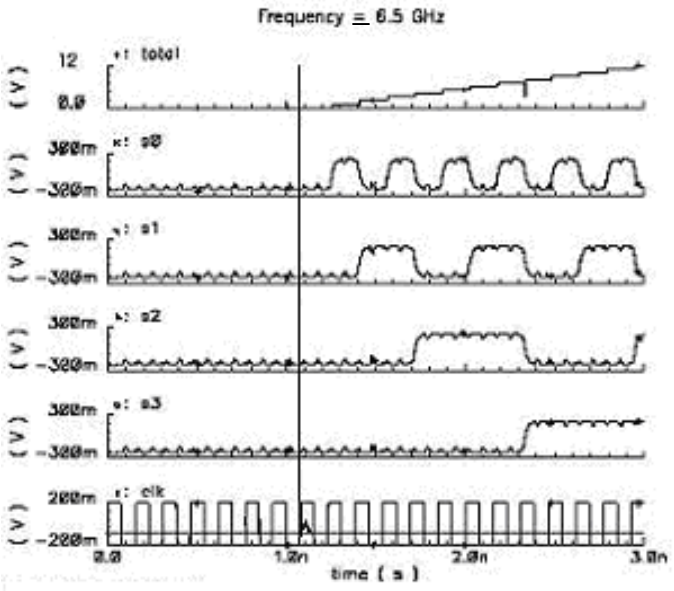


Figure 2.11 - Output waveform of an 8-bit fully pipelined adder.

2.6 New Pipelined Accumulator Architecture

An 8-bit fully pipelined accumulator has been redesigned using a new full adder circuit and resettable flip flops to reduce the power consumption. It is expected to run at almost the same speed as the old accumulator with lower power requirements. Also the area requirements are reduced.

2.6.1 Full adder circuit

A new full adder has been designed by merging the gates to use a single current tail for calculating sum output and a single current tail for calculating the carry output. The equations used are:

$$\begin{aligned} CARRY &= A(B+C)+\overline{A}(B\cdot C) \\ SUM &= A \oplus B \oplus C \end{aligned} \tag{2.8}$$

The circuits have four levels of transistors stacked vertically including the current source transistor. The fastest changing input is assigned to upper level of transistors and the slowest changing inputs are assigned to lower level transistors to gain speed and reduce glitches in the output. This new architecture helps in reducing the power consumption to a great extent without suffering much loss in terms of speed. The current source transistor can be a FET instead of a bipolar to use Vcc = 3.3.V and maintain the current source to be out of saturation.

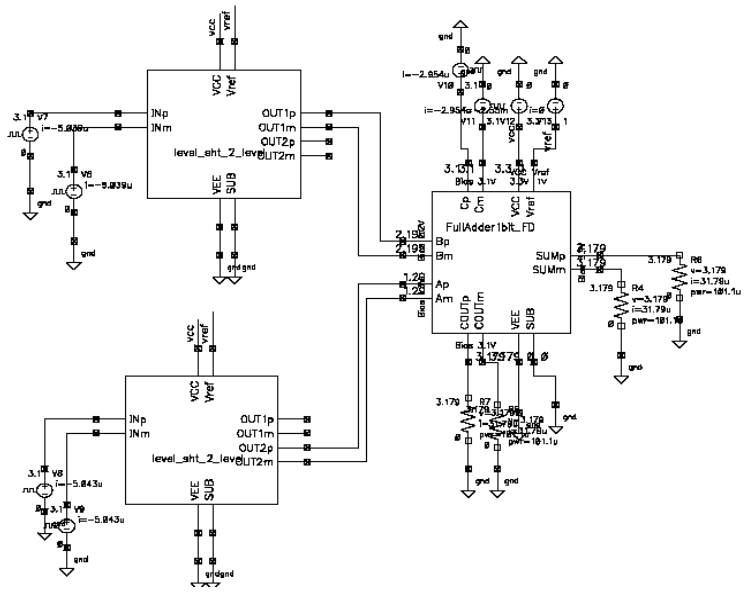


Figure 2.12 – Test bench of new full adder circuit.

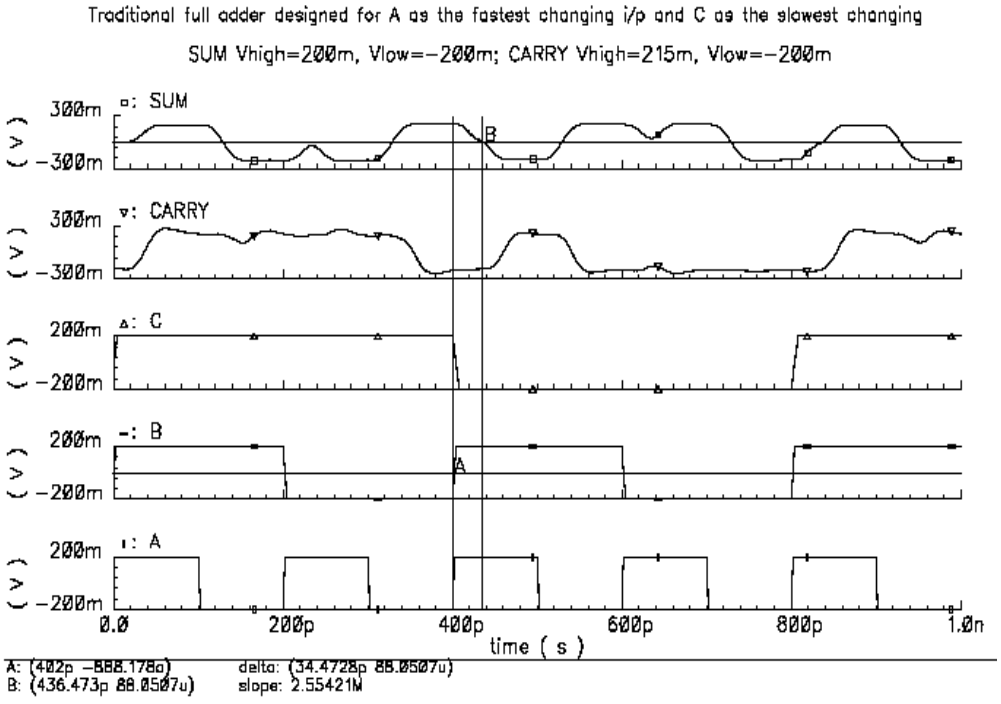
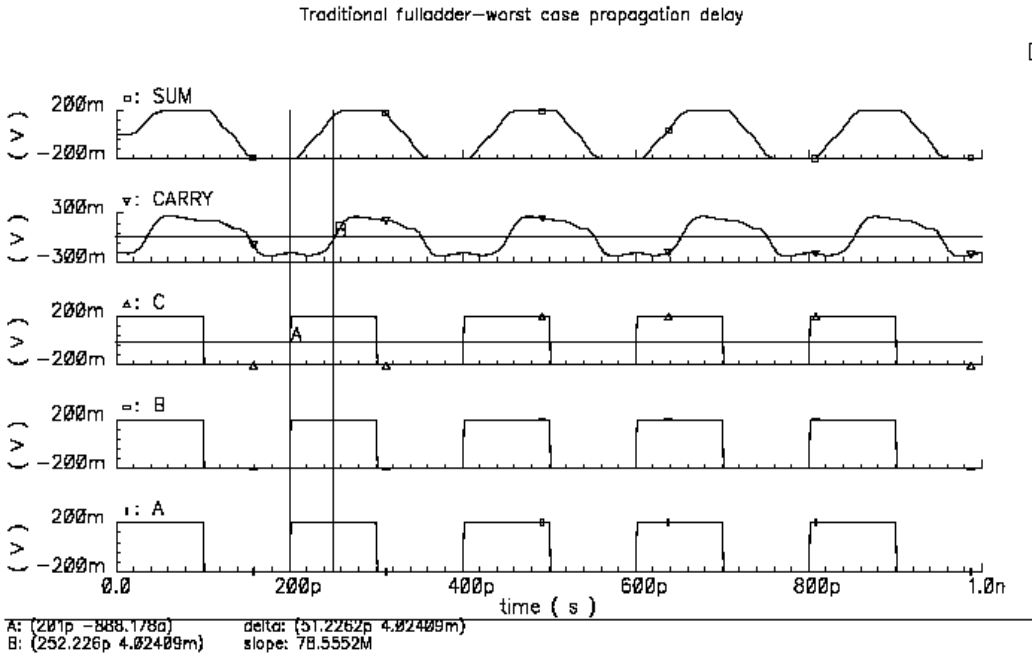


Figure 2.13 - Minimum propagation delay of traditional full adder is 35ps.



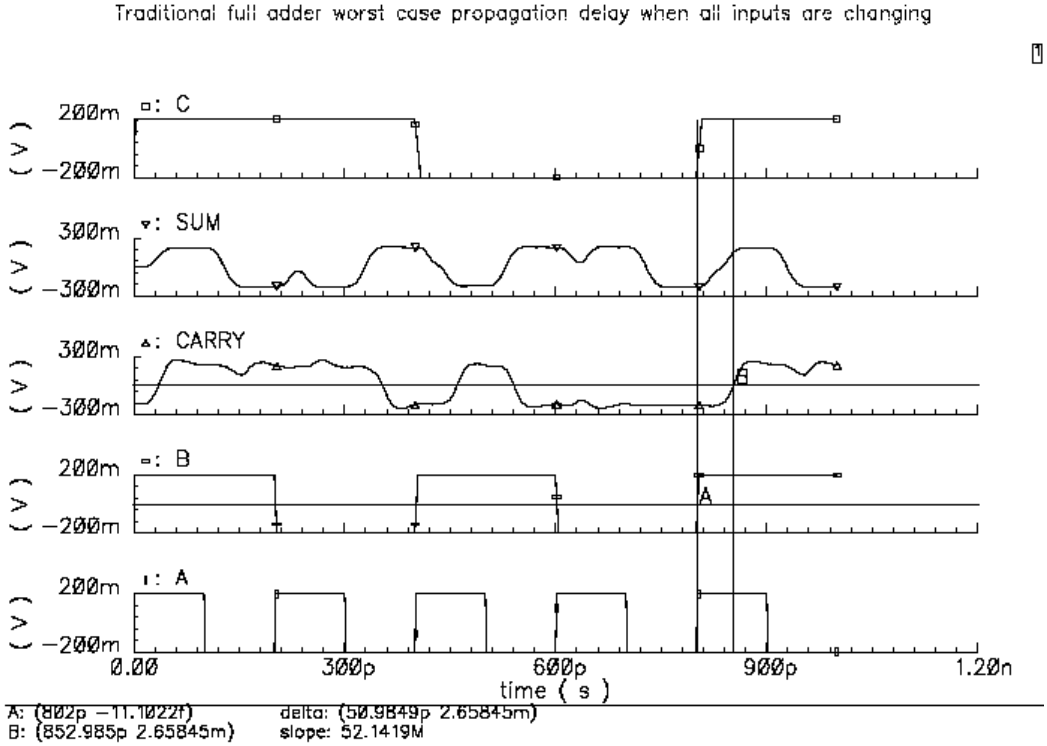


Figure 2.14 - Worst case propagation delay of traditional full adder is 51ps.

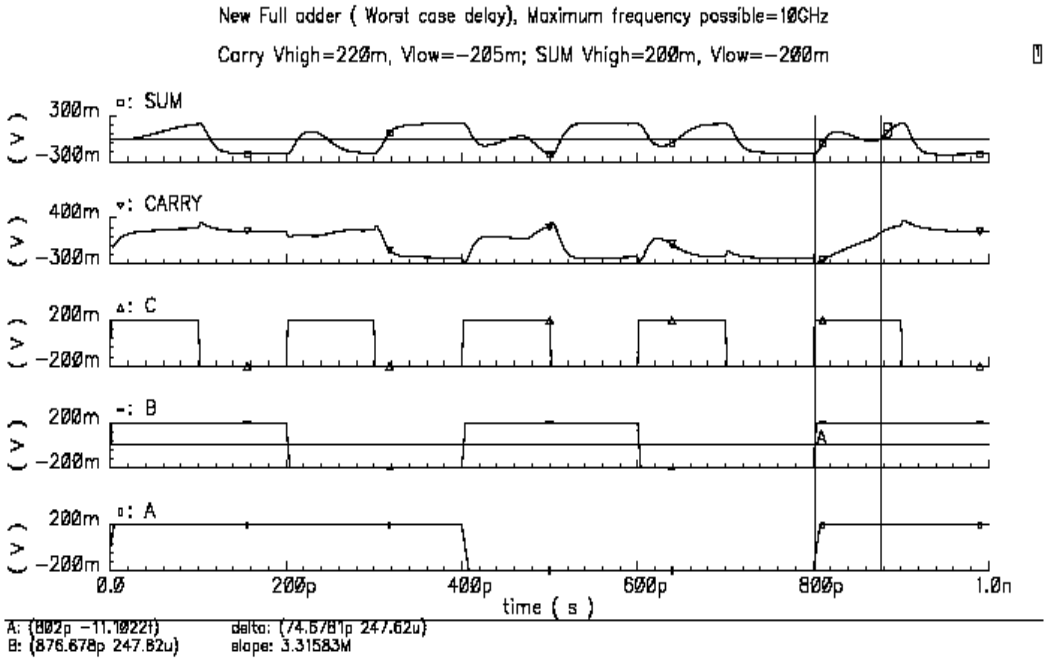


Figure 2.15 – Worst case propagation delay of the new full adder is 75ps. It takes at least 90ps to reach either Vhigh or Vlow.

In the Figure 2.15, Vhigh and Vlow indicate the maximum output swing.

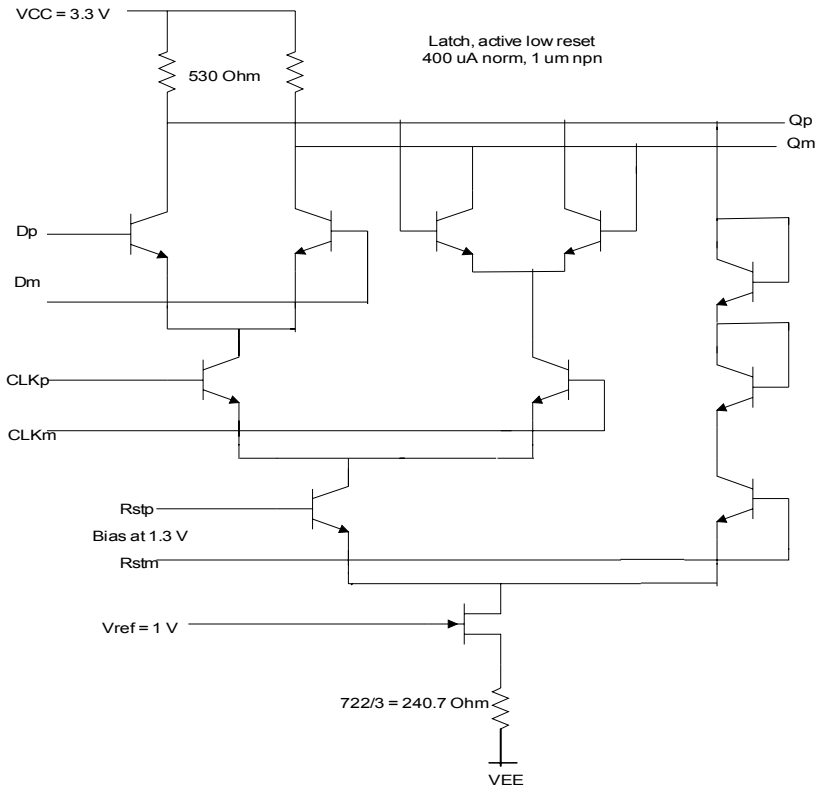


Figure 2.16 - Three level resetable latch.

The maximum speed attained by this full adder is about 10 GHz. The 8bit accumulator built with this full adder is expected to run at 6 GHz.

2.7 Accumulator with Carry look ahead adders

The proposed DDS architecture can incorporate various modulation schemes. The DDS modulation waveform configurations include chirp, step frequency, FM, MSK, PM, AM, QAM, and other hybrid modulations. To allow DDS to operate with modulations, the developed pipelined accumulator needs to be modified to allow variable frequency control words (FCW) as its inputs. In chirp or step frequency modulation the FCW input of the DDS changes continuously. To incorporate frequency modulation techniques in DDS, the latency period has to be reduced.

