Direct digital frequency synthesis

by

Lewelyn D'Souza

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Major Professor: Edward Lee

Iowa State University

Ames, Iowa

2000

Graduate College
Iowa State University

This is to certify that the Master's thesis of

Lewelyn D'Souza

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Frequency synthesis has entered an age where it is ubiquitous in applications such as military radios, satellite communications, radars, CB radios and consumer electronics. The reasons include tight control over the available spectrum by government and industry, the convenience of frequency synthesizers, increase in complexity of modulation.

Frequency synthesizer is defined as a system (an active electronic device) that generates one or many frequencies derived from a single input frequency reference, such that the output frequency is a rational multiple of the input.

1.1 Motivation

Reliable high frequency synthesis circuits today form the cornerstone of most communication systems. The GSM[1] system for cellular phone systems operates in two frequency bands around 900MHz. Optimal spectral efficiency is achieved with narrowband Time Division Multiplexing Techniques, which allows 124 channels to be placed in a 25MHz band. Frequency synthesis circuits have to be able to provide low values of SFDR and phase noise at these high frequencies, while allowing phase continuous frequency changes.

For a typical application, we can look at the structure of a standard transceiver (transmitter-reciever)[6]. A transciever is a building block of a wireless communication system which interfaces between the user and the transmission medium. The front end

[1]The Groupe Special Mobile or GSM was created in 1982 to set standards in wireless communication.

for this block, performs frequency conversions between high-frequency antenna signals and low-frequency baseband signals. Two configurations for this front end are shown in Figure 1.1, which shows a heterodyne or IF receiver, and Figure 1.2, which shows a zero-IF receiver.

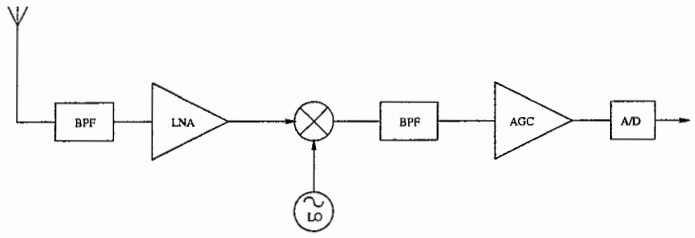

Figure 1.1 IF receiver architecture

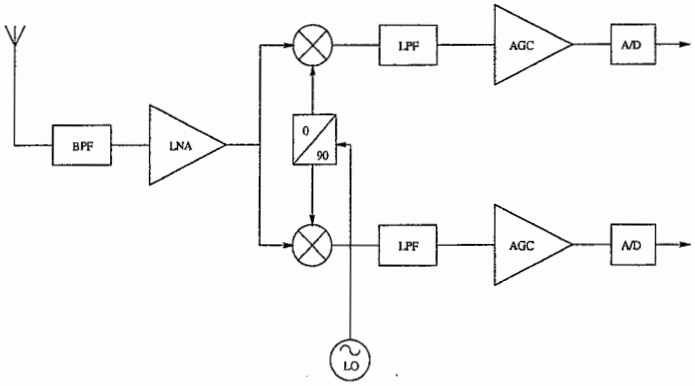

Figure 1.2 Zero-IF receiver architecture

Both these architectures amplify the antenna signal in a Low noise Amplifier, after removing unwanted signals in an RF Band Pass Filter. In the IF receiver, the signal is mixed down with the local oscillator signal to a relatively high intermediate frequency. The required channel is selected with another band pass filter, and digitized in the A/D converter after some automatic gain control. The transmitter architecture is similar, an IF frequency is used with a band pass filter to remove the image signal.

DDS synthesizers are ideal for quadrature modulation, since creation of I and Q is achieved by adding a second look up function with a phase mapping offset 90^{o}. Also intermediate frequency (IF) signals can be generated at same frequency as the reciever's IF's ([4], [5]). This allows sharing of certain IF filters and reducing tranciever size and weight.

Discrete realizations allow the choice of best technology for every building block. GaAs circuits provide excellent performance for Low Noise Amplifier and Power Amplifier blocks. And bipolar transistors, which have f_T values of several tens of GHz are used for the mixer and phase shifter blocks. But in order to integrate the whole receiver block, these two technologies cannot be combined to have the analog and digital portions on the same IC. BICMOS has the disadvantage that neither the bipolar nor the CMOS transistors are as good as the discrete technologies. Thus the frequency performance of the analog bipolar part will be worse, as will the power and area of the digital part. The realization of single chip trancievers is possible only if the high frequency analog processing circuits can be realized in a standard sub micron CMOS process, thus lending themselves to high volume production.

In the past, high end DDS components were restricted by cost to relatively low volume production. This work tries to minimize power/area/cost requirements of the ROM in standard DDS designs, so that they can be used for mainstream communication systems.

Some other areas which stand to benefit from such integration of high frequency synthesizers are digital radios and modems, frequency hopping systems, high performance test equipment, etc.

1.2 Review of different techniques

Three major types of frequency synthesis techniques are in current use.

1. **Phase locked loop synthesis**

   This is the most widespread technique, mainly due to its simplicity and economics, and is used in sophisticated radar systems as well as consumer electronics like car radios etc.

2. **Direct analog**

   This technique is more complicated than PLL, and hence is more expensive. Its main use is in medical imaging and spectrometers, fast-switching anti-jam communications and radar, where the advantages offered by this technique justify its cost.

3. **Direct digital**

   This technique has rapidly evolved to provide an economical, high performance tool for frequency synthesis, and is now used by instrument makers, satellite communications, radar, medical imaging, cellular telephony, etc. Since the output waveform is built from ground up, different types of modulation - frequency, phase and amplitude, are easily accomplished. The theory behind DDS is explained in chapter 2, while chapter 3 deals with the specific design idea used for the current effort. The implementation details are described in chapter 4.

The PLL and Direct Analog techniques are described below, and the Direct digital method is explained in the following chapter.

1.2.1 PLL synthesis

In this type[2], the reference frequency is multiplied by a variable number. The idea is to divide the output frequency by the required number, and adjusting the output frequency such that the *divided* frequency is equal to the reference frequency.

Figure 1.3 shows the block diagram of a simple PLL loop. The VCO output frequency is divided by a variable number N in the frequency divider. This divided frequency is compared to the reference frequency in the phase detector, which gives an output signal equal to the phase difference between its two inputs. The signal is low pass filtered by the loop filter, and is the control input to the VCO. Under conditions of lock, the two inputs of the phase detector have a constant phase relationship and thus equal frequency.

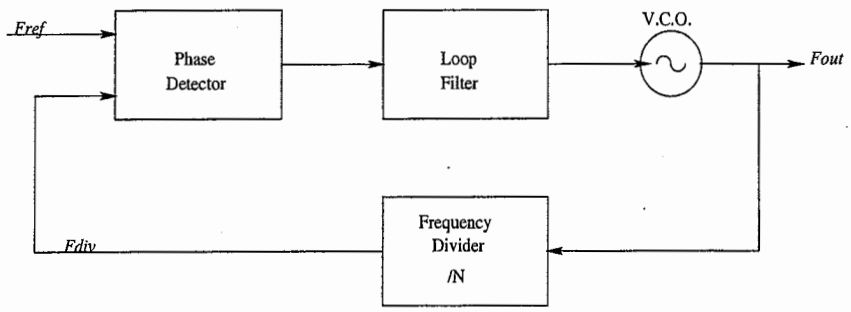

Figure 1.3 PLL block diagram

The output frequency therefore is

$$F_{out} = N.F_{ref} \tag{1.1}$$

If the output frequency increases, the phase difference between F_{div} and F_{ref} will drop and the phase detector output will decrease. The VCO is tuned by this action to a lower frequency till the correct frequency is reached. The loop filter suppresses undesired components in the phase detector output, and has an important effect on noise, acquisition of lock, response speed, loop stability etc.

[2]Also reffered to as the *indirect* type

The PLL is analyzed by considering the phase of the reference and the phase of the VCO signal as loop variables. Figure 1.4 shows this idea. The input signal has the phase $\theta_{ref}(t)$ and the VCO output has a phase $\theta_{out}(t)$ The prescalar divides the VCO frequency (and hence the VCO phase) by a factor N:

$$\theta_{div}(t) = \frac{\theta_{out}(t)}{N} \tag{1.2}$$

We assume that the loop is locked and that the phase detector gives an output voltage proportional to the difference in phase between its inputs:

$$v_{pd}(t) = K_pd.[\theta_{ref}(t) - \theta_{div}(t)] \tag{1.3}$$

where K_{pd} is called the phase detector gain factor and is measured in units of $V/rad.$

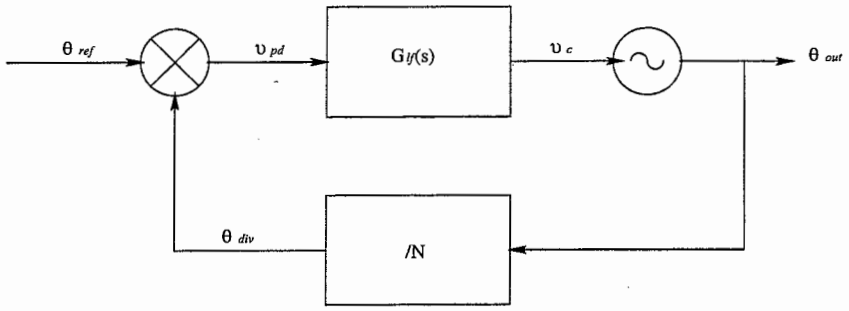

Figure 1.4 PLL state variable diagram

This phase error voltage v_{pd} is filtered by the loop filter with its transfer function $G_{lf}(s)$, which normally has a low pass charateristic. Noise and high frequency components are suppressed.

The VCO frequency is determined by the control voltage v_c. The deviation of the VCO center frequency from its center frequency is $\triangle\omega = K_{vco}.v_c$ where K_{vco} is the VCO gain factor in units of $[rads/Vs]$. Since frequency is the derivative of phase, the VCO operation can also be described as

$$\frac{d\theta_{out}(t)}{dt} = K_{vco}.v_c(t) \tag{1.4}$$

Taking Laplace transforms

$$\theta_{out}(s) = \frac{K_{vco}.V_c(s)}{s} \tag{1.5}$$

In Figure 1.5 the PLL is shown as a standard feedback network with a forward transfer function *G(s)* and feedback factor *H(s)*. The open loop transfer function equals

$$GH(s) = \frac{K_{pd}.G_{lf}(s).K_{vco}}{N.s} \tag{1.6}$$

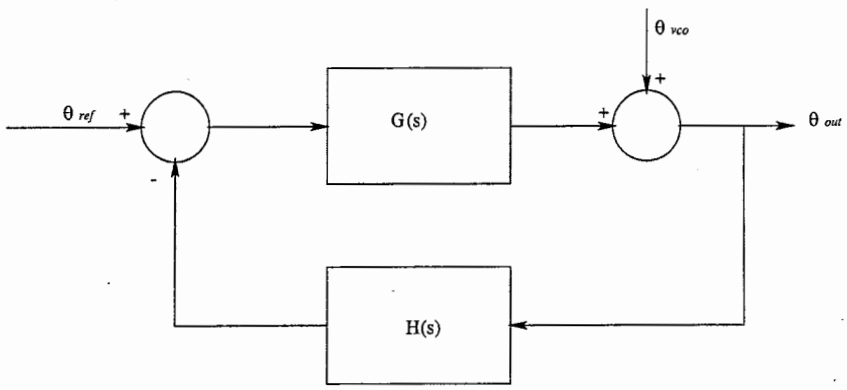

Figure 1.5 PLL Feedback system

The response of the phase-locked synthesizer to have the output frequency changed is inherently slower than other types of synthesis techniques. Changing the frequency is done by changing the divider modulus N, which results in slow change of the VCO control voltage as the loop acquires its steady state operation. The loop filter and the reference frequency play an important role in this process. Fast frequency change is possible only when the loop bandwidth is large, but the loop bandwidth must be limited to $1/10^{th}$ of the reference frequency.

The VCO determines the spectrum at frequencies further away from the carrier, and has to be designed carefully, for low noise high quality operation [1], [2], [3]. Once the loop is locked, it must be capable of following changes in the reference frequency or

the division modulus N without loosing the locked status, which is where a good phase detector design is required.

1.2.2 Direct analog

This type of synthesizer uses multiplication, mixing and division to generate the desired frequency from a single reference ie. arithmetic operations are performed on the input reference in the frequency domain.

To demonstrate the basic elements of the direct analog technique, consider a synthesizer which has to generate frequencies from 16Mhz to 16.99MHz, has a .01MHz step size, and has a 10MHz reference. Figure 1.6 shows the block diagram of such a synthesizer. The first stage generates frequencies from 16.0MHz to 16.9MHz, while the output of the second stage ranges from 16.0MHz to 16.99MHz. These different frequencies are generated by changing the position of the appropriate switches. For example: to generate 16.1MHz, 131MHz is successively mixed with 14MHz and 16MHz etc.

By adding more similar stages, the resolution of the synthesizer can be increased to any level required, allowing the same stage to be used without using any more references. Usually in these structures, the reference frequencies are generated direct analog methods itself rather than PLL. Reference frequencies required for the example explained above may be generated as follows : a 10MHz comb generates 10MHz comb lines 10, 20, 30, ..., 140 MHz. A 14MHz reference is generated by 140/10, or 70/5, etc.

