OPTIMIZING THE PERFORMANCE OF DIRECT DIGITAL FREQUENCY SYNTHESIZERS FOR LOW-POWER WIRELESS COMMUNICATION SYSTEMS

By

DAVID JAMES BETOWSKI

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER ENGINERING

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

DECEMBER 2004

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of DAVID JAMES BETOWSKI find it satisfactory and recommend that it be accepted.

____
Chair

____

____

ACKNOWLEDGMENT

The Author and Committee wish to extend a special thanks to those students and faculty members of the Washington State University School of Electrical Engineering and Computer Science who contributed in some way to this research:

Daniel G. Dwyer

Pao-Szu Wu

Suryanarayana B. Tatapudi

Jeffrey D. Lowe

Parag Upadhyaya

Zhihe “Bill” Zhou

George S. La Rue, Ph. D.

Deuk H. Heo, Ph. D.

Portions of this research were sponsored by the U.S. Air Force Research Laboratory (AFRL), the Center for the Design of Analog and Digital Integrated Circuits (CDADIC)*, and the Center for Neural Inspired Nano-Architectures (CNINA)†.

* www.eecs.wsu.edu/cdadic

† www.cnina.org

OPTIMIZING THE PERFORMANCE OF DIRECT DIGITAL FREQUENCY SYNTHESIZERS FOR LOW-POWER WIRELESS COMMUNICATION SYSTEMS

Abstract

by David James Betowski, M.S.
Washington State University
December 2004

Chair: Valeriu Beiu

A direct digital frequency synthesizer (DDFS) generates a highly accurate sine wave using feed-forward digital signal processing, overcoming many of the problems incurred with the traditional analog closed-loop frequency synthesizer, the Phase-Locked Loop (PLL). The most popular application of a DDFS is generating the variable carrier frequency required in portable wireless communication systems, which require both high accuracy and very low power consumption. A three-level abstraction analysis and design approach is presented.

The system level analysis focuses on selecting an appropriate phase-to-sine wave approximation circuit, which has traditionally been implemented with a large-sized ROM lookup table, resulting in a moderately high SFDR and very high power consumption. Recent solutions reduce the ROM size by using a segmented linear approximation to compute the sine wave in real-time. The solutions produce a very high SFDR (using up to 64 segments), but the circuit complexity remains high. A novel segmented parabolic approximation is introduced, and using only 16 segments, yields an 84 dBc SFDR. The circuit complexity is lower than other methods having a similar SFDR.

Both the new approximation and the DDFS, in general, require several high-speed arithmetic components, including 8 – 32-bit adders and a Wallace-tree based multiply-accumulate circuit. Pipelining the DDFS system may be necessary if high-speed operation is desired. A mathematical model is presented at the component level to determine the optimal number of stages with respect to the speed and power requirements.

Descending to the circuit level, four logic gate styles (complementary, pseudo-NMOS, dynamic, and differential cascode voltage switch) are constructed and simulated in three different CMOS processes to analyze the speed and power consequences of process scaling. Presented is a fresh investigation of significantly reducing the power-delay product by lowering the supply voltage to sub-threshold levels. It is shown that a 100-fold energy decrease does not require the added non-recurring engineering costs of scaling down to a smaller CMOS process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Dedication

Being the very first graduate of the new Master of Science in Computer Engineering degree at Washington State University, I dedicate this thesis to the following Computer Engineering faculty members: Dr. Valeriu Beiu, Dr. Jabulani Nyathi, and Dr. José G. Delgado-Frias. I express my sincere gratitude to all three professors for their support of my research, and extend a very special thanks to Dr. Delgado for creating this new degree and supporting my efforts to be the first graduate. In addition, I cannot even begin to extend enough thanks to the following persons for all of their support during my graduate career at WSU: Mitchell J. Myjak, Jennifer L. Streicher, and Mrs. Marynella and Dr. Cesario Zamora.

CHAPTER 1

INTRODUCTION

A frequency synthesizer is a quintessential component of any communication system. The fundamental component of an analog or digital communication system is a modulator, which multiplies a base signal containing the data by a carrier frequency that is generated by the frequency synthesizer. While many venerable communication systems, such as AM/FM and VHF/UHF broadcasting, utilize a carrier with a fixed amplitude and frequency, several modern systems utilize spread-spectrum modulation, where the frequency of the carrier varies several times per second [1]. Some well-known spread-spectrum based systems include code division and time-division multiple access (CDMA/TDMA) for cellular communications, direct-sequence spread-spectrum (DSSS) used in IEEE 802.11b wireless Ethernet [2], and frequency-hopping spread spectrum (FHSS) used in Bluetooth® [3]. The traditional method of implementing a frequency synthesizer is to use an analog phase-locked loop (PLL), a closed loop system that generates an output frequency based on a fixed reference frequency and a frequency division factor [4]. Because of the inherent closed-loop nature, PLLs have a low frequency switching speed, high phase noise, and stability issues, unacceptable to future generations of high-frequency spread spectrum technologies.

An alternative method of frequency generation is direct frequency synthesis (DFS), which takes a reference frequency and achieves the target frequency through feed-forward analog and digital signal processing techniques, such as filtering and mixing. An improvement of the DFS, and promising alternative to PLLs, is the mostly-digital Direct Digital Frequency

Synthesizer (DDFS) [5][6]. The basic block diagram of a DDFS is shown in Figure 1.1. The essential components are a phase accumulator (PA) with resolution of N bits [7], a phase-to-sine amplitude converter having a resolution of Q (less than N) [8], an M-bit digital-to-analog converter, and normally a low pass filter (not shown). Typically, an M-bit register is placed right before the DAC to pipeline the system and improve the clock frequency. The input to the system is a digital frequency control word, *FCW*, of length N, leading to an output frequency of:

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

According to 1.1, f_{out} ranges from 0 Hz to a maximum of $f_{clk}/4$ Hz. Within that range, f_{out} increases in increments of the frequency resolution, $f_{clk}/2^N$. The frequency switching speed is the delay between a change in FCW and the resultant change in f_{out}. In a non-pipelined system, $1/f_{clk}$ is the switching speed. For a pipelined system, the switching speed is the total system latency (number of pipeline stages times the clock period).

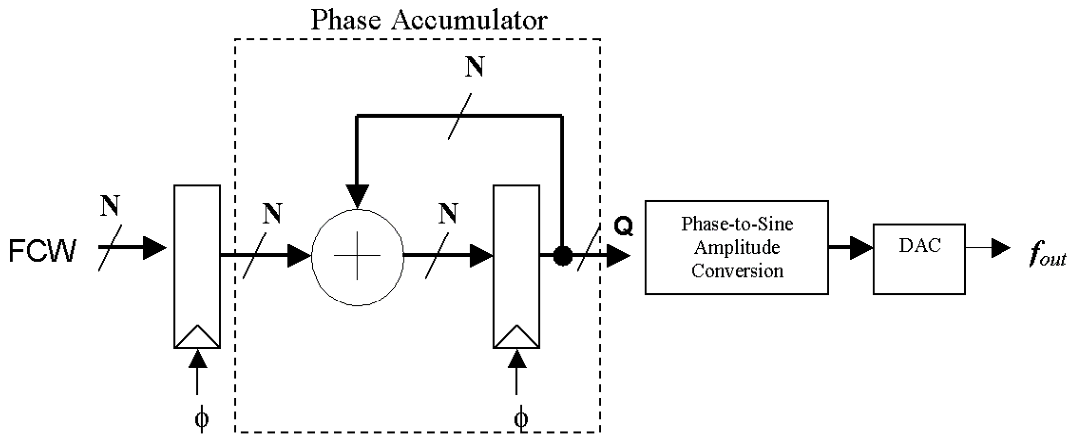


Figure 1.1: Block diagram of a DDFS.

1.1 Theory of Operation

The phase accumulator is a variable-increment N-bit counter. As shown in Figure 1.2, the output increases by *FCW* for every successive clock pulse, $\phi = 1/f_{clk}$. When *Phase* $> 2^N - 1$, the counter overflows and resets to 0. *T* is the duration of one period of the sine curve. For larger values of *FCW* the phase increases at a faster rate, hence a higher frequency wave will be generated, as shown in Figure 1.3.

The phase-to-sine amplitude conversion circuit reads the current value of the phase and outputs the corresponding amplitude for that particular point. The DAC and LPF are optional and used only if an analog output wave is desired.

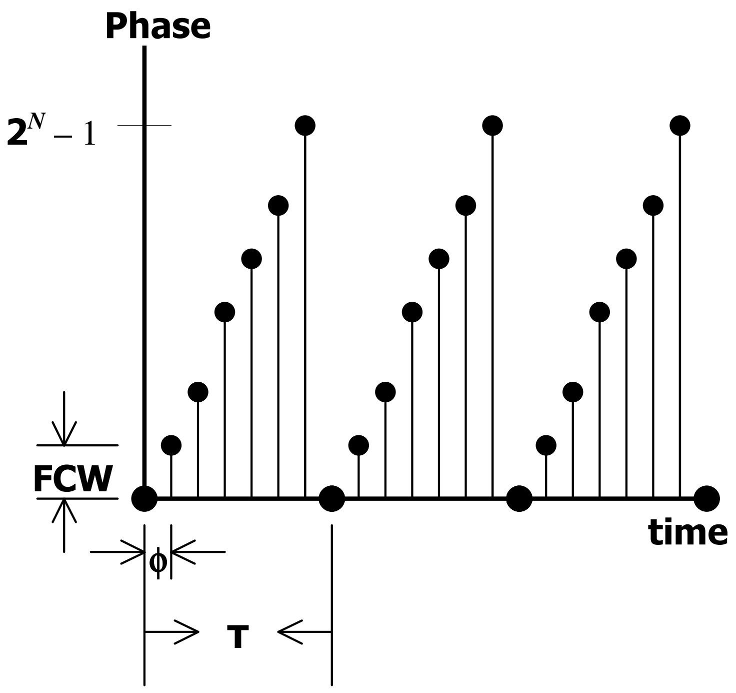


Figure 1.2: Output of phase accumulator.

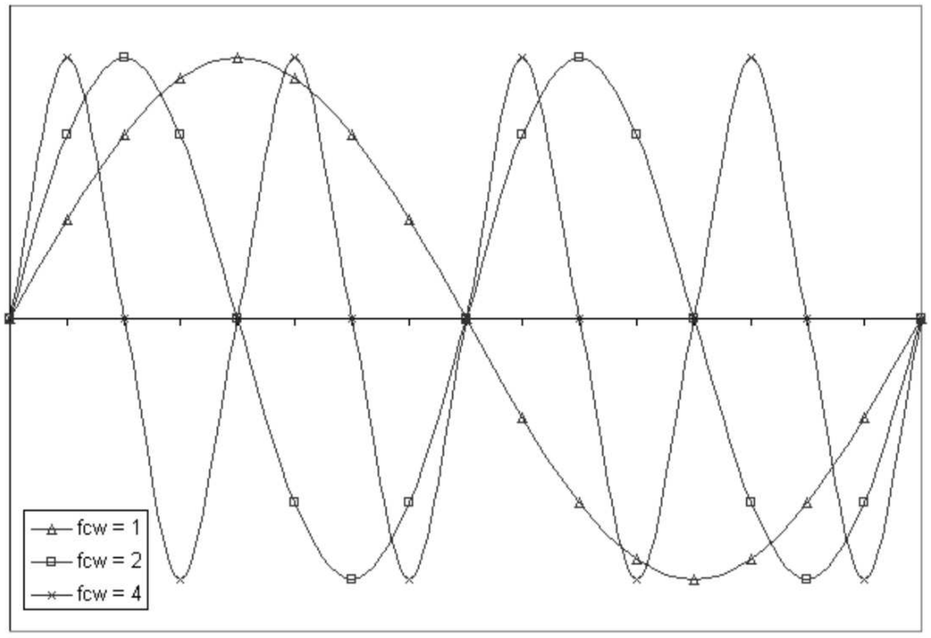


Figure 1.3: Effect of FCW values on output sine wave.

1.2 Design Challenges

There exist three primary performance metrics to characterize a DDFS

1. Accuracy
2. Power Consumption
3. Clock Frequency

The accuracy is measured in terms of both the spurious free dynamic range (SFDR) and the maximum approximation error. The maximum approximation error, MAE, is a time-domain metric, and is the difference between an ideal sine wave and the actual DDFS output wave. A smaller MAE is desired. The SFDR is a frequency-domain metric and is the difference between the magnitude of the largest fundamental and the magnitude of the spurious signal (or second largest fundamental). The SFDR (in dBc) is expressed as:

$$\mathrm{SFDR} = 20 \log_{10}(A_\mathrm{d}/A_\mathrm{s}) \tag{1.2}$$

where A_d is the amplitude of the first fundamental and A_s is the amplitude of the second fundamental. A high value of the SFDR is desired. Increasing the precision of N, Q, and M improve both the MAE and SFDR.

The clock frequency, f_{clk}, is the inverse of the maximum propagation delay through the DDFS, determining the maximum attainable output frequency. When a pipelined system is used, f_{clk} is the maximum propagation delay between two pipeline registers.

The power consumption (in Watts) is the total current that flows between the high and low potential voltage sources times the voltage difference between those two sources. While a low wattage is desired, the power consumption only provides a measurement of the total power used at one single instant. The proper measurement of the total power consumption over an interval is energy, also known as the power-delay product (PDP), and is calculated (in Joules) as:

$$PDP = \text{Total Power Consumption} \times (1/f_{clk}) \tag{1.3}$$

The PDP determines the power supply requirements (i.e. battery life) and the total heat dissipation of the device.

Ideally, a DDFS would achieve optimal values for all three metrics, but realistically only a maximum of two metrics may excel. To increase the accuracy, higher precisions of N, Q, and M are required, increasing the hardware complexity of the phase accumulator, the phase-to-sine amplitude conversion circuit, and the DAC, resulting in increased power consumption, and a slower clock frequency. Using high-speed optimization techniques, such as pipelining [7], a high clock frequency is attainable while maintaining high accuracy, but at a cost of higher power consumption due to the increased circuit complexity from implementing a pipelined system. Conversely, if low-power design techniques are utilized for a high accuracy system, then an optimal speed may not be achieved. Low-power and high-frequency operation are possible, but require low-precision arithmetic components, sacrificing the accuracy.

Because only two optimal characteristic metrics may be achieved, it becomes extremely important to fully understand the requirements of the target application for the DDFS. At the time or printing, the most popular applications of a DDFS are expected to be portable wireless and satellite communications, specifically cellular phones and portable wireless internet-enabled devices, which require a DDFS with high accuracy and low power consumption.

High accuracy is desired to achieve the most efficient use of allocated bandwidth in a finite and crowded wireless spectrum. When transmitting or receiving, each device occupies one or more channels within the allocated spectrum; the frequency-domain location of that channel is defined by its carrier, or center frequency, Ω, and bandwidth, W, as shown in Figure 1.4. Depending on the modulation/multiplexing technologies utilized, each portable device either has distinct ownership of a channel for the entire duration of the transmit/receive session, or changes channels several times per second in a TX/RX session [1]. To allow the maximum number of users, more channels must be fit into the allocated spectrum, further complicated by the requirement for increased channel bandwidth due to increased data transmission rate requirements. As the channels are spaced closer together, the potential for overlapping increases dramatically, which may result in interference and data errors. The error tolerance for the center frequency becomes lower, requiring a highly accurate synthesis of the carrier frequency.

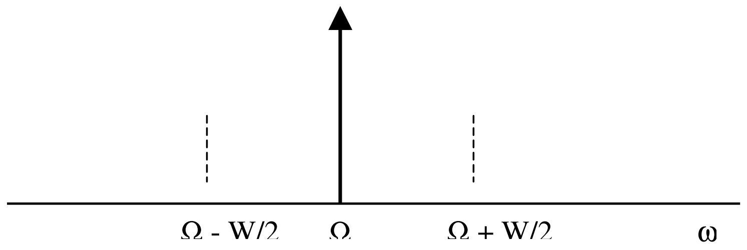


Figure 1.4: Frequency-domain representation of a channel.

Low power consumption is a highly important requirement, simply due to the inherent portable nature of these next-generation wireless devices. Consumers continue to demand cellular phones and wireless internet devices with a smaller footprint, yet with an increased duration of operation (aka “talk time”). Smaller footprints demand smaller batteries with reduced energy capacity. Recent developments in space satellite technology are yielding nano-satellites, which must offer the same high-speed data transmission rates as their larger counterparts, but have much smaller power supplies [9][10]. Batteries in portable devices may even be eliminated with micro-electrical-mechanical systems, MEMS, that generate microwatts of power from waste heat or vibrations. With reduced-capacity power supplies in the future generations of wireless devices, increased duration of operation is only possible if the electronics consume less power than today's devices. Furthermore, as more transistors are accommodated into a single die/package, heat dissipation increases. Unless consumers are willing to accept large heat sinks and/or noisy cooling fans on their portable electronics, low-power operation is mandatory.

Another explosively growing target application suited for a DDFS is remote sensing. Current embedded network sensors are based upon a low-power 4 to 8-bit microcontroller/microprocessor with networking capability, and are intended to be placed in many electronic and non-electronic devices to gather data and transmit it to a central or distributed computing system for analysis [11][12]. The majority of these sensors cannot be placed in a fixed location, and therefore, require some type of wireless data link to the central system. For instance, "smart dust" sensors placed in the air, water, and soil provide precise real-time environmental monitoring. The somewhat controversial radio frequency identification tags, RFID [13], are becoming immensely popular in product warehouses and retail for improved product tracking and inventory management, but were also used most recently at the 2004 Athens Olympic Games for precise timekeeping in the track-and-field events [14]. All of these devices require highly accurate wireless data transmission, although slow data transmission rates are tolerable. Ultra-low power operation is a definite requirement as these devices operate from a very small battery or even lack a battery. For instance, power for RFID tags is generated from induction [15], created by the magnetic field emitted by the tag scanner.