The 8 bit pipelined accumulator architecture has a large latency period of 7 clock cycles. This latency can be reduced to 1 clock cycle using an 8 bit accumulator with two 4-bit CLA adders (N=4 and M=2 in Figure 2.10). The latency can be reduced to zero using an 8-bit CLA adder with (N=8 and M=1). However, the architectures suffer a considerable reduction in speed. The maximum speed attained by N=4 and M=2 pipelined accumulator with CLA adders is same as the maximum speed attained by a 4-bit accumulator, which runs at maximum speed of 3.5 GHz. The speed of the 4-bit accumulator depends upon the speed of the 4-bit CLA adder and the reset flip-flops. All the gates used are two input gates. The basic building blocks in the adder circuit are NAND gate and EXOR gate. All other logic gates like NOR gate, AND gate and OR gate can be realized from NAND and XOR gates. Differential topology has been used to build the basic building blocks for the improvement of noise characteristics. SiGe npn transistors with an emitter length of 1um have been used. The power dissipation per gate is about 1.32mw. The total delay to obtain the output sum and carry is 6-gate delay for a 4-bit CLA adder, while the delay in a fully pipelined architecture is 3-gate delay. The generation of carries in the CLA can be summarized as:

$$C1 = g_0 + p_0 \cdot c_0 \tag{2.9}$$

$$C2 = g_2 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot c_0 \tag{2.10}$$

$$C3 = g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot c_0 \tag{2.11}$$

$$C4 = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0 \tag{2.12}$$

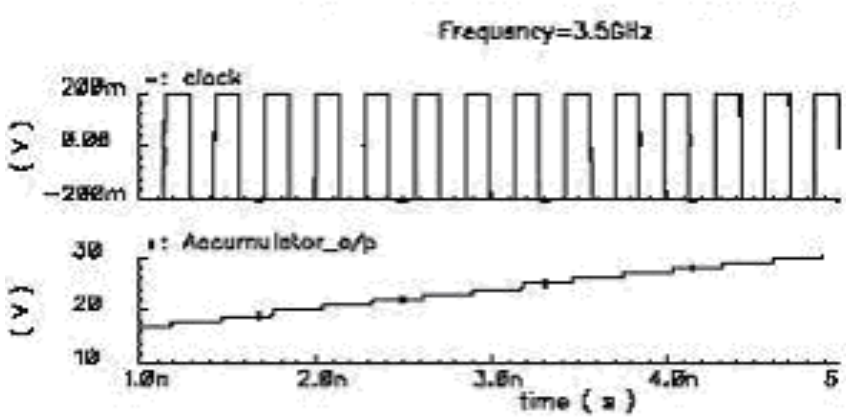


Figure 2.17 - Simulated output of an 8 bit pipelined accumulator with CLA adders (N=4 and M=2) operating at 3.5 GHz clock frequency.

where $g_n = a_n \oplus b_n$ is the carry generator and $p_n = a_n \cdot b_n$ is the carry propagate to calculate (n+1)*th* carry. The schematic of the 4 bit CLA adder has been shown in Figure 2.18.

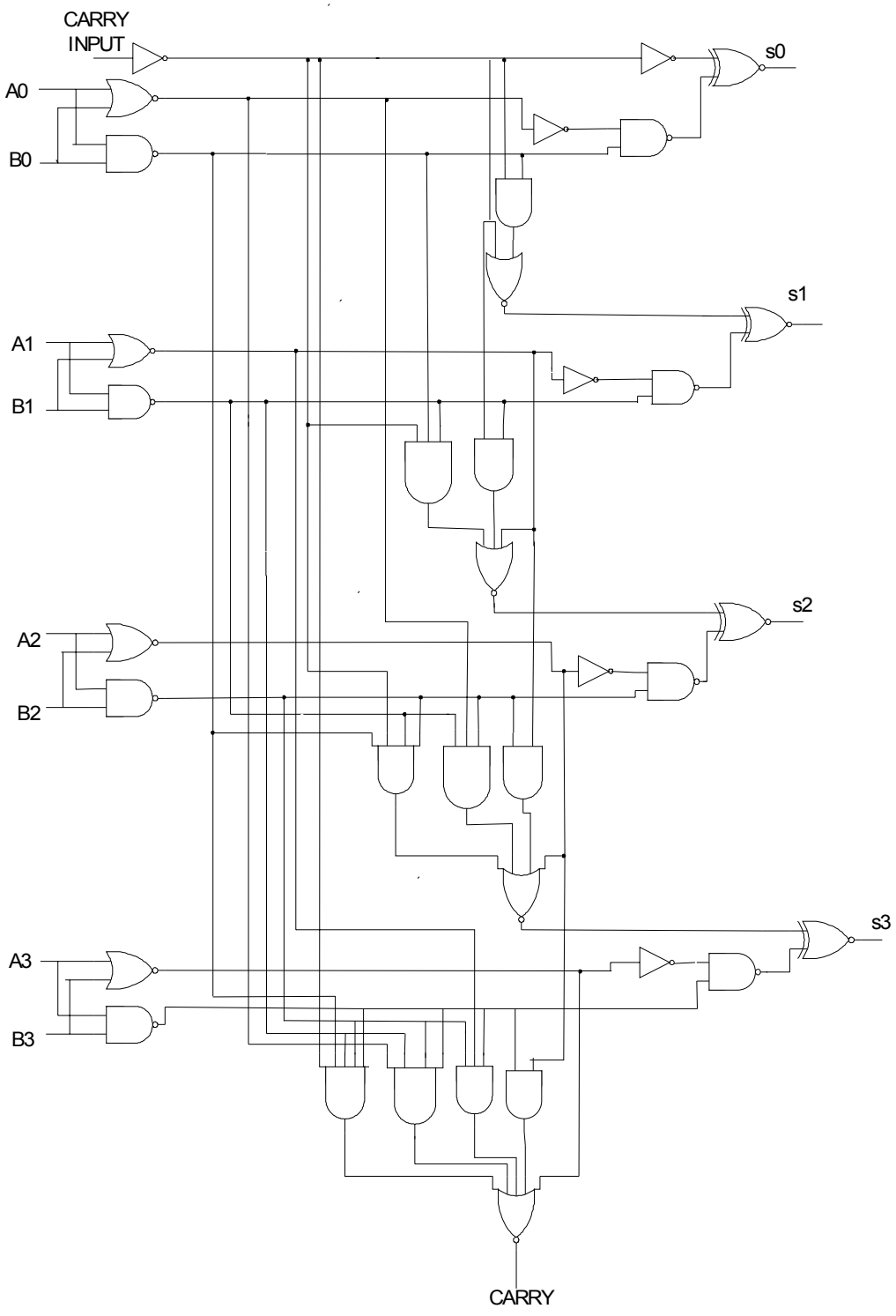


Figure 2.18 - The Gate level 4 bit CLA adder used for implementation of phase accumulator

2.7.1 Carry look ahead adder delay analysis

A CLA adder is constructed according to the carry propagate and carry generate logic. The basic gates have to be wired such that the delays due to pairs of gates are properly pipelined to achieve maximum speed. For the ith input bits of an adder the propagate term is denoted by pi and generate term by gi where,

$gi = xi \cdot yi$ has one gate delay denoted by 1D

$pi = xi \oplus yi$ has one gate delay denoted by 1D

c1=g0+p0.c0

(1D+2D)

= 3D

c2 = (g1+p1.g0) + (p1.p0.c0)

The delay according to the Equation is (1D+2D+3D),

Pipelining the delays will give (1D+2D) +3D

=3D+3D=4D

c3=g2+p2.g1+p2.p1.g0+p2.p1.p0.c0

= (1D+2D+3D+2D.2D)

Pipelining the delays will give

(1D+2D)+3D+ (2D.2D)

= (3D+3D+3D)

=5D

c4= g3+p3.g2+p3.p2.g1+p3.p2.p1.g0+p3.p2.p1.p0.c0

According the equation the delays are (1D+2D+3D+2D.2D+2D.2D.0D) (Considering all the gates to be two input gates).

= (1D+2D) +3D+ (2D.2D) + (2D.3D)

=3D+3D+3D+4D=3D+ (3D+3D) +4D

=3D+4D+4D= (3D+4D) +4D=5D+4D

=6D.

The equation can now be represented with brackets of precedence as

c4= ((g3+p3.g2) + (p3.p2.g1+ ((p3.p2).(p1.g0))))+((p3.p2).(p1.p0.c0))

2.8 Modifications to the 4 bit CLA Adder

2.8.1 Accumulator

In the new architecture of carry look ahead adder, two 3 input building blocks have been used. The two building blocks are A+B.C and A.B.C, where A, B, C are input signals to the gates. Apart from these, 2 input EXOR, OR and AND gates are also used. The 3-input gates are designed with the same power supply VCC=3.3 V, to reduce the total power consumption and area. Two different level shifters, shifting voltage levels from 3.3V to 2.2V and from 2.2V to 1.3V are required. Using three input gates the number of level shifters required is also reduced.

The generation of carry in the CLA can be summarized as:

$$c1 = g_0 + p_0 \cdot c_0$$
$$c2 = (g_1 + p_1 \cdot g_0) + (p_1 \cdot p_0 \cdot c_0)$$
$$c3 = ((g_2 + p_2 \cdot g_1) + ((p_2 \cdot p_1 \cdot g_0) + p_2 \cdot (p_1 \cdot p_0 \cdot c_0)))$$
$$c4 = ((g_3 + p_3 \cdot g_2) + ((p_3 \cdot p_2 \cdot g_1) + p_3 \cdot (p_2 \cdot p_1 \cdot g_0))) + ((p_3 \cdot p_2) \quad (p_1 \cdot p_0) \cdot c_0)$$

The brackets in the expression provide the order of precedence in which the operations are performed. This specific order has been chosen to obtain minimum power consumption and gate delay. To decide this order, comparisons have been made between

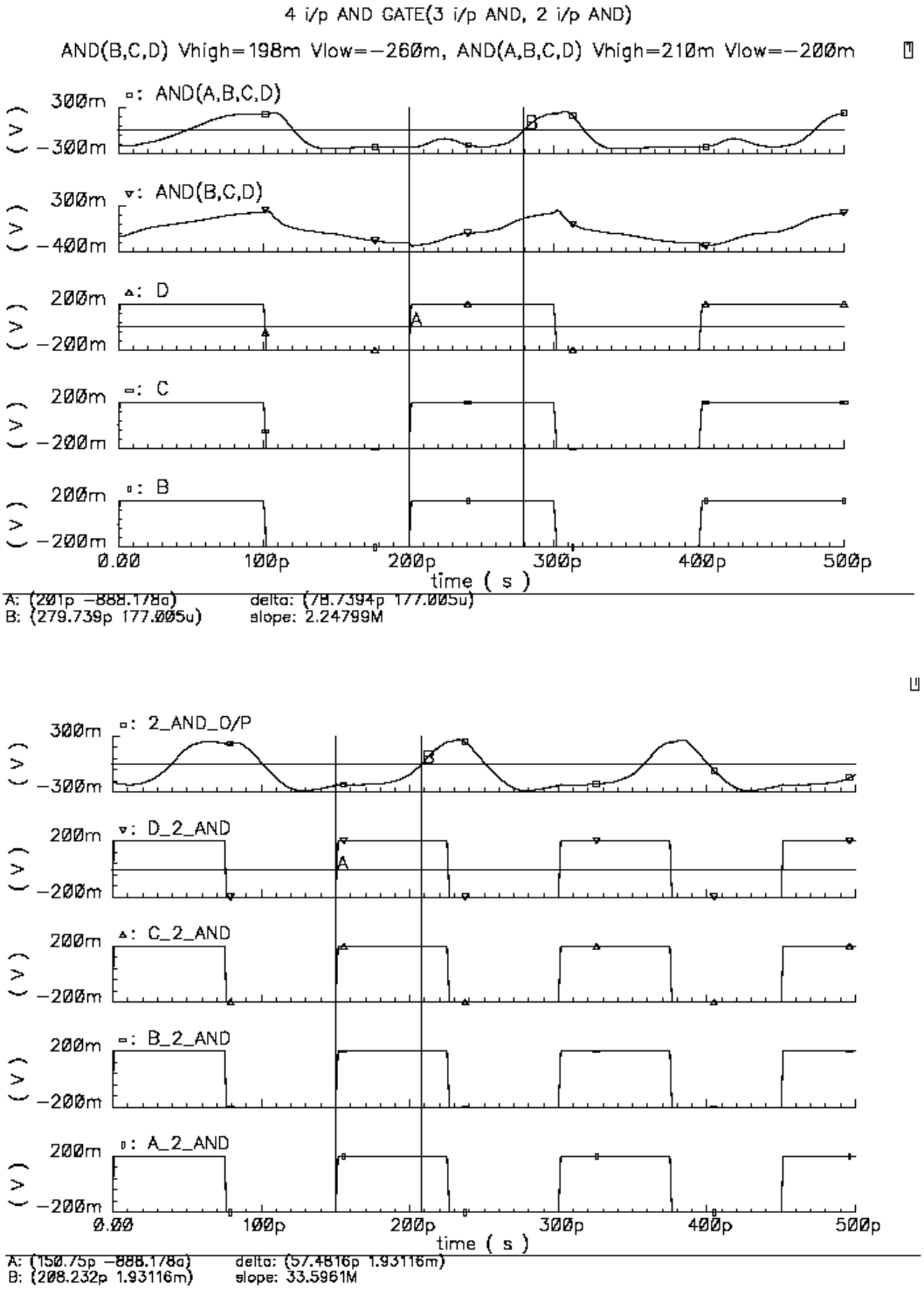


Figure 2.19 - The comparision shows that 2D2 (delay of two 2-input AND gates) is smaller than 1D3+D2 (delay of 3-input AND gate and a 2-input AND gate).

the delay of 4-input AND gate using a 3-input AND gate and a 2-input AND gate and that using two 2-input AND gates as shown in Figure 2.19.

For the three input gates the slowest changing input is assigned to the last stage of transistors for better speed performance.

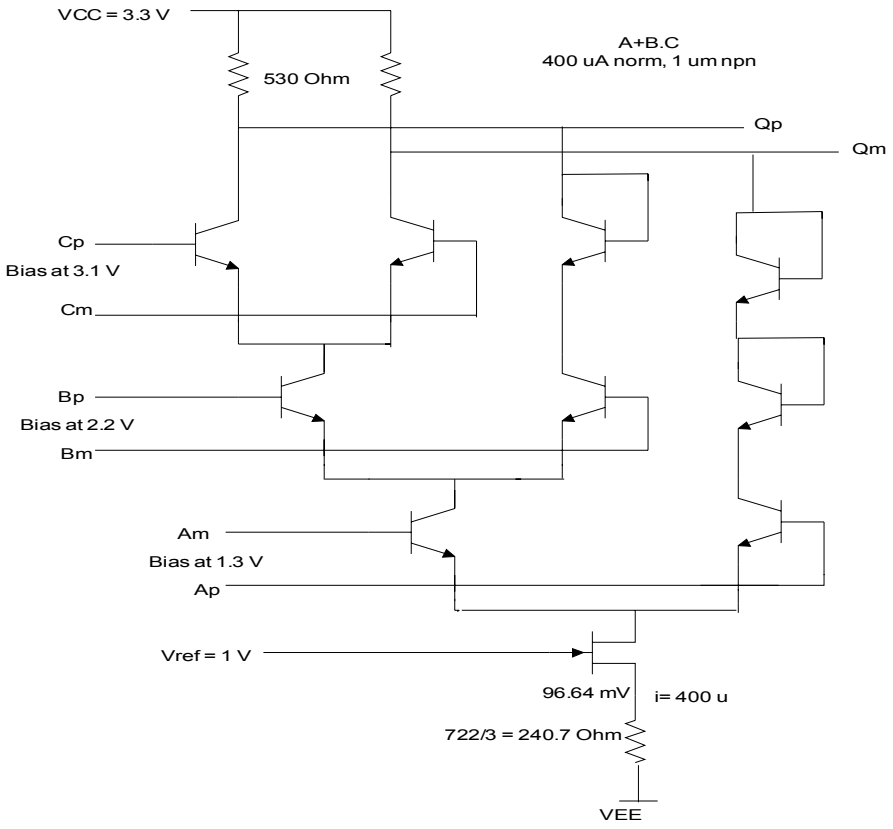


Figure 2.20 – Circuit diagram to implement logic function A+B.C.