Since the output frequency is basically a replica of the reference, the output spectrum can be relatively pure, provided that unwanted parts are efficiently filtered out by the bandpass filters. The switching speed on the other hand, depends on the switches, and the response time of the filters.

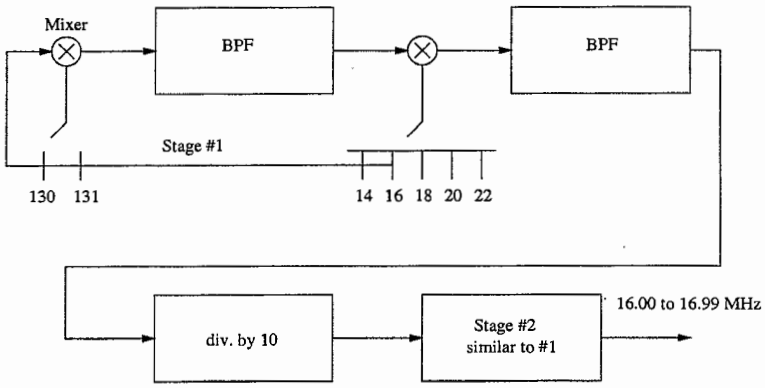

Figure 1.6 Direct Analog Synthesis

2 DDFS FUNDAMENTALS

2.1 Principle

Frequency synthesis using direct digital technique is based on the sampling theorem (Shannon) :

> Any stochastic signal with finite energy having a band limited spectrum i.e., it has no frequencies above a frequency ω_0, can be represented by its discrete samples in time, provided the sampling rate is at least $2F_0$ where $F_0 = \omega_0/2\pi$.

Figure 2.1 shows the fundamental blocks in a DDFS, based on an architecture originally proposed by Tierney et al[7]. DDFS operation is the exact reverse of sampling a signal: we construct digital samples of the signal required and convert it to analog for the final output. In order to generate a sine wave given by

$$Asin(\omega t + \phi) \tag{2.1}$$

we note first that the signal phase is a linear function: the gradient or slope of the phase $d\phi/dt$ is the angular frequency ω. This linear phase is generated by the phase accumulator part of the DDFS in figure 2.1. Figure2.2 shows the phase accumulation of a sinewave whose frequency is equal to 1/8 of the clock frequency. The circle shows the process of phase accumulation of $\pi/4$ at each clock cycle. The dots on the circle represent the phase value at a given time, and the sine wave shows the corresponding amplitude

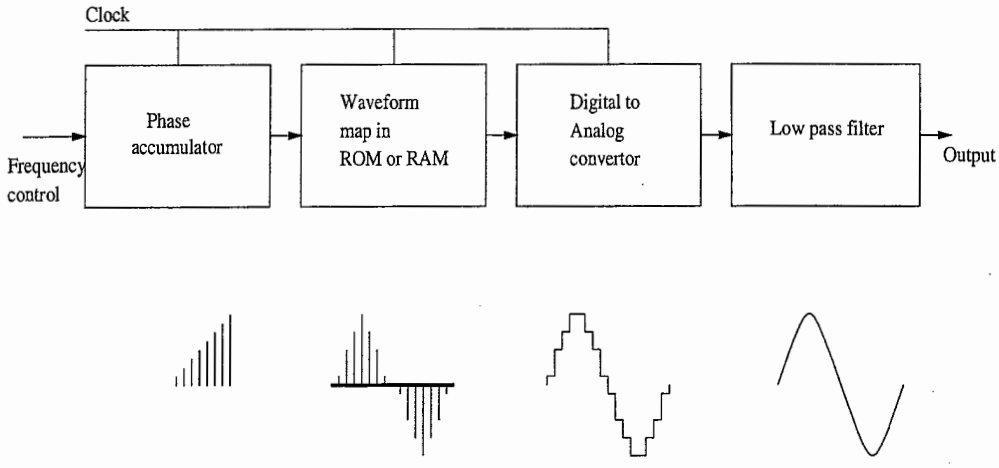

Figure 2.1 DDFS blocks

representation. The phase accumulator then, is a digital integrator that performs the function

$$S(n) = S(n-1) + W \tag{2.2}$$

where W is the input accumulator word[1]. The output of the accumulator is a linear output ramp whose slope is given by W, the input control word.

The phase to amplitude conversion occurs in a *sine lookup*. This transformation is nonlinear and some type of memory (ROM/RAM) is the conventional practice. The phase accumulator is usually large and all the bits are not sent to the sine lookup table. This is done in order to save the ROM size and DAC resolution. Typically, a DDFS will have 32 bits of phase resolution, and 12 bits would be fed to the sine lookup following it. There is a phase truncation error going from the phase accumulator to the rest of the DDFS, which is examined in detail in section 2.3.2. Each overflow of the accumulator represents a phase change of $0 - 2\pi$, hence a complete cycle of output sine wave.

Digital samples of the sine wave obtained at the output of the sine lookup is converted to an analog sine wave by the Digital to Analog Converter. A low pass filter at the output

[1]equivalent to $\pi/4$ phase increment in the figure 2.2

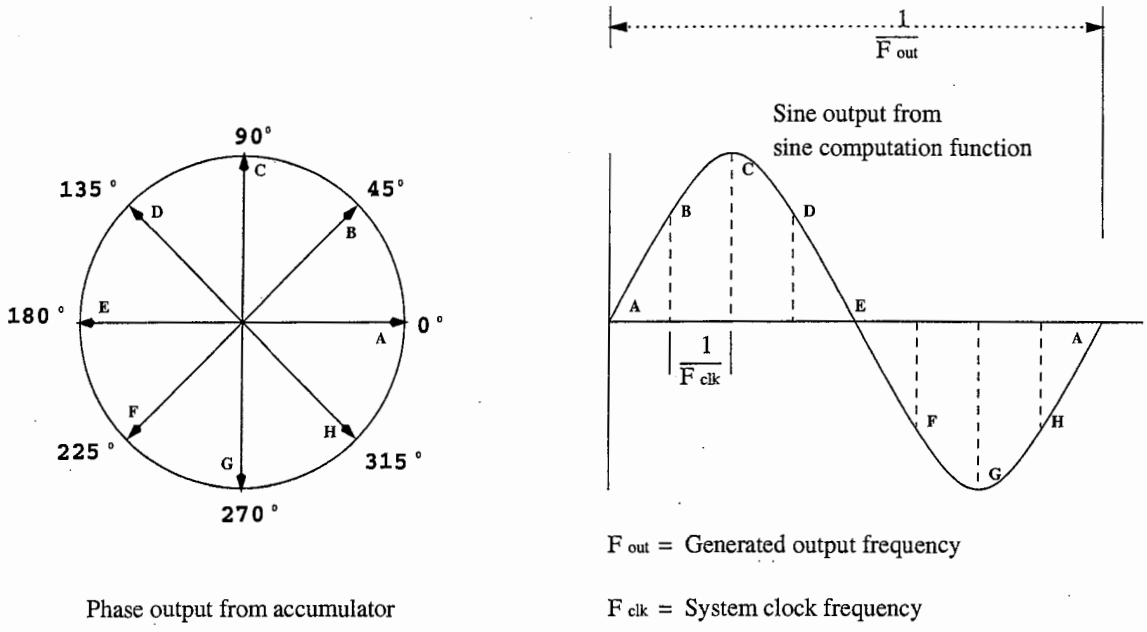

Figure 2.2 Sample sine wave

removes all aliasing frequencies to give the smooth sine of the required frequency.

Consider a 32 bit accumulator (N=32). If the clock frequency is $F_{clk} = 2^{32}/10$, then for an input word of $W = 1$ it will take 10s for the accumulator to overflow (i.e., the phase to change from $0 - 2\pi$). When $W = 2^{30}$, it takes exactly $4 * 10/2^{32}s$ (4 clock cycles) to overflow. We thus see that the period (hence the frequency) of the output depends on the input word W. The output frequency is given by:

$$F_{out} = 2\pi \frac{d\phi}{dt} = \frac{W}{2^N/T} = \frac{F_{clk}W}{2^N} \tag{2.3}$$

T = clock period = $1/F_{clk}$

W = input control word to accumulator

N = 2^n are the maximum accumulator states

2.2 DDFS parameters

Some parameters that are relevant in frequency synthesizer design are as follows:

2.2.1 Output frequency

The output frequency of the DDFS is theoretically limited to the Nyquist frequency[2]. Though theoretically we can synthesize upto $F_{clk}/2$, practically we can realize only frequencies upto $0.45F_{clk}$ due to difficulties in realizing the LPF.

2.2.2 Resolution

The smallest increment in frequency (step size) available at the output is called the resolution of the frequency synthesizer, and is given by $F_{clk}/2^N$, for an N-bit accumulator.

2.2.3 Speed

The switching speed of a DDFS is the time it takes the output to settle to a new output frequency from the time the input control word is changed. This delay depends on the propagation delay of signals through the accumulator, look up table logic, and the settling time for the output DAC. In case pipelining is used, the delay required to flush the new control word into the pipeline should also be considered. However, this pipeline delay is fixed, and can be accounted for by the system in use, as opposed to switching transients.

An issue here is the time required to propagate through the accumulator. Consider a 32 bit accumulator which has its 14 most significant bits connected to the ROM. If the LSB is changed, it takes this new bit 2^{32-14} clock cycles to propagate through the accumulator before it has an effect on the output. For a clock frequency of 100MHz, the frequency resolution is .025Hz. The time required to propagate the LSB change to

[2]i.e., at least two sample of the sine wave are required for synthesis

the output is $2^{18}/100M = 2.5ms$. It is clear that this time does not seriously affect the output frequency.

2.2.4 SFDR

The Spurious Free Dynamic Range or the SFDR defines the level of any discrete output frequency not related to the carrier. The SFDR is the ratio between the signal amplitude and the largest distortion component. Unlike noise, spurious signals are discrete spectral lines not related to the carrier, meaning they exhibit periodicity. The next section deals with the sources of spurious frequencies, and methods to minimize them in the DDS.

2.3 Spurious frequencies

The following sections explain some of the reasons for spurious signals at the output of the DDFS. A technique used in the present work to reduce the level of these spurious frequencies is also explained.

2.3.1 Phase error

In order to see how phase error is produced in the DDFS, consider the following simplest type of synthesizer (Fig 2.3). It uses only an accumulator and the MSB (i.e. the carry output signal) as its output.

The accumulator follows the equation

$$S(n) = S(n-1) + W \tag{2.4}$$

where $S(n)$ is the accumulator output at clock tick n and W is the input control. If we assume that the carry output is used as the output signal, and given that the accumulator performs a modulo 2^N arithmetic (for an N-bit binary accumulator), the average output

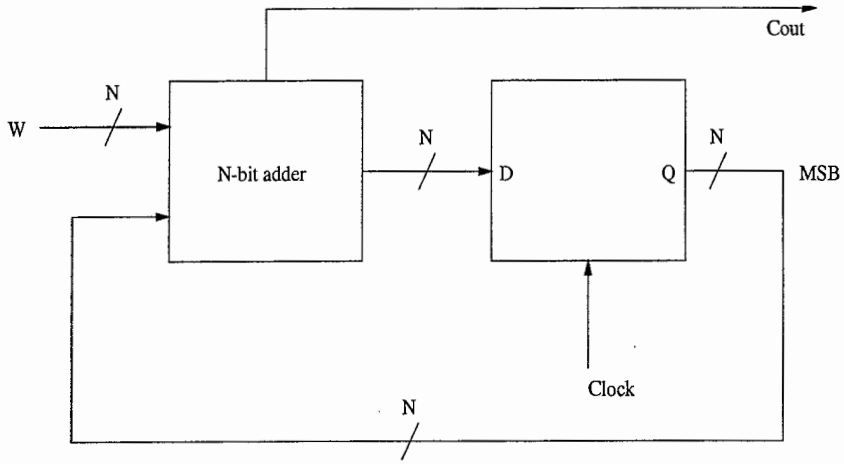

Figure 2.3 Single-bit DDFS

overflow frequency is given by

$$W_{av} = \frac{W}{T.2^N} = \frac{W.F_{ck}}{2^N} \tag{2.5}$$

where F_{ck} is the clock frequency. A typical output for N=4, W=4 and W=5 is

0 4 8 12 0 4 8 12 0 4 8 12 0 .....W = 4 (Fig 2.4)

0 5 10 15 4 9 14 2 7 12 1 6 11 0 .....W = 5 (Fig 2.5)

Note that when W divides 2^N, the output is periodic and smooth, while when W is not a factor of 2^N, a phase error is created between the ideal and actual output frequency. In the case above, the phase goes to zero in 5 cycles of the output, and this creates a new periodicity cycle, different from the desired output and is a source for undesired, spurious cycles.