Design Objective

Based on the aforementioned applications for a DDFS, it is obvious that the two major objectives in designing a DDFS to be used for today's popular applications are high accuracy and low power consumption. With the explosive proliferation of ultra-low power sensors, power consumption is expected to be the most important factor. Since the design of a DDFS is very complex, the DDFS must be analyzed and optimized at three levels of abstraction: system, component, and circuit. At the system level, the precisions of the phase accumulator and approximation circuits are determined, as well as the method for approximating the sine wave, and the high-level interconnection topology for the approximation circuit. Analysis at the component level involves selecting the proper architectures of the arithmetic building blocks, and determining if pipelining is appropriate to the DDFS system. At the circuit level, the logic gates are constructed and simulated at the schematic/layout levels. The optimal accuracy and power consumption are obtained by implementing the optimizations from all three levels.

An actual DDFS implementation will not be constructed, as utilizing all of the mentioned optimizations is significantly expensive. If the DDFS is implemented on a full-custom IC, all three abstraction levels may be optimized. However, if the DDFS is implemented on an FPGA or reconfigurable device, optimizations may only take place at the system and component levels. It should be noted that the system and circuit level optimizations presented are not limited to a DDFS system, and may be used in any type of digital IC if its application requirements are similar to those of the DDFS.

CHAPTER 2

THE SYSTEM LEVEL

Analyzing the DDFS at the system level, the required number and precision of the components is determined such that bit widths are minimal but allow the desired frequency resolution and SFDR. At this high of a level, there exists no single number to measure the power consumption or speed, but these metrics are relative to the number of required arithmetic and logic components and their precision, and the size of the ROM (in bits), if required. Fewer and less precise components and a smaller ROM indicate lower power consumption and potentially higher speed. The SFDR and MAE will be determined using Matlab® simulations.

Phase Accumulator

When accuracy is a top-priority, the precision of the phase accumulator is easily determined by solving for N based on the desired frequency resolution.

$$N = \log_{10}(f_{clk} \,/\, resolution) \,/\, \log_{10}(2) \tag{2.1}$$

In most circumstances, however, f_{clk} is not known until testing of the physical DDFS IC. It is imperative that N remains larger than Q, so N is usually set at 16, 24, or 32-bits. 32-bit resolution is most desired for modern applications.

2.1 Phase-to-Sine Amplitude Conversion

The accuracy and power consumption are directly influenced by the method of phase-to-sine amplitude conversion, as these circuits tend to occupy the majority of on-chip area. The output precision M should be no greater than the precision of the DAC, as any extra bits will simply be discarded. Traditionally, the approximation circuit has been implemented with a full-size ROM lookup table [5], with the size of the ROM defined as

$$\text{ROM Size (bits)} = M \times 2^{Q} \tag{2.2}$$

where M is the output precision and consequently the width of the ROM data bus. Q, the input precision and width of the address bus, is a function of the target SFDR, such that [16]:

$$\text{SFDR} \approx 6.02Q - 3.92 \tag{2.3}$$

For example, to achieve a moderate SFDR of 60 dBc, $Q = 11$, and if $M = 12$, the lookup table size is 24,756 bits. Assuming that four transistors are required to realize each bit, then over 99,000 transistors are required!

One effective method of reducing the ROM size is to exploit the quarter-wave symmetry properties of the sine curve [17]. As shown in Figure 2.1, the approximation circuits need only generate one quarter of the sine wave, and MSB1 and MSB2 are used as sign bits to mirror the sine curve horizontally and vertically, respectively. This method produces a four-fold decrease in the ROM size.

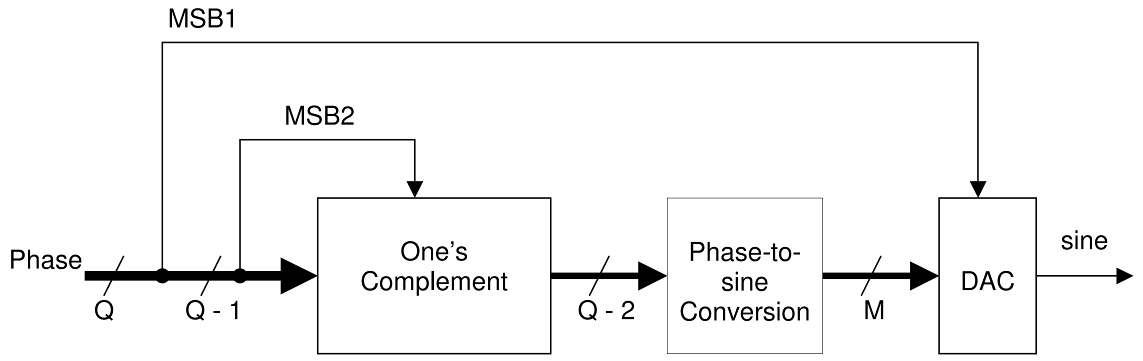


Figure 2.1: Block diagram of phase-to-sine approximation utilizing quarter-wave symmetry.

Table 2.1 presents a few ROM-based solutions that use differing techniques to reduce the ROM size. Of particular interest is the first-order parabolic approximation [18]. This method generates a first-order parabola that fits the sine curve using the circuit shown in Figure 2.2. The circuit uses a small error-correction ROM table and a multiplier and adder. The reduced circuit complexity, though, sacrifices the accuracy, as the SFDR is 28.7 dBc, relatively low for today's applications.

Table 2.1: Comparison of several ROM-based sine approximations.

Approximation Method	Year	MAE	SFDR (dBc)
Sine-phase difference [19]	1988	0.21125	19.06
Double trigonometric [20]	1998	0.11723	27.91
First-order parabolic [18]	2000	0.05600	28.68
Piece-wise 3-segment [21]	2001	0.05500	37.49
Meitzler/Millard [22]	2003	N/A	83

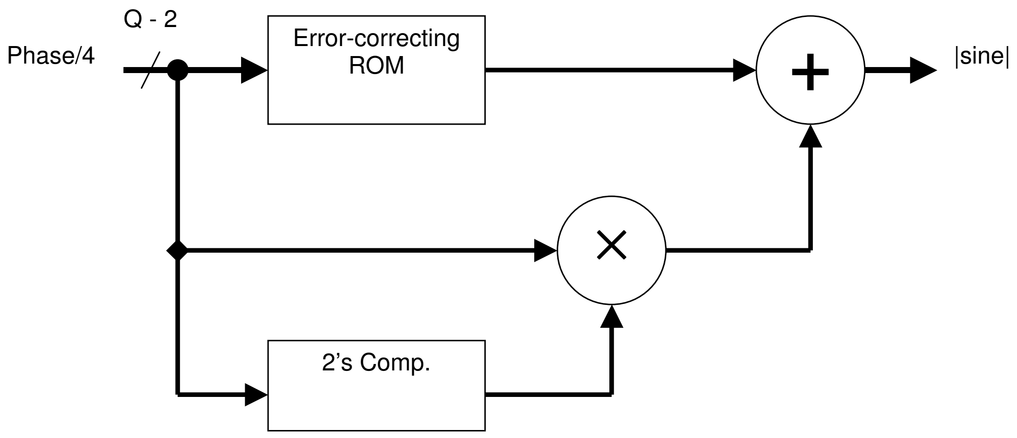


Figure 2.2: Block diagram of 1st-order parabolic approximation circuit.

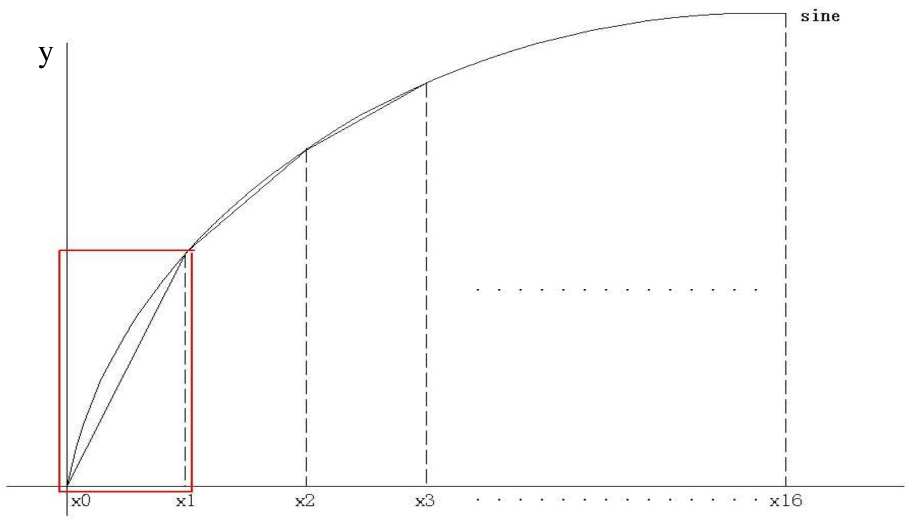


Figure 2.3: Approximating a sine curve with linear segments.

Several recent approximation methods have significantly reduced the ROM size while providing highly accurate results. These solutions use segmented linear interpolation [23], as illustrated in Figure 2.3. The *x*-axis is divided into *S* equal segments (x1 – x0, … x16 – x15), and a trendline in the form of $m_i x + c_i$ is used to 'best fit' the sine curve for each segment. Each segment, denoted by the subscript *i*, has its own unique slope, m_i, and y-intercept, c_i.

Based on these characteristics, the hardware implementation requires two ROM-lookup tables to store m_i and c_i, a multiplier to realize m_ix, and an adder. Such a circuit, however, is no less complex than any of the previous ROM-based solutions, therefore, special methods have been taken to reduce the complexity. First, because each segment only requires one value of c_i, the ROM may be implemented with an S-to-1 multiplexer (of up to M-bits wide), where the individual data pins are tied to logic 1 or 0 to realize c_i; the control bus requires $\log_2 S$ bits. Secondly, m is constrained to the interval of [-1, 1] and, may be approximated as the sum of P inverse powers-of-two (where $P = \log_2 S$) as shown in 2.4, and therefore, m_ix may be expressed by 2.5.

$$m_i = \sum_{j=0}^{P-1} m_{ij},\ m_{ij} \in \{-2^0, -2^{-1}, -2^{-2}, \ldots, 0, \ldots, 2^{-2}, 2^{-1}, 2^0\} \tag{2.4}$$

$$m_i x = \sum_{j=0}^{P-1} x m_{ij} \tag{2.5}$$

Since x is multiplied by a single power-of-two, the operation is implemented by shifting x to the right by $-\log_2 |m_{ij}|$ places, and inverting the result if m_{ij} is negative. Therefore, m_ix may be realized by adding S shifted operands, consistent with the theory of binary multiplication. A hardware block diagram of the segmented linear approximation is shown in Figure 2.4. Physically realizing m_ix requires P multiplexers with S inputs of D-bit width, where D is usually between 10 – 16 bits. For each input, x is shifted and inverted accordingly, as denoted by >>. x is actually known as the offset angle, $(x - x_i)$, and is the lower $Q - 2 - P$ bits of the phase output. The upper P bits are used as the multiplexer control bus, and select the proper c_i and m_j for each segment. The adder to combine the slopes and y-intercepts requires $P + 1$ inputs and has an output precision of M. The only ROM used in this approximation is the c multiplexer, and ranges in size from 224 – 960 bits.

The reported SFDR ranges from 48 dBc (for $S = 8$), to 72 dBc (for $S = 16$) [24]. For increasing the SFDR and decreasing the MAE, the number of segments has to be increased. Such an approach was recently detailed in [25], with the SFDR being improved to 84.2 dBc (for $S = 32$), and 96.2 dBc (for $S = 64$). Obviously, these segmented linear approximations require more and larger multiplexers as S increases, and the complexity of the multiple-input adder approaches that of a multiplier.

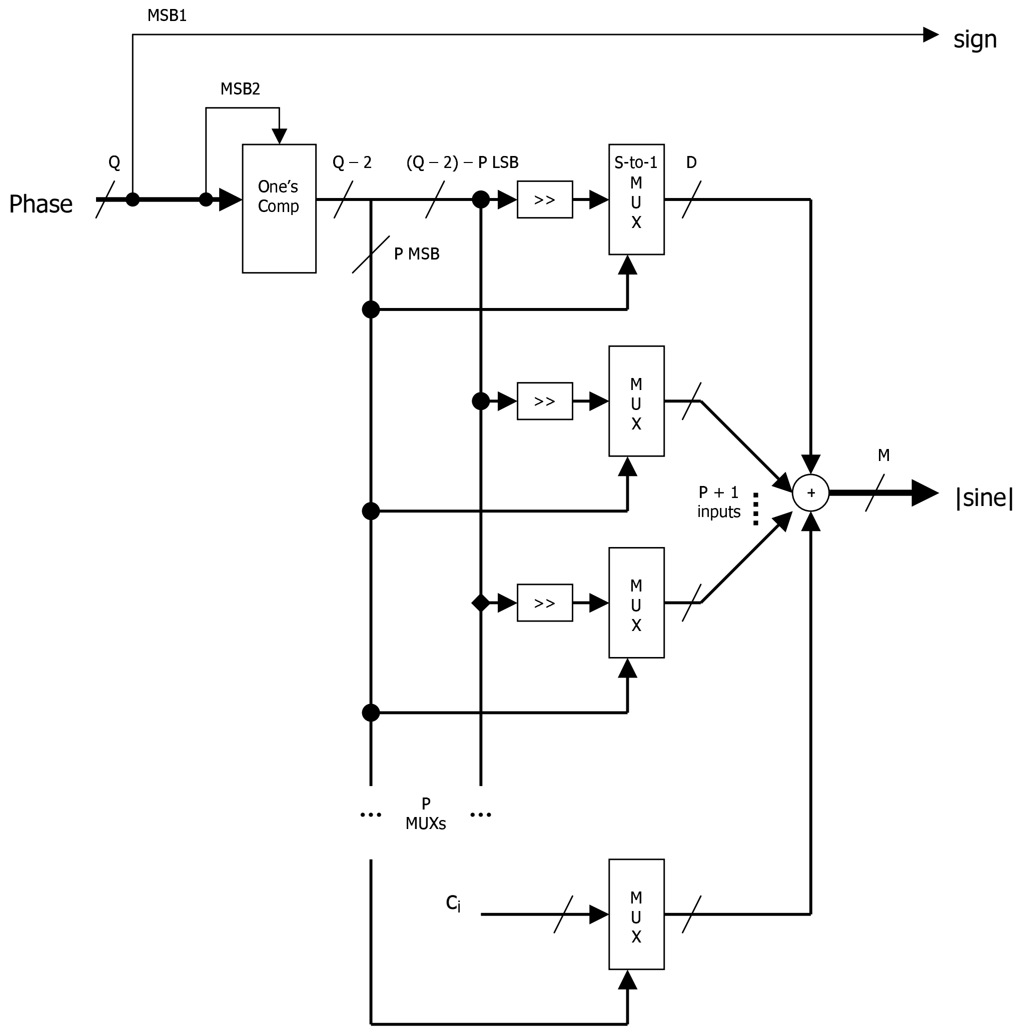


Figure 2.4: Hardware implementation of a segmented linear approximation circuit.

2.2 An Improved Parabolic Sine Approximation

The segmented linear approximation provides a high SFDR, but still has a relatively high hardware complexity. While attempting to eliminate the use of a multiplier, the structure of the multiple-input adder is very similar to a multiplier. Furthermore, the multiplexers occupy significant area, and will continue to consume significant area as the segment count increases.

The previously mentioned first-order parabolic is revisited. The SFDR is low since the parabola must approximate the entire quarter-wave. The segmented linear approximation has a high SFDR since a best-fit trendline is computed for each segment, but requires many segments for an accurate sine approximation. The positive aspects of both approximation methods can be combined into an improved segmented parabolic approximation [26]. The x-axis is still divided into S segments, and opposed to a trendline, a parabolic approximation in the general form of:

$$-x^2 + m_i x + c_i \tag{2.6}$$

is used for each segment. Because a parabola inherently follows a sine curve better than a straight line, fewer segments are required. A methodical procedure, optimized for lowering the Maximum Absolute Error (MAE) has been followed, and is applicable for arbitrary segment counts and arbitrary precision. This procedure involves simulating and observing a segmented linear approximation and determining how to reduce the approximation errors.

Before this process may begin, three design parameters must first be known:

- The input precision, $Q - 2$, which should use the smallest reasonable value. $Q = 12$ bits has been chosen for the approximation circuit presented here.
- The desired number of segments, S, which is a power of two. In an effort to minimize hardware complexity, the solution presented uses 16 segments.

- M, the final output precision of the phase-to-sine converter, not including the sign bit. The presented solution assumes that a 12-bit binary DAC with built-in sign inversion is to be used.

Once the precisions have been obtained, a reference wave with an amplitude precision of M is constructed using Matlab. The x-axis is divided into the S equal segments, and then m_i and c_i are determined for a $m_i x + c_i$ trendline such that the endpoints lie on the reference wave. The result should be similar to Figure 2.5, which uses only a two-segment example for clarity.