2.8.2 Carry look ahead adder delay analysis

Consider a 3 input gate, for example A+B.C, from the figure the fastest changing input should be connected to C and the slowest changing input should be connected to A, for optimum speed. We can assign numbers to the amount of delays, the delay from input A to output can be assigned number 2. The delay from B to output as 1 and from C to output as 0. The connections have to be made according to the inputs indicated by the delay number on each gate as shown below.

$$C_1 = g_0 + p_0 \cdot C_0$$
$$(2) \quad (0)\ (1)$$

$$C_2 = (g_1 + p_1 \cdot g_0) + (p_1 \cdot p_0 \cdot C_0)$$
$$(2) \quad (0)\ (1) \qquad (1)\ (0)\ (2)$$

$$C_3 = (\,(g_2 + p_2 \cdot g_1\,)) + ((\,p_2 \cdot p_1 \cdot g_0) + p_2 \cdot (p_1 \cdot p_0 \cdot C_0))$$
$$((2) \quad (0)\ (1)) \qquad ((0)\ (1)\ (2) \qquad (1)\ (0)\ (2))$$
$$(2) \qquad (1) \qquad (0)$$

2.9 Cosine Weighted Digital to Analog Converter [7]

The high-speed DDS utilizes a cosine-weighted DAC operating in current mode, which does not require op-amp buffer at the output and its speed would not be limited by the bandwidth of the op-amp. The DAC contains a current-cell matrix [4] as shown in Figure 2.21. Each DAC cell outputs a current proportional to cosine value of the corresponding phase indicated by the a+b bits. The sinusoidal output is obtained by summing the output currents from all the cells through an external pull-up resistor. An npn transistors with minimum emitter length of 1um can be used to switch the current cells. The bias current should be carefully chosen considering the speed, power

consumption and DAC output full scale voltage swings. The accuracy of the bias current is ensured using an optimized band-gap reference design with calibrating capability. Dynamic performance of the DDS rapidly degrades with frequency due to transient glitches in the DAC. These glitches can be minimized by using thermometer decoding scheme that ensures the minimum number of cells switching simultaneously. In addition, all switching control signals should be buffered to ensure differential synchronous switching for all cells.

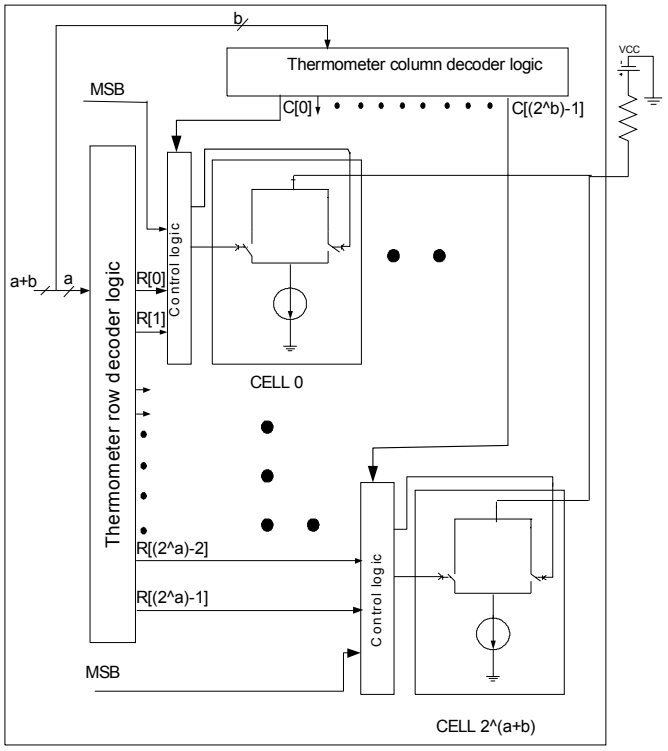


Figure 2.21 - Nonlinear DAC with current cells.

The DAC control logic is shown in Figure 2.22, where the signal MSB represents the MSB bit of the phase accumulator output. Signals A and B are differential pair signals which control the current switches in the DAC cell.

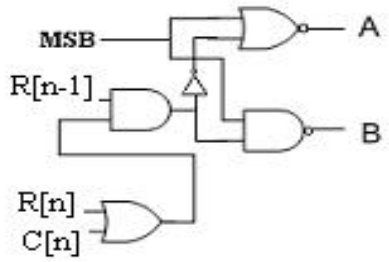


Figure 2.22 - Control logic of the nth DAC cell.

CHAPTER 3

NOISE ANALYSIS OF DDS OUTPUT SPECTRUM

DDS (direct digital synthesizer) has its applications in radios, instrumentation, and radar systems etc. Though large and unpredictable spurious responses have troubled old designs, innovations have improved DDS performance, and the worst-case spurs are now smaller and predictable. Careful frequency planning allows us to place the worst-case spurs outside the bandwidth of interest, so that they can be easily filtered. Most DDS applications use only a fraction of their output spectrum and attenuate the remainder with external filters. The bandwidth of interest is typically from 0 Hz to about 40% of the sampling frequency. The sub-Nyquist limitation is due to the transition band of the external image-rejection filter. Some applications can use the image band and eliminate an upconversion stage, but the reduced power in the image lowers the SNR. Image use also requires bandpass filtering rather than a lowpass filter. The DAC's zero-order sample-and-hold imparts a Sinc (sin(x)/x) attenuation envelope to the fundamental, images, and harmonics in the DDS spectrum.

A DDS has four principal spur sources: the reference clock, truncation in the phase accumulator, angle-to-amplitude mapping errors, and DAC error terms, including nonlinearities and quantization noise as shown in Figure 3.1. The spur frequencies' predictability allows you to develop an effective frequency plan.

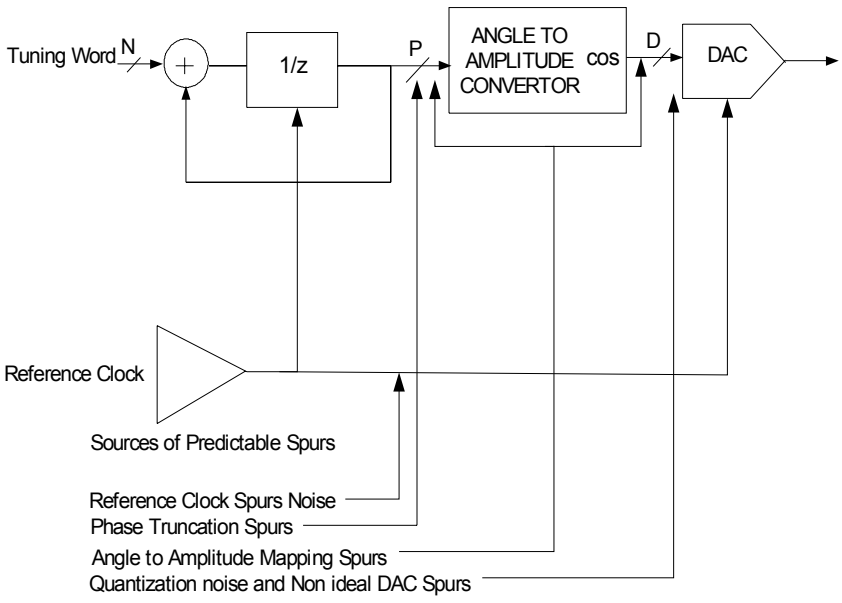


Figure 3.1 - DDS has four principle sources of spurs.

3.1 Reference clock

A DDS functions like a high-resolution frequency divider with the reference clock as its input and the DAC as its output. The spectral characteristics of the reference clock directly impact those of the output. Though phase noise and spurs on the reference clock also appear at the DAC output, they do so at a reduced magnitude due to the frequency division. The improvement, expressed in decibels, is 20 log(N), where N is the ratio of input to output frequencies. For example, two trials dividing a 300-MHz clock down to 80 and 5 MHz result in a difference in their phase-noise plots of 20 log(16)=24 dB. The DDS's internal reference-clock path is the dominant contributor of phase noise from the DDS. Modulating the clock amplitude generates spurs in its output spectrum. If a 400-MHz RF carrier with 10% AM (amplitude modulation) by a 100-kHz sine-wave signal is observed at reference clock and 10.119-MHz DDS outputs, this effect can be demonstrated. The Figure 3.2 [2] which superimposes the reference-clock and DAC-

output spectra, shows the amplitude reduction in AM clock spurs at the DDS output. The attenuation calculation, 20 log (400 MHz/10.119 MHz), predicts a 32-dB improvement, although the plot shows more. The additional spur attenuation is due to the fact that the modulated sine wave of the reference-clock signal encounters a limiter or squaring circuit at the DDS input. The limiter stage converts the sine wave to a square wave, and the AM spurs are thus converted to PM (phase modulation) spurs. The AM-to-PM conversion results in an additional attenuation of spurs that depends on the characteristics of the limiter circuit but is typically on the order of –6 dB.

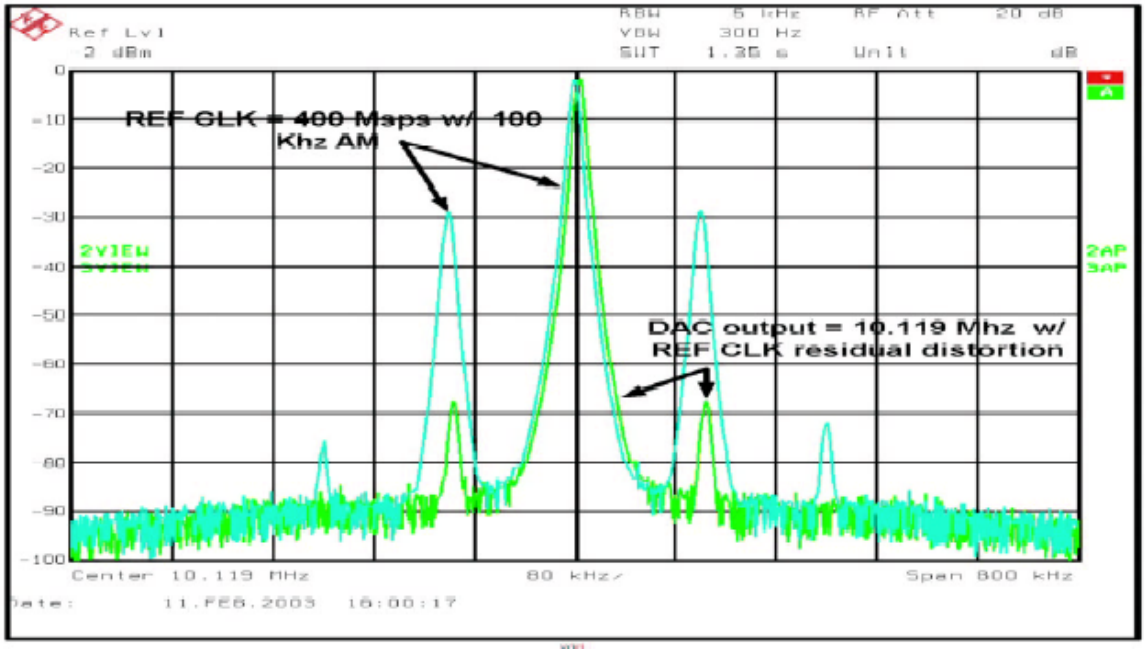


Figure 3.2 - Spurs caused by modulation of the reference clock amplitude are also reduced by 20log(N). The DDS's input limiter, which converts the amplitude modulation into a phase-modulation term, provides additional suppression of this spur.

The reference clock imposes limits on DDS performance in ways that are often recognizable. The reference clock usually causes those DDS spurs that maintain their relationship to the carrier as you change the output frequency. Also, there is some degree

of noise at the input of any circuit. A high-slew-rate reference clock spends less time traversing the region where noise can cause jitter.

3.2 Phase truncation

Consider a DDS with a 32-bit phase accumulator. If the design maintained all 32 bits throughout, the DDS core would occupy a large die area and dissipate significant power. Truncating the value from the phase accumulator i.e., passing only the accumulator's most significant bits to the angle-to-amplitude mapper reduces the power dissipation and die area as well as the phase resolution of the angle-to-amplitude mapper.

The DDS's phase-truncation spur mechanism models as a noise source summed with an

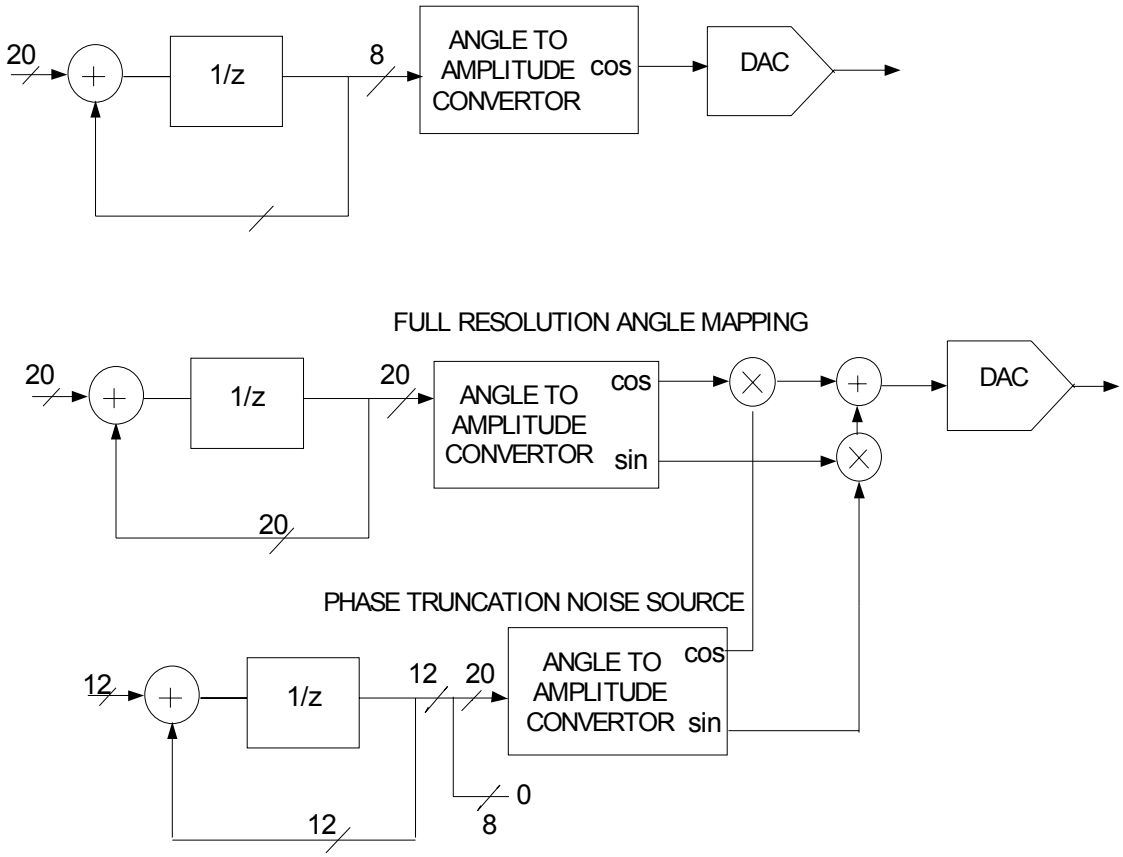


Figure 3.3 - The DDS's phase truncation spur mechanism models as a noise source summed with an otherwise ideal synthesizer.

otherwise ideal synthesizer as shown in Figure 3.3. The example truncates a 20-bit phase accumulator to 8 bits. M is the phase accumulator's tuning-word width. Each update of the reference clock adds the value of M to the accumulator output. The output is divided into two sections, the truncated phase word, P, which is sent to the mapper, and the discarded bits, D=M–L. As the value in the discarded section accumulates, it eventually overflows into the truncated phase word. One effect of this overflow is production of phase-modulation spurs. A second effect is that those overflows maintain the full-frequency resolution of T. Note, if no bits in the discarded portion are set to logic one, then no phase-truncation spurs occur.