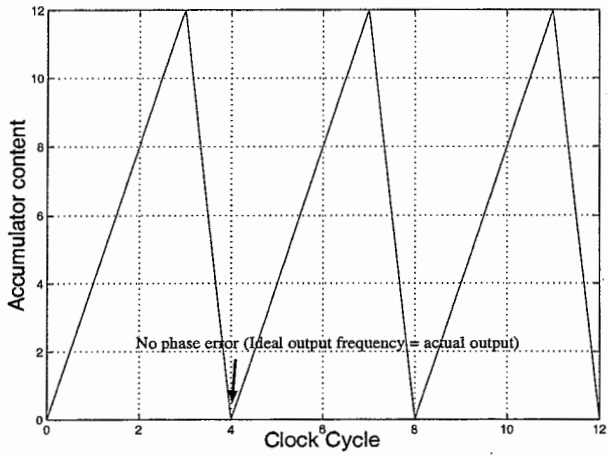

Figure 2.4 When W divides N

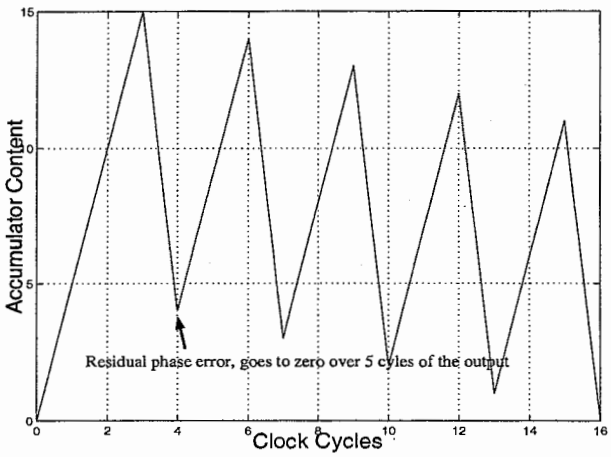

Figure 2.5 When W does not divide N

2.3.2 Quantization errors

Since we use finite arithmetic to represent the signals in the DDFS, there is always some quantization error, which generates spurious signals. The errors and the corresponding spurious frequencies produced have been studied extensively. A mathematical model of a direct digital synthesizer is shown in Figure 2.6 ([8], [9], [10]).

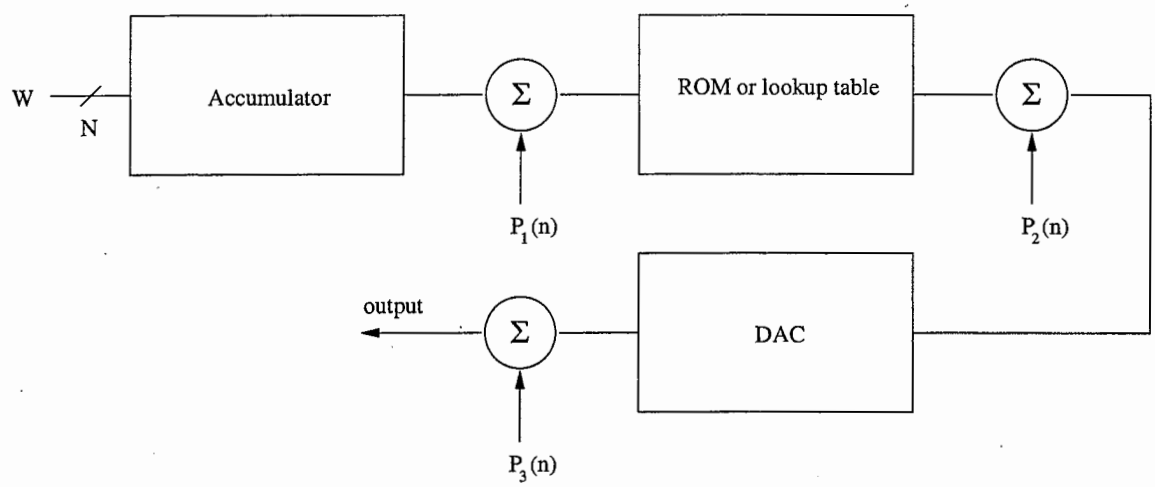

Figure 2.6 DDFS quantization model

The first source of noise $P_1(n)$ is introduced because of the truncation in the bits which are fed from the accumulator to the ROM. The lookup table (ROM) usually uses some sort of compression algorithm, and $P_2(n)$ represents the error due to this, while $P_3(n)$ represents the analog inaccuracies of the DAC, including its quantization effects.

If the accumulator is composed of N bits and only W bits are connected to the ROM, then the accumulator can be considered as a device that generates only W integer digits and $N - W = F$ fraction bits. All the numbers that are generated in the lower F bits do not affect the output directly, only propagate to the W higher bits. This is a major source of error generation, since there are a large number of multiples of $2\pi/2^N$ which are generated in the accumulator for which there is no corresponding storage on the

ROM.

By the nature of the accumulator operation, the error is periodic, and the periodicity of this error determines the position of the spurious signal that it will generate in the output. Only output frequencies for which only W bits (where W is the number of bits fed to the ROM from the accumulator) are controlled do not generate phase quantization. However, their number is negligible relative to the the total number of frequencies. For e.g.: N = accumulator bits = 32,

W = bits passed form accumulator to ROM = 14,

Total number of frequencies generated = 2^{31},

Frequencies that do not generate phase quantization = 2^{13}.

Any other frequency creates a residual phase that propagates in the accumulator and generates a new cycle that will be expressed as a spurious signal.

Also, although these frequencies do not generate phase quantization in the DDS artihmetic, there is always a phase quantization error since the real value of the phase is represented by a finite bit word and these frequencies will generate spurs too.

As an illustrative example, consider the following. Let N = 4, W = 2. Here, the accumulator will reach the exact state $2^N(2\pi)$ after 8 clocks. The output frequency will be $F_{clk}/8$ and the accumulator states will be

0, 2, 4, 6, 8, 10, 12, 14, 0,...

However if W = 6, then the cycle will be

0, 6, 12, 2, 8, 14, 4, 10, 0,...

In this case, 3 cycles of the fundamental output frequency (of frequency $\frac{6}{16}F_{clk}$) were needed to come back to the original state, and this will create spurs at one-third (and its harmonics) of the output frequency given by $6F_{clk}/2^4$.

This is explained in the following manner: The phase accumulator performs the operation of a digital integrator followed by a modulo 2^N operator. While the accumulator generates its main output, given by $W/2^N$ (for a normalized clock), there is another pe-

riodicity (and generally periodicities) being generated, denoted by P, the one for which the phase relation $\phi(i) = \phi(i + P)$ for all i holds. While the main output periodicity is given by

$$\frac{2^N T}{W} \tag{2.6}$$

where T is the clock time, the other periodicity will be determined by

$$\frac{2^N}{gcd(W, 2^N)} \tag{2.7}$$

where gcd(a,b) = is the greatest common divisor of a and b. Only W = $2^G (G < N)$ generates a solution gcd(2^N,W) = W However for the majority of W, this is not the case, and the secondary periodicity will show up at the output as a spurious signal. In the case of N = 4, W = 6,

$$\frac{2^4}{gcd(16, 6)} = \frac{16}{2} = 8 \tag{2.8}$$

Thus the main output period is $\frac{16}{6}$T = 2.666T. The period of the secondary term is P=8T. The spectrum of the digital sine, at the output of the ROM, assuming a perfect ROM, will now consist of a cardinal periodicity of 2.666T, but also one that is 3 signal cycles, or 8T. A spurious signal at +1/3 the fundamental (and its harmonics) will exist.

2.4 Reduction of spurs

Various techniques have been proposed to reduce the spurs caused by the finite wordlength representation of phase and amplitude samples [11], [12], [13]. Figure 2.7 shows the Nicholas Phase Accumulator technique ([11], [8]) used to reduce the effect of spurs on the output of the DDFS. This method doesn not actually destroy the periodicity of the error sequences, but it spreads the spur power into many peaks. k is the number of phase bits taken from the phase accumulator output of m bits which are actually applied to the sine lookup, and W is the input word. In figure 2.7 for $reset = 1$, the carry input

toggles between 0 and 1 periodically, and has the effect of adding a $1/2LSB$ weight to the phase accumulator. This modifies the existing m bit accumulator to emulate the operation of a phase accumulator with a word length of $j+1$ bits under the assumption that the least significant bit of the frequency control word is one. For reset = 0 (and carry input = 0) the accumulator operates normally.

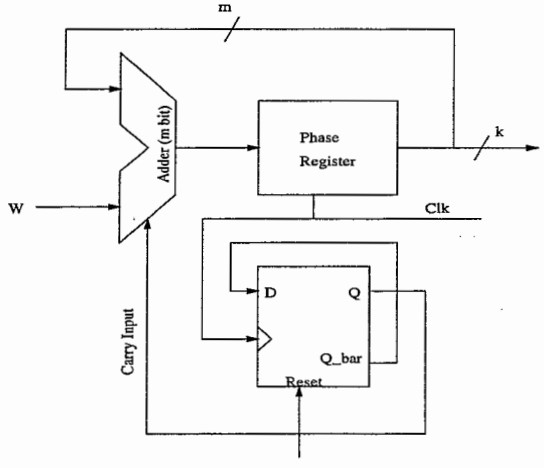

Figure 2.7 Nicholas Accumulator

2.5 Why Direct Digital Synthesis?

2.5.1 Resolution

Consider a DDFS with an accumulator of 32bits, and clocked at 100Mhz. The frequency resolution is $F_{clk}/2^{32} = .025Hz$, which is excellent. The size of the accumulator controls the frequency resolution, and increasing the accumulator size is simple and adds little cost or complexity to the design.

For a PLL synthesizer, there are two blocks which operate at output frequency, the VCO and the frequency divider (Figure 1.3). High frequencies are difficult to achieve in the frequency divider, which is actually a programmable counter. Hence we require

a prescalar circuit before the divider, so that the divider can now operate at a lower frequency. However, the frequency resolution increases by the number that the prescaler divides the output frequency. In order to achieve higher resolution then, the reference frequency has to be reduced, which forces the loop bandwidth to be low. Low bandwidth for the loop implies higher response time to frequency change. Hence, increasing the resolution in a PLL synthesizer is not very easy.

For a Direct analog type of synthesizer, frequency resolution depends on the number of multiply-mix-divide blocks, and though increasing resolution is easier than the PLL type, extra stages have to be more accurate and hence expensive.

2.5.2 Switching speed

DDFS provides the fastest switching speed among all the synthesis techniques described previously. Section 2.2.3 describes that in a DDFS the time required for the output to settle at its new value when the input control word changes is s function of the logic and pipelining delays. This is typically on the order of nanoseconds for a GHz synthesizer. For a Direct Analog synthesis, the switching speed depends on the speed of the switches and the response time of the filters, and can be in microseconds. PLL synthesis however offers the lowest speed of all, since changing the frequency is done by changing the divider modulus, which results in a slow change of the VCO control voltage as the loop acquires its steady state operation. Fast frequency change is possible only when the loop bandwidth is large, and the loop bandwidth is normally limited to one tenth of the reference frequency.

2.5.3 Phase continuous switching

This relates to the behavior of the phase of the output signal during a transient state. For phase continuous switching, the phase transition should exhibit almost no transient and should ideally look as shown in Figure 2.8 *b*. This feature is useful when

we need to generate linear frequency modulation (chirp signals), since it generates very little "noise".

Theoretically, a DDFS produces phase continuous switching, because a change in frequency changes only the slope of the accumulator output. In practice, two factors affect this smooth phase transition.

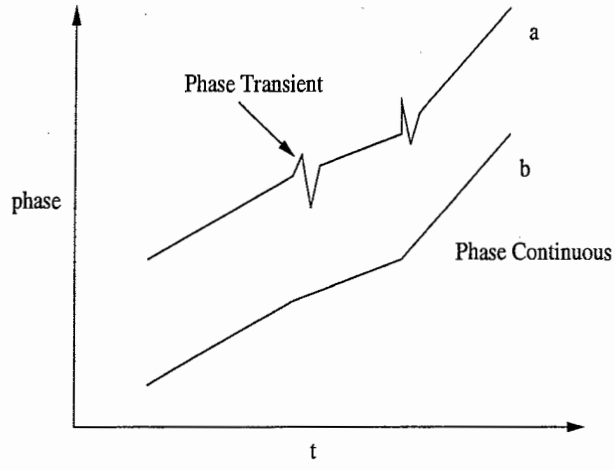

Figure 2.8 Phase switching in transition

Firstly, the change in frequency is a step function, and happens within one clock cycle. Thus the output of the DAC has a smooth transition, but its output is usually followed by a low pass filter. The low pass filter, has its own finite bandwidth, and hence the transition is smooth only if the step size in frequency is within the bandwidth of the filter. If the step is close to the bandwidth of the DDFS, then a transient will be observed. Secondly, the accumulator usually uses pipelining, and it is necessary to wait for the pipeline to fill before a new frequency can be updated. If a new frequency is switched before the pipe is filled, a major transient is likely.

2.5.4 Ease of modulation

Since in a DDFS, there is total control over the parameters, it is easy to add frequency, phase and amplitude modulation. Figure 2.10 shows how various types of modulation can be applied.

Frequency is changed by changing the accumulator input word, which is the basic function of the DDFS. Because of the nature of the DDFS, phase continuous output is easy to achieve (see section 2.5.3). In order to add/subtract a phase shift, we insert an adder between the accumulator and the ROM. Figure 2.9 shows how this concept works to shift the phase. Amplitude is varied by inserting a multiplier between the ROM and the DAC. The design strategy of the current work combines the ROM and the DAC into a single block, and the method of amplitude modulation described above cannot be used, since there is no clear distinction between the ROM and the DAC anymore.