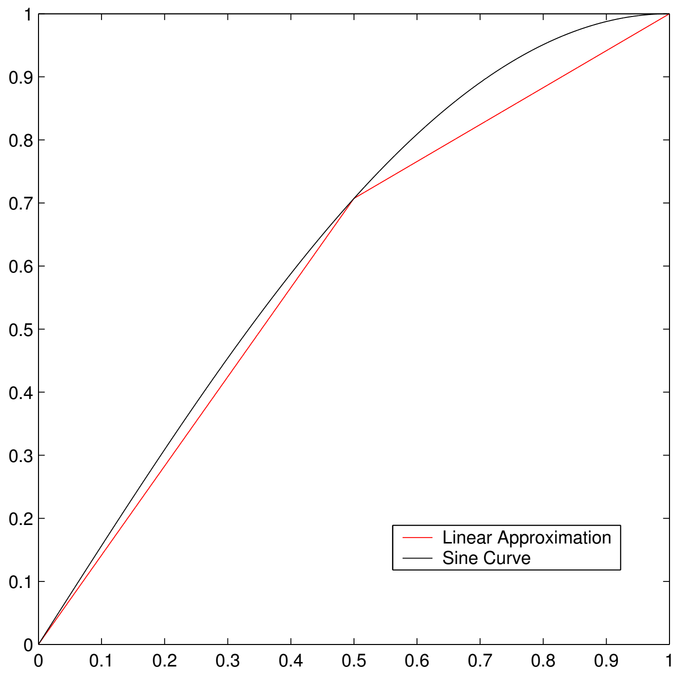


Figure 2.5: A 2-segment linear approximation.

Calculating the difference between the reference wave and the 2-segment linear approximation, it is revealed that the MAE is roughly parabolic in nature, as can be seen in Figure 2.6. This MAE is best approximated with a parabolic correction factor (shown in red on Figure 2.6):

$$-(x - \Delta_i)^2/2^{k\_i} \tag{2.7}$$

The parabola is centered in the given segment using Δ_i, which has a magnitude of:

$$\Delta_i = 2^{(px - 1)}, \\ \text{where } p_x = Q - 2 - \log_2 S \tag{2.8}$$

The division by 2^{k_i} is used to shift the parabola to obtain the best possible fit with the MAE.

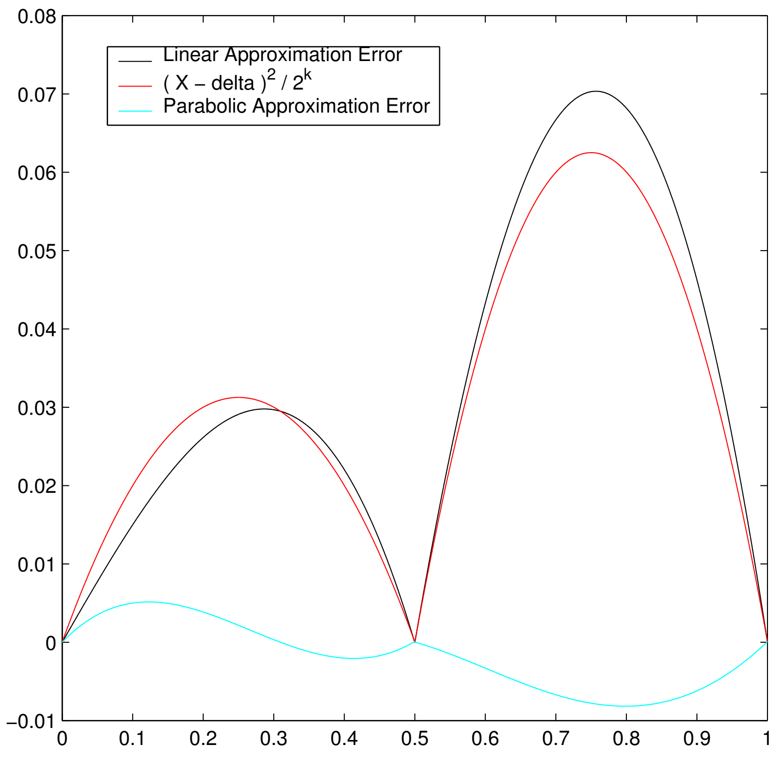


Figure 2.6: Comparison of approximation errors for a 2-segment approximation.

The MAE may then be improved by subtracting the parabolic correction factor from the linear approximation, combining 2.7 and 2.9, such that the quantitative description of the approximation is:

$$m_i x + c_i - (x - \Delta_i)^2 / 2^{k\_i} \tag{2.9}$$

This equation is expanded to:

$$-x^2/2^{k\_i} + (2\Delta_i/2^{k\_i})x - \Delta^2/2^{k\_i} + m_i x + c_i \tag{2.10}$$

Since both the m_i and $2\Delta_i/2^{ki}$ coefficients of x are constants, they may be added together to form a new (i.e., corrected) slope, m_i*. Also $\Delta_i{}^2/2^k$ can be added into c_i to form c_i*. The final equation reflecting these modifications is:

$$-x^2/2^{k\_i} + m_i^*x + c_i^* \tag{2.11}$$

The MAE of this improved approximation is shown in blue in Figure 2.6 and is much smaller than the MAE of a linear approximation. The SFDR can be calculated by taking the Fast-Fourier Transform of 2.11. Experimenting with Matlab by changing the values of M, Q, and S, and observing the resultant MAE and SFDR allows the optimal precisions to be determined. Note that c_i* need only be as precise as M, since additional precision would simply be lost. To increase the magnitude appropriately, c_i* can be internally padded with zeros on the least significant end.

Using the previously stated design parameters: $Q = 12$, $M = 12$, and $S = 16$, m_i, c_i, and k_i are obtained and detailed in Table 2.2. Figure 2.7 shows a comparison between the reference sine wave and the approximated sine wave for a full 16-segment approximation. The top portion of Figure 2.8 displays the linear approximation error, parabolic correction factor, and parabolic-corrected approximation error. A closer view of the parabolic-corrected approximation error is shown in the bottom portion of Figure 2.8, revealing a maximum approximation error (MAE) of 7.6×10^{-4}. Note that the MAE is largest in the 16th segment, partly due to the fact that the coefficient c_{16} is slightly altered to keep the output from overflowing (i.e., due to the targeted output precision). An analysis of the final output sine wave in the frequency domain is shown in Figure 2.9. The resultant SFDR is 84.2 dBc, assuming that the DAC causes no profound reduction of the SFDR.

Table 2.2: Coefficients for a 16-segment parabolic approximation.

Segment number, i	m_i	c_i	k_i
1	805	1	7
2	803	403	5
3	788	800	5
4	773	1190	4
5	743	1568	4
6	706	1932	4
7	678	2276	3
8	628	2599	3
9	572	2897	3
10	511	3167	3
11	445	3407	3
12	376	3613	3
13	303	3785	3
14	227	3921	3
15	150	4018	3
16	103	4074	2

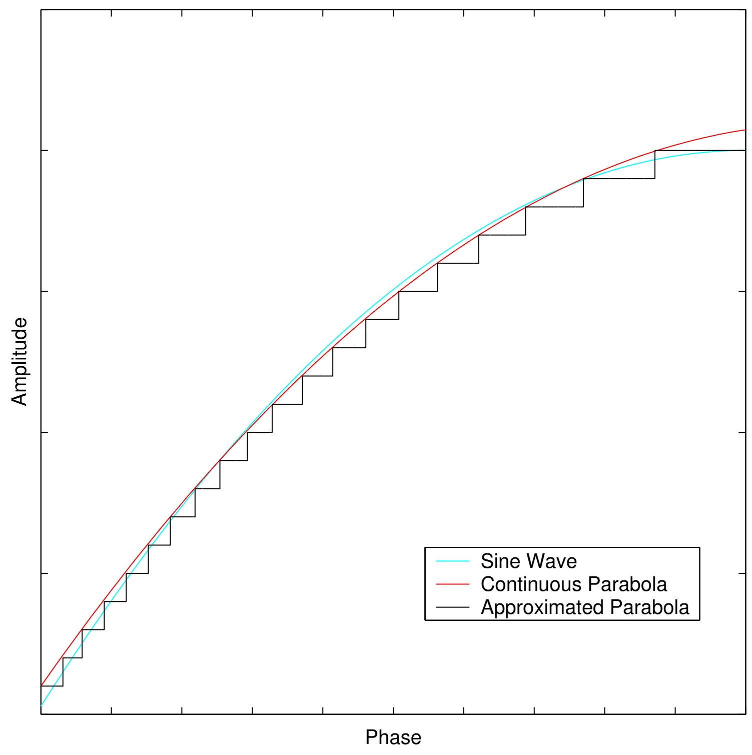
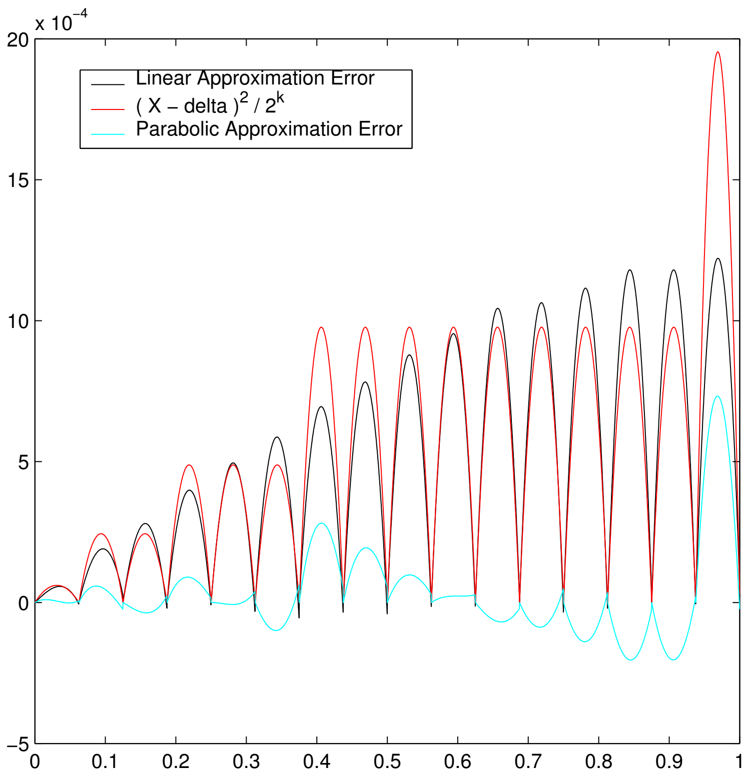


Figure 2.7: A 16-segment parabolic approximation.



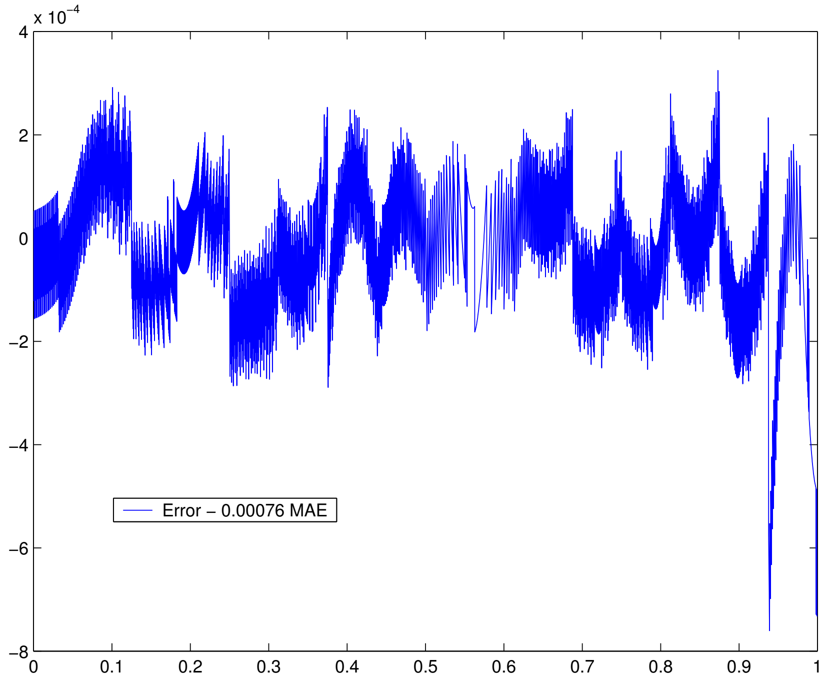


Figure 2.8: Maximum approximation errors of a 16-segment parabolic approximation (top) and a closer view of the parabolic MAE (bottom).

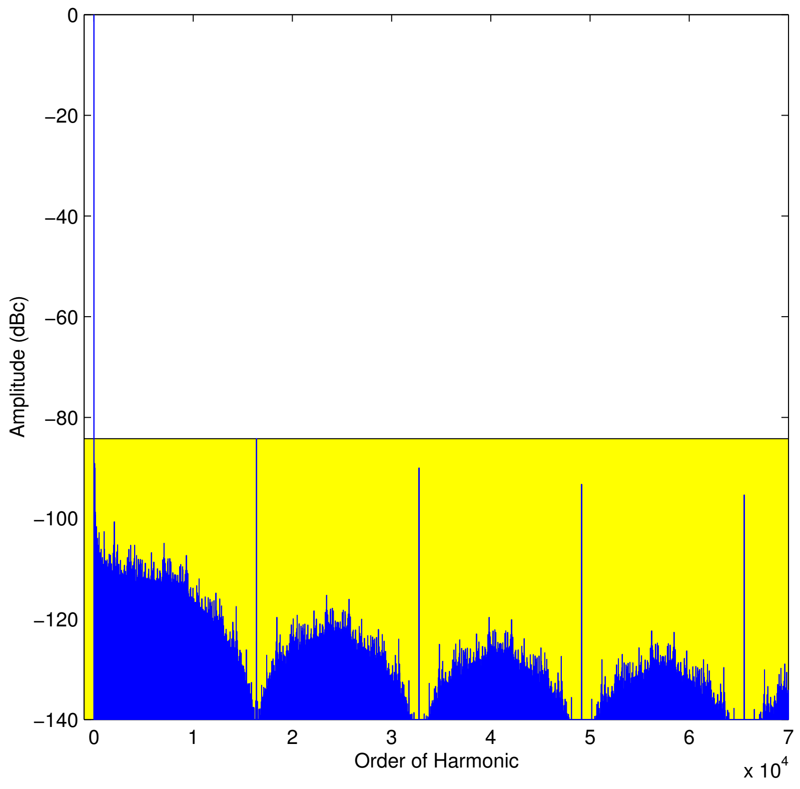


Figure 2.9: Frequency-domain analysis of a 16-segment parabolic approximation.

Hardware Implementation

To physically implement this approximation, 2.11 is rewritten as:

$$(m_i^* - x/2^{k-i})x + c_i^* \tag{2.12}$$

Based on 2.12, this solution requires an adder, a subtractor (adder with one inverting input), and a multiplier. Unlike the segmented linear approximation, *x* is not being multiplied by a constant, and therefore, the multiplexer/multi-input adder solution cannot be used. An actual multiplier is required, but since the multiplication is followed by an addition operation, a multiply-accumulate circuit (MAC) may be used to streamline operations. Two ROM-lookup tables are required to realize m_i^* and c_i^*, but again may be implemented with 16-to-1 multiplexers whose inputs are hard-wired to logic 1 or 0. Dividing x (the offset angle) by 2^{k_i} is implemented in the same manner as the segmented linear approximation. k_i represents the number of rightward shifts of x required to fit the parabolic correction factor as close as possible to the linear MAE. Since each segment has its own value of k, an additional 16-to-1 MUX is required, with each data input connecting to the shifted x for that particular segment. Like the segmented-linear approximation, the multiplexers share a 4-bit control bus, resulting in an 8-bit wide x. To minimize the hardware complexity and ROM-size, the smallest possible precisions of the arithmetic components should be used. These precisions are determined by simulating 2.12 in Matlab, adjusting the precisions of m, c, k, and the multiplication and addition/subtraction operations, and observing the effect on the MAE and SFDR.

A hardware block diagram of the approximation using the constants in Table 2.2 and having an 84 dBc SFDR is shown in Figure 2.10. The precisions of all components are listed, with the m and c ROM having 10 and 12-bit widths, respectively, and the MAC having 8-bit × 10-bit + 12-bit precision. Not shown is the shifting operations between x and the MUX, although these do not add to the hardware complexity since k is fixed. The final MAC output precision is 21-bits, but only the upper 12-bits are passed on to the DAC. The total ROM size is 400 bits.

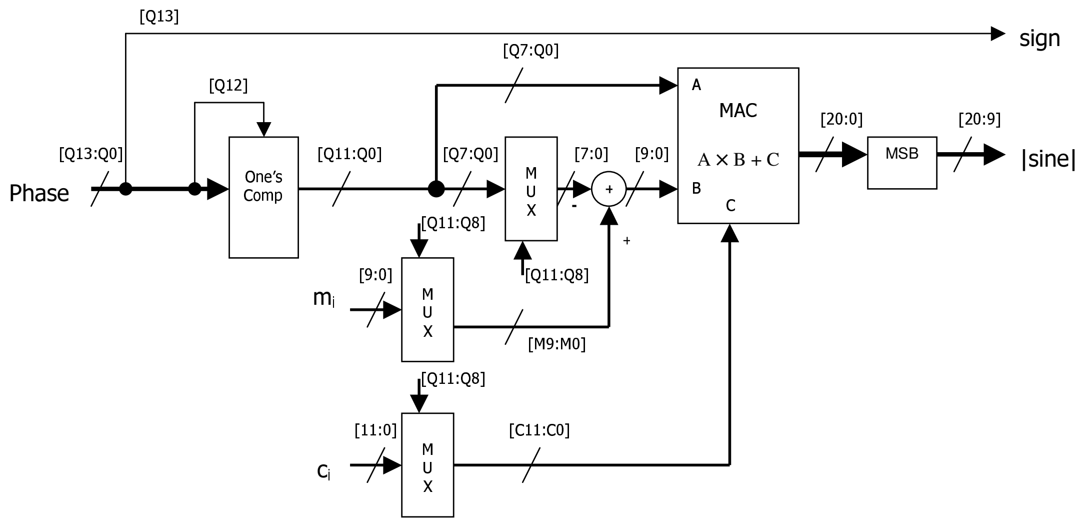


Figure 2.10: Hardware implementation of a 16-segment parabolic approximation circuit.