3.2.1 Mathematical Analysis of Phase truncation spurs [8]

When the input frequency control word, FCW, is represented in M-b, the M-b phase value of the phase accumulator is updated as

$$\Phi[n+1] = (\Phi[n] + FCW) \bmod 2^M \tag{3.1}$$

and truncated to an L-b phase value. Then, the truncation error of can be obtained in a recursive equation given by

$$\Psi[n+1] = (\Psi[n] + R) \bmod 2^{M-L} \tag{3.2}$$

where R is a least significant (M-L)-b value of FCW given by

$$R = FCW - \left[\frac{FCW}{2^{M-L}}\right] \times 2^{M-L} \tag{3.3}$$

Finally, the cosine output of the DDFS becomes

$$x[n] = \cos\left(\frac{2\pi(\Phi[n] - \Psi[n])}{2^M}\right) \tag{3.4}$$

$$= \cos\left(\frac{2\pi\Phi[n]}{2^M}\right)\cos\left(\frac{2\pi\Psi[n]}{2^M}\right) \tag{3.5}$$

$$+ \sin\left(\frac{2\pi\Phi[n]}{2^M}\right)\sin\left(\frac{2\pi\Psi[n]}{2^M}\right) \tag{3.6}$$

In this case, the cosine corresponding to $\Phi[n]$only has a desired frequency component. $\Psi[n]$ is periodic as well as $\Phi[n]$, and the period of $\Psi[n]$is $R/2^{M-L}$. In the case of$L>2$, the period of $\cos(2\pi\Psi[n]/2^M)$and $\sin(2\pi\Psi[n]/2^M)$ is determined by$\Psi[n]$. Thereby, the periodic truncation error creates spurs at the harmonic frequencies of $R/2^{M-L}$. If the input frequency control word is decomposed as $FCW = N\times 2^{M-L}+R$, the L-b truncated phase value is increased by N or N+1, and the rate of the increment by N+1 is given by $R/2^{M-L}$. The mean value of the increment observed at the L-b phase value is given by $N+R/2^{M-L}$. Accordingly, the conventional M-b phase accumulator with the L-b truncation can be equivalently decomposed into two accumulators for L-b and (M-L)-b. The update of the (M-L)-b accumulator corresponding to the truncation error is equal to

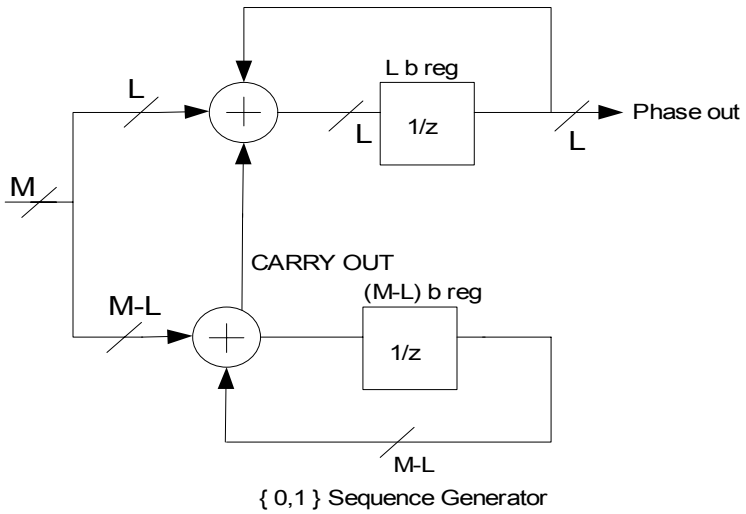


Figure 3.4 - A conventional phase accumulator representation with M bits truncated to L bits.

Equation 3.2. The L-b accumulator is increased by N+1 when the (M-L)-b accumulator is overflowed. Otherwise, it is increased by N. Thus, the (M-L)-b accumulator can be simply considered as a {0, 1} sequence generator controlled by the (M-L)-b control word of R. In the conventional phase accumulator, the {0, 1} sequence from the (M-L)-b accumulator is periodic, because the truncation error is periodic. Figure 3.5 and Figure 3.6 shows the spectrum of the {0, 1} sequence generated by (M-L)-b accumulation of the conventional phase accumulator. These harmonic tones contribute to the spurs in the final output. If the periodicity of the {0, 1} sequence is eliminated, the power of the spurs can be reduced significantly.

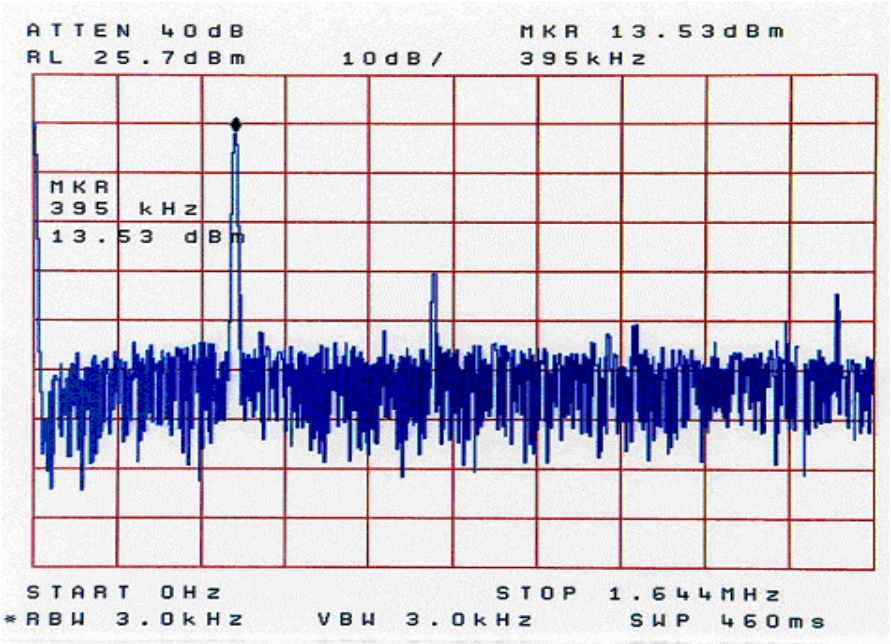


Figure 3.5 – Spurs observed in the output spectrum due to phase truncation, Spectrum for FCW = $2^{9}+1$,

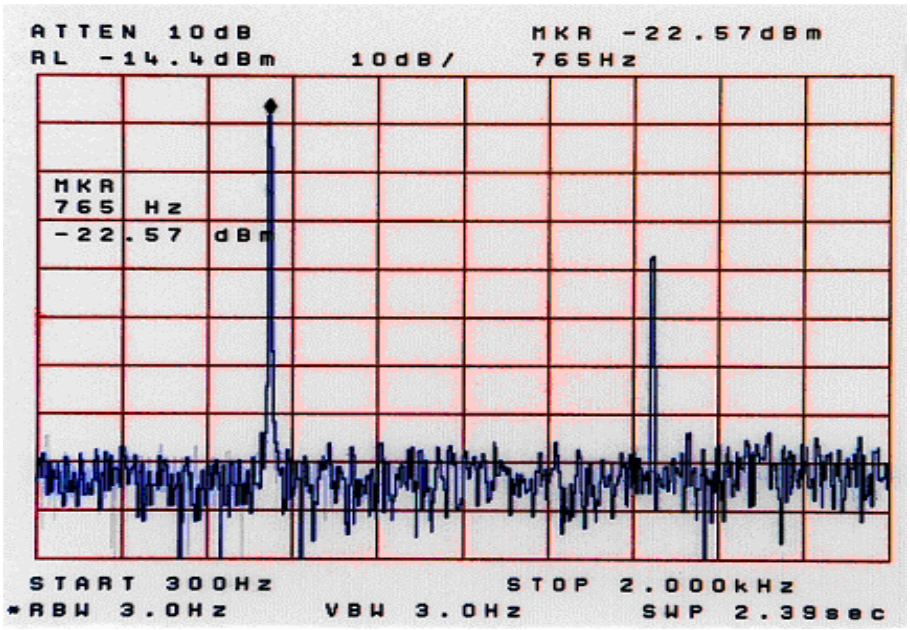


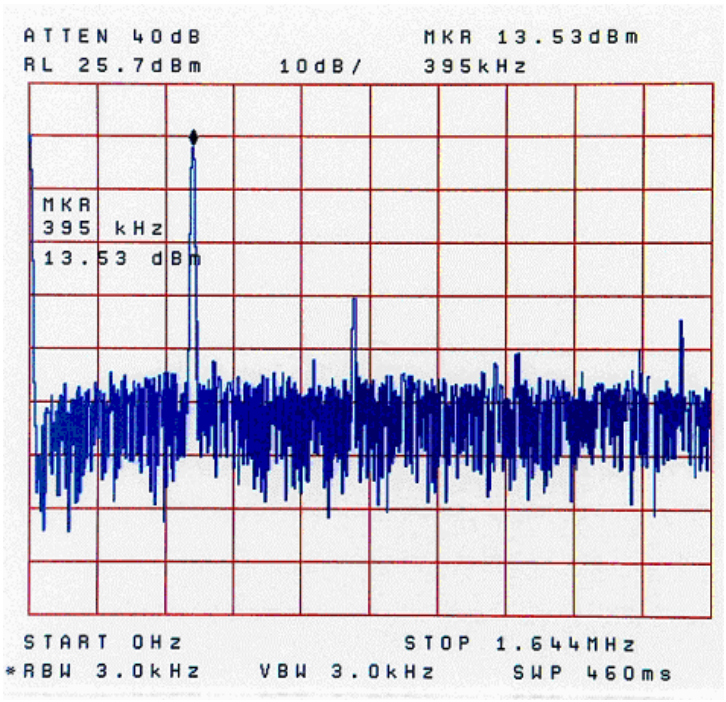
Figure 3.6 – Spurs observed in the output spectrum due to phase truncation, Spectrum for FCW = 1.

The phase truncation spurs for a phase accumulator of 15 bits truncated to 9 bits, DAC resolution of 8 bits at 25 MHz and FCW = 2^9+1 is shown in Figure 3.6. Figure 3.7 shows phase truncation spurs for FCW=1. It can be observed that the decibel level of phase truncation spurs do not change with FCW. Phase-truncation spurs are proportional to the LSB weight in the phase word and, therefore, are not typically issues. Conversely, in applications in which a DDS drives a PLL, phase-truncation spurs within the loop bandwidth of the PLL will be amplified by 20 log(N) dB, where N is the PLL-multiplication factor. The accumulator's phase-word output should be 3 or more bits wider than the DAC resolution. It can be calculated that the worst-case spurious magnitude attributable to the phase-word width is –6.02P dBc, where P is the number of bits in the phase word. So, a 12- to 19-bit phase resolution produces a –72- to –114-dBc spurious magnitude. But the 8 bit DAC that has been used to observe the results has

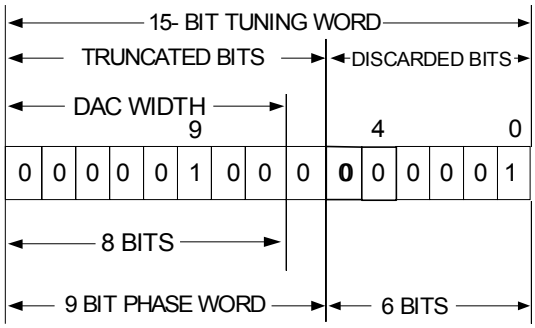
about 5~6 effective number of bits that result in about 30~36 dB quantization noise floor which has been observed during the test. Thus the worst case spurious response is 30~36 dBc instead of the expected 48 dBc. The tuning-word width, its fraction disposed of through truncation, and the reference-clock frequency combine to reveal the frequency offset of the truncation-spur phase-modulated sidebands.

$$f_s = f_{REF} \frac{M}{2^N} \tag{3.7}$$

where f_S is the spur offset frequency, f_{REF} is the reference-clock output frequency, M is the decimal value of the discarded bits, and N is the number of discarded bits. These sidebands appear on both sides of the fundamental. If the offset frequency is greater than the output frequency or greater than the difference between the output and the Nyquist frequencies, the sidebands fold around dc, Nyquist, or both. Figure 3.7 shows phase-truncation spurs that were generated from a DDS with 9-bit phase resolution and a 15-bit tuning word. The first phase-truncation sideband spurs are approximately –30 dBc as it would be expected from the resolution of DAC.



(a)



$$\text{PHASE TRUNCATION SPUR SPACING} = \frac{\text{DECIMAL EQUIVALENT OF DISCARDED BITS}}{2^{\text{NUMBER OF DISCARDED BITS}}} \times f_{REF}$$

b)

Figure 3.7 - The phase truncation spurs are offset from the fundamental by 390 kHz and its harmonics (a) The reference clock rate and analyzing the 15 bit tuning word reveal this fact. (b) The calculation shows how to predict the 390.625 KHz spur spacing.

3.3 Over sampling

The carrier to noise spectral density varies with the FCW selected by the user. The worst case spurious magnitude is given by -6.02N where N is the number of bits in the phase register used to address the ROM. An improvement in the output spectral density can be observed by reducing the FCW. For a DDS with N=15 bit, FCW=1 and f_{clk}=25 MHz

$$f_{out} = \frac{25M}{2^{15}} \times (1) = 762.93Hz$$

The Nyquist frequency is given by $2 f_{out}$. The over sampling ration is defined as

$$OSR = \frac{f_s}{2f_{out}} = \frac{2^N}{2FCW} = \frac{2^{N-1}}{FCW}$$

For FCW = 1, the over sampling ratio is 2^{14}. For every doubling of over sampling ratio over worst case (OSR =1, FCW = 2^{14}) an improvement of 3dB can be observed in the in-band quantization noise spectrum. In this case an improvement of 14 x 3 = 42 dB will be observed. The quantization noise floor due to the 8bit DAC/ADC (effective 5~6 bits) is at -30dBc, thus the quantization noise floor with OSR = 2^{14} is below the carrier by

$$-30dBc - 3dB \times 14 \cong -72dBc$$

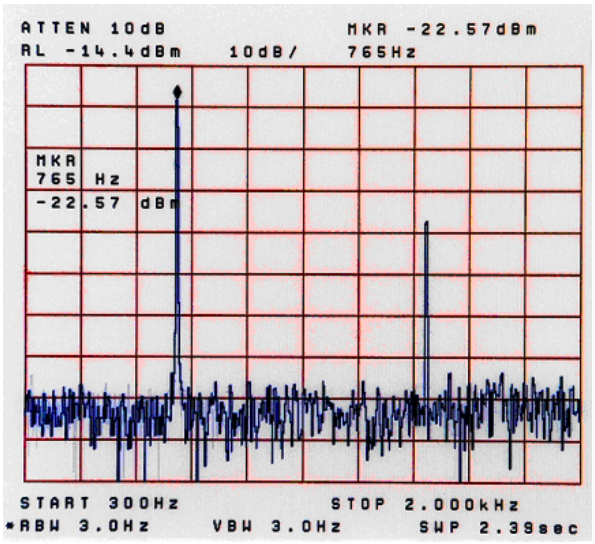


Figure 3.8 - Spectrum of the DDS with a traditional ROM showing the carrier signal at about 762.93Hz and worst case spur for FCW = 1, phase accumulator = 15 bits, DAC resolution = 8 bits, clock freq. = 25MHz.

3.4 Angle-to-amplitude mapping

DDS designs can implement the phase-to-amplitude block algorithmically, which reduces die area and power consumption or as a ROM look-up table. The algorithmic approach enhances hardware efficiency, but its approximations may generate higher spur levels.

The phase-to-amplitude conversion of a time-sampled sine wave is

$$Va_i = V_p \sin(\varphi_i) \tag{3.8}$$

where Va_i is the sample amplitude, Vp is half of the DAC's full-scale voltage, and Φ_i is the value of the sample's phase word.