If all these modulation techniques are used, then the DDFS output will be given as

$$A(a)sin[\omega(W)t + \phi(b)] \tag{2.9}$$

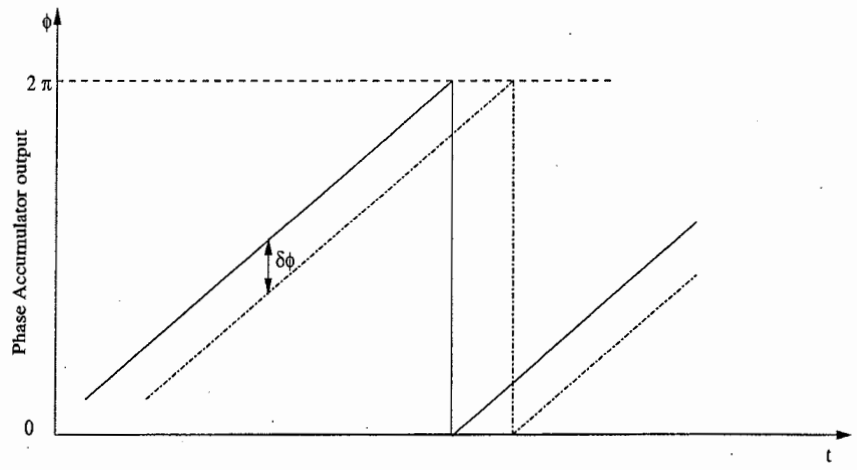

Figure 2.9 Phase control in DDFS

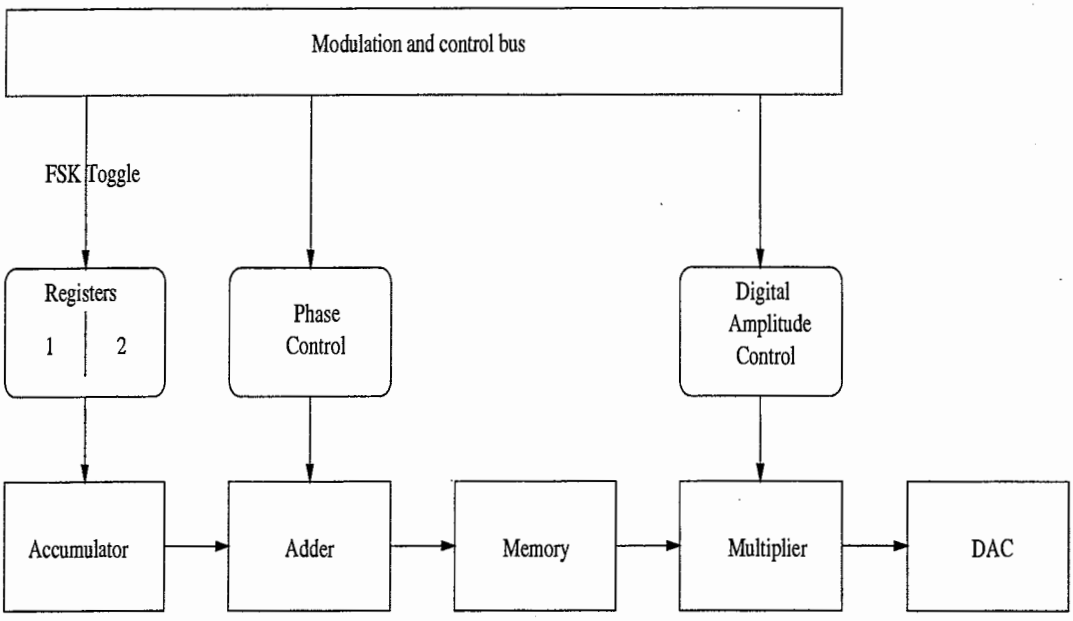

Figure 2.10 Modulation techniques in DDFS

3 ROM-LESS SYNTHESIS

3.1 Introduction

In order to have good spectral purity in the output sine wave, the resolution of the ROM (ie the phase resolution) as well as the DAC, should be high. This requires that the ROM should be large. Various techniques have been used to reduce the ROM size ([10], [14], [15], [16]). However, compression of the ROM requires additional digital circuitry which consumes power, hence reducing ROM size does not neccesarily mean reducing the power dissipation.

The architecture used for the present design combines the ROM lookup table and the DAC into a single non-linear DAC, whose function is to convert the phase bits from the accumulator directly to an analog sine wave output. If the phase resolution[1], and the amplitude resolution of the non-linear DAC is the same as in a conventional DDFS, the performance will be equivalent to a DDFS with ROM. In addition, if the non-linear DAC requires the same power as a linear DAC, the power consumed in the ROM can be eliminated [17].

3.2 Quadrant compression

A simple compression was used in the present design in order to reduce the sine wave information required to be stored by the nonlinear DAC. It is based on the fact that a

[1]i.e. number of accumulator bits fed to the sine lookup portion

Table 3.1 Quadrant Table

Phase	MSB	MSB - 1	Sine
$0 < A < 90$	0	0	sin A
$90 < A < 180$	0	1	sin (90-A)
$180 < A < 270$	1	0	-sin A
$270 < A < 360$	1	1	-sin (90-A)

sine wave is symmetrical in four quadrants (Figure 3.1).

Thus, given information about the first quadrant, the complete sine wave can be constructed. Two MSB's from the output of the accumulator are used to provide the control bits which decide the quadrant being generated at any given time. This relationship is illustrated in table 3.1. The actual implementation of this scheme is illustrated in Figure 3.2.

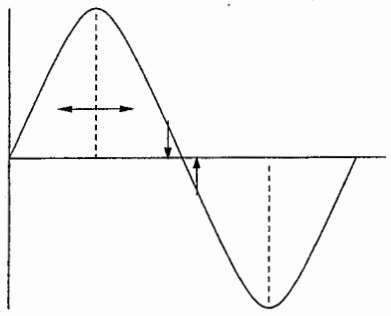

Figure 3.1 Sine Quadrant Symmetry

3.3 Non-linear DAC design

The design methodology for the DAC is now presented. From Figure 3.2 we can see that the nonlinear DAC output depends on the complementer output $St(n)$ and the MSB of the phase accumulator output $\phi(n)$. Let i be the amplitude resolution of the

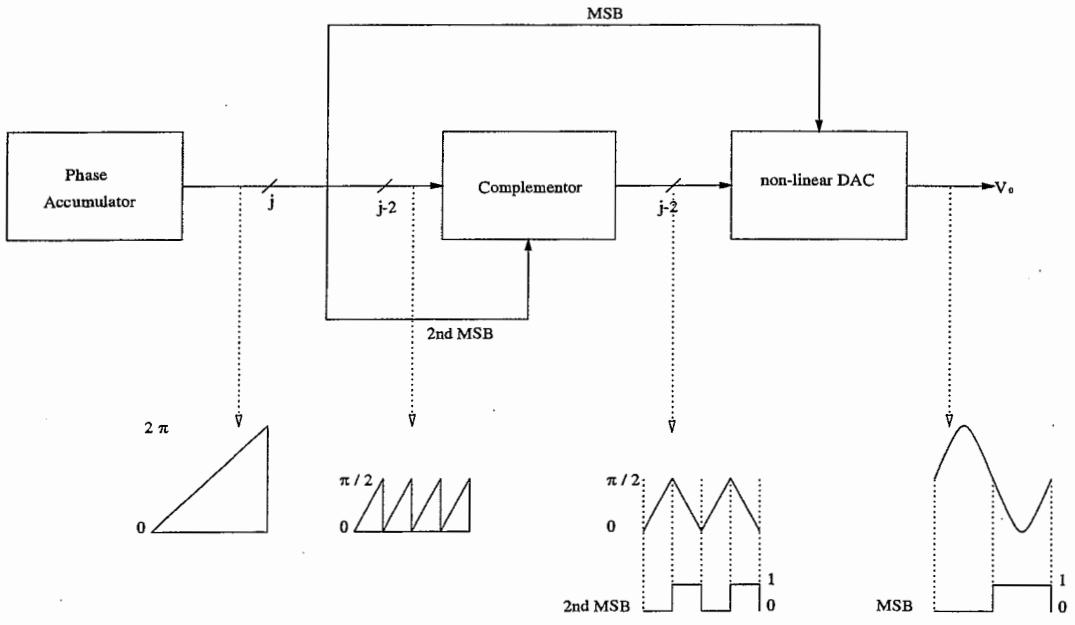

Figure 3.2 DDFS with nonlinear DAC - quadrant compression

output sine wave for one half cycle (i.e. the sine wave reaches a maximum value of $2^i - 1$). The ideal output V_o is then given as

$$V_{o,ideal} = \begin{cases} (2^i - 1)sin(\pi \frac{.5+St(n)}{2^{j-1}}) & for \quad MSB = 0 \\ - \ (2^i - 1)sin(\pi \frac{.5+St(n)}{2^{j-1}}) & for \quad MSB = 1 \end{cases} \tag{3.1}$$

where $0 \leq St(n) \leq 2^{j-2} - 1$. The integer j represents the part of phase accumulator output that is used as input to the nonlinear DAC. As discussed in section 2.1 the accumulator may have a certain size for good phase resolution, but only part of this output is actually fed to the DAC, since it is difficult to get good DAC accuracy at higher resolution. In practice, a 1/2 LSB offset is added to the equation above, so that the output step of the DAC around the zero crossover point is actually displaced by 1/2 LSB (Fig 3.3), and skew at that point when the sine wave is generated by reflection, is minimized. Another advantage of adding the offset is that for the complementer, we can now use simple one's complement (ie, xor gates) instead of using the two's complement

method.

The DAC is realized as a matrix of 2^{j-2} cells, which means it can produce a sine wave output for one quadrant of a phase input which goes from 0 to $2^j - 1$. Each of the DAC cells will produce an incremental analog output step with a bit change in the input change so that it fits the sine wave (Figure 3.3).

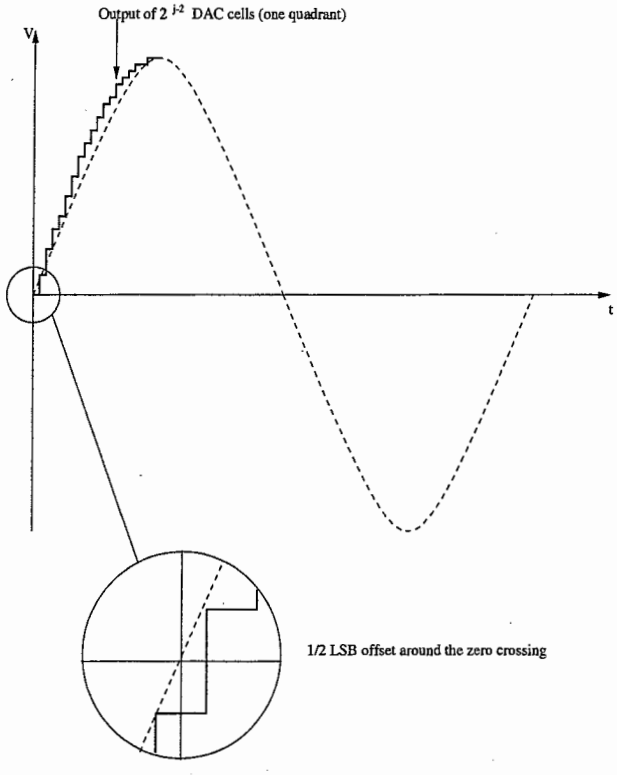

Figure 3.3 Non Linear DAC output

For a given complementer output, the output of the DAC is obtained as the sum of all DAC cell outputs from 0 to St(n) and is written as

$$|V_o| = \sum_{st(n)}^{k=0} C_k \tag{3.2}$$

where C_k represents the DAC cell output. These values for C_k can be calculated as:

$$C_k = \begin{cases} (2^i - 1)sin(\pi \frac{.5}{2^{j-1}}) & for \quad k = 0 \\ (2^i - 1)sin(\pi \frac{k+.5}{2^{j-1}}) - \sum_{m=0}^{k-1} o_m & for \quad 1 \leq k \leq 2^{j-2} - 1 \end{cases} \tag{3.3}$$

The values for C_k thus obtained have to be rounded off to the nearest integer, since the DAC cells will be designed to give currents in integer multiples, and though this will introduce additional quantization error, the matching of DAC current cells for integer will be much better, in addition to making layout easier.

It is seen that the maximum value of the DAC cell ($C_{k,max}$) is obtained when the sine output is zero, where the slope is maximum. This value is given by $(2^i - 1)\pi/2^{j-1}$. It is seen that increasing the phase resolution (j) by one will halve the $C_{k,max}$ whereas increasing the amplitude resolution by one will double the $C_{k,max}$. Increasing the phase resolution will also double the *number* of DAC cells, but since each cell is now 1/2 the original size the overall increase in area is not significant. On the other hand, for a DDFS using a ROM, doubling the phase resolution will double the ROM size. The DAC is realized as a matrix of current cells with values proportional to C_k, and the outputs of the current cells when selected are fed into 50Ω (off chip) resistors.

3.4 System level simulations

The values for the DAC cells were calculated using a C program, and system Level simulations were performed using matlab, to find out the the value of the spurious output with all the quantization errors. It was found that for a phase resolution of 12 bits (j=2) and amplitude resolution of 10 bits (i=10)[2], the C_k values required are in multiples of unit current to 4 times the unit current. Figure 3.4 shows the spectrum of the output for quarter frequency of the clock, as seen the SFDR is about 77db. The spurious performance degrades as the output frequency approaches the clock. The code for the C/matlab programs are given in appendix A.