2.3 Hardware Complexity Comparison

Table 2.3 displays a comparison of a few most recent methods of sine approximation. The SFDR of the improved 16-segment parabolic is the second highest, and has the second smallest ROM size. To determine which approximations provide a fair compromise between accuracy and circuit complexity, a new measurement is introduced. The Cost is calculated as the ROM size divided by the SFDR. The 16-segment linear approximation offers the lowest cost, although the SFDR is relatively low for today's standards. When a high SFDR is required, then the 16-segment segmented parabolic approximation provides the lowest cost.

Table 2.3: Comparison of SFDR, ROM size, and cost for several recent sine approximations.

Approximation Method	SFDR (dBc)	ROM Size (bits)	Cost (bits/dBc)
16-segment linear [24]	66	224	3.39
32-segment linear [25]	84	448	5.33
64-segment linear [25]	96	960	10
Meitzler/Millard [22]	83	5376	64.77
16-segment parabolic	84	400	4.76

The use of the ROM size as a measure of circuit complexity is actually deceiving, as the linear and segmented parabolic approximations, and even the Meitzler/Millard, depend not only on ROM, but on complex arithmetic components, as listed in Table 2.4. Obviously, the segmented parabolic approximation requires a complex MAC and an additional full adder, both occupying significant area, but theoretically consuming less power than a ROM lookup table of the same physical dimensions. While the linear approximations lack a complex multiplier, the area occupied by a multiple-input adder is similar to a multiplier. Furthermore, the area occupied by the large number of multiplexers required by the linear approximations is significant, and consume nearly the same area as a ROM lookup table. For the 32-segment linear approximation, five D-bit 32-to-1 multiplexers are required (where $D = 10 – 16$), while the segmented parabolic approximation requires only one 8-bit 16-to-1 MUX, one 10-bit 16-to-1 MUX and one 12-bit 16-to-1 MUX.

Table 2.4: Comparison of hardware complexity for recent sine approximations.

Approximation Method	Q (bits)	M (bits)	Number of Arithmetic Components MUX	Adders	Multipliers
Meitzler/Millard	12	12	2 12-bit 2-to-1	2 2-input 12-bit	--
16-segment Linear	10	11	5 *D*-bit 16-to-1	1 5-input *D*-bit	--
32-segment Linear	14	11	6 *D*-bit 32-to-1	1 6-input *D*-bit	--
64-segment Linear	16	12	7 *D*-bit 64-to-1	1 7-input *D*-bit	--
Segmented Parabolic	12	12	1 8-bit 16-to-1, 1 10-bit 16-to-1, & 1 12-bit 16-to-1	1 2-input 8-bit & 1 2-input 21-bit	1 8×10 bit

CHAPTER 3

THE COMPONENT LEVEL

Once the required number of adders and multipliers and their precision(s) is determined at the system level, the appropriate architectures are selected with regards to the speed (clock frequency) and power requirements. Based on Figures 1.1 and 2.10, the most complex components of the segmented parabolic DDFS system are the multiply-accumulate unit and the phase accumulator. At this level, the arithmetic component architectures are characterized in terms of gates, where a gate denotes a simple or complex logic component that performs an operation fundamental to a multiple bit adder or multiplier, such as XOR or AND/OR. The power consumption is represented by the gate count, where the more gates equals higher power consumption. The speed is estimated by identifying the worst-case path through the adder of multiplier, and is expressed in terms of gate delays (or layers), which is the total number of gates in the worst-case path.

3.1 Multiply-Accumulate Circuit

Because of the inherent complexity of a multiplier circuit, the 8 × 10 + 12 MAC poses a very significant speed and power consumption bottleneck. A traditional multiplier uses array multiplication, yet the resultant structure has a long propagation delay and occupies a large area [27]. An alternative solution is to use a Wallace-Tree multiplication structure [28]. Based on the theory of binary multiplication, multiplying an 8-bit operand A by the 10-bit wide B expands to:

$$\begin{array}{cccccccc}
 & & & & PP_{0,9} & \dots & \dots & PP_{0,0} \\
 & & & PP_{1,9} & \dots & \dots & PP_{1,0} & \\
 & & \dots & \dots & \dots & \dots & & \\
+ & PP_{7,9} & \dots & \dots & PP_{7,0} & & & \\
\hline
Z_{20} & \dots & \dots & \dots & \dots & \dots & Z_1 & Z_0
\end{array}$$

where Z denotes the final product, and PP_{ij} denotes one bit of a partial product, resultant from an AND-ing operation ($A_0B_0 \dots A_7B_9$). This operation is essentially the addition of the eight shifted 10-bit partial products. Since a full adder can process only two operands, the 8 partial products must be reduced using carry-save arithmetic. The reduction is handled by using a series of interconnected binary full adders, heretofore referred to as 3|2 counters, as shown in Figure 3.1. The propagation delay through this multiplier is four 3|2 counter delays plus one AND operation delay. The addition of the 12-bit operand, C, adds one additional layer of 3|2 counters. However, the multiplication-addition result is in carry-save format, where the result is represented by twenty individual Sum and Carry bits. To concatenate these bits into the standard sum format, a 21-bit (20 Sum + 1 Carry) full adder is required.

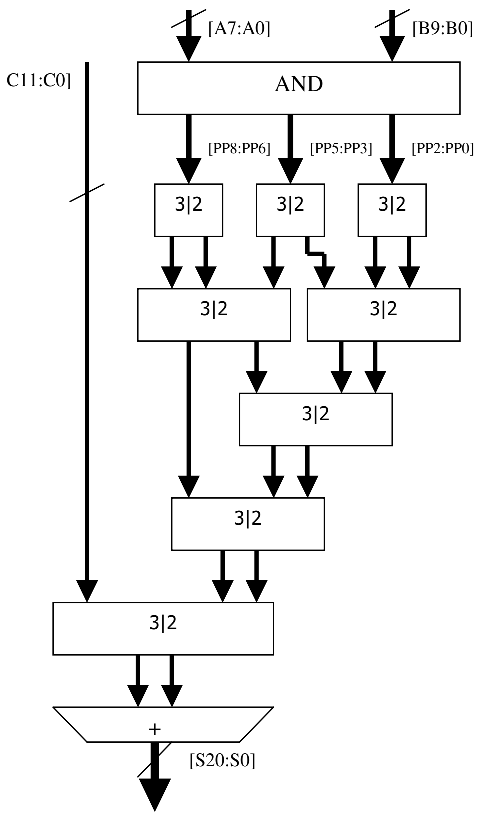


Figure 3.1: The 8 × 10 + 12 multiply-accumulate circuit

3.2 Adder Architectures

Based on Figures 1.1, 2.10, and 3.1, the most fundamental arithmetic component of the DDFS is the adder. There are two basic types of adders that comprise the design: single-bit operand binary adders and multiple-bit operand full adders. The multiplication of the MAC is performed by a Wallace tree connectivity pattern of 3|2 counters. The truth table of a single 3|2 counter is shown below:

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Therefore, the logical equations of a 3|2 counter are $S = A \oplus B \oplus C$ and $Cout = AB + BC + AC$. The half adder (HA), or 2|2 counter, has logical equations $S = A \oplus B$ and $Cout = AB$. Implemented using conventional complementary logic, a 3|2 counter has a minimum delay of two gates.

The phase accumulator, MAC, and approximation circuit rely on more complex full adder circuits that process operands of 8 to 32 bits wide. The simplest implementation of a full adder is a ripple carry adder (RCA) [29], shown in Figure 3.2, which is a serial chain of 3|2 counters plus an initial 2|2 counter. Because one 3|2 counter is required for every bit in the addition operation, $N - 1$ 3|2 + 1 2|2 counters are required, resulting in a total delay of $2N$ gates, and a total gate count of $7N$ gates, when using simple 2-input AND/OR/XOR gates. Therefore, a 32-bit RCA has a 64-gate delay and gate count of 224 gates. The RCA offers the smallest gate count out of any full adder architecture, but has the longest delay, which is considered very slow, even for low-power applications.

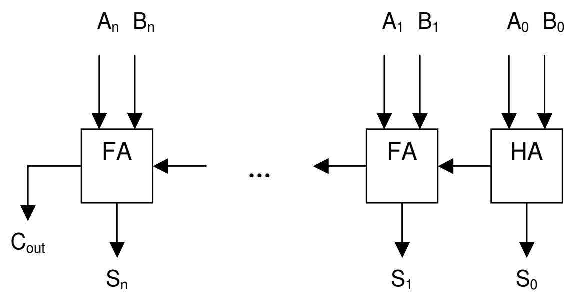


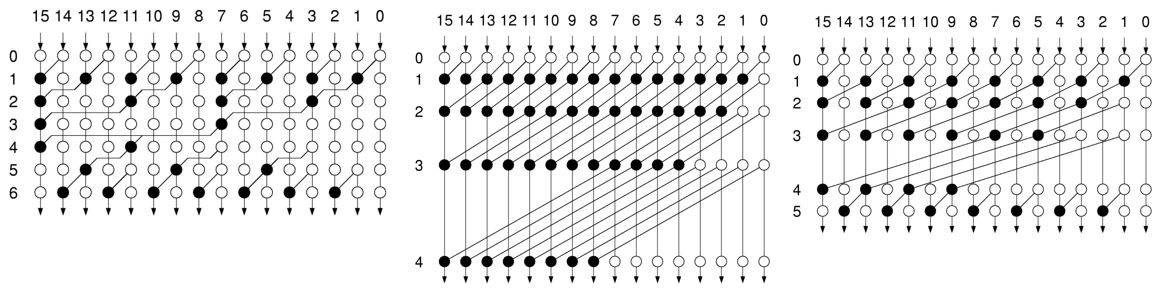
Figure 3.2: Architecture of a ripple carry adder (RCA).

For the DDFS to achieve high-speed operation, the selection of the full adder architecture is usually limited to those with a lookahead structure, otherwise known as parallel-prefix. The highest performance parallel-prefix adders (PPA) are the Brent-Kung, Kogge-Stone, and Han-Carlson architectures [29]. At the component level, the topology of these adders is depitcted by a prefix diagram to clearly indicate the gate count, delay, fan-in, and fan-out. A prefix diagram for a 16-bit Brent-Kung, Kogge-Stone, and Han-Carlson adder is shown in Figure 3.3. Each black circle in the diagram denotes an associative operation, while each white circle is simply a buffer, where the output is the same as the input; the lines represent the path of the signals. A prefix diagram has the dimensions $N \times m$, where N is the precision of the adder and m is total number of rows containing at least one black circle. Descending in hierarchy, the associative operations of a black circle are comprised of two logic equations to calculate the propagate and generate signals, such that:

$$P_{i,j} = P_{i-1,j}P_{i-1,j-1}$$

$$G_{i,j} = G_{i-1,j} + P_{i-1,j}G_{i-1,j-1}$$

where *i* is the layer number, and *j* is the bit number. Not shown in the diagrams is a top layer that computes the initial values of P and G, where $P_j = A_j \oplus B_j$ and $G_j = A_jB_j$, and a final layer responsible for calculating the final sum, where $S_j = P_{0,j} \oplus G_{m-1,j-1}$.



Brent-Kung Kogge-Stone Han-Carlson

Figure 3.3: Prefix diagrams for three parallel-prefix adder architectures [29].

Based on the prefix diagrams, the gate count is the number of black circles times the number of gates in each black circle, plus two gates for every bit in the top layer, and one gate for every bit in the final layer. The worst-case delay is obtained by multiplying *m* by the number of gate delays per black circle and adding the delay of the top and final layers. However, to accurately characterize the speed and power consumption of a PPA, it becomes necessary to consider the symbolic Fan-In (FI) and Fan-Out (FO). The FI is the total number of distinct inputs to a black circle, while the FO is the number of subsequent black and white circles driven by that particular circle. FI and FO are a representation of the total input and output parasitics, respectively, of the equivalent logic gates for each circle. As FI or FO increases, the equivalent total load capacitances and resistances increase, hence increasing the actual power consumption and the maximum propagation delay. To decrease the delay when FI or FO is large (usually greater than 2), higher speed gates may be used, but at the expense of further increasing the power consumption.

Based on the prefix diagrams for a $N = 16$ adder, the fastest architecture is the Kogge-Stone model, requiring only 6 symbolic delay layers, including the first and last layers not shown in the diagram, and having a 12-gate delay assuming each layer has a 2 gate delay. For any value of N, both FI and FO are bounded to a maximum of 2. Of the three architectures, the Kogge-Stone is the most hardware intensive, requiring 49 black circles and, consequently, 146 gates (assuming 1 gate to implement each logic equation). Conversely, the Brent-Kung model provides the lowest hardware complexity requiring only 26 black circles; however, the maximum delay rises to eight symbolic layers. While FI is finitely bounded to 2 for any N, FO is logarithmically bounded to $\log_2 N$, indicating slower operation and/or a higher power consumption as N increases. Compromising between the lowest hardware complexity of the Brent-Kung model and the high-speed and bounded FI and FO of the Kogge-Stone, the Han-Carlson offers seven symbolic delay layers, but requires only 32 black circles.

To assist in selecting an appropriate architecture, an unofficial approximation of the area-delay product (ADP) may be calculated. This function is a rough estimate of the PDP, and is defined as the Gate Count times the Delay. To account for the effects of fan-out, the result is multiplied by 1 for those architectures with a bounded FO of 2. When the fan-out is unbounded, it is assumed that the delay and/or power consumption are affected by at least 2%. Therefore, when the FO is greater than 2, the ADP is multiplied by 1 plus two-percent of the maximum FO. Table 3.1 lists the resultant ADP for 16-bit Kogge-Stone, Brent-Kung, and Han-Carlson adders (a lower value is better). The Kogge-Stone has the highest ADP due to its large hardware complexity. The Brent-Kung has a relatively large ADP as well, somewhat due to the effects of fan-out, but mostly due to the long delay. The Han-Carlson has the lowest ADP thanks to its lower hardware complexity.

Table 3.1: Area-Delay Product for 16-bit parallel-prefix adders.

Architecture	Area-Delay Product (Gates $\times$ Gate Delay)
Kogge-Stone	1752
Brent-Kung	1696
Han-Carlson	1568

Observation of the prefix diagrams reveals that the delay is $\log_2 N$ constrained, significantly smaller than a serial adder with a linear or quadratically constrained delay. The gate count, however, is a function of $N \log_2 N$, significantly larger than a serial adder. Table 3.2 lists several popular serial and parallel-prefix adders and actual equations expressing the total number of gates and gate delays [29]. The equations of the Han-Carlson adder have unknowns (Γ, Φ, Δ) since the Han-Carlson is actually a hybrid of the Kogge-Stone and Brent Kung and cannot be reasonably expressed in equations. To simplify the comparison between adder architectures and select an appropriate model based on power and speed requirements, an asymptotic model may be used. Based on the equations in Table 3.2, for a parallel-prefix adder, it is reasonable to assume that the area and delay, A and T, respectively, for any parallel-prefix adder structure can be represented as:

$$A = \alpha\, N \log_2 N \tag{3.1}$$

$$T = \beta \log_2 N \tag{3.2}$$

where α and β are coefficients for the particular type of adder, and may be derived from Table 3.1. For a Han-Carlson adder, it is reasonable to assume that $\alpha = ½$. It should be noted that these asymptotic models are very liberal in their approximations, and may require precise tweaking of α and β. With A and T, a more precise area-delay product, AT, (corresponding to power consumption) and power delay product, AT^2, (energy) may be calculated.

Table 3.2: Gate and layer count for various serial and parallel-prefix adder architectures [29].

Adder	Number of Gates	Number of Gate Delays
Ripple Carry	$7N + 2$	$2N$
Carry-Skip	$8N + 6\sqrt{(N-1)} - 6$	$4\sqrt{(N-1)}$
Carry-Select	$14N - 5k - 5$	$2k + 2$
$k = \lceil 1/2\sqrt{(8N-7)} - 1/2 \rceil$		
Carry-Increment	$10N - k + 2$	$2.8\sqrt{N}$
$k = \lceil 1/2\sqrt{(8N-7)} - 1/2 \rceil$		
Brent-Kung	$10N - 3\log_2 N - 1$	$4\log_2 N$
Kogge-Stone	$3N\log_2 N + \mathrm{N} + 8$	$2\log_2 N + 4$
Han-Carlson	$\Gamma N\log_2 N + \Phi$	$2\log_2 N + \Delta$

3.3 The Phase Accumulator

When having a high clock-frequency is a major consideration, the maximum delay through the phase accumulator poses significant limits. The total PA delay is the adder propagation delay plus the delay of one register.

Improving the Adder

Constructing a parallel-prefix adder minimizes the delay, and is imperative even for low-power implementations, but power consumption remains high. However, unlike an adder in a general-purpose microprocessor, it happens that only certain portions of the adder must operate at a high speed. The phase-to-sine approximation uses $Q = 8 - 16$ bits, requiring that these upper sum bits be available as fast as possible. The remaining $N - Q$ sum bits need not be available as quickly, but the upper Q sum bits depend on the carry bits from those lower $N - Q$ bits. Based on this requirement, there exists critical and non-critical paths through the adder. The critical path is identified by viewing the prefix diagrams and tracing the path from the output of those Q bits all the way back to the inputs. Any black circles in the way of the path form the critical path and must operate at a high speed. The black circles not included in the critical path may operate at a slower speed (hence using less power). Figure 3.4 displays an example of an illustration of the critical and non-critical paths for a 16-bit Kogge-Stone where $Q = 4$. The critical path is comprised of all those black circles within the blue outline. The physical realization of this lower power adder is achieved at the circuit level by constructing the non-critical P, G, and XOR logic gates with transistors smaller than those of the critical gates.