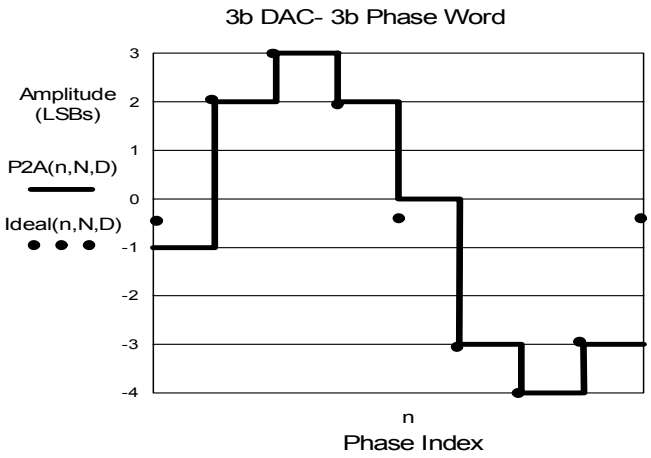


Figure 3.9 - Limited phase resolution results in skipping DAC codes. The dots at each phase increment show the calculated Va_i.

It's unlikely that the Va_i that the system calculates corresponds exactly to a DAC code, so the DDS selects the nearest code, resulting in a residual error. If the phase word has too few bits, the Va_i calculation may skip over DAC codes (Figure 3.9). Conversely, retaining more bits in the phase word reduces these errors. To guarantee that all DAC codes are available to the phase-to-amplitude converter, a good rule of thumb is to set the phase word to a minimum of 3 bits wider than the DAC. A Fourier transform of the time-domain sine plot would display a corresponding spectral plot with discernible frequency spurs. The error can be interpreted as modulating signal acting on the sine wave. The resulting spurs' frequency locations can be determined and their amplitudes can be approximated, although the amplitudes are subject to some architectural dependencies. In some DDSs, the amplitude of the worst-case spurs ranges from –12 to –24 dB below the DAC-quantization-noise level. The quantization noise (SNR) is proportional to the DAC resolution:

$$SNR = 6.02N + 1.76(dB) \tag{3.9}$$

where N is the DAC resolution in bits.

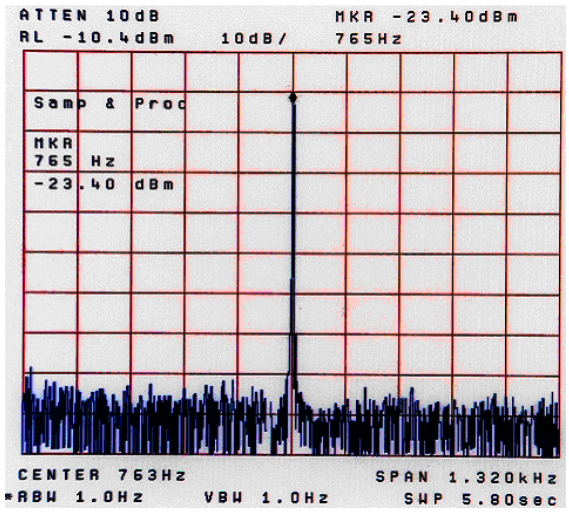


Figure 3.10 - Spectrum showing the carrier signal and quantization noise floor for FCW=1.

A normalized spectral plot of a simulated look-up-table based DDS displays the spurious response that the amplitude-error signal causes (Figure 3.10). The DDS tuning word is 15 bits, the phase word is 9 bits, the reference clock is 25M samples/sec, tuning word, FCW=1 and the DAC resolution is 8 bits(effective 5 – 6 bits due to board errors). The location of spurs due to quantization noise depends on the tuning word and the DDS architecture. Their power level is below the DAC's expected SNR, which is a goal of the design. A DAC with greater resolution would decrease the magnitudes of these spurs.

Instead of performing a Fourier transform of the amplitude-error signal, a relatively simple method determines the frequency of the most pROMinent spurs for a given tuning word. This method uses a test tuning word with only one bit set. The resulting spectrum consists of the test carrier and its spurs, the offsets and spacing of which harmonically relate to the carrier frequency. The spectral region where these spurs reside is

$$f_s = f_{REF} \frac{\pi}{2^{b\ n}} \tag{3.10}$$

where b represents the location of the single bit asserted in the tuning word, and n is the DAC resolution in bits. This analysis method proceeds through nine steps:

Accurately measure and record the reference-clock frequency, f_{REF}, to a ±1 Hz tolerance.

Counting from the MSB of the tuning word, assert only the bth bit. The bth bit is defined as bth≥n+8; that is, if the DAC width is 10 bits, then set 18th bit or higher counting from the MSB.

Calculate the tuning-word frequency, f_C, from f_{REF} and the tuning word. Use Equation 3.10 to locate the frequency region of the pROMinent spur set to measure. Note that the spacing between these sets of spurs is two times the tuning-word frequency. Measure and record the frequency of each individual spur in the worst-case set. Individually divide the frequencies of the worst-case spur set by the tuning word. Round the results to the nearest whole number. The results will probably be consecutive odd numbers. Let this be step 1.

These results in step 1 are harmonically related to the tuning word. When you change the tuning word, predict the locations of the corresponding spurs by multiplying the new tuning word by each value in step 1. Let this be step 2. Terms from step 2 greater than Nyquist, $f_{REF}/2$ will alias. To locate the alias if the product is above Nyquist but below f_{REF}, subtract the f_{REF} from the product; the difference is where the alias resides. To locate the alias if the product is above f_{REF}, divide by f_{REF} and analyze the remainder of the quotient. If the remainder is below 0.5, multiply it by f_{REF} Otherwise, subtract it from 1 and then multiply by f_{REF}. let this be step 3. Repeat steps 2 and 3 for every value found in step 1.

3.5 Quantization noise, DAC nonlinearities

A DAC's quantization noise and distortion determine its SNR. You can calculate a first-order approximation of SNR by taking the ratio between the quantization-noise power, integrated over the Nyquist bandwidth, and the power in the fundamental. As a result, SNR is proportional to the DAC resolution in bits, as given in Equation 3.9. This SNR calculation describes an ideal DAC. Real DACs also have nonlinearities due to process mismatches and imperfect bit-weight scaling. Nonideal switching characteristics also add distortion and nonlinearity. The most prominent DAC spurs are usually due to nonideal switching characteristics, which, along with any nonlinearity in the transfer function, appear as lower order harmonics of the fundamental. Both quantization noise and the nonideal DAC properties produce a response that consists of harmonically related spurs of the fundamental. This relationship is the key to understanding how to predict the frequency location of the prominent spurs.

Harmonics alias because the DAC is a time-sampled system. As a result, the carrier's harmonics, the reference clock, and the reference clock's harmonics create numerous sum- and difference-mixing products. The well-defined mathematical relationship of these products makes predicting the spur locations possible. Harmonics beyond the first Nyquist zone are mapped back to the first Nyquist zone.

For example, a DDS tuned to 25.153 MHz with a reference clock of 100M samples/sec generates low-order odd harmonics close to the fundamental. Once the harmonic series exceeds the Nyquist frequency, they alias back into the first Nyquist zone in a predictable way (difference product). This DDS has a 14-bit DAC. The SFDR (spurious-free dynamic range) within the 4-MHz bandwidth is better than –73 dBc.

Increased over-sampling, by raising f_{REF} to 400M samples/sec eliminates the alias products of the third, fifth, and seventh harmonics within the first Nyquist zone.

Significant benefits arise in DDS applications from running the DDS and comparator at a simple subharmonic of the reference clock. These benefits include reduced jitter and a simpler reconstruction filter. Because f_{REF} and f_C are related by an integer ratio, the DAC quantization noise and spurs caused by other error sources fall exactly on top of the harmonics of f_C. In such cases, harmonics don't produce jitter, because they are phase coherent with the carrier.

CHAPTER 4

A NOVEL DDS ARCHITECTURE WITH IMPROVED COMPRESSION RATIO AND QUANTIZATION NOISE

In this section a novel DDS ROM compression technique is presented. The technique is based on two basic properties of the sine function, (a) piecewise linear characteristic of the sinusoid for infinitesimal difference in phase angle (b) variation in the slope of the sinusoid with the phase angle. Storing only some of the values of the sinusoid and constructing the entire sinusoid using compression techniques reduce the memory requirements in DDS. In the proposed architecture, the first property of sinusoid is used to interpolate the values of sinusoid that have not been stored in the ROM, thereby greatly reducing the memory requirement. The second property has been used to increase the number of values that can be interpolated as the slope of the sinusoid decreases, without degradation in the performance. The variation in the number of values interpolated has been achieved through a nonlinear addressing scheme. The ROM incorporated in the proposed architecture for a phase resolution of 15 bits is 1216 bits resulting in a spectral purity of -90.6 dBc. The proposed architecture has better compression ratio than the Nicholas architecture [9] with the same hardware efficiency and spurious response. The linear, nonlinear and traditional architectures have been implemented in a Xilinx Spartan II FPGA. Measured spectra show that the compressed and uncompressed ROMs end up with the same noise floor and thus the proposed ROM

compression algorithms are verified. In the proposed technique the sine wave has been assumed to be piece-wise linear between the values stored in the ROM. The technique requires a small multiplier, which does not present a bottleneck to low power and high-speed applications. The results show that the technique can achieve a considerable reduction in ROM size with reduced circuit complexity without degradation in the spurious response. The technique has been implemented on Xilinx Spartan II FPGA.

4.1 Back ground of ROM Compression Algorithms

Different algorithms have been proposed for compression of look up tables. A simple architecture for ROM compression was first proposed by Hutchison [10], which was improved by Sunderland and Nicholas [9]. The architecture proposed by Nicholas [9] has been considered to be most efficient due to its simplicity and minimum hardware requirements. Nicholas architecture provides reduction in the look up table storage requirements by replacing the storage requirements of one large ROM of size 2^{A+B+C} words with two smaller ROMs of sizes 2^{A+B} words and 2^{A+C} words, whose outputs are added together to reconstruct the sine function.

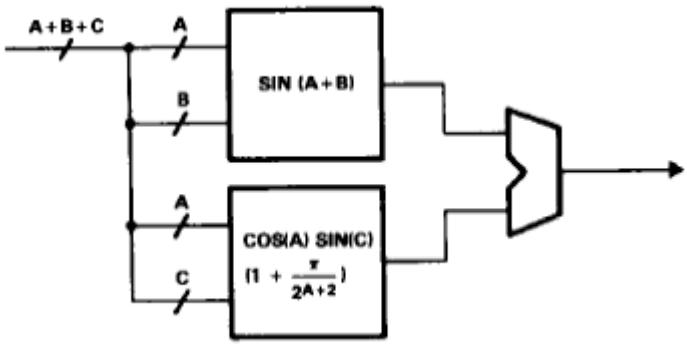


Figure 4.1 – Block Diagram of Coarse-Fine ROM structure.

These are called the coarse ROM and fine ROM. Certain values of the sine function have

been stored in the course ROM which is addressed by the MSB bits of the phase accumulator's output. The values of sine function between the coarse ROM values have to be interpolated. These values are obtained using the values stored in a second ROM called fine ROM. In the Nicholas architecture, none of the values of the sine function are calculated. All the values of the sine function are indirectly stored. As the speed of technology is increasing, calculation of values is faster as well as easier; hence the memory requirements can be reduced by calculating the values that have to be interpolated rather than storing them as adopted by Nicholas architecture. This is the basis for the proposed architecture where all the values that have to be interpolated are calculated thereby greatly reducing the memory requirements. Further, the number of values to be calculated can be increased, which reduces the number of values that have to be stored, by exploiting the properties of sinusoid. The architecture proposed by Bellaouar [11] and [12] also uses the course and fine ROM structure long with a multiplier to calculate the values that have to be interpolated.

4.2 Some of the algorithms commonly used in all ROM reduction techniques

4.2.1 Sine Function Symmetry

The simplest way to reduce the ROM size to ¼ is to exploit the sine function symmetry [9] about $\pi/2$ and π. By incorporating this technique into the architecture, only values between 0 and 90^0 are needed to be stored. The architecture to construct the entire sine function using the values between 0 - 90^0 is as shown in Figure 4.2. All the values are represented in 2's complement format. A ½ LSB offset has been introduced in to the value that has to be complemented so that a 1's complementor can be used in place of 2's

complementor.

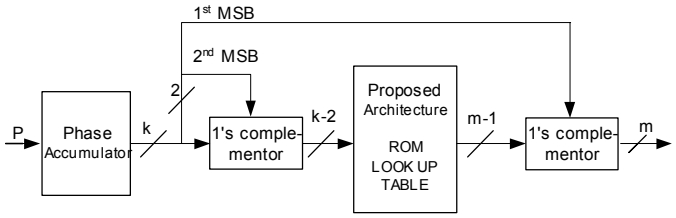


Figure 4.2 - Architecture for constructing sine function using symmetry around Π/2 and Π.

4.2.2 Sine Phase Difference Algorithm

The sine phase difference algorithm [9] is one of the popular techniques used in almost all the compression algorithms proposed. According to this algorithm the ROM stores the values obtained from the function

$$f(x) = \sin\left(\frac{\pi x}{2}\right) - \frac{\pi x}{2} \tag{4.1}$$

instead of the value of function sin (πx/2). Two bits of amplitude are saved in the ROM by incorporating this algorithm without degradation in the spurious response, because $\max\left[\sin\left(\frac{\pi P}{2}\right) - P\right] \approx 0.21 \max\left[\sin\left(\frac{\pi P}{2}\right)\right]$, but this algorithm requires an additional adder at the output of the ROM to calculate the function

$$f(x) + x \tag{4.2}$$

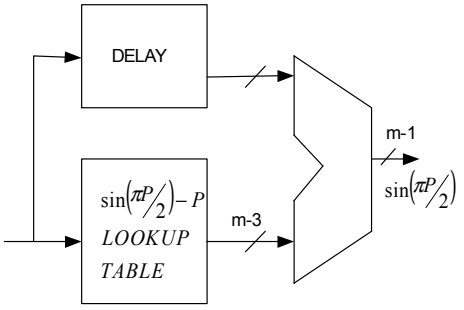


Figure 4.3 - Block Diagram of Sine-phase difference algorithm.

4.3 Proposed Algorithm

4.3.1 Generic Architecture

The proposed technique is based on two basic properties of sine function, (a) the piecewise linear characteristic of sinusoids for infinitesimal differences in phase angle (b) variation in the slope of the sinusoid with phase angle. In the architectures of DDFS using ROM compression techniques, certain values of the sinusoid are stored in the ROM, and entire sinusoid has to be constructed using these values. For these the values, which are not stored in the ROM, have to be interpolated. In our architecture, the two most popular techniques of storage reduction (i) sine phase difference algorithm and (ii) sine function quadrature symmetry have been used. The sine phase difference algorithm helps to reduce the width of the word which has to be stored in the ROM by 2 bits. Incorporating this algorithm in to our architecture requires 2 additional adders. In the proposed architecture, sine phase difference algorithm has been used to reduce the ROM from 1408 bits to 1216 bits. Exploiting the sine function quadrature symmetry reduces the memory requirements by 4 times. At any clock cycle the value in the phase register indicates the phase of the sine function. As all the values of the sine function are not stored in the ROM, out of the N bits of the phase address register only the first m bits are used for addressing. The remaining d bits (d = N-m) indicate the relative position of the sine value that has to be interpolated from the value that has been stored in the ROM. The value in the phase registers R at any instant reads,

$$R = FCW \times (i-1) \tag{4.3}$$

where FCW is the input frequency control word to the phase accumulator and i is the number of clock cycles. The word length of R is N bits. The phase of the sine value that has to be calculated is given by

$$B = \frac{2 \times \pi \times R}{2^N} \tag{4.4}$$

The equation used for calculating the value that has to be interpolated is

$$\sin(B) = \frac{(\sin C - \sin A)}{(2^d)} \times (s[d-1:0]) + \sin(A) \tag{4.5}$$

where $\sin(B)$ is the sine function value that has to be calculated and $\sin(A)$ and $\sin(C)$ are sine values stored in the ROM where $\sin(C) > \sin(A)$. d = N - m gives the number of values that have to be interpolated if FCW = 1. The LSB bits s[d-1:0] of the phase register, indicate the relative position of the sine value that has to be interpolated from the sine value that has been stored in the ROM. In other words, it represents the number of clock cycles required for the value in the phase register to change from A to B for FCW=1. The output of the ROM has a word length of k bits.