[2]The amplitude resolution including the sign bit for the DAC is actually 11bits

Another objective of the simulations was to make sure that the DAC cell values calculated using the equations 3.3 are correct. This is important especially because of a pseudo-randomization technique was used for the thermometer decoder[3], to reduce the effect of gradients.

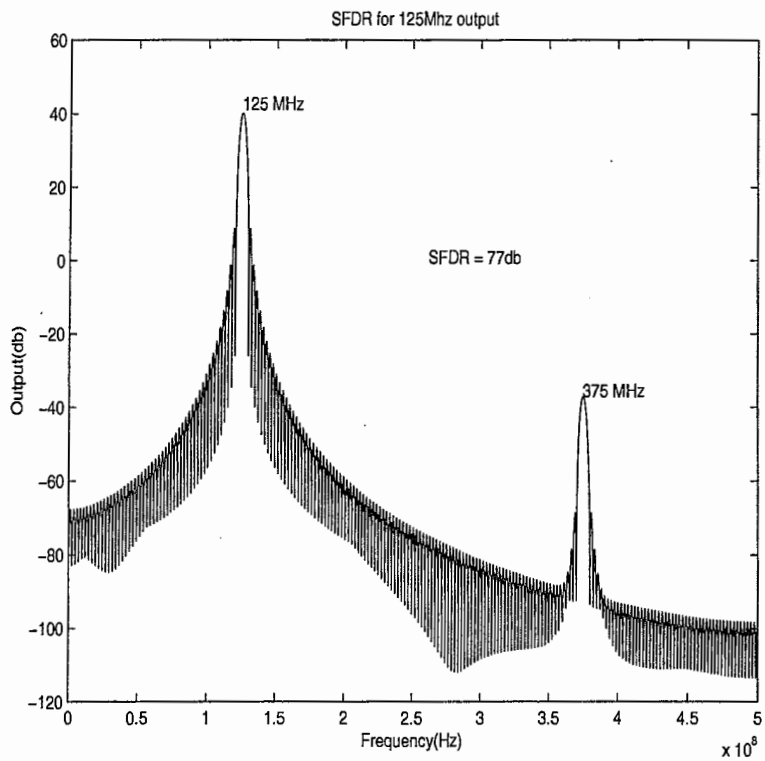

Figure 3.4 SFDR at 1/8th the clock frequency

[3]This randomization technique is explained in chapter 4

4 IMPLEMENTATION

This chapter describes the detailed design of each of the blocks in the DDFS. The aim was to be able to have the clock frequency at 1GHz, and use various techniques to reduce the spurious output as much as possible.

4.1 Block level design

The various blocks in the DDFS are shown in Figure 4.1. The accumulator has a resolution of 16 bits, out of which 12 bits of phase resolution are fed for the DAC (section 3.4). The DAC is implemented as a matrix of cells which are selected by two thermometer code decoders, row and column [18]. The 2 msb's of the phase of output of the accumulator is used to perform the quadrant compression. Hence the row and column decoders for the DAC are 5 bits each.

4.2 Accumulator

The phase accumulator cannot do the complete 16 bit addition in one clock cycle because of the delay caused by the carry propagating through the adder. In order to achieve high frequency operation for the accumulator, a pipelined structure was used [19]. The delay is now only limited to the delay through a flip flop and that through the adder. In this way, increasing the size of the accumulator does not increase the delay of the accumulator, but it does increase the tuning latency of the synthesizer. A block level diagram of the pipeline used in this architecture is shown in Figure 4.2. Only 12

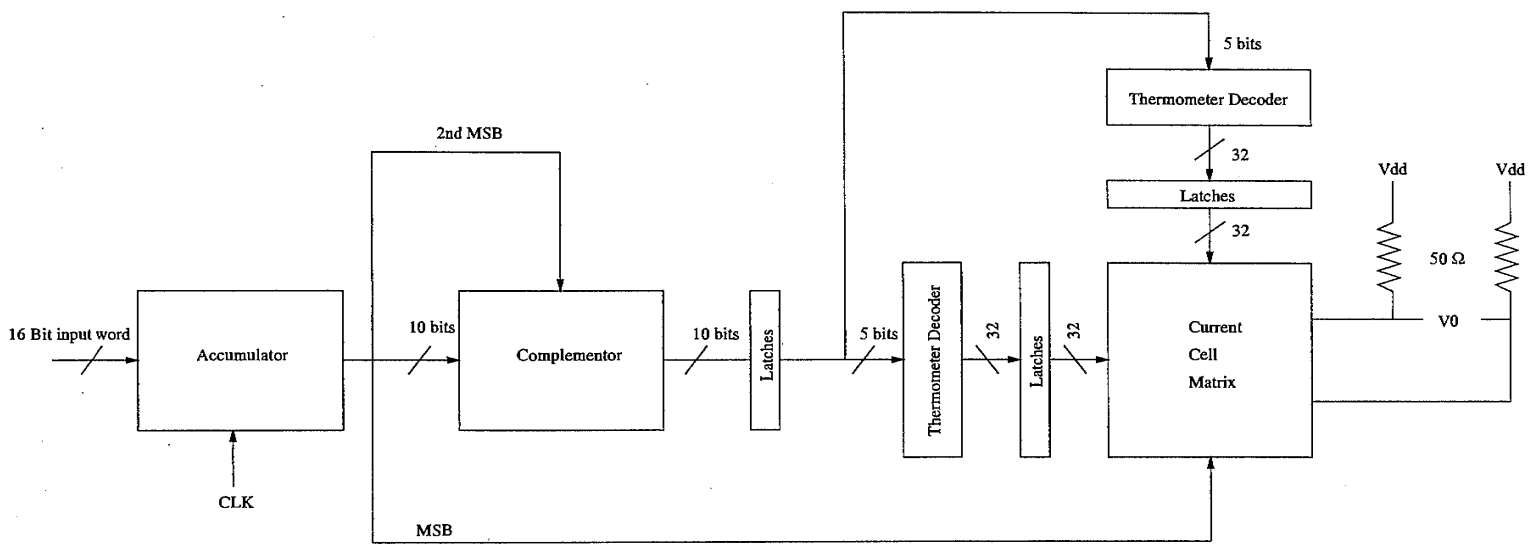

Figure 4.1 Blocks in DDFS implementation

most significant bits of the accumulator are passed to the rest of the circuitry, so the flip flops for the 4 lsb's are actually not implemented, since only the carry is important for the subsequent pipeline stages.

In Figure 4.2, when the Reset = 1, the input to the LSB adder toggles between 0 and 1, and has the effect of randomizing quantization errors in the accumulator. This changes the nature of the spurious response, distributing power in the peaks to the noise floor. The ultimate result is an improvement in the spurious response.

Thus the two basic blocks in the accumulator are the 1 bit adder, and the D flip flop. A conventional CMOS full adder architecture based on transmission gate theory ([20], [21], Figure 4.3) was used for the 1 bit adder, while the latch structure used for constructing the D Flip Flop is given in Figure 4.4 [22]. Two of these latches are used to form a positive/negative edge triggered flip flop. This latch uses ratioed logic, and acts as a negative level triggered latch. The W/L ratios of MN_1 and MP_1 are determined so that the voltage of node n1 remains below V_{Tn} of MN_2, regardless of the input D, during the high period of the clock. Thus, pull up or pull down of the output Q does not happen, because MN_2 and MP_2 remain in cut off when the clock is high. Therefore the signal path from input D to the output Q is not transparent and the latch enters into its hold mode.

When the clock changes state from high to low, the latch enters into its evaluation mode. If the input D is low, node n1 is pulled up to Vdd only by MP_1. The pulldown strength of MN_2 should be sufficiently larger than the pull up strength of MP_2 so that the output low voltage ($V_{OL,Q}$) is lower than the input low voltage (V_{IL}) of the following stage.

It was found that the latch given in Figure 4.4 consumes lot of power, and is required only at the adder inputs where it has to drive the nest stage Flip Flop and one of the adder inputs. An alternative structure used for the rest of the flip flops which don't need to drive as large a capacitance are given in 4.5. The basic latch in this flip flop consists

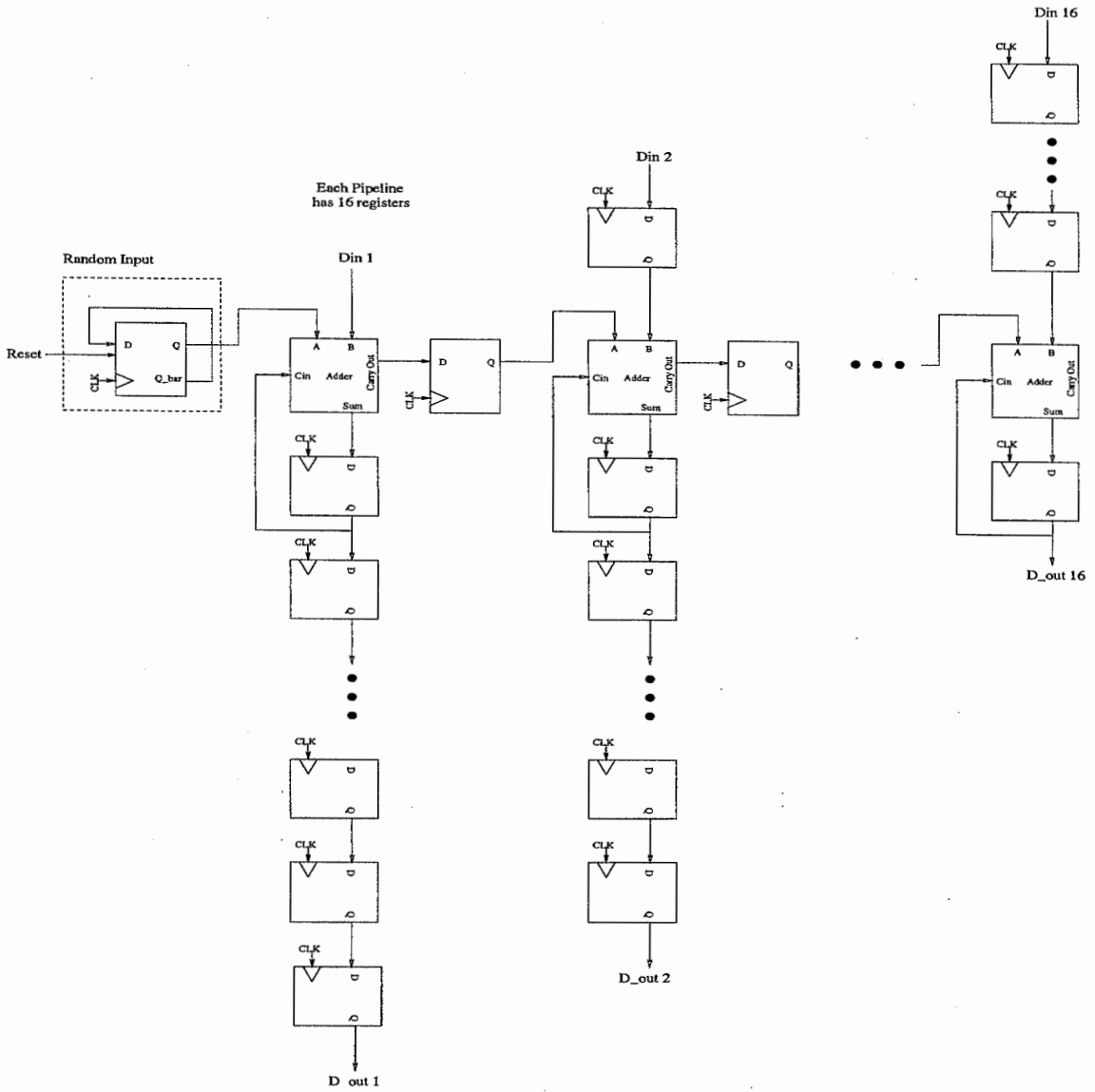

Figure 4.2 Accumulator Pipeline

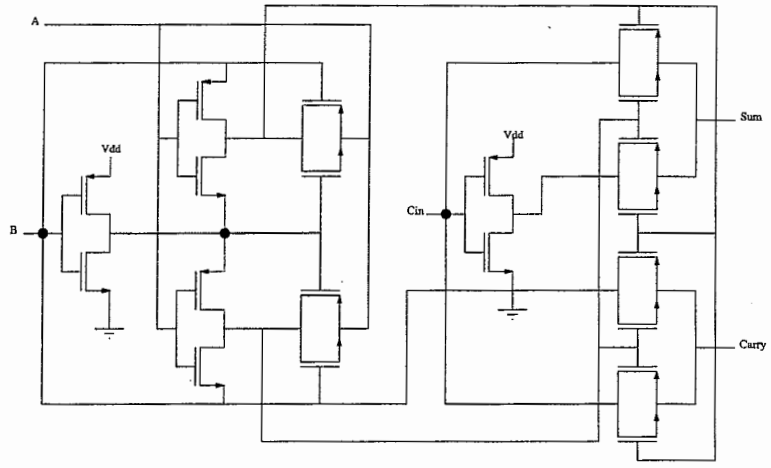

Figure 4.3 Full adder used in accumulator

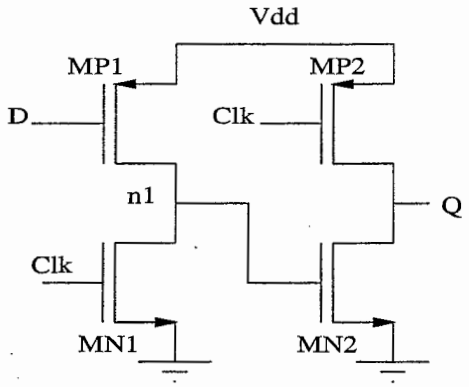

Figure 4.4 Higher drive latch used in accumulator

of two inverters, where the first inverter is clocked.