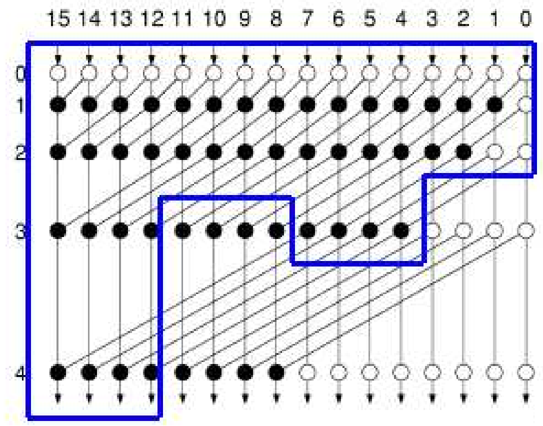


Figure 3.4: Example of the critical path of a Kogge-Stone adder (inside the blue outline) with high-speed delivery of the upper 4 bits.

Pipelining

When a smaller PA delay is desired, pipelining may be utilized at the expense of further increased power consumption [7]. As displayed in Figure 3.5, the PA is divided into *p* pipeline stages of *n* bits each, such that *np* = *N* (or equivalently *p* = *N/n*). Each adder outputs *n* + 1 bits: *n* sum bits, and one carry output bit. These bits are stored in an end register. The *n* stored sum bits then feed back into the adder, and the latched carry output bit connects to the carry input of the adder in the next pipeline stage. To store the upper *Q* = 10 – 12 bits, additional end registers must be placed on the *p* – 1, *p* – 2, … stages of the pipeline. *p* clock cycles are required to fully initialize or flush the pipeline. While a non-pipelined PA requires only two *N*-bit register banks, the total number of flip-flops for a multiple stage pipelined PA is obtained based on equation 3.3:

$$FF_{Total} = \frac{p(p+1)n}{2} + p(n+1) + End\_FF - 1 \qquad (3.3)$$

To determine the end register requirements for a Q-bit phase-to-sine conversion circuit assuming $Q < N$:

$$End\_FF = \begin{cases} Q-n, if\ A=0\ and\ Q-n \geq 0 \\ An, if\ A \neq 0\ and\ Q-n \geq 0 \\ 0, else \end{cases} \tag{3.4}$$
$$A = \sum_{j=0}^{j<\lceil Q/n \rceil} j$$

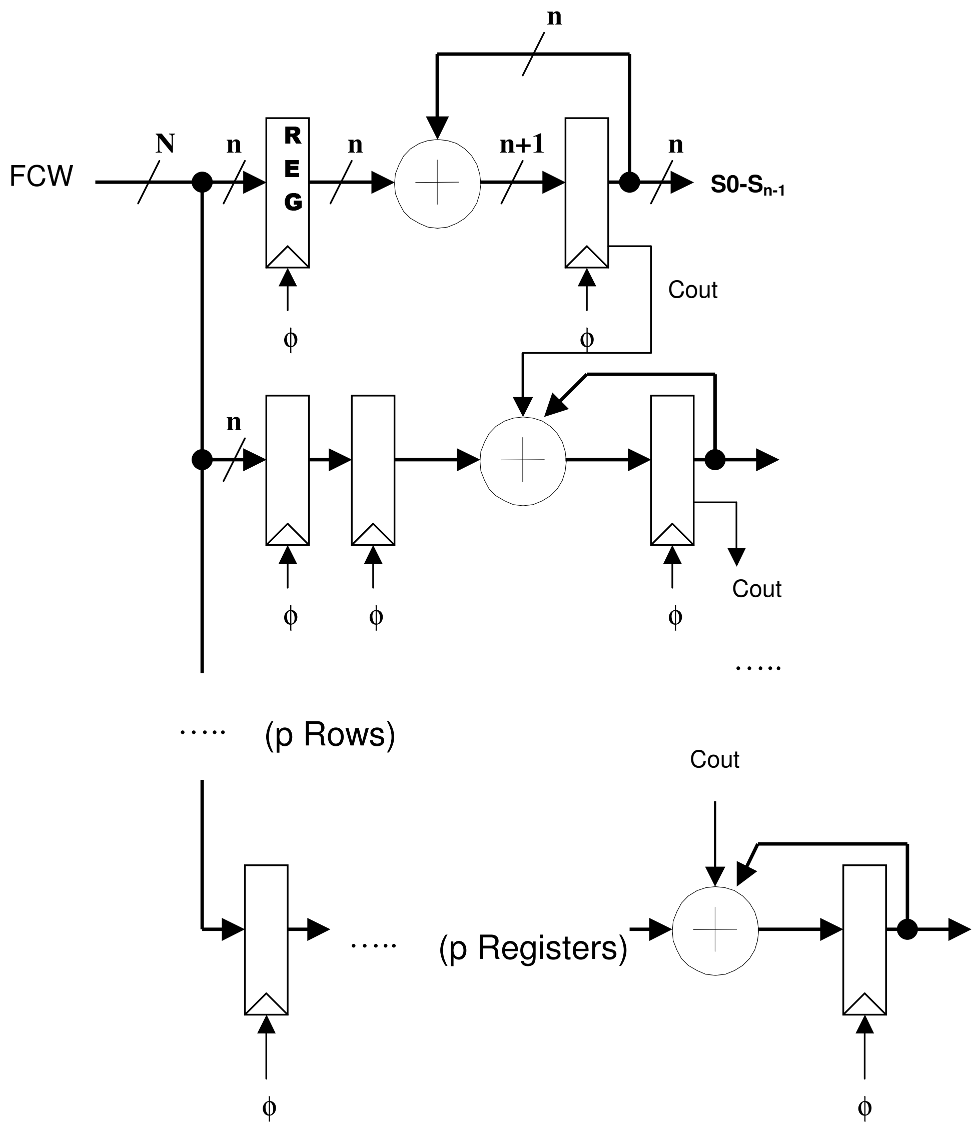


Figure 3.5: Configuration of a parallel-pipelined phase accumulator.

It follows that a parallel-pipelined PA requires almost twice as many flip-flops as a single-stage pipelined PA. Based on equations 3.1 – 3.4, it is possible to estimate an optimal *n* with respect to minimal power. The total area of the PA can be represented as:

$$A(N, n) = \varphi FF_{Total}(N, n) + \alpha n \log_2 n \qquad (3.5)$$

The latency L of one pipeline stage is:

$$L(N, n) = \beta \log_2 n + \gamma \tag{3.6}$$

The total delay through the PA (from the 1st to mth stage) is:

$$T(N, n) = p\, L(N, n) \tag{3.7}$$

Here φ and α are the area coefficients of a flip-flop and gate that comprises the black cell, respectively, and β and γ are the associated delay coefficients. More precisely, φ is the ratio of the power dissipation of a flip-flop to that of an adder gate. Since the actual power dissipation is not known until completion and simulation of the IC layout, φ may be estimated as the ratio of the number of transistors composing a single flip-flop to the number of transistors in a single adder gate. High-speed flip-flops produced with current CMOS processes are composed of 15 – 25 transistors, and a single adder gate consists of 4 – 8 transistors using standard CMOS, pseudo NMOS, dynamic, or static threshold logic [27] – [32]. Consequently, it is estimated that $\varphi = 2\ldots4$. In a standard 0.25 μm CMOS process, adder gates with a delay of 50 – 100 ps [30][31], and flip-flops with a delay of 125 – 300 ps [32] have been reported. Therefore, it may be concluded that $\gamma = 1\ldots6$.

The measures AT and AT^2 will be used for finding optimal designs for the PA. Using equations 3.5 – 3.7, the optimal n values can be determined. The simplest method is to create surface plots of L, T, and AT for all practical values of N and n. For a PA with $Q = 12$ constructed with a Han-Carlson connectivity pattern adder, and with $\varphi = 4$ and $\gamma = 2$, the AT surface plot is shown in Fig. 3.6. Interpreting the surface plot, maximizing n minimizes power consumption since fewer flip-flops are used in the design. Of course, increasing *n* also increases the latency.

To determine the optimal *n* for a given *N*, it is useful to take a cross section view of the *AT*, delay, and latency surfaces, and superimpose them on the same plot. Results for *N* = 16, 24, and 32 are shown in Figures 3.7 – 3.9. Note that the *AT* and latency cross-sections have been normalized in order for all three views to be visible on the same plot. As observed from the three figures, the lowest power consumption occurs at *n* = *N*. This result is expected, since the minimum number of flip-flops is used, of course this configuration has the largest latency. If a smaller latency is desired, but within low power consumption limits, all the plots show that *n* = *N*/2 is an excellent power-latency combination. The smallest latencies are achieved for *n* = 1, with the power consumption jumping to nearly 32 times the minimal value!

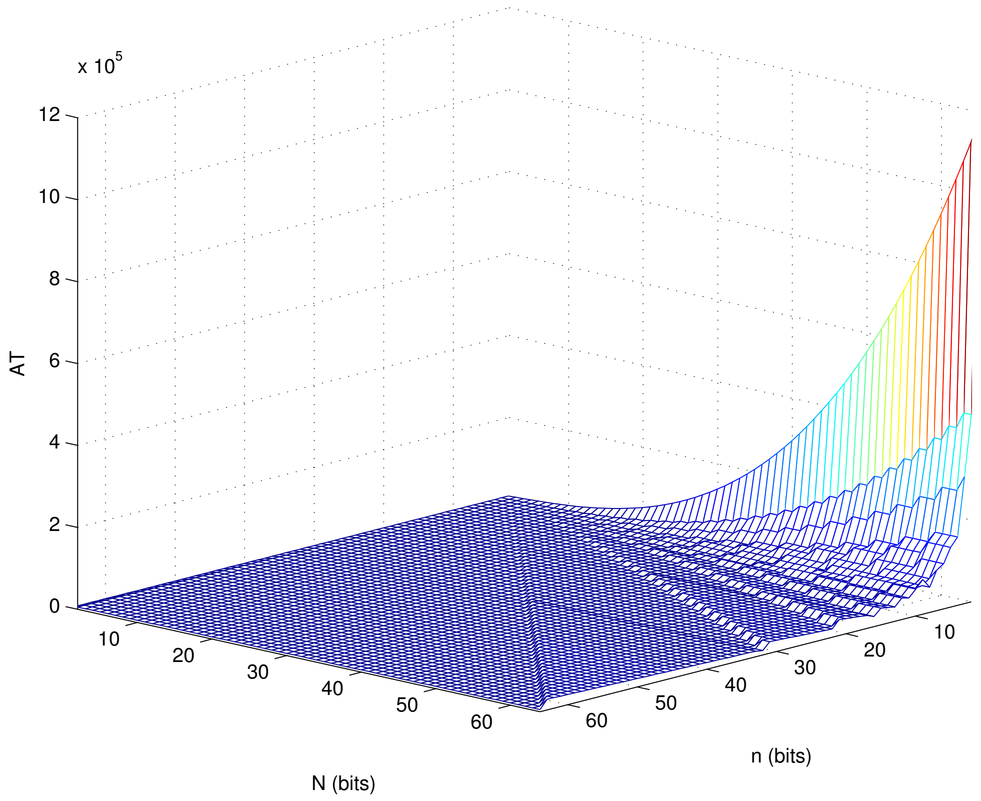


Figure 3.6: Surface plot of *AT* (power consumption) for a Han-Carlson phase accumulator with $\varphi = 4$, $\gamma = 2$.

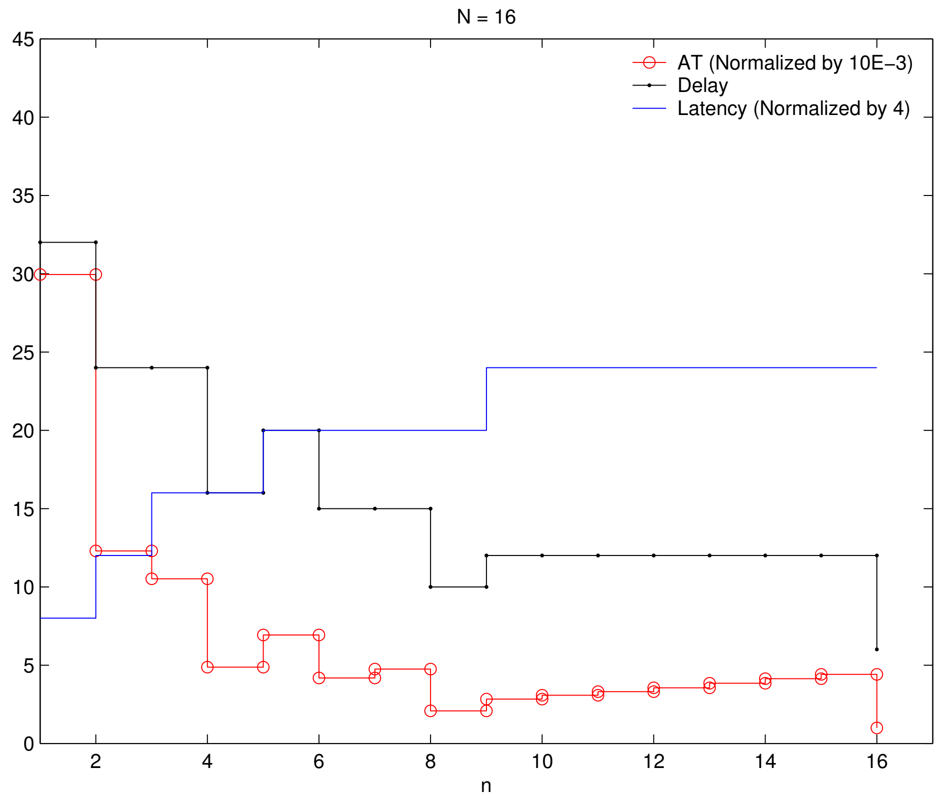


Figure 3.7: Superimposed cross section views of AT, delay, and latency for a 16-bit Han-Carlson phase accumulator with φ = 4, γ = 2.

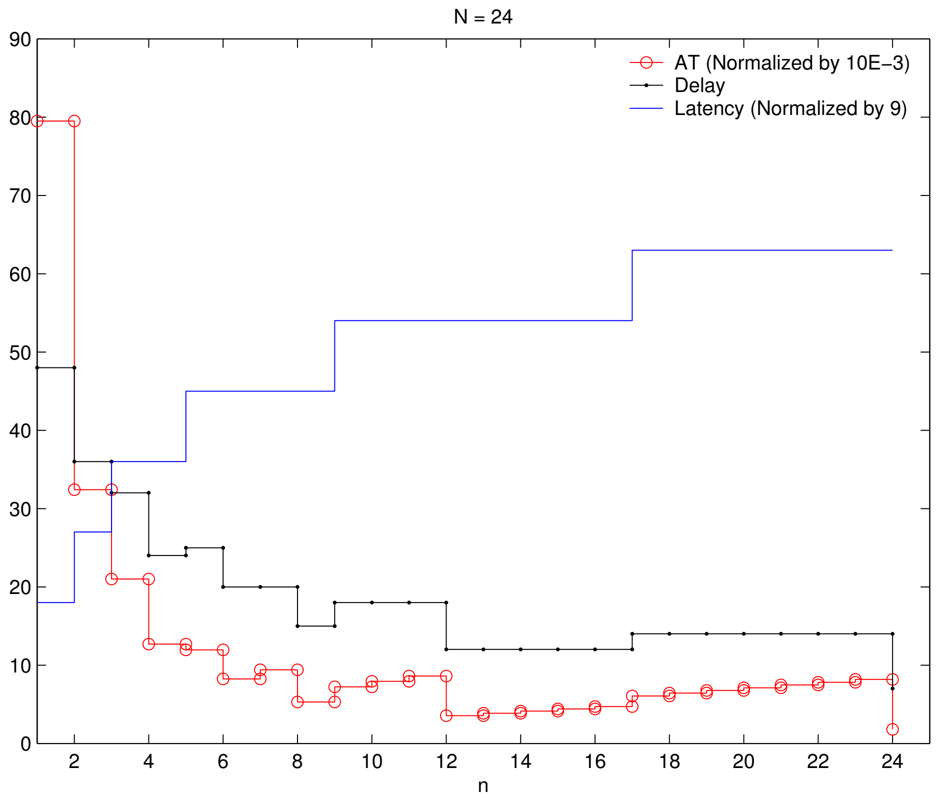


Figure 3.8: Superimposed cross section views of AT, delay, and latency for a 24-bit Han-Carlson with φ = 4, γ = 2.

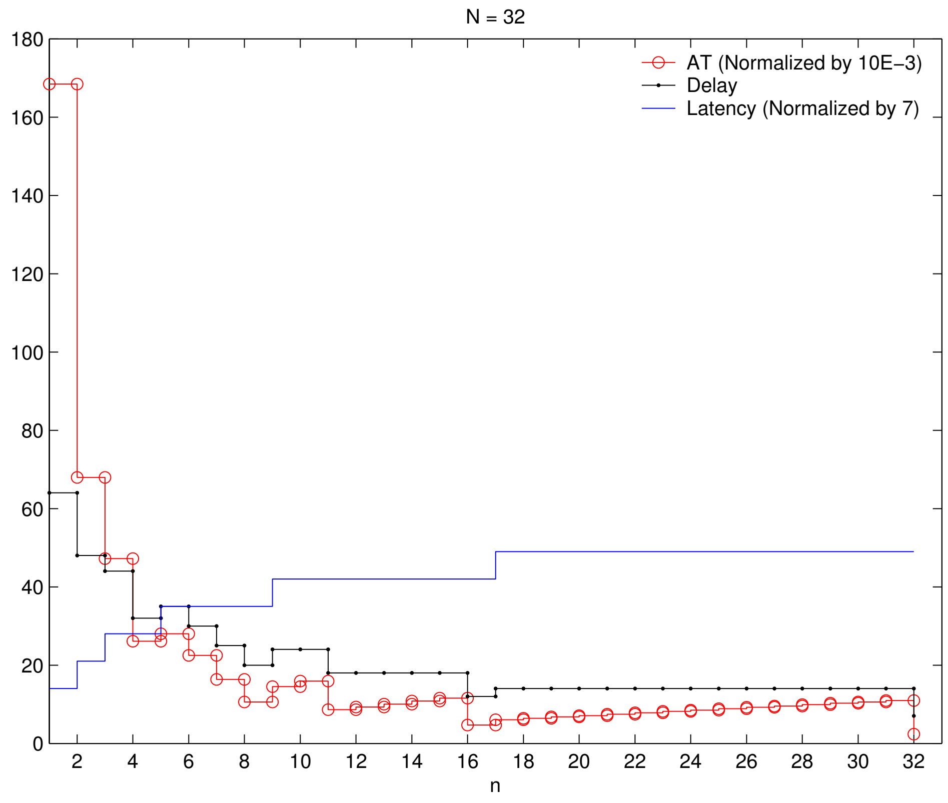


Figure 3.9: Superimposed cross section views of AT, delay, and latency for a 32-bit Han-Carlson phase accumulator with φ = 4, γ = 2.