In the proposed architecture the sine function is assumed to be piecewise linear. The division in Equation 4.5 causes increase in the quantization noise due to the rounding effect. In order to avoid the division in Equation 4.5 which involves hardware, the formula has been modified to

$$\sin(B) = (\sin(C) - \sin(A)) \times (s[d-1:0]) + \sin(A) << (d) \tag{4.6}$$

This change makes the output word length n = k + d bits. This increases the precision of representation and does not increase the quantization noise levels. The k + d bits are later truncated to k bits to meet the specifications of the DAC. The spurious response has been

analyzed after truncating the word length to k bits. The memory requirements here are $2^m \times k$ bits.

The expression $(\sin(C) - \sin(A)) \times (s[d-1:0])$ of Equation 4.6 is calculated by using a Multiplier. This architecture is termed as linear architecture and has been implemented in MATLAB for the dimensions provided in the Table I. The size of the subtractor was chosen by taking into account the maximum difference between successive ROM samples. The values in the ROM have been optimized to give minimum error. These values are the averaged values that have been obtained after a number of simulation trials.

TABLE I: Dimensions of DDS with single ROM

Phase accumulator size s[14:0]	15 bits
Addressing lines to ROM s[12:5]	8 bits
No. of LSB bits for interpolation s[4:0] or d	5 bits
Width of the ROM with sine phase difference algorithm	12 bits
output word length n = k + d	14 + 5 = 19 bits
Truncated output word length	14 bits
Subtractor size	7 bits
Multiplier size	7 x 5

Memory requirements have further been reduced by exploiting the second property of sine function. The slope of sine ranges from 1 to 0 as the angle changes from 0^0-90^0. So the slope of the sine wave changes gradually between successive ROM values at the beginning and reduces to almost 0 at 90^0. Hence more ROM values can be stored in

the region of the sinusoid having higher slope and a smaller number of ROM values can be stored in region of sinusoid with lesser slope. Therefore, the number of values that have to be interpolated are less in the high-sloped region than the number of values that have to be interpolated in the low-sloped region of the sinusoid. For the architecture shown in Figure 4.5, the sinusoid is partitioned in to two parts; $0\text{-}45^0$ and $45^0\text{-}90^0$. The sine values for phases between $0^0\text{-}45^0$ are stored in ROM1, and those for $45^0\text{-}90^0$ in ROM2. As the change in the slope of the sinusoid, is less between $45^0\text{-}90^0$, the number of ROM values stored to calculate sine value after 45^0 can be reduced to almost half without significant drop in the spurious response. So the number of addressing lines to ROM2 changes accordingly to m-1, where m is the number of addressing lines to ROM1. The number of values that have to be interpolated between $45^0\text{-}90^0$ will increase (twice the number that have to be interpolated in the single ROM architecture) and the relative position of the value that has to be interpolated is indicated by LSB d_2 (d_2 = N-m+1) bits. Equation 4.6 gets modified to

$$\sin(B)_2 = (\sin(C)_2 - \sin(A)_2) \times (s[d_2 - 1 : 0]) + \sin(A)_2 << (d_2) \qquad (4.7)$$

$$\sin(B)_1 = (\sin(C)_1 - \sin(A)_1) \times (s[d_1 - 1 : 0]) + \sin(A)_1 << (d_1) \qquad (4.8)$$

Subscripts 2 and 1 indicate that the values under discussion are from ROM2 and ROM1 respectively. d_2 is one bit greater than d_1(d_1 = d) so shifting the $\sin(A)_2$ value by d_2 will make the output word length k+d+1 bits. In order to have uniform output the width of ROM2 is made k-1 instead of k. The memory requirements for this architecture are $2^{m-1} \times k + 2^{m-2} \times (k-1)$ bits.

4.4 Quantization Error Analysis

The number of bits in each word of the ROM will determine the amplitude quantization error. The finite amplitude quantization in the sine ROM values leads to degradation of the output spectrum. So the quantization noise is discussed in detail in this section. Without phase truncation, the output of the DDS is

$$\sin\left(\frac{2\pi P(n)}{2^N}\right) - E_A(n) \tag{4.9}$$

where P (n) is the N-bit value in the phase accumulator register at the n^{th} clock cycle and $E_A(n)$ is the quantization error due to finite ROM values at the n^{th} clock cycle. The amplitude quantization errors with an uncompressed ROM can be assumed to be uniformly distributed and uncorrelated within each quantization step if the step size is small, we have

$$\frac{-\Delta_A}{2} \le E_A \le \frac{\Delta_A}{2} \tag{4.10}$$

with the quantization step size $\Delta_A = \frac{1}{2^k}$ where k is word length of values stored in the ROM.

The sinusoid signal power is

$$P_A = \frac{A^2}{2} \tag{4.11}$$

and the error power is

$$E\{E_A^2\} = \frac{1}{\Delta_A}\int_{-\Delta_A/2}^{\Delta_A/2} E_A^2\, dE_A = \frac{\Delta^2}{12} \tag{4.12}$$

Assuming that the amplitude quantization error gets its maximum absolute value $\Delta_A/2$ at

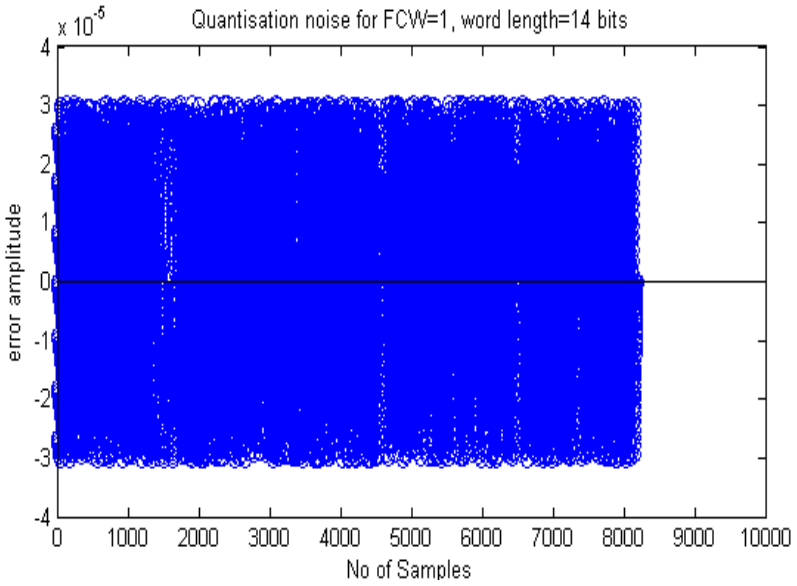


Figure 4.4 - Graph showing $\frac{-\Delta_A}{2} \le E_A \le \frac{\Delta_A}{2}$ for memory length of 14 bits.

every sampling instance, and all the energy is in one spur, the carrier to spur ratio is

$$\frac{C}{S} = 10\log_{10}\left(\frac{4P_A}{\Delta^2}\right) = (-3.01 + 6.02k)dBc \tag{4.13}$$

However, in the worst case the sum of the discrete spurs is approximately equal to

$$\frac{C}{S} = 10\log_{10}\left(\frac{P_A}{E\{E_A^2\}}\right) = (1.76 + 6.02k)dBc \tag{4.14}$$

We have used the last formula to analyze the error due to finite amplitude quantization in the proposed architecture. $E\{E^2_A\}$ is the mean of the variance of the error. Therefore,

$$E\{E_A^2\} = \frac{1}{N}\sum_{j=1}^{j=N} e_j^2 \tag{4.15}$$

where j represents at the j^{th} clock cycle. N is the total number of clock cycles to construct one period of the sine function and e_j is

$$e_j = \sin(B) - \sin(B)_{cal} \text{ at the } j^{th} \text{ clock cycle} \tag{4.16}$$

Figures 4.6 and 4.7 show e_j, which is the error between the actual sinusoid function and that obtained from the proposed architecture for Linear and Nonlinear addressing respectively, at each clock cycle. The values of error shown in the graphs are used for calculating the mean of the variance of the error.

4.5 Proposed Architecture with Non-linear Addressing (Two ROMs)

This section discusses the implementation of the generic algorithm with two ROMs. For the architecture with two ROMs, the sinusoid is partitioned in two parts; 0-45^0 and 45^0-90^0.The sine values for phase between, 0-45^0 being stored in ROM1, and 45^0-90^0 in ROM2. This architecture is simulated for dimensions of ROM1 equal to 2^6 x 12 bits and ROM2 equal to 2^5 x 11 bits. The width of ROMs has been reduced from 14 bits to 12 bits in ROM1 and 13 to 11 bits in ROM2 by using the sine-phase difference algorithm. Let the 15 bit output of the phase accumulator register be represented by variable s[14:0] where 14:0 indicate bit 14 is MSB and bit 0 is LSB. The first 2 bits out of the 15 bits of the phase accumulator register indicate the quadrant of the sine wave. Hence the remaining 13 bits are used to address the sine wave between 0-90^o. FROM the ROM sizes mentioned, the number of values that have to be interpolated are approximately 64 between 0-45^0 and 128 in between 45^0-90^0 if FCW = 1. The address lines required to address ROM1 are 7 i.e., s [12:6] and that required for ROM2 are 6 i.e., s [12:7]. According to Equations 4.7 and 4.8, the values of ROM1 are shifted by 6 and the values of ROM2 are shifted by 7 before being added to the output of the multiplier to obtain the output sine value. The position of sine value that has to be interpolated according to the value in the phase register is given by 6 LSBs s [5:0] for sine phase between 0 to 45^o and by 7 LSBs s [6:0] for sine phase between 45^o to 90^o. The whole sine

value is represented with a precision of 15 + 6 = 21 bits. This value is truncated to 14 bits before sending to the DAC. This technique requires a 7x8 bit multiplier. The increase in the hardware overhead is the trade off for the great reduction in the memory requirement. The sinusoid can further be partitioned to $0\text{-}30^0$, $30^0\text{-}60^0$, $60^0\text{-}90^0$ instead of $0\text{-}45^0$ and $45^0\text{-}90^0$, to obtain further reduction in the memory requirement. The values stored in the ROM are optimized to minimize the mean square error by computer simulations. Worst-case spur amplitude of about -88 dBc is achieved with this method. But the simulation results show that a ROM of 2^6 x 13 + 2^5 x 12 bits provides a spurious rejection of -90.3 dB, which provides the best compression ratio.

Incorporating sine-phase difference algorithm in nonlinear addressing the word length of ROM can be reduced by 2 bits. Introduction of this technique in to ROM requires two extra adders. The values stored in the ROM are given by the expression

$$((2^k - 1) \times \sin(A)) - (ci) << (k - m) \tag{4.17}$$

where $A = \dfrac{2 \times \pi \times 2^{N-m} \times (ci)}{2^N}$

and ci is the value in the m addressing lines to the ROM. The value ci has to be added back to the value in the ROM before the value that has to be interpolated is calculated. The input to adder1 is the output from the ROM and the m addressing lines (ci) to ROM1 are shifted left by k-m places to make the word length same. To calculate the step size that is an input to the multiplier, two successive values of ROM (sin A and sin C) are required. The value $(2^k - 1) \times \sin \dfrac{2 \times \pi \times 2^{N-M} \times (ci+1)}{2^N} - (ci+1) << (k-m)$ is stored in the ROM instead of sin C whose value is given by $(2^k - 1) \times \sin \dfrac{2 \times \pi \times 2^{N-m} \times (ci+1)}{2^N}$. So a value of

$ci+1$ shifted left by k-m bits should be added to the ROM value before the interpolated value is calculated. Hence 2 adders are required to calculate sin A and sin C.

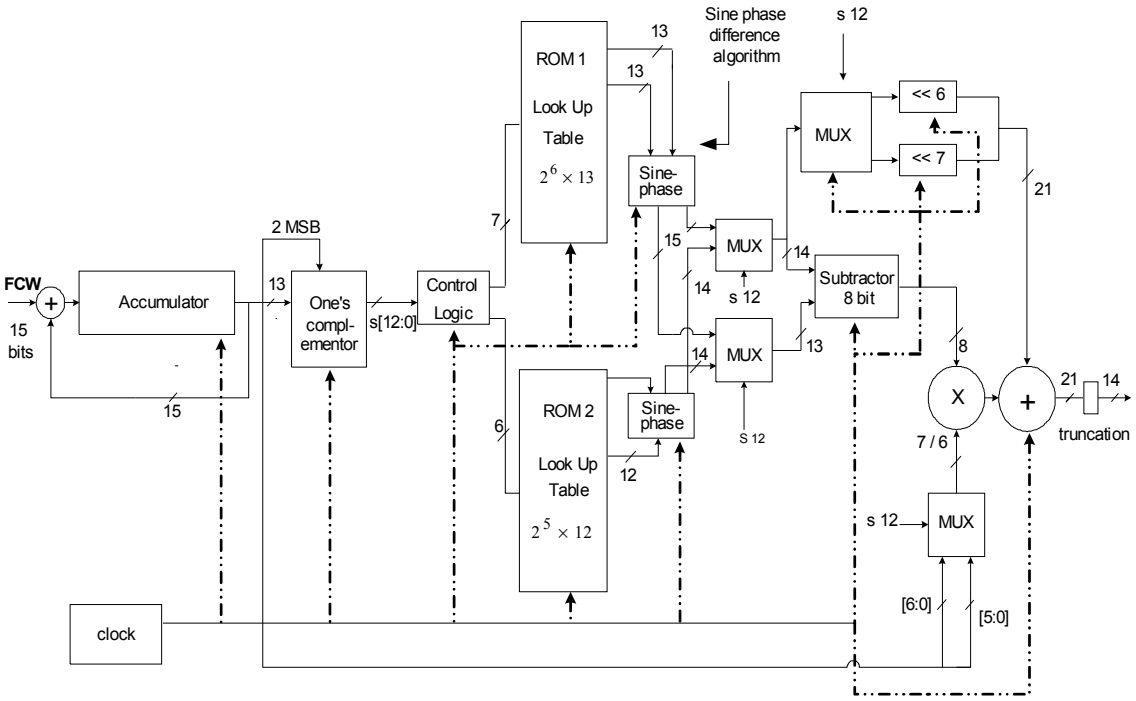


Figure 4.5- Phase to Sine conversion architecture with non-linear addressing.

4.6 Simulation Results

A computer program has been written in MATLAB to simulate the proposed architecture. The results henceforth are mentioned for both the nonlinear and the linear addressing architectures. The results are for phase bits P = 15. According to [9], architecture having uncompressed memory with symmetry considerations, having above mentioned phase bits should have a worst-case spur of -90 dBc, which is obtained from the simulations as well. For the proposed linear addressing architecture with multiplier, ROM size is taken to be 2^8 x 12 (3072 bits). The worst-case spur thus obtained in this case is also 90 dBc. For nonlinear addressing the ROM size used is 2^6 x 13 for ROM1 and 2^5 x 12 for ROM2 adding up to 1216 bits, thus saving 1856 bits as compared to the

linear addressing scheme. As can be seen from the error graphs in Figures 4.6 and 4.7 the error obtained is of the order of 10^{-5} which yields a worst-case spur of -90 dBc. The ROM samples have been optimized to reduce the mean square error thus used to obtain the worst-case spur.

The Table II discusses the variation in the spurious response for different word lengths for ROM1 and ROM2. The widths or word lengths of 13 and 12 for ROM1 and ROM2 respectively resulted in a magnitude of worst-case spur of –95 dBc. To attain the performance goal of -90dB of spurious rejection the word lengths of 12 and 11 bits for ROM1 and ROM2 are optimal which result in a worst-case spur of -88dBc. FROM the graph it can be observed that as the word length of the ROMs is reduced the performance is degraded, which is as expected. It has been observed that word lengths of 12 and 11 bits are optimal for ROM1 and ROM2 respectively. Hence the algorithm has been simulated keeping the word lengths constant at these values for different lengths of ROM1 and ROM2.