Figure 4.6 shows the simulation of the 16bit accumulator at a 1GHz clock frequency, for a digital input word of 7 (111000..0). The clock is in the waveform is the nonideal clock at the output of the clock drivers, and the transistor parasitics were simulated by entering the drain/source area and peripheries. The waveform shows the output of the lower five bits of the accumulator, as it steps through 7, 14, 21 etc.

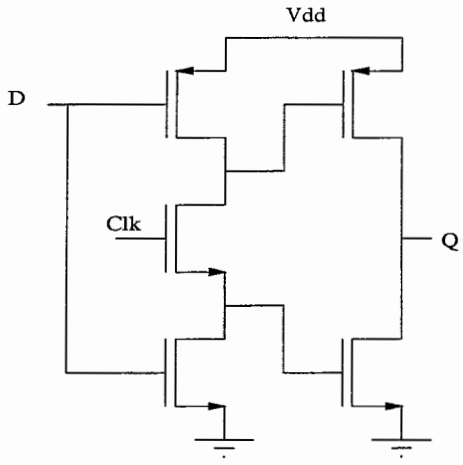

Figure 4.5 Lower power latch used in accumulator

4.3 Complementer/Thermometer code decoder

The complementer is realized as a row of xors (Section 3.3) and requires a set of flip flops at its output, in order that it can drive the thermometer code inputs. The capacitance at the input of the thermometer code decoder is minimized by using a pseudo nmos structure [23]. The decoder was designed with standard K map minimization techniques. The outputs of the thermometer decoder have to drive 32 DAC cell inputs, and the row and column decoder outputs should arrive at the input of the DAC cell at the same time, so that current in the DAC cell can be switched correctly. A row of flip flops with a higher drive ability was placed at the output of the decoders, to have the

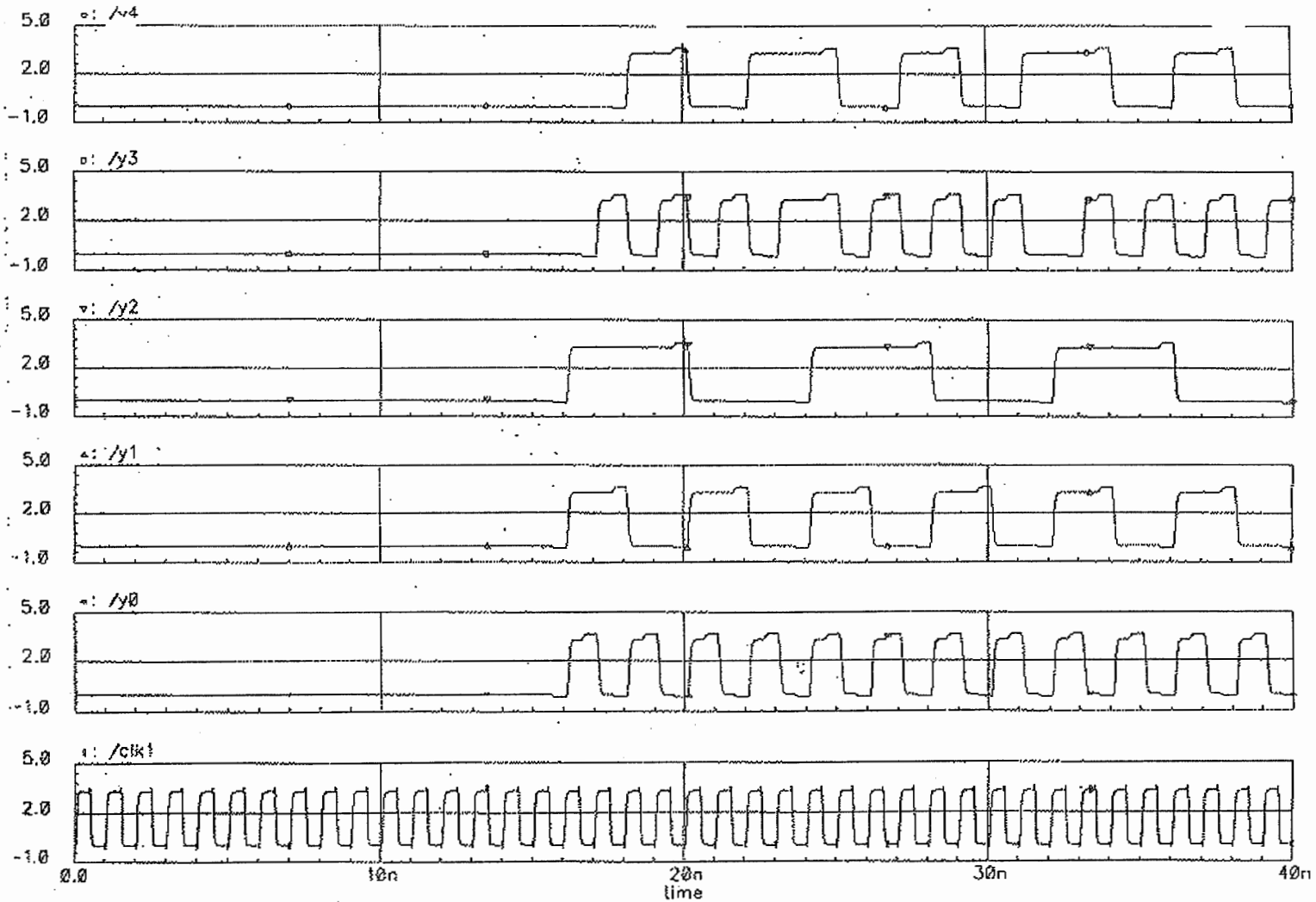

Figure 4.6 Accumulator simulation

decoder respond within the 1ns clock period, and ensure correct timing at DAC current cell inputs. The two stages of latching thus introduced (Figure 4.1) cause a pipeline delay of two clock cycles beyond that present in the accumulator.

4.4 Non-linear DAC

The Non Linear DAC can be realized as a matrix of $2^{12-2}(1024)$ cells, where the current delivered by each cell is determined by equation 3.3 in section 3.3. Because of the nature of the output required (i.e., each step size is potentially different from its neighbors) a fully segmented approach (Figure 4.8) is mandatory, and has the advantage of giving good DNL and glitch energy performance [24]. Various steps are taken in the layout in order to minimize the errors due to process gradient errors, since the size of the DAC is large as compared to a binary weighted approach. An approach used to reduce the glitch when going from the positive to the negative cycle of the sine wave is to duplicate the cells for realizing one quadrant, as illustrated in Figure 4.8. The DAC cell logic required to select the correct cell with the given input code is shown in Figure 4.7.

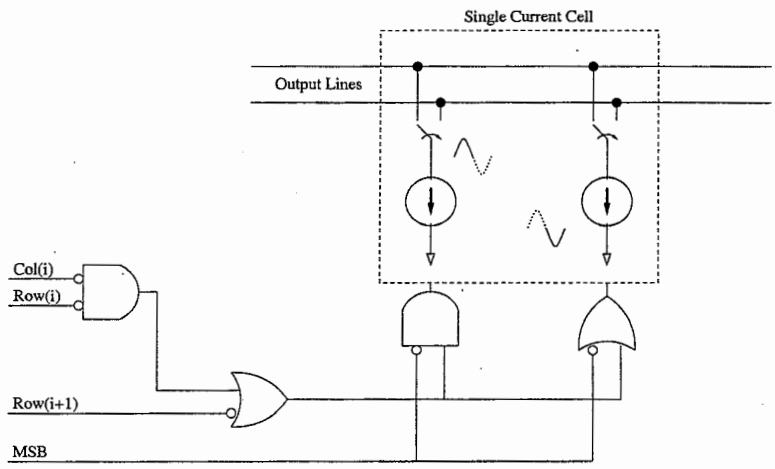

Figure 4.7 DAC cell selection logic

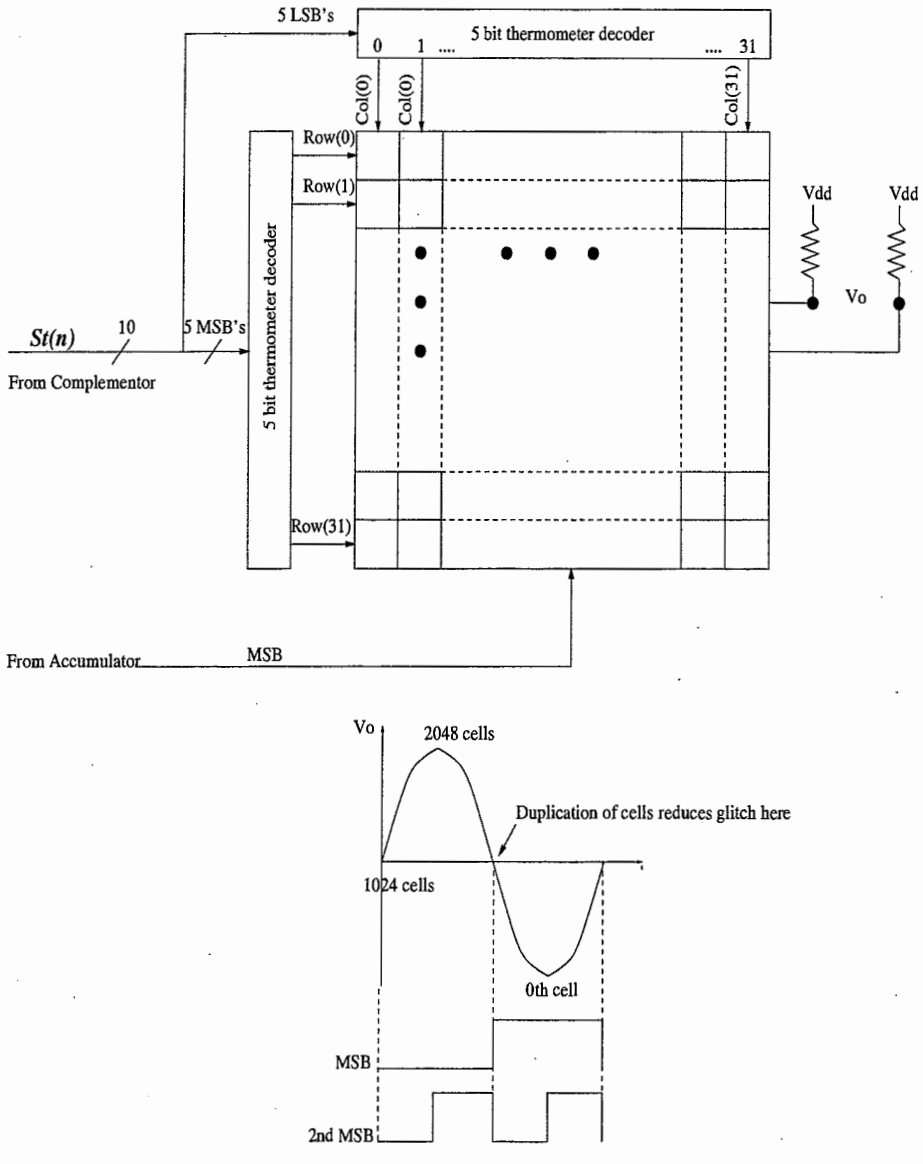

Figure 4.8 Duplication of Sine Quadrant

4.4.1 Current cell design

For a 2v pp differential voltage, the number of current cells which are conducting goes from 1 to 2048, and the unit current can be very approximately determined as 10uA per cell. A tradeoff here exists in that the response time for the DAC cell depends on the current value, and is higher for higher values of current. However, increasing the current limits the swing, along with increasing the power dissipation. An initial circuit used to simulate the unit current cell is shown in figure 4.9. This is basically a current switch, which switches the entire bias current from one branch to the other depending on the control input.

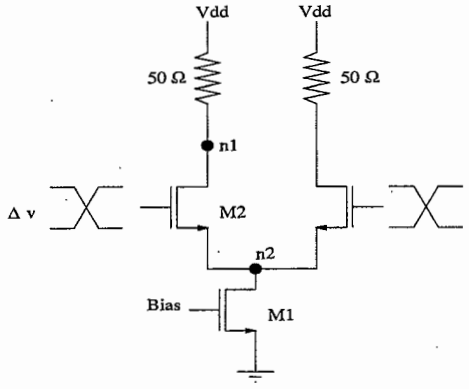

Figure 4.9 Simple DAC Cell

The dynamic performance of this structure is limited by the following factors:

- The pole on the output node caused by the parasitic capacitance of 2048 current cells
- Feed through of the control signals to the output lines

In figure 4.9, the dominant node which determine the dynamic characteristics is the output node n1. The pole associated with this node is given by the following.

$$p_1 \approx \frac{1}{2\pi R_L(C_L + C_{dtot2})} \tag{4.1}$$

where R_L represents the external 50ω load, and C_L is the load capacitance, while C_{dtot2} represents the total drain capacitance of all the current cells connected at n1, plus the pad capacitance.

The dominant pole at p_1 can be minimized by using minimum dimensions for the switching transistor. It will be seen later that a cascade transistor is necessary to prevent signal feed through, and this can be made minimum size.