3.4 Other Design Considerations

Pipelining the phase accumulator definitely provides a significant improvement to the overall clock frequency. However, the PA is not the only major speed constraint. Assuming that an *M*-bit register is placed before the DAC (where *M* is the DAC precision), the propagation delay through the phase-to-sine approximation circuit must be equal to or less than the latency of a single PA pipeline stage, otherwise a higher clock frequency cannot be used. Coupled with the fastest parallel-prefix adder, the Kogge-Stone, the minimum delay of the MAC circuit is 18 gates. Assuming a 2-gate delay for each multiplexer, the total approximation circuit delay is at least 25 gates. Therefore, pipelining the approximation circuit is imperative. Because there are no feedback components, pipelining the MAC (and the other components) is relatively trivial. With some minor modifications, the same type of mathematical model used with the PA may be used to determine the optimal number of pipeline stages.

With the introduction of pipelining, questions arise to the issue of data hazards, as in a pipelined microprocessor. Because the DDFS always executes sequentially, there exists no data hazards, and no need to "stall" execution. However, deepening the pipeline increases the total system delay, consequently reducing the frequency switching speed. When the value of FCW changes, *p* clock cycles (where *p* = Total Pipeline Stages) are required until the output based on the new FCW is seen at the output of the DAC. Should the target application have a rapidly changing FCW, then a deep pipeline is probably not appropriate.

Like the phase accumulator adder, the 21-bit MAC adder may be modified to reduce power consumption. A critical path exists for those upper 12-bits of the output used by DAC. The black circles that comprise the remaining non-critical path of the lower 9-bits may be completely eliminated, as those bits are not required anywhere else in the circuit.

CHAPTER 4

THE CIRCUIT LEVEL

Observing the DDFS from both the system and component levels, the adder is obviously the most common and largest component of DDFS, since a segmented parabolic implementation requires three 8, 21, and 32-bit adders. Second to the pipeline registers, the adders single-handedly influence the overall speed and power consumption. Circuit level analysis descends into the black circles of the parallel-prefix adders to design, simulate, and characterize the comprising logic gates. While this section focuses exclusively on those black cell logic functions, the analysis techniques may be applied to any type of CMOS logic gate in a full-custom IC. To fully understand the motivation behind these techniques, a fundamental understanding of the power consumption of an IC is required

Power Consumption

The total power consumption (in Watts) of a single logic gate is calculated as:

$$P_{Total} = V_{DD}\, I_{TOTAL} \tag{4.1}$$

where I_{TOTAL} is the total current that flows between V_{DD} and GND. $\mathrm{P_{Total}}$ can also be expressed as:

$$P_{Total} = P_{Static} + P_{Dynamic} + P_{Leakage} \tag{4.2}$$

P_{Static} is the power dissipated when there exists one or more short-circuit or low-impedance paths between V_{DD} and GND. There is no standard equation for P_{Static} as it is based on the number of short-circuit paths, their respective impedances, and the duration for which there exists a short-circuit. The impedance is a function of the width of the transistor(s) that form the short-circuit path(s), where wider transistors have lower impedance.

$P_{Dynamic}$, which is the power dissipated during 0→1 or 1→0 transitions due to the charging and discharging of load and parasitic capacitances through parasitic resistances, is calculated as

$$P_{Dynamic} = C_L V_{DD}^2 f \alpha \tag{4.3}$$

where C_L is the total equivalent load and parasitic capacitance, proportional to the on-chip area, f is the operating frequency (the clock frequency in a pipelined system), and α is the switching activity factor. $f \propto RC_L$, where R is the sum of the interconnect and drain-source resistances between C_L and V_{DD} or GND. When $\alpha = 1$, every transistor in a gate switches during every 0→1 or 1→0 transition. Realistically, such an occurrence is very uncommon, and usually $\alpha = 10\%$.

$P_{Leakage}$ is the power dissipated when current flows between the drain and source of a MOSFET while it is operating in the cutoff region. The leakage current for a gate is calculated as:

$$I_{Leakage} = W_{eff} I_s e^{\frac{V_{TH}}{V_o}} \tag{4.4}$$

where W_{eff} is the effective width of the cell, I_s is the effective zero-threshold leakage current, and V_o is the sub-threshold slope [33].

The power delay product, $P_{Total} \times (1 / f)$, determines the total energy usage, directly related to the battery life and the heat dissipation. The objective is to minimize either P_{Total} or *Delay*, but realistically only P_{Total} is minimized due to the linear relationship between *Delay* and $P_{Dynamic}$.

In order to reduce P_{Total}, one or more of the three components must be decreased. P_{Static} is minimized by reducing the number of short-circuit paths and increasing the impedance of those paths that cannot be eliminated. Decreasing f linearly decreases $P_{Dynamic}$, but this option is not usually possible when maximum speed operation is always desired. A more realistic method is to decrease C_L, by minimizing the hardware complexity and interconnect lengths, and also by minimizing transistor widths (but at the possibility of decreasing f). Decreasing V_{DD} causes a quadratic decrease, at the expense of reduced noise margins and the possibility of incorrect outputs.

4.1 Implementation

For a parallel-prefix adder, the two separate logical functions represented by a black cell with a symbolic fan-in of 2 are [29]:

$$P = P_n P_{n-1}$$

$$G = G_n + P_n G_{n-1}$$

P is implemented with a two-input NAND gate. Using discrete logic gates, G requires two NAND gates, experiencing a delay at least twice as long as P. To conserve both area and speed, G may be implemented with a single complex logic gate where the delay, although larger than P, will be smaller than using two discrete gates. Because G is inherently the slower and greater power consumer of the two functions, the design and testing process presented is exclusively for an implementation of G, but can be applied to any type of logic gate. There exist four popular Boolean logic styles for implementing G in a CMOS process.

The most traditional implementation uses complementary logic, or hereafter referred to as CMOS [27], shown in Figure 4.1. G (or rather the inverse) is realized using the NMOS pull-down network, and the complementary PMOS pull-up network provides the logic 1 output. Based on the logic structure, there cannot exist a constant short-circuit or low-impedance path between V_{DD} and GND, eliminating P_{Static}. Due to the complementary network, any input transition must switch at least two transistors, such that the driver gate sees a large input capacitance. Additionally, having complementary transistors increases the total output load capacitance, C_L, resulting in a large $P_{Dynamic}$, and increases the series resistance for charging or discharging the load capacitance, thereby decreasing the speed. Except in rare circumstances, the output voltage for logic 1 always swings to V_{DD}, and GND for logic 0.

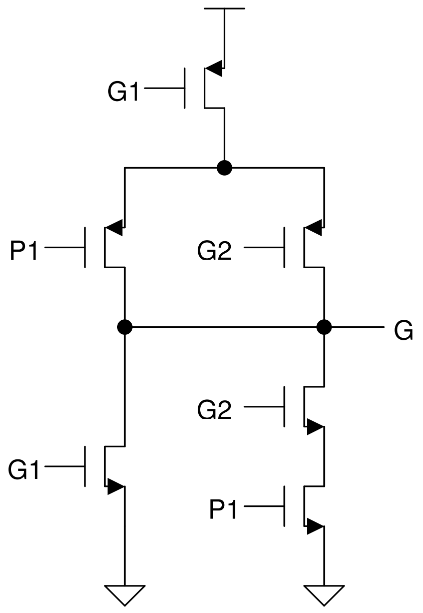


Figure 4.1: A complementary logic (CMOS) implementation of G.

Figure 4.2 shows a pseudo-NMOS implementation of G, where the complementary pull-up network has been replaced by a single always-on PMOS transistor [27]. The width of this transistor is normally smaller than the equivalent width of the CMOS pull-up network, reducing C_L, the total input capacitance, and the series resistance. Because this transistor is always on, however, there exists a low impedance path between V_{DD} and GND during logic 0 outputs. This situation poses two major challenges. First, P_{Static} is large, and when averaged, dominates P_{Total}. Second, the noise margins are smaller than a CMOS gate. For a logic 1 output, G swings to V_{DD}, but during a logic 0, G does not swing to GND, but:

$$V_{out} = V_{DD}\,[R_N / (R_N + R_P)] \tag{4.6}$$

where R_N and R_P are the equivalent series resistances of the NMOS and PMOS transistors, respectively. If V_{out} is greater than 10% of V_{DD}, the transistors in the load devices connected to the output may instead recognize a logic 1 output. To keep V_{out} below 10% of V_{DD}, the widths of the PMOS and NMOS transistors should be reduced, and while P_{Static} is reduced, the speed is reduced.

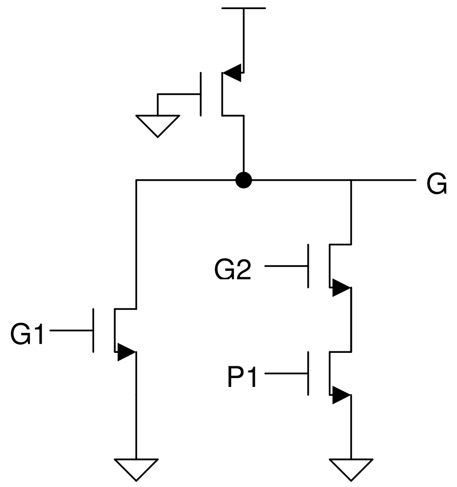


Figure 4.2: A pseudo-NMOS logic implementation of G.

Shown in Figure 4.3 is a dynamic logic gate, and an inverter used in the clock distribution network [27]. The pull-down network is connected to V_{DD} and GND by a PMOS and NMOS transistor, respectively, whose gates are connected to a common clock, ϕ. The configuration of this gate results in two phases of operation, pre-charge and evaluate. During pre-charge, ϕ is at logic 0, turning on the PMOS and turning off the lower NMOS, charging C_L to V_{DD}. The gate enters the evaluate phase when ϕ is logic 1, turning off the PMOS and turning on the NMOS. If the input combination results in a logic 0, C_L discharges through the NMOS transistors that are turned on so that G = GND; any logic 1 output leaves G $\approx$ V_{DD}, as C_L slowly discharges through the load and parasitic resistances. With the two-phase operation, $P_{Static} = 0$, as there never exists a path between V_{DD} and GND. However, it must be assured that the inputs change only during pre-charge. A $0 \rightarrow 1$ output transition cannot occur during evaluate, as there exists no connection to V_{DD} to charge C_L to logic 1. Because most digital systems are pipelined, it becomes trivial to synchronize the inputs to meet this requirement. With the reduced C_L, $P_{Dynamic}$ is smaller than a CMOS gate, and the speed is improved. $P_{Dynamic}$ is usually experienced less frequently: only during a $1\rightarrow0$ transition upon entering evaluate, and a $0\rightarrow1$ transition entering pre-charge.

Because this gate is synchronous, two major problems non-existent to the other gates presented are introduced. First is the issue of clock skew, when the clock signal arrives at the load gates before arriving at the driver gate [34]. Should the propagation delay of the clock interconnect between the load and driver be equal or greater than the pulse width, the load gates prematurely evaluate the inputs before the driver has actually had a chance to produce those correct inputs. Skew becomes even more significant as clock frequencies increase; therefore, it becomes necessary to have a very carefully designed clock distribution network. The second major problem is with the fundamental concept of a clock distribution network itself. In modern-day integrated circuits, clock distribution and communication account for up to 60% of the overall power consumption. The increased power consumption is due to the fact that the transistors of the clock buffers are constantly switching as long as a clock signal is present. When calculating the power consumption of a dynamic gate, it is necessary to include the power consumption of the clock buffer(s) driving the gate. Additionally, assume that the inputs to the dynamic gate are held constant for several clock periods such that G = GND. Consequently, G must swing between V_{DD} and GND upon entering and exiting pre-charge. Therefore, when averaged over the duration of those clock cycles, the $P_{Dynamic}$ of this gate is greater than an asynchronous gate whose output remains fixed at GND for the same duration. For a constant logic 1 output, at the end of evaluate, G must swing back to V_{DD} due to the charge leakage, although this swing is smaller in magnitude. It becomes possible that the combined power consumption of the gate and clock buffer(s) plus $P_{Dynamic}$ during idle inputs may exceed the power consumption of an asynchronous logic gate.

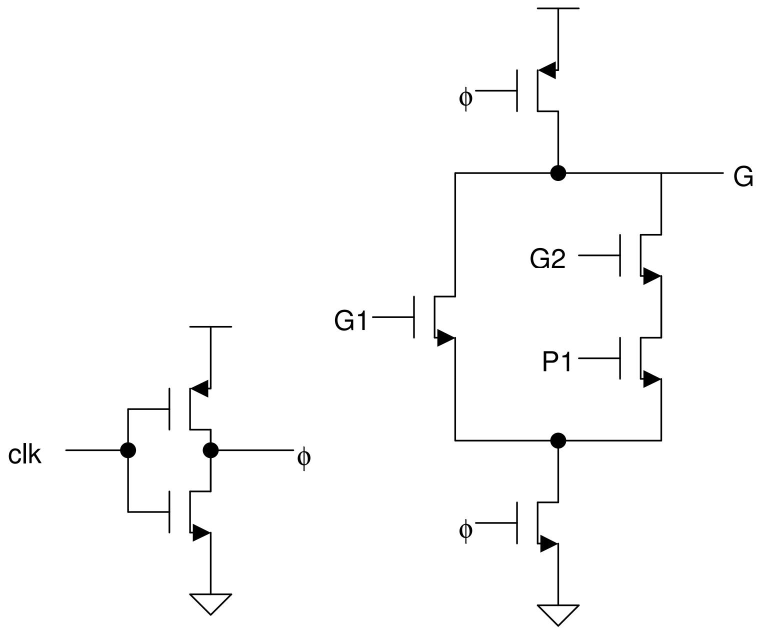


Figure 4.3: A dynamic logic implementation of G (right) and the clock driver (left).

A differential cascode voltage switch (DCVS) gate [35][36], shown in Figure 4.4, attempts to provide the reduced C_L of pseudo-NMOS and dynamic gates, but with asynchronous operation and no P_{Static}. The fundamental component is the differential PMOS stack comprised of two identically matched transistors whose gates are driven by the output of the inverse, insuring a purely differential output. Because both pull-down networks must be exactly symmetric, the pull-down function cannot be implemented with the NMOS network used in the previous three gates. G and nG are realized using pass-gate (PG) logic [36], a modified version of pass-transistor logic that prevents floating nodes. No static power dissipation exists for DCVS-PG gates, and C_L is reduced due to the absence of a complementary pull-up network.

Because of the dual logic networks, however, $P_{Dynamic}$ is expected to be greater than that of a pseudo-NMOS or dynamic gate. Additionally, the total input capacitance is twice as large, and the gates driving G1 and nG1 also experience input resistance. As a result, lower speed operation may be expected.

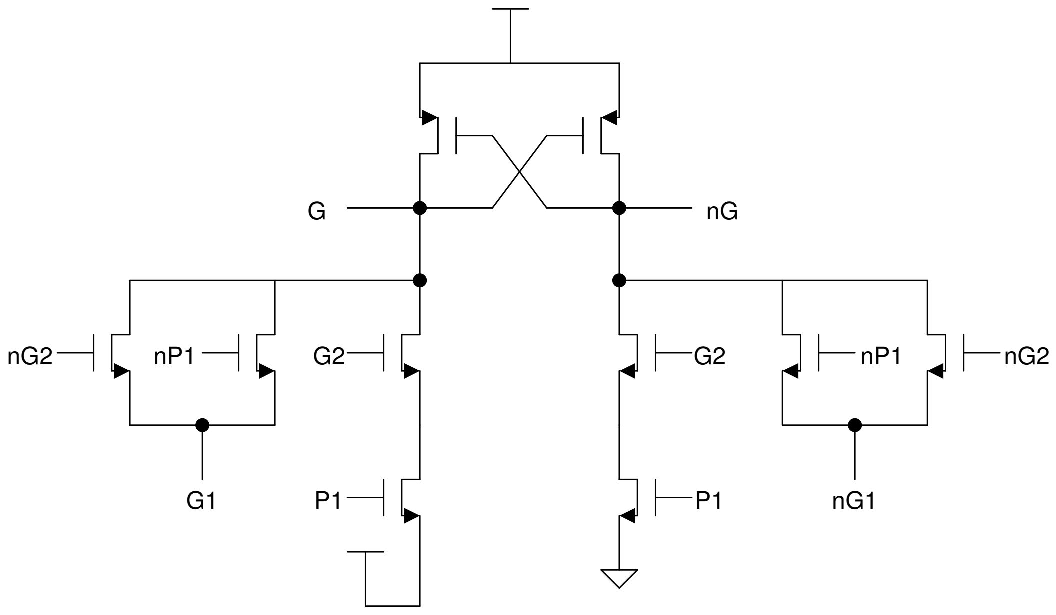


Figure 4.4: An implementation of G using differential cascode voltage switch pass gate (DCVS-PG) logic.