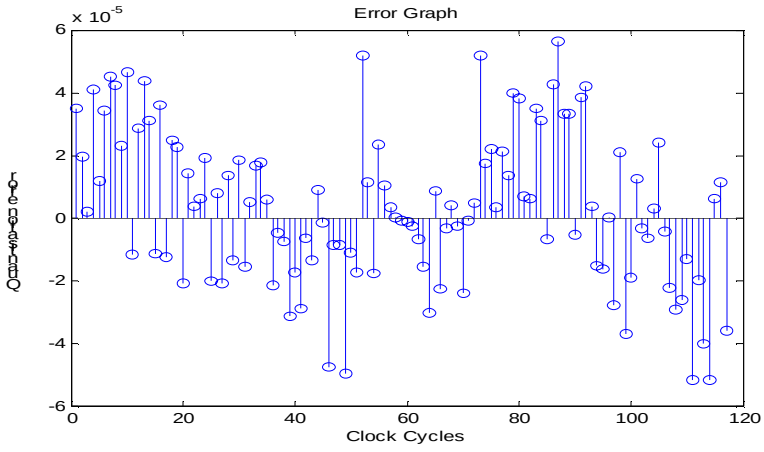


Figure 4.6 - Error versus samples for P = 15 bit, ROM is 2^8 x 12 bit, FCW = 69, linear addressing, error of the order of 10^{-5}.

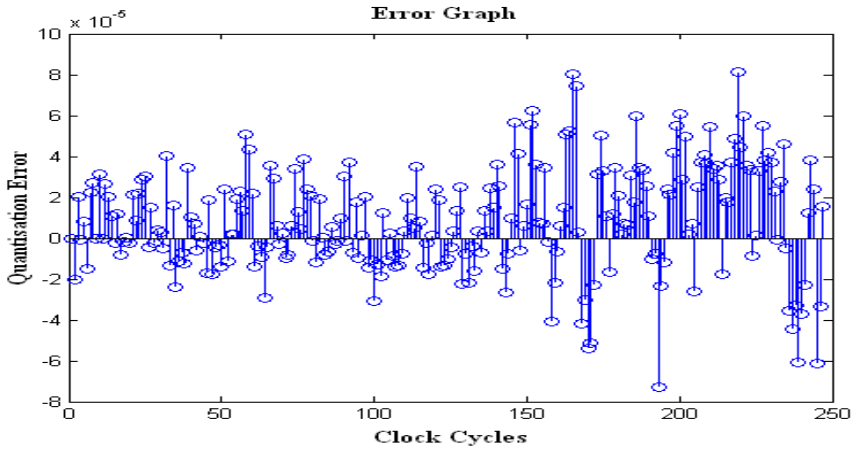


Figure 4.7- Error versus samples for P = 15 bit, nonlinear addressing, FCW = 33.

TABLE II: Memory word lengths versus worst case spur for non-linear addressing

Memory word length for ROM1	Memory word length for ROM2	Worst case spur (dBc)
8	7	-63
9	8	-72
10	9	-76
11	10	-83
12	11	-88
13	12	-95

From the Table III, it can be observed that memory lengths of 2^7 and 2^6 for ROM1 and ROM2 meet the requirement of -90dB of spurious rejection. It can be observed that even though the memory lengths are increased to 2^8 and 2^7 for ROM1 and ROM2 respectively there is not much improvement in the performance. One reason for this might be the constant word length of the ROMs, which limits the performance. On the other hand when the memory lengths are decreased to 2^6 and 2^5 for ROM1 and ROM2 respectively we get an -86 dB of spurious rejection, which is not much reduction in performance but doubles the compression ratio to 102.4:1. The word length is an important limitation on performance. Hence by increasing the word length to 13 and 12 bits respectively for

ROM1 and ROM2 with memory lengths 2^6 and 2^5 respectively a spurious rejection of -90.6 dB can be achieved which gives a compression ratio of 94.3:1. This memory dimension is taken as standard for comparison and considered to be the optimal.

The graph in Figure 4.9 indicates the possible combinations of ROM1 and ROM2 dimensions giving the same performance. The graph also shows that increasing the memory word length beyond an extent will not result in further improvement in performance. The bottom curve in Figure 4.9 is for an SFDR of -85 dBc, and the top curve is for an SFDR of -95 dBc.

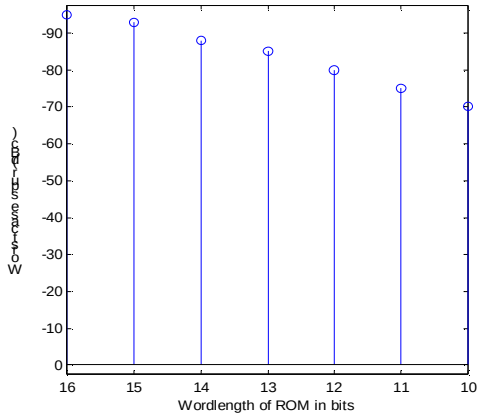


Figure 4.8 - Memory word lengths versus worst case spur for linear addressing.

TABLE III: Memory lengths versus worst case spur for non-linear addressing

Memory length for ROM1	Memory length for ROM2	Worst case spur (dBc)
16	8	-71
32	16	-80
64	32	-86
128	64	-88
256	128	-88

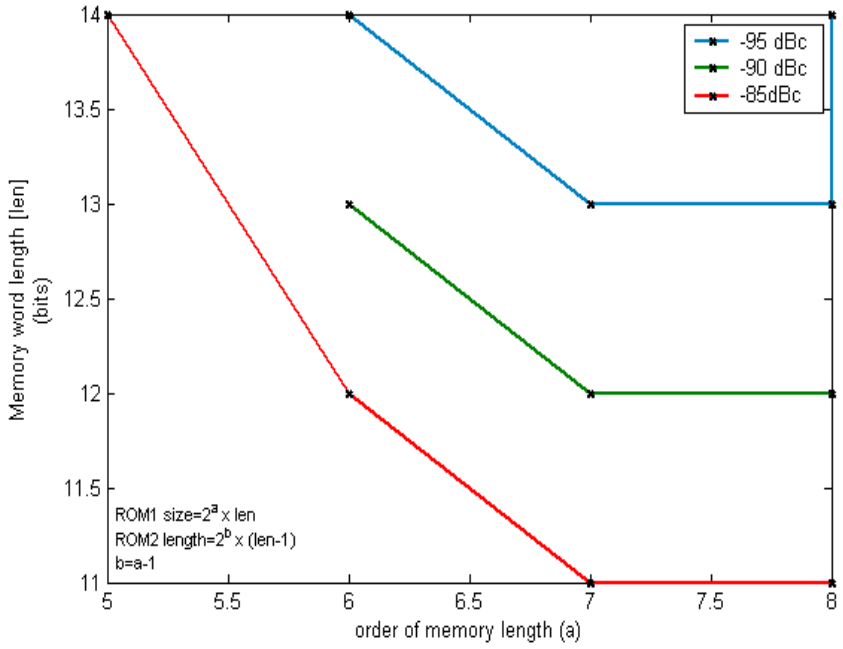


Figure 4.9 - Different word-lengths and memory lengths of ROM1 and ROM2 with same spurious response.

TABLE IV: COMPARISION BETWEEN THE NICHOLAS' ARCHITECTURE AND OUR PROPOSED ARCHITECTURE FOR P = 15 BITS

Method	**Worst spur (dBc)**	**ROM size (bits)**	**Compression ratio**
This work	-90.3	2^6 x 13 + 2^5 x 12	94.3:1
Nicholas' ROM	-90	2^8 x 9 + 2^8 x 4	37:1

One of the algorithm discussed in [12], which is a slight modification of Nicholas architecture, claims a compression ratio of 128:1 for a phase accumulator register of P=12 bits. There is 50% reduction in the ROM size by performing interpolation around a value in the middle of the interpolation range which gives almost same performance as that of Nicholas architecture by modifying the phase segmentation. But no details have been provided for P=15 bits. The inherent property of our architecture is that the

TABLE V [11]: COMPARISON BETWEEN VARIOUS PROPOSED ARCHITECTURES FOR P=12 BITS

Method	Needed ROM	Total Compression Ratio	Spectral Purity	Comments
Uncompressed Memory	2^{12} x 8 bits	1:1	-72.245	reference
¼ Sine-Wave Symmetry	2^{10}x8 bits	4:1	-72.245	2 adders required
Hutchison's Technique	2^{7} x 6 bits 2^{10} x 2 bits	11.64:1	-72.245	(3,4,3) phase segmentation, 2 adders required
Sunderland's Architecture	2^{7} x 8 bits 2^{4} x 3 bits	27:1	-72.245	(3,4,3) phase segmentation, 2 adders required
Nicholas Architecture	2^{4} x 7 bits 2^{4} x 3 bits	51.2:1	-72.245	(3,3,4) phase segmentation, 2 adders required
Bellaouar's Architecture	2^{5} x 8 bits 2^{5} x 5 bits	78.8:1	-72.245	(5,5) phase segmentation, a 5x5 multiplier and 2 adders
Proposed Architecture Linear Addressing	2^{4} x 9 bits	56.89:1	-71	5 x 7 multiplier, 2 adders
Proposed Architecture Non linear Addressing	2^{5} x 9 bits 2^{4} x 8 bits	78.8:1	-71	6 x 7 multiplier, 2 adders

compression ratio greatly increases for increasing size of phase accumulator. The architecture in [12] might not give similar results for a larger phase accumulator because the basic algorithm is similar to Nicholas architecture and the compression ratio of Nicholas architecture reduces with increasing phase accumulator length. The architecture proposed in [13], is also a linear interpolation ROM compression algorithm with only a single ROM with 64 memory locations. But the output resolution of this architecture is smaller than what we have achieved in this work and with the increase of the output resolution their error also increases. So although, a higher compression ratio is achieved the SFDR will be limited by the number of input bits to the DAC (i.e. DAC quantization noise). They have also mentioned degradation of output SFDR if very short sequences of oscillations are synthesized prior to changing the FCW. This is not a problem with the proposed architecture and hence will allow a faster frequency switching. The ROM-less

DDFS is an attractive design in terms of area, power consumption and speed but the SFDR achieved is still lower than the ROM based architectures.

4.7 FPGA Measurements

The linear, nonlinear, and traditional DDS architectures have been coded using Verilog RTL and are implemented using Xilinx Spartan II FPGA. The PCB board noise is at about -30dBc. An 8-bit DAC has been used for the implementation which has 5~6 effective number of bits due to the noise floor observed during the test. We therefore modified our architecture to have a 15-bit accumulator that is truncated to 9 phase bits and a compressed ROM that generates an 8-bit sinusoidal waveform. Choosing 9 ROM phase bits will help fully utilize the number of DAC bits and meanwhile can still demonstrate the ROM compression algorithm. The clock frequency applied is 25 MHz. For frequency control word FCW = 1, the characteristic equation for DDS gives the output frequency as

$$f_{clk} = 25\ MHz$$
$$f_{out} = \frac{25\ M}{2^{15}} \times (1) = 762\ .93\ Hz \tag{4.18}$$

TABLE VI: FPGA Specifications

FCW	1
f_{clk}	25 MHz
N	15 bits
P	9 bits
DAC width	8
Effective ROM length	128

Out of the available 9 phase bits for ROM addressing, 2 bits are used for the sine quadrature symmetry. Thus for the traditional architecture, 7 phase bits are used to

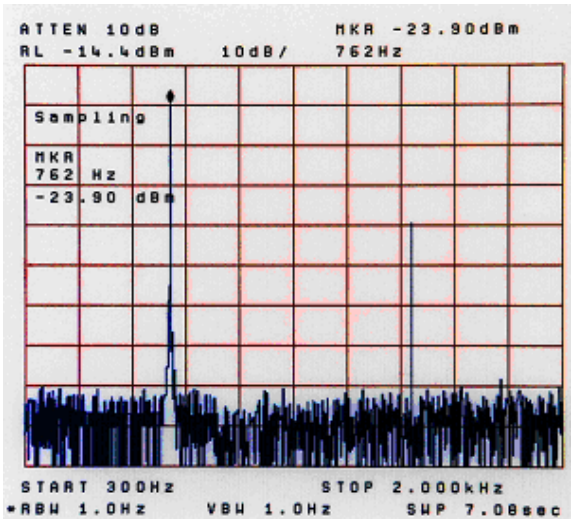


Figure 4.10 - Spectrum of DDS with linear architecture for FCW = 1, phase accumulator = 15 bits, DAC resolution = 8 bits, clock freq. = 25MHz.

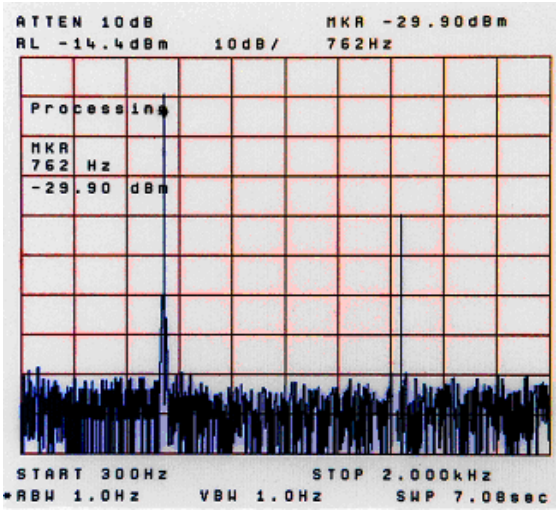


Figure 4.11 - Spectrum of DDS with non-linear architecture for FCW = 1, phase accumulator = 15 bits, DAC resolution = 8 bits, clock freq. = 25MHz.

address the ROM. The ROM size used is 2^7 x 8 bits, ROM length is 128 bits, and ROM width is 8 bits. For the linear architecture, the ROM size is selected to be 2^5 x 8 bits; 5 bits are used to address the ROM which has a width of 8 bits. Out of the 9 phase bits (after phase truncation) 2 MSB are used for quadrature symmetry and the 2 LSB are fed to the multiplier. The compression ratio obtained in this case is 4:1. For the nonlinear

architecture, as shown in Figure 4.5, the ROM sizes are given by ROM1 (2^3 x 8), ROM2 (2^2 x 7); 4 bits of the phase word are used to address ROM1 and 3 bits are used to address ROM 2, which has widths of 8 and 7 bits respectively. The compression ratio achieved in this case is 11.13:1 when compared with the traditional architecture.

As we can see in Figure 4.10 and Figure 4.11, our proposed linear and nonlinear compressed ROMs end up with the same quantization noise floor at about -72dBc as the traditional architecture without ROM compression in Figure 3.8. Thus, the ROM compression has not degraded the output noise performance. We have also measured DDS output spectra for other input FCWs and clock frequencies. The measured data verified the performance of the proposed ROM compression algorithm under various conditions.

APPENDIX A

MATLAB CODE

In this appendix the MATLAB code for the implementation of traditional, linear, nonlinear ROM architectures of DDS has been provided.