The feed through of signals to the output is reduced by using two techniques:

- The coupling of the switching control signals to the output lines through the parasitic gate-drain capacitance of the switches is given as

$$\Delta V \approx \frac{nC_{gd2}}{C_L + C_{dtot2}} \Delta V_{G2} \tag{4.2}$$

  where ΔV_{G2} is the control voltage swing, C_{dtot2} is the total parasitic drain capacitance of n switching transistors being commuted simultaneously. Since the n value depends on the code, these errors are also code dependent, and cause distortion at the output. This problem is minimized by reducing ΔV_{G2} to the minimum necessary amplitude. This effectively translates into having the control signal swing by the $\sqrt{2}(V_{gs} - V_T)$ for the switching transistors. This was implemented by having two flipflops at the input of the current cell inside each cell, which operate at a lower power supply (2.2v) so that the output swing is limited.

- A cascade transistor in inserted in series with the switching transistor, and this isolates the drain of the witching transistors from the output (Figure 4.10). For a low to high transition of the control signal, while the switching transistor is forming a channel, the cascaded transistors are off, and the signal path form the drain of the switching transistor to the output node is open. For a high to low

transition, there is some coupling at the start, but the switching transistor switches off rapidly, the voltage at the source of the cascade transistor again rises, turning it off, and isolating the output node for the remaining of the transition of the control signals.

Figure 4.11 shows the output current in the two branches when three DAC cells having values $20\mu A$, $40\mu A$ and $30\mu A$ switch simultaneously. Figure 4.12 shows a similar result for 256 cells each of which have a current of $20\mu A$. It was found that the DAC output current settles to within a 1-3 percent error value within 1 nanosecond, with total drain parasitics for the DAC matrix included in the simulation setup.

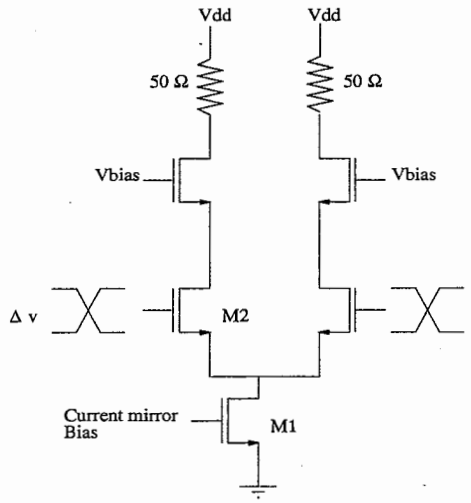

Figure 4.10 Cascade current cell

4.5 Layout considerations

Considerations at the layout stage included

- Matching considerations for analog circuits
- Reducing errors due to gradients

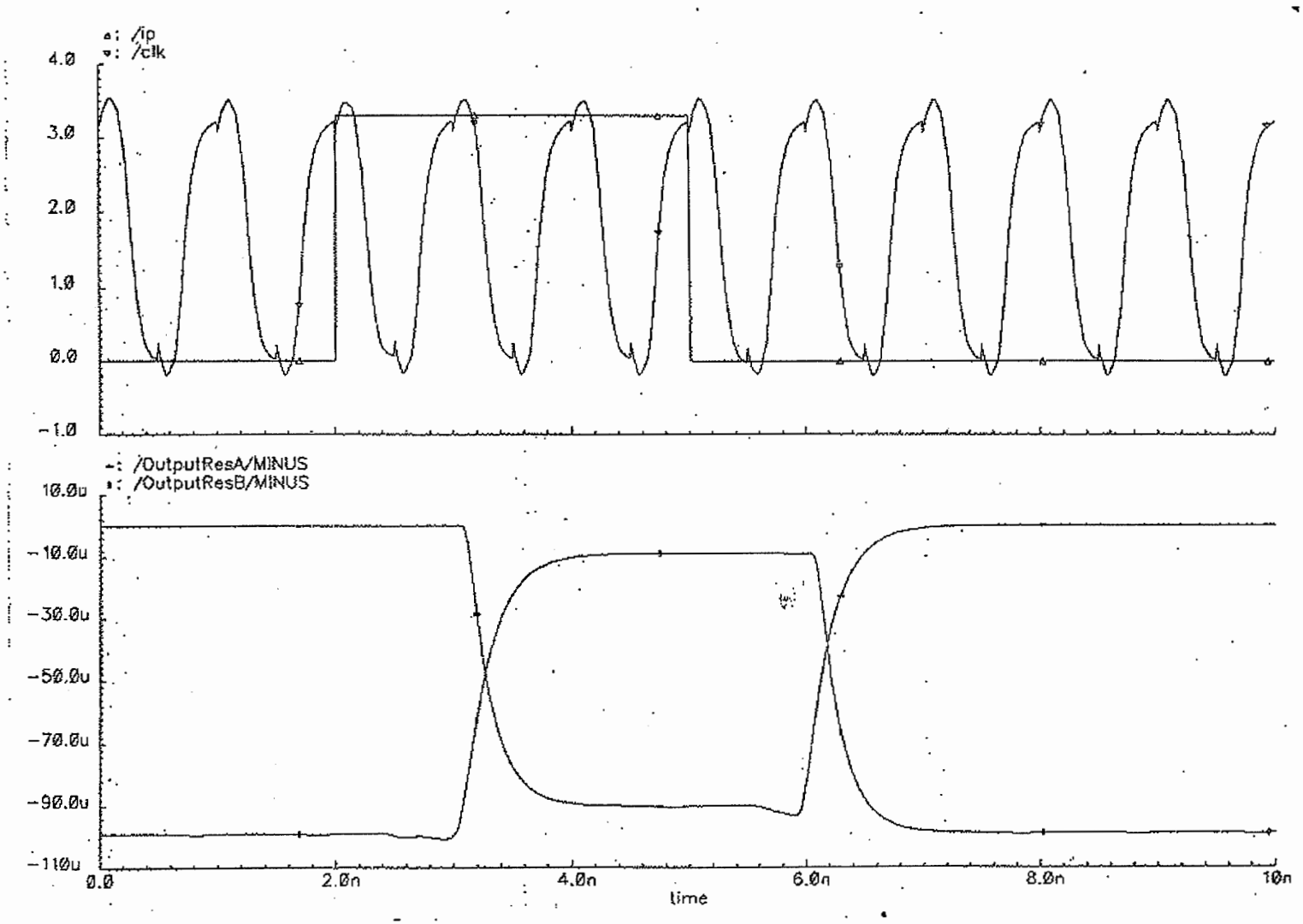

Figure 4.11 DAC cell simulation - 3 cells

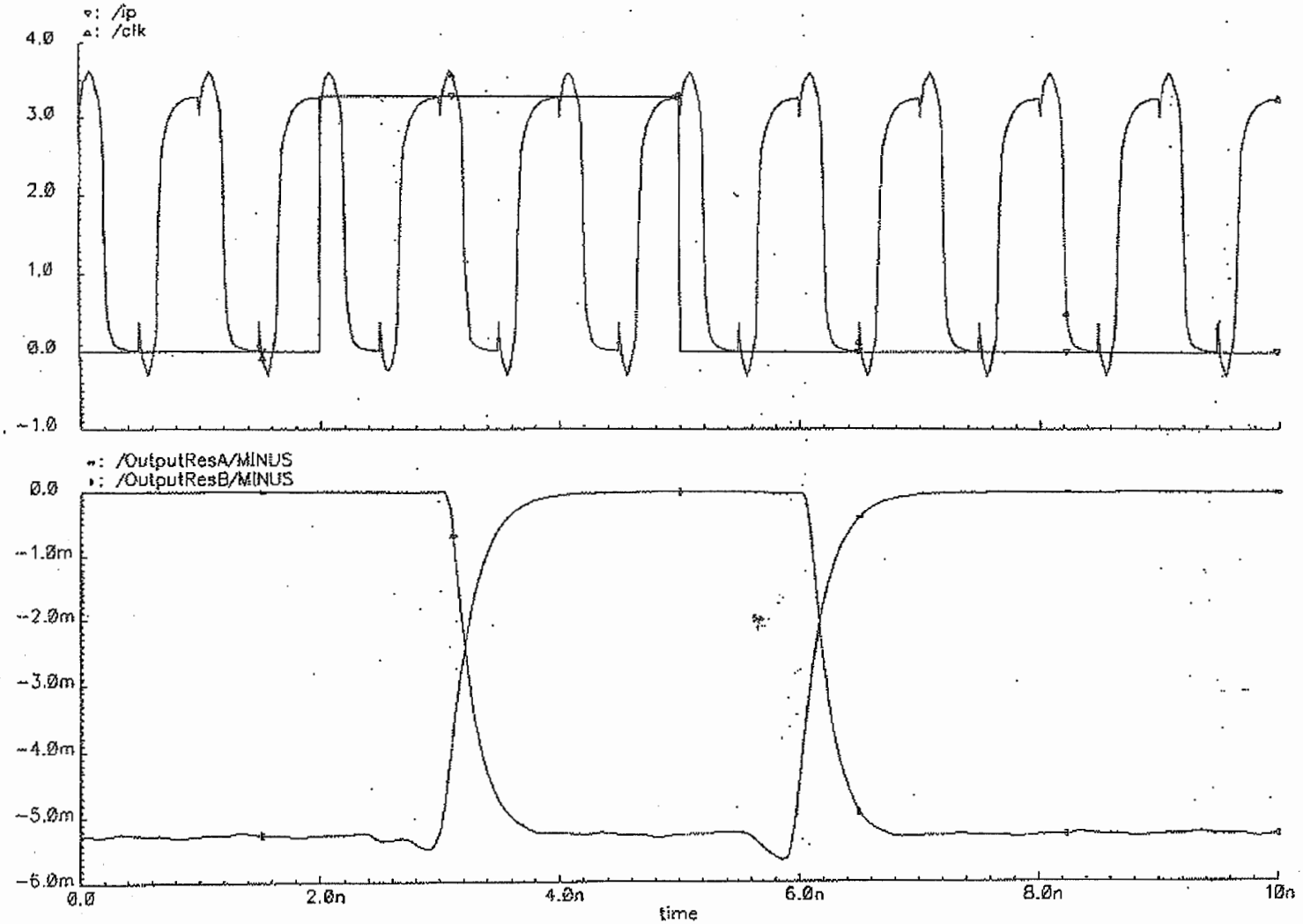

Figure 4.12 DAC cell simulation - 256 cells

- Minimizing effects due to bond wire inductances
- Clock distribution
- Power supply noise

The accumulator is about $1/8^{th}$ the size of the DAC, and hence the floor plan was mainly decided by the DAC. In order to have equal clock delays to the register inputs due to interconnect, the clock is distributed in a tree fashion, and the clock drivers are placed so that a set of registers uses a common driver, in addition to a global driver just after the input from the pad. Also, since the four lsb's of the accumulator are not actually connected to the remaining circuitry, the pipeline registers are not required, saving area.

The analog and digital power rails are taken out separately, as are analog and digital ground lines. The substrate line is also separate from all of these lines. Large decoupling MOS capacitors are distributed across the rails in order to reduce supply variations affecting the circuit operation. It was found that connecting a signal to more than one bond pad reduces the effect of bond wire inductance.

The DAC was laid out in a highly symmetrical matrix arrangement, where each cell is exactly identical to the other in layout. Since the cells need a current varying from 10μA to 40μA, each cell contains all the four unit current sources, but those that are not required are connected as MOS capacitances. Figure 4.13 illustrates this concept. A dummy transistor on either side of the active sources is also used so that each of the active sources sees a similar environment in layout.

Since the DAC occupies a large area, gradients in the layout could affect the INL performance. A technique used to reduce this is to shuffle the order of the the rows and columns. Boundary effects are avoided by surrounding the active DAC cells with a layer of dummy cells (Figure 4.14).