4.2 Test Procedure

This section details the testing procedure used to determine the actual speed and power consumption of the four logic gates when used in the context of a parallel-prefix adder. The gates are constructed and simulated in the schematic view of a leading EDA software program. To obtain an accurate characterization, the tests must take place under the reproducible worst-case conditions for the gate.

Setup

A diagram of the test setup for each gate is shown in Fig. 4.5. Assuming a Han-Carlson or Kogge-Stone adder, the maximum fan-out of G is 3, so each gate drives a load of three identically sized gates of the same style. To create the largest possible C_L, the output connects to the widest input transistors in the load. Since the gates have been constructed at schematic view, the interconnects are assumed to be ideal and have no effect on the final results presented here. In a physical layout, however, the interconnects significantly affect the overall performance of a gate, and the layout simulations have the possibility of producing completely different results.

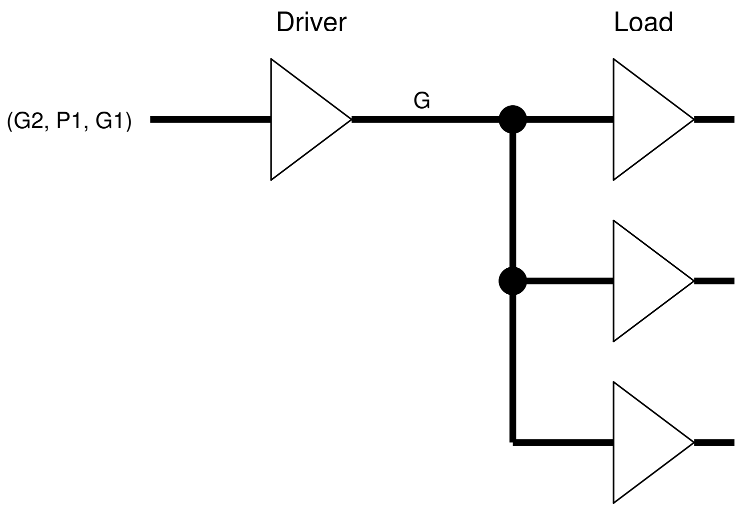


Figure 4.5: Driver gate with a fan-out of 3.

To maintain consistency in the test process, the widths of the pull-down transistors of the CMOS gate will be used for the other three gates. The minimum NMOS width is set at six times the minimum feature size, λ, of the CMOS process. However, the sizing of the pull-up transistors varies for each type of gate. For the CMOS gate, the width of the PMOS transistors are chosen such that the high-to-low, t_{HL}, and low-to-high, t_{LH}, propagation times are equal (typically $W_P \geq 2W_N$). For the pseudo-NMOS gate, the PMOS is sized such that $V_{out} = 10\% \ V_{DD}$ during a logic 0. The dynamic gate has the PMOS sized such that t_{HL} and $t_{Pre\text{-}charge}$ are equal. t_{HL} is defined as the delay between the rising edge of ϕ following an input change during pre-charge phase, and the resultant output change in the evaluate phase. $t_{Pre\text{-}charge}$ is the delay between a falling edge of ϕ and the resultant output change to logic 1 during a pre-charge phase. For the DCVS-PG gate, equal t_{HL} and t_{LH} are desired; however, as will be shown later, these times cannot be equalized with the particular connectivity of this gate. Therefore, the PMOS width is set to the minimum allowed process width of a PMOS.

Two sets of simulations are performed for each gate: one to determine the worst-case maximum delay and the other to determine the worst-case power dissipation. For the delay, the input pattern (for $G_1P_1G_2$) $010 \rightarrow 011$ has been selected, as changing the LSB causes the output to change between logic 0 and 1, respectively. Since only one input is changing, the equivalent series resistance of the pull-up and pull-down networks is high, resulting in slower operation. Additionally, those transistors connected to G_2 have the largest widths, equating to the largest input capacitance. The worst-case power dissipation is measured using the input pattern $000 \rightarrow 111$, which causes a $0 \rightarrow 1$ output transition. Because all three transistors are switching, the equivalent series resistance of the pull-up and pull-down networks is minimal, allowing for maximum current flow. With the exception of the dynamic gate, the input pattern change has a period of 2 ns, which is slow in relation to the maximum speed of the CMOS process, but allows ample settling time for the final outputs. For the dynamic gate, ϕ has a period of 2 ns, and the input pattern change occurs during a selected pre-charge phase.

The total power consumption is measured by multiplying the total voltage swing, which is V_{DD}, times the average current, I_{Avg}. I_{Avg} is measured by averaging the total current that flows through V_{DD} over a single 2 ns input pattern or clock period. For the dynamic gate, the inputs have been configured such that the selected clock cycle includes a $0 \rightarrow 1$ transition during the pre-charge phase. Additionally, the current of the clock network must be considered; therefore, the current flowing through the clock buffer is added to those measurements.

The effects of process scaling are to be investigated for these simulations to analyze and foresee the effects on speed and power consumption. Table 4.1 displays the predicted changes to the minimum values of L, V_{DD}, and V_{TH} for recent and future CMOS processes [37]. The speed and power consumption are expected to increase and decrease, respectively, as λ becomes smaller. As L and the minimum device widths decrease, along with the oxide thickness, the parasitic and interconnect capacitances become smaller, lowering C_L and the total input capacitance. With the channels becoming narrower (as L becomes smaller), the drain-to-source resistances are reduced. However, the decreases in channel widths and threshold voltages significantly increase $P_{Leakage}$. As shown by equation 4.4, any such decrease in V_{TH} exponentially increases $P_{Leakage}$. Although $P_{Dynamic}$ and P_{Static} are magnitudes larger than $P_{Leakage}$ in today's CMOS processes, future decreases of λ will certainly increase it to significant values.

Table 4.1:Characteristics of present and future (highlighted) CMOS processes [37].

Year	Minimum L (μm)	V_{DD} (V)	V_{TH} (V)
1998	0.25	3.0	0.50
2001	0.18	1.8	0.45
2002	0.13	1.2	0.40
2003	0.10	1.2	0.21
2004	0.09	1.2	0.20
2005	0.08	1.1	0.20
2007	0.065	1.1	0.18
2010	0.045	1.0	0.15
2018	0.018	0.7	0.11

To investigate the effects of process scaling, three CMOS processes (characteristics shown in Table 4.2) have been chosen: a 0.25 μm and 0.18 μm non-epitaxial process from TSMC, and an 0.13 μm the Berkeley Predictive Technology Model (BPTM) process [38]. This 0.13 μm process is not a physical process but a conservative 1999 extrapolation of a 0.13 μm process.

Table 4.2: Specifications of CMOS processes to be used in simulations.

Process	L (μm)	V_{DD} (V)	V_{TH} (V)
TSMC 0.25 μm	0.30	2.5	0.50
TSMC 0.18 μm	0.18	1.8	0.49
BPTM 0.13 μm [38]	0.13	1.2	0.313

Simulation Results

Table 4.3 displays the maximum delay, current, and PDP for the simulations of the four logic gates in the three CMOS processes. Table 4.4 displays how these parameters improve (or decrease, to be exact) as the process becomes smaller. Based on the results, the dynamic gate provides the best PDP and optimal delay and current. However, the power consumption is somewhat inaccurate as these simulations assume ideal clock interconnects. While the results of all the gates are affected by this assumption, clock interconnects tend to be the longest on-chip interconnects. The DCVS-PG gate, while offering the lowest power consumption, offers the longest delay of any of the four gates. This poor performance can be attributed to the fact that the pass-gate logic networks must drive a large capacitive and resistive load. Since the PMOS transistors are minimum sized, it is thought that increasing the widths improve the speed. The opposite has been observed because G (and nG) experiences a larger C_L due to the increased gate-to-source capacitance of the inverse PMOS. Considering only asynchronous gates, the pseudo-NMOS offers the shortest delay, but the largest power consumption, while the CMOS gate offers a moderately low delay and power consumption.

Table 4.3: Simulation results of the four types of logic gates for all three CMOS processes.

Gate	Process	Delay (ps)	Current (μA)	PDP (fJ)
CMOS	TSMC 0.25	177.37	187.9	83.3
	TSMC 0.18	112.17	75.1	15.2
	BPTM 0.13	72.89	32.09	2.81
Pseudo-NMOS	TSMC 0.25	144.04	181.82	65.5
	TSMC 0.18	78.40	159.3	22.5
	BPTM 0.13	54.29	69.88	4.55
Dynamic	TSMC 0.25	127.84	129.31	41.3
	TSMC 0.18	87.07	54.11	8.48
	BPTM 0.13	50.30	20.98	1.27
DCVS-PG	TSMC 0.25	281.70	128.71	90.6
	TSMC 0.18	185.59	47.86	16.0
	BPTM 0.13	109.53	15.12	1.99

Table 4.4: Improvement of gate performance with respect to process (normalized to TSMC 0.25 μm device).

Gate		Improvement		
		Delay	Current	PDP
CMOS	TSMC 0.25	1.00	1.00	1.00
	TSMC 0.18	1.58	2.50	5.49
	BPTM 0.13	2.43	5.86	29.68
Pseudo-NMOS	TSMC 0.25	1.00	1.00	1.00
	TSMC 0.18	1.84	1.14	2.91
	BPTM 0.13	2.65	2.60	14.38
Dynamic	TSMC 0.25	1.00	1.00	1.00
	TSMC 0.18	1.47	2.39	4.87
	BPTM 0.13	2.54	6.16	32.64
DCVS-PG	TSMC 0.25	1.00	1.00	1.00
	TSMC 0.18	1.52	2.69	5.67
	BPTM 0.13	2.57	8.51	45.61

4.3 Ultra-Low Power/Energy Design

The aforementioned gate designs have been optimized for high-speed operation, attaining speeds of up to 18 GHz per gate. But as explained in Chapter 1, the most popular DDFS applications prioritize very low power consumption and energy, due to the reduced size (or total elimination) of power supplies, and the requirement for reduced heat dissipation. Many of these applications, such as RFID tags and smart dust sensors, require a very low output frequency (kHz to MHz), negating the requirement for attaining a maximum clock frequency. While process scaling appears effective in reducing the PDP, by decreasing C_L and $V_{DD,}$ this method may be impractical, and even unnecessary for ultra-low power applications. For example, an 8-bit embedded microprocessor clocked at 4 MHz benefits from the reduced power consumption when using a smaller process, but reaps no speed benefits. Then arises the issue of affordability, as the overhead and non-recurring engineering (NRE) monetary costs for using a smaller CMOS process are very expensive. Many embedded devices, especially consumable ones, must be extremely affordable (sometimes under $1 USD). But most important is the issue of leakage currents, as $P_{Leakage}$ is expected to have a greater effect on P_{Total} as the channel lengths and threshold voltages decrease as processes become smaller.

Instead of migrating to a smaller CMOS process in an effort to reduce the PDP, it would be wise to consider if the power consumption of a relatively large and cheaper process could be significantly reduced [33], [39] – [42]. Referring back to equation 4.3, lowering V_{DD} quadratically decreases $P_{Dynamic}$. A decrease in V_{DD} also linearly decreases P_{Static} and $P_{Leakage}$, should $I_{Leakage}$ remain constant. Hypothetically speaking, what if the V_{DD} of a mature CMOS process were lowered to reduce both $P_{Dynamic}$ and P_{Static}? Because this process is mature, the V_{TH} is higher than a smaller process, meaning that $I_{Leakage}$ will be smaller. The resultant V_{DD} decrease also significantly decreases f, preventing high-speed operation, but it should be possible to select a V_{DD} that still allows the low minimum clock frequencies required by these ultra-low power applications. This decrease in f then furthermore decreases $P_{Dynamic}$.

Design Procedure

Reducing V_{DD} is not as trivial as applying a lower supply voltage to an existing circuit. Because the gates were designed using a normal V_{DD}, the transistors must be sized accordingly to prevent incorrect logic outputs. To obtain the optimal V_{DD} and transistor sizes, a methodical procedure has been developed, and includes analyzing the effects of process scaling.

The first step is to explore the basic speed and power effects when using a lower V_{DD}. The 0.25 μm CMOS process is characterized with a five-node ring oscillator, shown in Figure 4.6. The NMOS transistors of the inverters are sized to 6λ, keeping consistent with the previous gate simulations. The delay (the length of one period) and power consumption of the ring oscillator are measured for various values of the supply voltage, starting with one-half the normal process V_{DD}, 2.5 V, and decreasing it down to 1/20th. Figure 4.7 displays the average current and delay for these V_{DD} values. It is noticed that the current significantly decreases when V_{DD} is slightly greater than 500 mV, which happens to be V_{TH}. The delay significantly increases when V_{DD} is slightly below V_{TH}. Figure 4.8 provides a closer view of this V_{DD} range, displaying the significant non-linear current increase when $V_{DD} > 500$ mV, and the significant non-linear delay increase when $V_{DD} < 400$ mV. This same phenomena also occurs for the 0.18 μm and 0.13 μm CMOS processes, although with different values of V_{DD}. Based on these results, it seems desirable to select a V_{DD} between 400 and 500 mV such that minimal power consumption is obtained, but without an extremely long delay. Since the range of V_{DD} is less than V_{TH}, the NMOS of the inverter always operates in the cutoff region, and the PMOS always operates in triode. During a logic 1 input, although current is flowing through the PMOS, most of the current "leaks" through the NMOS, as the gate voltage is relatively high, causing the inverter to register a valid logic 0 output. While optimal PDP values may be obtained with any V_{DD} in the range of 400 – 500 mV, V_{DD} should be set halfway at 450 mV, because noise margins significantly decrease when operating at sub-threshold voltages. If V_{DD} were somehow "pushed" slightly out of the 400 – 500 mV range, the delay or power consumption could significantly increase. For the 0.18 and 0.13 μm processes, the V_{DD} has been set to 450 and 300 mV, respectively.

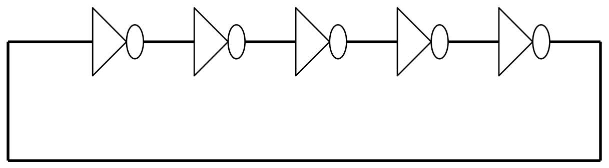
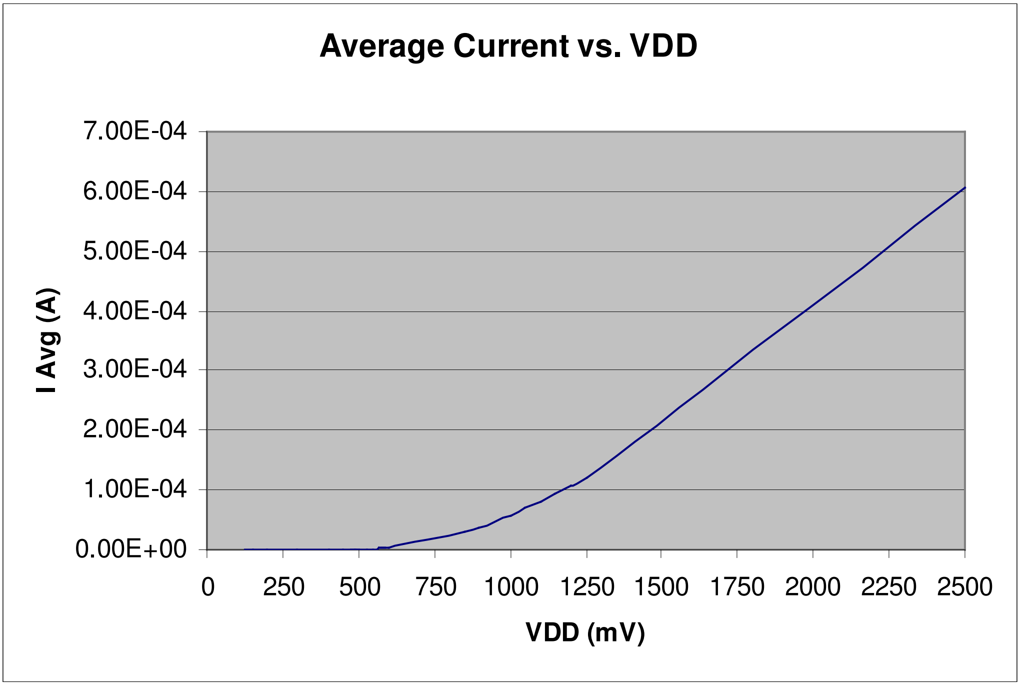


Figure 4.6: A five-node ring oscillator.