A.1 TRADITIONAL ROM DDS ARCHITECTURE

```
%Accumulator
FCW=input ('FCW:');    %FCW is the frequency control word provided by the user
s=0;                   %s is the variable storing the accumulator values
i=0;                   %i is variable indicating the number of clock cycles
j=1;                   %j is  variable to break the phase accumulator loop
sample=0;              %Sample is the sine value corresponding to the phase word in the
                       %accum.
ROM=load('uncom10.txt');      % load values in the txt file 'uncom10.txt' in to ROM
z=0;
F=0;
while s <=(65536/2) & j~=0    %Phase accumulator s is 15 bit wide. When one complete
                              %sine wave is completed the value in s might not be 0 but
                              %less than 2^15 so variable j is %used to break the loop.
j=s;
s=s+FCW;                                  %increment the phase accumulator
```

```
   if s>=32767
     s=s-32767;                    %when s>2^15 the next cycle of sine wave starts
 end

 i=i+1;                            %indicates the number of clock cycles from the initial reset
sample(i,:)=ROM((s+1),:)/(16383);     %gives the sine value between 0-1. width of ROM
                                      %is 15 bits
error(i,:)= (sin ((pi/2)/8192*(i-1)*FCW)- sample(i,:));
                                      %Eq to calculate error during each clk cycle
j=s;
end                                   %phase accumulator loop ends
%error=sort(error(1:(i-1)/4));
plot(sample(1:100));                  %plots the sine value on Y axis and no. of clock cycles
                                      %on x-axis
%for loop to calculate the total signal power and error power
for N=0:1:(((i-1)/(4*FCW))-1)     %checking output for only 1st quadrant of sine between
                                  %0o-45o
z=z+(error((N+1),:))^2;           % gives the total error power during all the clock cycles
end
%Value of N after the for loop, gives total power in all samples during all the clk cycles
 f=20*log10(sqrt(N/(z*2)));       %f value gives the worst case spur value in dBc
 nfft=(i-1);
fout=sample(1:nfft);
```

```
psd=20*log10(abs(fftshift(fft(fout(1:nfft))))+0.0000001);
%for N=0:1:i-2
%    F=F+(psd((N+1),:))^2;
%end
%signal=max(psd);
%psd=psd-signal;
%stem(psd(1:nfft));
```

A.2 LINEAR ROM DDS ARCHITECTURE

```
FCW=input ('FCW:');    %FCW is the frequency control word that is provided by the user
s=0;                   %s is the variable storing the accumulator values
i=0;                   %i is variable indicating the number of clock cycles
q=0;
m=0;
sample=0;              %Sample is the sine value corresponding to the phase word in the
                       %accumulator
ROM=load('6f.txt');    % load values from the ROM stored in the notepad file '6f.txt'
                       %y=load('2615.txt');
l=1;                   %l is variable to break the phase accumulator loop
step=0;
stepsize=0;
j=0;
k=0;
z=0;
```

```
qut=0;
F=0;
FC=0;
FC2=0;
a=0;
u=0;
s1=0;
while s <=(65536/2) & l~=0  %Phase accumulator s is 15 bit wide. When one complete
                            %sin wave is completed the value in s might not be 0 but
                            %less than 2^15 so j is used to break the loop
 l=s;
si=s;
sb=dec2bin(s,15);           %sb has the binary value of s packed in 15 bits, width of
                             %phase accumulator
u=sb([3 4 5 6 7 8 9 10 11 12 13 14 15 ]);
                             %u does not have first 2 bits as they only indicate
                            %quadrant
u=bin2dec(u);
MSB=sb([1 2]);               %MSB indicates the quadrant
MSBd=bin2dec(MSB);           %decimal value of the quadrant
 switch MSBd
case 1
    cc=bitcmp(u,13);
```

```
      u=cc;
case 3
       cc=bitcmp(u,13);
       u=cc;
       end
       ub=dec2bin (u,15);
d=ub([ 11 12 13 14 15 ]);        %d is the distance between the positions of the sine value
                                 %that has to be calculated and that has been stored in the
                                 %ROM.
c=ub([ 3 4 5 6 7 8 9 10  ]);     %c is used to address the ROM values
ci=bin2dec(c);
di=bin2dec(d);
i=i+1;
m=ROM(ci+1);                     %m is assigned the sine value addressed by c
if(k~=m)
  j=k;
  k=m;
end
diff=ROM(ci+2)-ROM(ci+1);   % diff has difference between successive ROM samples
step=di*(diff/32);               %calculates the step size
if i==248*4                      %for FCW=33, i=248*4 gives one complete cycle of sine
break
end
```

```
switch (MSBd)                    %each case indicates one of the four quadrants
case 0
sample(i,:)=((ROM(ci+1))+ step)+16;          %value 16 is added to reduce the error and
                                             %has been decided after many iterations of
                                             %simulations
samp=dec2bin(sample(i,:),19);
sampl= samp([1 2 3 4 5 6 7 8 9 10 11 12 13 14 ]);     % only 14 MSBs of the sine value
                                                      %calculated are considered
sample(i,:)=bin2dec(sampl);
sample(i,:)=(sample(i,:))/(2^14-1);              % sample has sine value between 0-1.
error(i,:)=(sin ((2*pi)*FCW*(i-1)/(2^15)+ 2*pi/2^16 )- sample(i,:));
                                                 %2*pi/2^16 is due to ½ LSB shift.
case 1                                           %for the second quadrant
sample(i,:)= ROM(ci+1)+ step+16;
samp=dec2bin(sample(i,:),19);
sampl= samp([1 2 3 4 5 6 7 8 9 10 11 12 13 14 ]);
sample(i,:)=bin2dec(sampl);
sample(i,:)=sample(i,:)/(2^14-1);
error(i,:)= (sin ((2*pi)*FCW*(i-1)/(2^15) + 2*pi/2^16)- sample(i,:));
case 2                                           % for III quadrant
sample(i,:)=(ROM(ci+1)+ step)+16;
samp=dec2bin(sample(i,:),19);
sampl= samp([1 2 3 4 5 6 7 8 9 10 11 12 13 14 ]);
```

```
sample(i,:)=bin2dec(sampl);
sample(i,:)=-1*(sample(i,:))/(2^14-1);     %sine is negative in III quadrant
error(i,:)=(sin ((2*pi)*FCW*(i-1)/(2^15)+ 2*pi/2^16 )- sample(i,:));
case 3                                          % for IV quadrant
sample(i,:)=(ROM(ci+1) + step)+16;
samp=dec2bin(sample(i,:),19);
sampl= samp([1 2 3 4 5 6 7 8 9 10 11 12 13 14 ]);
sample(i,:)=bin2dec(sampl);
sample(i,:)=-1*(sample(i,:))/(2^14-1);
error(i,:)=(sin ((2*pi)*FCW*(i-1)/(2^15)+ 2*pi/2^16 )- sample(i,:));
end
s=s+FCW;
if s>=65536/2;
s=s-65536/2;
end
l=s;
end
plot(sample(1:100));                        %Plots the sine wave
title('ROM samples')
xlabel('no of samples')
%title('scaled sine value 2^12*sin(90*2^6*(N)/2^(12))')
%xlabel('Number of samples')
nfft=i/4;
```

```
fout=sample(1:nfft);
psd=20*log10(abs(fftshift(fft(fout(1:nfft))))+0.0000001);
%% ERROR CALCULATION
stem(error(1:247*4));                    %shows the error plot
for N=0:1:246*4
F=F+(error((N+1),:))^2;                  %F indicates the total error power
end
f=20*log10(sqrt(N/(F*2)));               %Gives the value of worst case spur, N is the value
                                          %of the total signal power as the amplitude of
                                          %signal is 1.
```

A.3 NON LINEAR ROM DDS ARCHITECTURE

MATLAB CODE

```
FCW=input ('FCW:');          %FCW is the frequency control word provided by the user
s=0;                         %s is the variable storing the accumulator values
i=0;                         %i is variable indicating the number of clock cycles
q=0;
m=0;
sample=0;                    %Sample is the sine value corresponding to the phase word in the
                             %accumulator
ROM1=load('6f1.txt'); % load values from the ROM1 stored in the notepad file '6f1.txt'
ROM2=load('6f2.txt'); % load values from the ROM2 stored in the notepad file '6f2.txt'
l=1;                         %l is  variable to break the phase accumulator loop
step=0;
```

```
stepsize=0;
j=0;
t=0;
k=0;
z=0;
qut=0;
F=0;
FC=0;
FC2=0;
a=0;
u=0;
while s <=(65536/2) & l~=0        %Phase accumulator s is 15 bit wide. When one
                                  %complete sin wave is completed the value in s
                                  %might not be 0 but less than 2^15 so j is used to
                                  %break the loop
l=s;
si=s;
sb=dec2bin(s,15);
%sb2=dec2bin(s,14);
FCL=dec2bin(FCW,15);              %FCL has the binary form of FCW
if s<=4160                        %if accumulator phase is less than 45^o
c=sb([ 3 4 5 6 7 8 9 10]);        %8 MSBs are used to address the ROM1
ci=bin2dec(c);
```

```
else
c=sb([3 4 5 6 7 8 9 ]);                    %7 MSBs are used to address the ROM 2 as it is
                                           %half the size of %ROM1
ci=bin2dec(c)-63;                          % to start the addressing from 0. the first values in
                                           %ROM2 should have an address 0.
end
 MSB=sb([1 2]);
 MSBd=bin2dec(MSB);
  if s<=4160                               %if phase <=45^o
   d=sb([ 11 12 13 14 15 ]);               %last 5 bits indicate the distance between the positions
                                           %of the sine %value that has to be calculated and that
                                           %has been stored in the ROM1.
 else
   d=sb([10 11 12 13 14 15]);              %last 6 bits indicate the distance between the positions
                                           %of the sine value that has to be calculated and that
                                           %has been stored in the ROM2
 end
 di=bin2dec(d);                            %di had decimal value of the distance
i=i+1;
if s<=4160                                 %if phase<=45^o  take values from ROM1
     m=ROM1((ci+1));
    diff=ROM1(ci+2)-ROM1(ci+1);
else
```

```
    m=ROM2((ci+1));                %if phase>45^o  take values from ROM2
    diff=ROM2(ci+2)-ROM2(ci+1);
  end
   if(k~=m)
  j=k;
   k=m;
    end
  t(i,:)=ci;
 if s<=4160
   step=di*(diff/32);
else
   step=di*diff/64;
end
 if i==252                %for FCW=33, i=252 shows more than complete I quadrant of sine
end
  switch (MSBd)
   case 0
        if s<=4160
       sample(i,:)=(ROM1(ci+1)+ step);
        else
       sample(i,:)=(ROM2(ci+1)+ step);
        end
       samp=dec2bin(sample(i,:),19);
```

```
      sampl= samp([1 2 3 4 5 6 7 8 9 10 11 12 13 14 ]);
       sample(i,:)=bin2dec(sampl);
       sample(i,:)=sample(i,:)/(2^14-1);
    error(i,:)=(sin ((2*pi)*FCW*(i-1)/(16384*2))- sample(i,:));
    case 1
   if s<=4160
  sample(i,:)=(ROM1(ci+1)+ step);
  else
 sample(i,:)=(ROM2(ci+1)+ step);
 end
samp=dec2bin(sample(i,:),20);
sampl= samp([1 2 3 4 5 6 7 8 9 10 11 12 13 14 ]);
sample(i,:)=bin2dec(sampl);
 sample(i,:)=sample(i,:)/(2^14-1);
error(i,:)= (sin ((2*pi)*FCW*(i-1)/(16384*2))- sample(i,:));
 case 2
      if s<=4160
      sample(i,:)=(ROM1(ci+1)+ step);
      else
     sample(i,:)=(ROM2(ci+1)+ step);
      end
  sample(i,:)=(-1)*sample(i,:)/(2^20-1);
     error(i,:)= (sin ((2*pi)*FCW*(i-1)/(16384*2))- sample(i,:));
```

```
          case 3
           if s<=4160
           sample(i,:)=(-1)*(ROM1(ci+1)+ step);
           else
          sample(i,:)=(-1)*(ROM2(ci+1)+ step);
           end
         sample(i,:)=sample(i,:)/(2^20-1);
      error(i,:)= (sin ((2*pi)*FCW*(i-1)/(16384*2))- sample(i,:));
   end
  s=s+FCW;
  if s>=65536/2;
    s=s-65536/2;
    end
  l=s;
  end
stem(sample(1:(i-1)));
title('ROM samples')
xlabel('no of samples')
%title('scaled sine value 2^12*sin(90*2^6*(N)/2^(12))')
 %    xlabel('Number of samples')
nfft=i/4;
fout=sample(1:nfft);
psd=20*log10(abs(fftshift(fft(fout(1:nfft))))+0.0000001);
```

```
%% ERROR CALCULATION
stem(error(1:251));
 for N=0:1:250
F=F+(error((N+1),:))^2;
end
f=20*log10(sqrt(N/(F*2)));
```

REFERENCES

[1] J. Tierney et al., "A digital frequency synthesizer," IEEE Trans Audio Electro acoustics. vol. AU-19, pp.48-57, 1971.W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[2] David Brandon, "DDS Design", Analog Devices, May 13, 2004. http://www.edn.com/article/CA415103.html.

[3] Gutierrez-Aitken, et al., "Ultrahigh-Speed Direct Digital Synthesizer Using InP DHBT Technology", IEEE J. Solid-State Circuits, vol. 37, No.9, pp.1115–1119, Sept. 2002.

[4] S. Mortezapour and E. K. F. Lee, "Design of Low-Power ROM-Less Direct Digital Frequency Synthesizer Using Nonlinear Digital-to-Analog Converter," IEEE J. Solid-State Circuits, vo1.34, no.10, pp. 1350-1359, 1999.

[5] Jiangdong Jiang and Edward K.F. Lee, "A ROM-less Direct Digital Frequency Synthesizer Using Segmented Nonlinear Digital-to-Analog Converter", IEEE Conference on Custom Integrated Circuits, pp. 165 – 168, 2001.

[6] J.del Pino, A. Hernaddez, J. Garcia, B. Gonzalez and A. Nunez, "Performance analysis and Propagation Delay Time Estimation of Logic Families with HBTs", Centre for Applied Microelectronics, University of Las Palmas, Spain. www.dice.ucl.ac.be/~anmarie/patmos/papers/S11/11_4.pdf.

[7] Dayu Yang and Foster F. Dai, "A 10GHz Nonlinear Cosine-Weighted DAC in High-Speed DDS", IEEE 5th Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems, Atlanta, GA, September, 2004.

[8] Yongchul Song, Beomsup Kim, "A 14-b Direct Digital Frequency Synthesizer with Sigma Delta Noise Shaping" IEEE J Solid State Circuits, vol 39, pp 847-851, May 2004.

[9] H. T. Nicholas, et al., "The optimization of direct digital frequency synthesizer performance in the presence of finite word length effects," Proc. 42nd Annual Frequency Control Symp. USERACOMM (Ft. Monmouth, NJ), May 1988, pp. 357-363.

[10] B. H. Hutchison, Jr., Frequency synthesis and applications, New York: IEEE Press, 1975.

[11] Bellaouar, M. Obrecht, A. Fahim, and M. I. Elmasry, "A low-power direct digital frequency synthesizer architecture for wireless communications," IEEE J. Solid-State Circuits, vol. 35, pp 385-390, March 2000.

[12] M.M. El Said, M.I. Elmasry, "An improved ROM compression technique for direct digital frequency synthesizers", Proc. of IEEE Symp. On Circuits and Systems, vol.5, pp.437-440, May 2002.

[13] J.M.P. Langlois, D. Al Khalili, "A Novel approach to the design of direct digital frequency synthesizers based on linear interpolation", IEEE Trans. on Circuitand System II: Analog and Digital Signal Processing, vol.50, n.9, pp.567-578, Sept. 2003.

[14] Charles G. Ekroot and Stephen I. Long "A GaAs 4-bit Adder-Accumulator Circuit for Direct Digital Synthesis," IEEE J. of Solid State Circuits, vol. 23, no. 2, pp. 573-580, April, 1988.

[15] Malinky Ghosh, Lakshmi S. J. Chimakurthy, F. Foster Dai, and Richard C. Jaeger, "A novel DDS architecture using nonlinear ROM addressing with improved compression ratio and quantization noise", IEEE International Symposium on Circuits and Systems (ISCAS), pp.705-708, Vancouver, Canada, May 2004.

[16] Foster F.Dai, Lakshmi S. J. Chimakurthy, Dayu Yang, Jun Huang, and Richard C. Jaeger, "A Low Power 5 GHz Digital Synthesizer Designed In SiGe Technology", IEEE 5th Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems, Atlanta, GA, September, 2004.

[17] D. A. Sunderland, R. A. Strauch, S. S. Wharfield, H. T. Peterson, C. R. Cole**, "**CMOS/SOS frequency synthesizer LSI circuit for spread spectrum communications**",** IEEE J Solid-State Circuits, vol. 19, no. 4, pp 497-506, Aug 1984.

[18] A. Essenwanger, V. A. Reinhardt, "Sine output DDSs. A survey of the state of the art," Frequency Control Symposium, pp 370 -378, May 1998.

[19] Sodagar, G. R. Lahiji, "A pipelined ROM-less architecture for sine-output direct digital frequency synthesizers using the second-order parabolic approximation," Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on Circuits and Systems, vol. 48, no. 9, pp 850-857, Sept. 2001.

[20] J. Vankka et al., Direct Digital Synthesizers-Theory, Design and Applications, Kluwer Academic Publishers, 2001.

[21] F. Curticapean and J. Niittylahti, "Low-power direct digital frequency synthesizer," Proc. IEEE 43rd Midwest symposium on Circuits and Systems, 2000, on CD-ROM.

[22] H. T. Nicholas III and H. Samueli, “A 150 MHz direct digital frequency synthesizer in 1.25 um CMOS with -90dBc spurious performance,” IEEE J. Solid-State Circuits, vol. 26, pp. 1959-1969, Dec. 1991.

[23] J. Vankka et al., “A direct digital synthesizer with an on chip D/A converter,” IEEE J. Solid-State Circuits, vol. 33, no. 2, pp. 218-227.