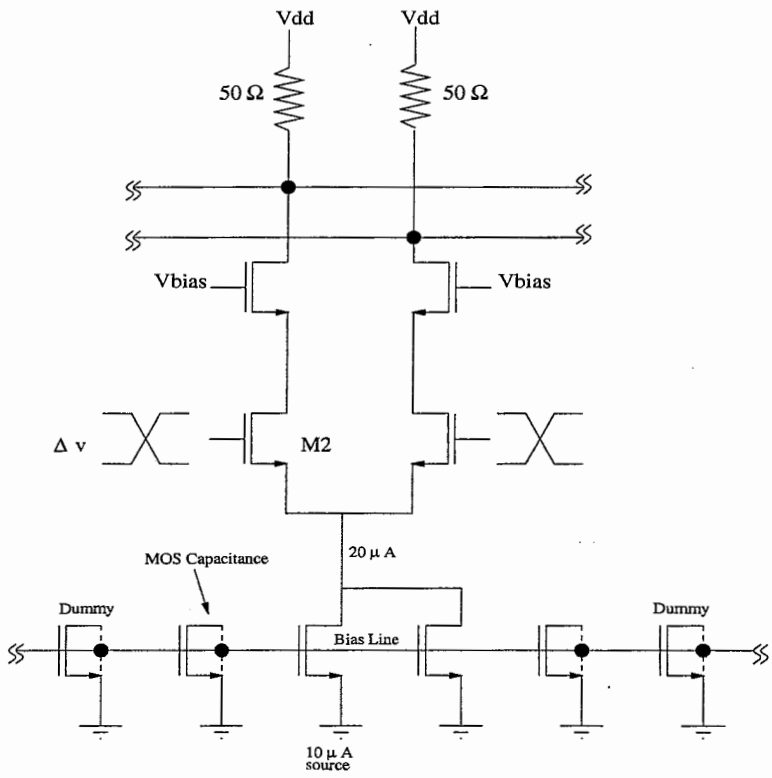

Figure 4.13 DAC cell matching

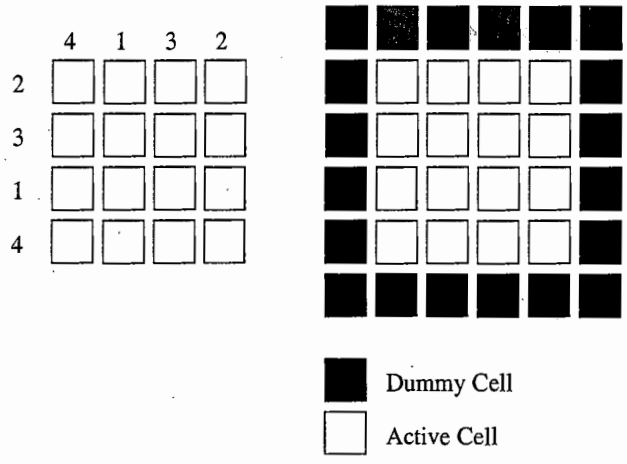

Figure 4.14 Randomization and dummies

5 CONCLUSION

In this work, a design technique for DDFS using a nonlinear DAC is studied. This technique has its main advantage in the fact that for the same bit resolution provided by an architecture using a ROM, lower power and area are required. The design technique for a non linear fully segmented current mode DAC is explained, and various circuit techniques for keeping spurious frequencies at the output as low as possible are explored. A spurious performance of 77DB at 125MHz ($1/8th$ the clock frequency) was observed in matlab simulations. Also, the estimated power dissipation of the entire chip was about 1W.

Since the problem of spurious output due to quantization errors has been extensively studied and various solutions are available to this problem, the main area of further research would be in the design of the DAC to give good accuracy and speed performance at the resolution required.

Because of power and area savings, this approach can find uses primarily in the portable wireless communication systems area. The disadvantage of using DDFS in modulation schemes however, is that amplitude modulation cannot be easily accomplished. However, there are techniques using analog mixers etc. which can be used for this purpose.

APPENDIX C AND MATLAB CODE

In this appendix, the C code used to generate the DAC cell values, and the matlab code used to perform system level simulations, is included.

C CODE

```
#include <stdio.h>
#include <math.h>
#define PIval 3.141593
#define Jval 12
#define Ival 10
#define Xval 32
#define Yval 32

void main()
{
/* this file calculates Ck values and then forms the random matrix */
/* these are the final values for the dac cells */

/* An array for the Ck values,*/
 float CK[Xval][Yval], Crandom_int[Xval][Yval],Crandom[Xval][Yval];
 int i,j,k=0;
```

```
 int x,y,zero_count=0;
 float C_accum=0;
 double sinvalue, temp;
 int tmp;
 /* This is the pseudo random sequence */
 int Corder[Xval]={8,5,7,6, 24,21,23,22 ,16,13,15,14, 4,1,3,2,
32,29,31,30, 20,17,19,18, 28,25,27,26, 12,9,11,10};
 /* This is the straight sequence
    int Corder[Xval]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
18,19,20,21,22,23,24,25,26,27,28,29,30,31,32};*/

CK[0][0]=.5*rint(2* (pow(2,Ival)-1) * (PIval*.5/pow(2,Jval-1)) ) *.5;
C_accum = CK[0][0];
k=1;

for(x=0; x<Xval; x++)
{
 for(y=0; y<Yval; y++)
 {

  if(x == 0 && y == 0) continue;

  sinvalue = fabs( sin(PIval*(k+.5)/pow(2,Jval-1)) );
  CK[y][x] = rint(2*(pow(2,Ival)-1) * sinvalue)*.5 - C_accum;
  /* count # of zero cells */
  if(CK[y][x] == 0) zero_count ++;
  C_accum += CK[y][x];
```

```
/* K value increment */
  k++;
 }
}

/* This loop exchanges the columns in the random sequence*/
for(i=0; i<Xval; i++)
 {
   for(j=0; j<Yval; j++)
      {
Crandom_int[j][i] = CK[j][Corder[i]-1];
      }
 }

/* This loop exchanges the rows in the random sequence*/
for(i=0; i<Xval; i++)
 {
   for(j=0; j<Yval; j++)
      {
Crandom[i][j]=Crandom_int[Corder[i]-1][j];
      }
 }

/******* Printout ******/

for(x=0; x<Xval; x++)
```

```
 {
  for(y=0; y<Yval; y++)
  {
   printf("%.2f ",Crandom[x][y]);
  }
 }
}
```

Matlab Code

The matlab code simulates the thermometer code decoding with the randomization, so that given a matrix of DAC cell values generated by the C program, sine waves at the output of different frequencies can be obtained, and the SFDR can be found.

```
clear;
ip=[];
ip_mat=zeros(32,32);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Read the DAC cell values from the output of C program %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fid=fopen('ck_values_new');
for loop_var = 1:1,
digit = fscanf(fid,'%f');
if(line == -1), break, end
 ip=[ip digit];

end
fclose(fid);
```

```
for i=1:32,
 for j=1:32,
  ip_mat(i,j)=ip((i-1)*32+j);
 end
end
ck_val=ip_mat;
ck_val
pause
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The Pseudo random sequence
Rand=[8 5 7 6 24 21 23 22 16 13 15 14 4 1 3 2  32 29 31
30 20 17 19 18 28 25 27 26 12 9 11 10];
%%%Rand=[1:32]; %% For straight thermometer code

%% Build the thermomter code matrix
for tval=1:32,
x=[];
  for i=1:tval,
  x=[x 1];
  end
     for j=tval+1:32,
     x=[x 0];
     end
  therm=[therm x'];
  end
therm=therm';
```

```
%%%%%%%%%%Initialize%%%%%%%%%%
c_accum=zeros(1,1024);
accum_index=1;
one_col=ones(1,32)';
zero_col=zeros(1,32)';
Final_decode=[];
Final_decode1=zeros(32,32);
Final_decode2=zeros(32,32);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for col_dec=1:32,

for row_dec=1:32,

%% Columns of ones for the columns already turned on
for i=1:(col_dec-1),
Final_decode=[Final_decode one_col];
end

%% Thermomter code value for the 'active' column
row_decode=therm(row_dec, 1:32);
Final_decode=[Final_decode row_decode'];

%% Rest of the columns are turned off
for i=1:(32-col_dec),
Final_decode=[Final_decode zero_col];
end
```

```
%% Final Decode ready here, interchange rows and columns %%
for i=1:32,
Final_decode1(1:32,i)=Final_decode(1:32,Rand(i));
end

for i=1:32,
Final_decode2(i,1:32)=Final_decode1(Rand(i),1:32);
end

%%%%%%%%%%% Form C_ACCUM %%%%%%%%
c_accum_mat=Final_decode2.*ck_val;

accum=0;
for i=1:32,
for j=1:32,
accum=accum+c_accum_mat(i,j);
end
end
c_accum(accum_index)=accum;
accum_index=accum_index+1;
%%%%%%%%%%  CLear for next run  %%%%%%%%%%%%
Final_decode=[];
Final_decode1=zeros(32,32);
Final_decode2=zeros(32,32);
row_decode=[];
end  %----------> row loop
end  %----------> col loop
```

```
%% Form the sine wave by quadrant reflection %%
z=c_accum;
z1=fliplr(z);
full=[z z1 -z -z1];
full=full./1023;

%for loop=1:4,
%tmp(loop)=full(loop*1024);
%end

%% Form more than one sine wave, for the FFT %%
tmp=full;
y=tmp;
for loop=1:50,
y=[y tmp];
end
len=length(y);

%% Take FFT
Y=fft(y,4096);
Yabs=abs(Y);
ind=find(Yabs==0);
Yabs(ind)= NaN*ones(size(ind));
Fyy=20*log10(Yabs);
f=1e9*(0:4095)/4096;
figure
```

```
plot(f,Fyy(1:4096))
```

REFERENCES

[1] J. Craninckx and M. Steyaert, "A fully integrated CMOS DCS-1800 frequency synthesizer", *IEEE Journal of Solid State Circuits*, vol. 33, pp. 2054-2065, December 1998.

[2] Seog-Jun Lee and Beomsup Kim, "A fully integrated low-noise 1-GHz frequency synthesizer design for mobile communication application", *IEEE Journal of Solid State Circuits*, vol. 32, pp. 760-5, May 1997.

[3] Thamsirianunt and Tadeusz, "CMOS VCO's for PLL frequency synthesis in GHZ digital mobile radio communications", *IEEE Journal of Solid State Circuits*, vol. 32, pp. 1511-24, Oct 1997.

[4] O'Brien P., McGrath S., Burkley C.J. , "The generation of bandwidth efficient modulation schemes using direct digital synthesis", *Personal, Indoor and Mobile Radio Communications*, proceedings, PIMRC '92., pp. 393-396

[5] Bjerede B., Lipowski J., Petranovich J., Gilbert S., " An intermediate frequency modulator using direct digital synthesis techniques for Japanese Personal Handy Phone (PHP) and Digital European Cordless Telecommunications (DECT)", *44th IEEE Vehicular Technology Conference*, vol.1, pp. 467-471 1994

[6] J. Craninckx and M. Steyaert, *Wireless CMOS Frequency Synthesizer Design*, Kluwer Academic Publishers, Boston MA,1998

[7] J. Tierney, C. Rader, and B. Gold, "A Digital Frequency Synthesizer", *IEEE transactions on Audio and Electroacoustics*, March 1971, pp. 48-57

[8] H.T. Nicholas and H. Samueli, "An analysis of the Output Spectrum of DDFS in the Prescence of Phase Accumulator Truncation", *Proceedings of the 41st Annual Frequency Control Symposium*, May 1987, pp. 495-502.

[9] Fransisco Cercas, "DDFS for Frequency Hopping Spread Spectrum Systems", Master's thesis, International School of Technology-Lisbon, 1988.

[10] H. Nicholas, H. Samueli, and B. Kim, "The optimization of DDFS performance in the prescence of Finite word Length effects," *42nd Annual Frequency Control Symposium*, pp. 357-363, 1988

[11] Jouko Vankka, "Spur Reduction Techniques in Sine Output Direct Digital Synthesis", *IEEE International Frequency Control Symposium*, 1996, pp. 951-959

[12] M. Bozic, A. E. Jones, J.G. Gardiner, "Elimination of Non-Harmonic Spurious in Direct Digital Synthesis," *IEE Colloquium on Direct Digital Frequency Synthesis*, Nov 1991, digest 172, pp. 1/1-1/6.

[13] V.S.Reindhart, "Spur Reduction Techniques in Direct Digital Synthesizers", *Proceedings of IEEE International Frequency Control Symposium*, 1993, pp. 230-241.

[14] D. Sunderland, R. Strauch, S. Wharfield, H. Peterson, C. Cole, "CMOS/SOS frequency synthesizer LSI circuit for spread spectrum communications", *IEEE J. Solid State Circuits*, vol. SC-19, pp. 497-505, Aug 1984

[15] L. Weaver and R Kerr, "High resolution phase to sine amplitude conversion," U.S. Patent 4905177, Feb 27, 1990.

[16] A. Madisetti, A. Kwentus, A. Wilson, Jr., "A Sine/Cosine Direct Digital Frequency Synthesizer using an angle rotation algorithm," in *IEEE ISSCC Dig. Tech Papers,* 1995 pp. 262-263

[17] Siamak Mortezapour, Edward K. F. Lee, "Design of Low-Power ROM-less Direct Digital Frequency Synthesizer Using Nonlinear Digital-to-Analog Converter", *IEEE Journal of solid state circuits*, Vol 34, Oct 1999, pp. 1350-1359.

[18] David A. Johns, Ken Martin, *Analog Integrated Circuit Design*, John Wiley and Sons Inc., New York NY, 1997.

[19] Jouko Vankka, Mikko Waltari, Marko Kosunen, and Kari A. L. Halonen, "A Direct Digital Synthesizer with an On-Chip D/A converter", *IEEE Journal of solid state circuits*, Vol. 33, February 1998, pp. 218-227.

[20] E. Abu-Shama, M. Bayoumi, "A New cell for Low Power Adders", *IEEE International Symposium on Circuits and Systems*, part 4 (of 4) May 12-15 1996, pp. 49-52.

[21] Nan Zhuang, Haomin Wu, "A New Design of the CMOS Full Adder", *IEEE Journal of Solid State Circuits*, pp. 840-844, May 1994.

[22] Chang Byungshoo, Joonbae Park, Wonchan Kim, "A 1.2 CMOS Dual-Modulus Prescaler Using New Dynamic D-Type Flip Flops", *IEEE Journal of Solid State Circuits*, v.31, May '96 pp. 749-52.

[23] Neil H. E. Weste, Kamran Eshraghian, "*Principles of CMOS VLSI design- A Systems Perspective*", second edition, John Wiley and Sons Inc., New York NY, 1993.

[24] Jose Bastos, Augusto M. Marques, Michel Steyaert, Willy Sansen, "A 12-Bit Intrinsic Accuracy High Speed CMOS DAC", *IEEE Journal of Solid State Circuits*, Vol 33, N0. 12, Dec 1999, pp. 1959-1969.