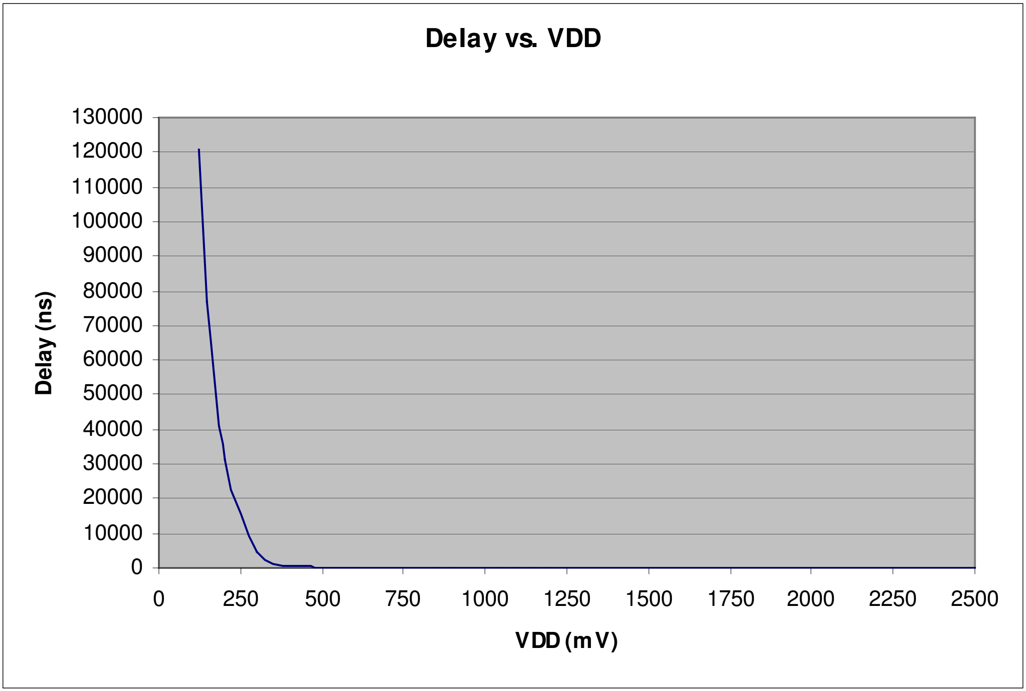
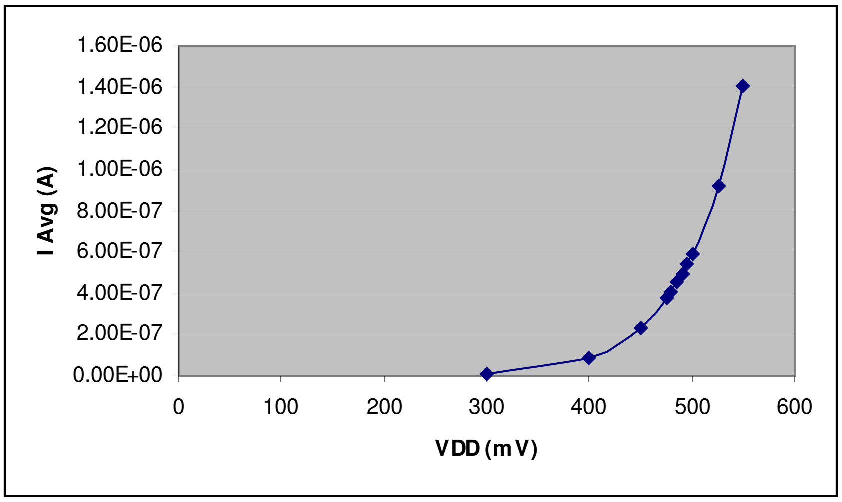


Figure 4.7: Plots of the current (top) and delay (bottom) of a 0.25 μm ring oscillator for several values of V_{DD}.



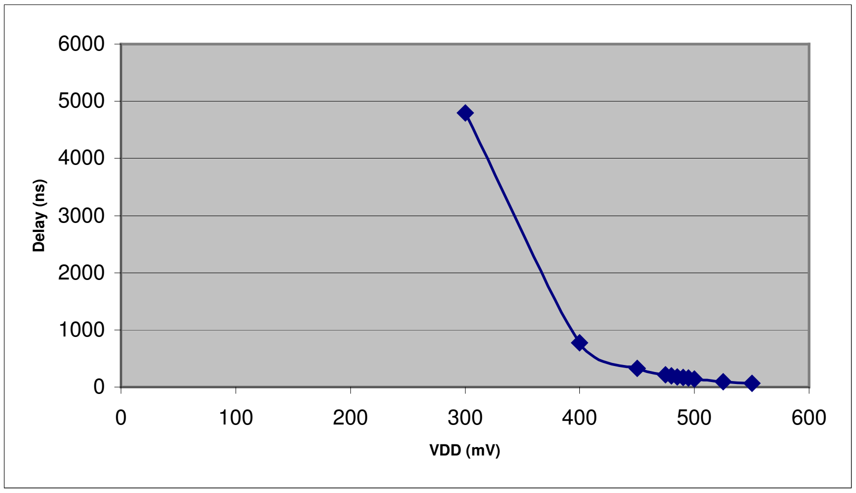


Figure 4.8: Closer view of the current (top) and delay (bottom) plots for a 0.25 µm ring oscillator.

Now that a V_{DD} has been obtained, the PMOS transistor must be sized accordingly, such that the rise and fall times of the ring oscillator period are equal. For ultra-low power applications that require a shorter delay, pseudo-NMOS style logic is to be considered. A separate ring oscillator is constructed such that the gates of all the PMOS transistors are hard-wired to GND. The width of the PMOS must be selected such that $V_{out} \leq 10\%\ V_{DD}$. The results of the simulations with both types of ring oscillators for all three CMOS processes are shown in Table 4.5. The values of the delay, current, and PDP are for one-single inverter, or one-fifth of the total oscillator values. Of course, the BPTM 0.13 μm oscillators provide the lowest PDP, but the difference between the larger 0.25 μm oscillators is only four-fold.

Table 4.5: Simulation results of the two types of ring oscillators for the three CMOS processes using the selected sub-threshold supply voltages.

Process	Inverter Type	V_{DD} (mV)	Delay (ns)	Current (nA)	PDP (fJ)
TSMC 0.25 μm	CMOS	450	59.38	57.20	1.528
	Pseudo NMOS	450	36.60	96.00	1.581
TSMC 0.18 μm	CMOS	450	35.34	54.12	0.8607
	Pseudo NMOS	450	15.10	137.68	0.9355
BPTM 0.13 μm	CMOS	300	34.40	40.86	0.4217
	Pseudo NMOS	300	18.80	77.68	0.4381

Once the widths of the PMOS transistors have been obtained, the logic gate transistors may be sized accordingly. The P transistors of the CMOS logic gate are sized such that the equivalent PMOS width equals the width of the ring oscillator PMOS. For the pseudo-NMOS gate, the actual PMOS width matches the pseudo-NMOS ring oscillator. Since the dynamic and DCVS-PG gates could not be characterized with a ring oscillator, the same widths from the normal supply voltage simulations are used. However, the clock buffers of the dynamic gate are dimensioned proportionally to the low-voltage CMOS ring oscillators. The gates are subjected to the exact same test cases as their normal V_{DD} counterparts, except the input combinations and clock have a 2 μs period.

Simulation Results

Table 4.6 displays the simulation results for these logic gates. Table 4.7 displays how the performance of each gate improves as the process becomes smaller. Obviously, the 0.13 μm process always provides the optimal performance. While the overall current decreases as the processes become smaller, this decrease is not as dramatic as with the normal V_{DD} simulations.

Table 4.6: Simulation results of the four logic gates for all three CMOS processes, using sub-threshold supply voltages.

Gate	Process	Delay (ns)	Current (nA)	PDP (J)
CMOS	TSMC 0.25	61.99	29.43	8.21E-16
	TSMC 0.18	73.56	16.69	5.52E-16
	BPTM 0.13	19.6	8.84	5.20E-17
Pseudo-NMOS	TSMC 0.25	52.38	86.25	2.03E-15
	TSMC 0.18	46.26	65.38	1.36E-15
	BPTM 0.13	15.43	63.45	2.94E-16
Dynamic	TSMC 0.25	55.9	14.95	3.76E-16
	TSMC 0.18	45.9	10.46	2.16E-16
	BPTM 0.13	11.96	4.63	1.66E-17
DCVS-PG	TSMC 0.25	181.61	24.34	1.99E-15
	TSMC 0.18	84.77	11.48	4.38E-16
	BPTM 0.13	24.93	3.66	2.74E-17

Table 4.7: Improvement of gate performance with respect to gate type (normalized by 0.25 μm device).

Gate	Process	Delay	Current	PDP
CMOS	TSMC 0.25	1.0000	1.0000	1.0000
	TSMC 0.18	0.8427	1.7633	1.4860
	BPTM .13	3.1628	3.3292	15.7941
Pseudo-NMOS	TSMC 0.25	1.0000	1.0000	1.0000
	TSMC 0.18	1.1323	1.3192	1.4937
	BPTM 0.13	3.3947	1.3593	6.9218
Dynamic	TSMC 0.25	1.0000	1.0000	1.0000
	TSMC 0.18	1.2179	1.4293	1.7406
	BPTM 0.13	4.6739	3.2289	22.6377
DCVS-PG	TSMC 0.25	1.0000	1.0000	1.0000
	TSMC 0.18	2.1424	2.1202	4.5423
	BPTM 0.13	7.2848	6.6503	72.6688

Like the normal V_{DD} simulations, the dynamic gate provides the lowest PDP for every process. Since the clock buffers drive ideal interconnects, the potentially high power consumption of the clock network is not included in the results. As the propagation delay is relatively long, clock skew is not as significant a problem as it is for a very high-speed circuit. However, since the clock operates at a lower frequency, the issue of charge leakage is introduced. During a logic 1 output in the evaluate phase, the load and parasitic capacitances must retain the charge for a longer duration. Because of the lower V_{DD}, the maximum amount of charge, Q, is smaller than Q for a normal V_{DD}, since $Q = C_L V_{DD}$. With the increased clock period, and the reduced Q, C_L has more time to discharge to voltages that may yield a logic 0. The charge leakage problem may be resolved by either using larger transistors to increase C_L or by pipelining to shorten the clock period. Both methods come at the expense of increased power consumption, and potentially a longer delay if the transistor size is increased. Resultantly, the PDP of the overall circuit may be greater than one utilizing asynchronous gates.

While the pseudo-NMOS gate has the highest PDP, it does not suffer from the charge leakage problems of the dynamic gate, and provides the shortest delay of any of the asynchronous gates. However, the noise margins of this gate are the smallest of any of the gates, since G does not swing to GND during a logic 0 output. The noise immunity is further reduced due to the low V_{DD}. Both the CMOS and DCVS-PG gate provide rail-to-rail voltage swings, with the latter offering a lower PDP for the 0.18 and 0.13 μm processes. While the differential structure provides common mode noise rejection, the voltage degradation caused by extensive use of pass-transistors in a very low voltage environment may produce incorrect outputs. The CMOS gate offers the best noise margins of any of the gates, but has a mid-range PDP.

Choosing the proper gate solely depends on the target application requirements. Due to the charge leakage issues associated with clocking a dynamic gate at a low frequency, and the added power consumption for countering these issues, such a gate should be avoided in ultra-low power applications. When high-speed operation is desired, a pseudo-NMOS offers the best performance, at the expense of a higher PDP, but offers the worst noise immunity out of any of the gates. If robust operation is a priority, then the CMOS gate by far offers the best noise immunity and a moderately low PDP, at the expense of a relatively shorter delay. The CMOS gate is probably the most appropriate for use in low-power wireless sensors and embedded systems. The device power supplies, especially MEMS, induction loops, and solar cells, tend to be less reliable, providing noisy and fluctuating voltages. Furthermore, portable wireless devices are operated in uncontrollable, and often hostile, environments that further affect the noise margins.

Another application-influenced decision is whether to use a smaller CMOS process to further decrease the PDP. As shown by tables 4.6 and 4.7, a smaller process provides a smaller PDP. But is the increased monetary expense of migrating to a smaller process worth the decreased PDP? Figure 4.9 displays a comparison of the PDP of a normal V_{DD} gate (blue) to its low-voltage counterpart (crimson). Table 4.8 displays the total PDP improvement of a gate from its 0.25 μm implementation to its low-voltage 0.13 μm equivalent. For the CMOS gate, the PDP is 1600 times smaller! If this 1600-fold improvement is most imperative, then using a smaller process is worth the expense. The most relevant applications when very minimal energy usage is desired are space satellites and non-recoverable sensors, as the power supplies cannot be recharged/replaced following deployment. These applications inherently have a high budget and can afford using a bleeding-edge CMOS process. However, is this high cost justifiable for low-cost portable applications where the battery is replaceable? Table 4.9 explains the overall PDP improvement by listing the low-voltage gates and how their PDP improves: 1) With respect to the corresponding normal V_{DD} device and 2) With respect to the corresponding low-voltage 0.25 μm gate. The PDP of 0.13 μm CMOS gate improves approximately 16-fold over the 0.25 μm device. Yet, the PDP of the low-voltage 0.25 μm gate improves 101-fold over the 2.5 V device. Therefore, out of the total 1600-fold PDP decrease of the CMOS gate, the most profound effect comes from lowering V_{DD} to 450 mV within the 0.25 μm process. Even with the two smaller processes, the most significant decrease to the PDP occurs simply by lowering the V_{DD} within that process. Therefore, if increased battery life is important, but also the product cost, then lowering the supply voltage of a larger and cheaper CMOS process is appropriate.

Table 4.8: Overall PDP improvement over the normal V_{DD} 0.25 μm gates.

Gate	Overall PDP Improvement
CMOS	1602.94
Pseudo-NMOS	222.92
Dynamic	2487.74
DCVS-PG	3311.42

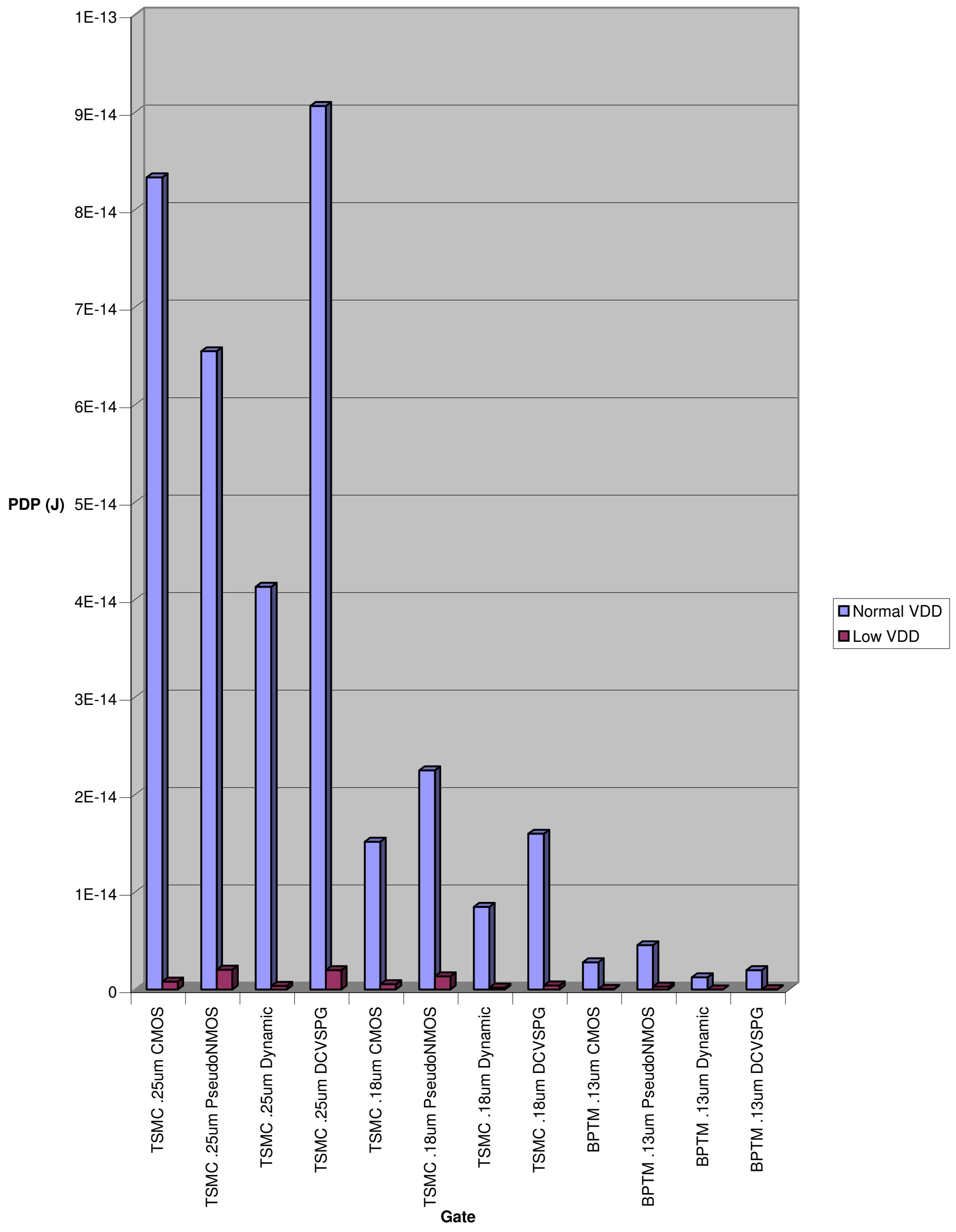


Figure 4.9: Comparison of the PDP of a low-voltage gate to its normal V_{DD} counterpart.

Table 4.9: Improvement of the PDP for low-voltage gates.

Low-voltage Gate	PDP Improvement	
	Normalized to normal V_{DD} gate of the same process	Normalized to the low-voltage 0.25 µm gate
0.25 µm CMOS	101.49	1
0.25 µm Pseudo-NMOS	32.21	1
0.25 µm Dynamic	109.89	1
0.25 µm DCVS-PG	45.57	1
0.18 µm CMOS	27.45	1.49
0.18 µm Pseudo-NMOS	16.52	1.49
0.18 µm Dynamic	39.25	1.74
0.18 µm DCVS-PG	36.51	4.54
0.13 µm CMOS	54.00	15.79
0.13 µm Pseudo-NMOS	15.50	6.92
0.13 µm Dynamic	76.23	22.64
0.13 µm DCVS-PG	72.60	72.67

Future Work

Before physically implementing these ultra-low power/energy gates, additional simulations must be performed. Extremely important is how the performance of these gates is affected by temperature and process variations. Because V_{DD} is extremely low, it is expected that these gates will fare worse than their normal V_{DD} counterparts, operating slower than reported and potentially producing incorrect logic outputs. Several solutions have been proposed to counter these problems. One such method is adaptive body biasing (ABB) where the substrate of the P transistors is connected not to V_{DD} but to a higher voltage, and the substrate of the N transistors is connected to a voltage below GND, effectively reducing V_{TH} [41], [43] – [44]. These voltages are then controlled by a closed-loop adaptive biasing circuit, adjusting them based on the current temperature, operating frequency, and V_{DD}. To maintain precise control, several ABB circuits may be required on the die, occupying valuable on-chip area. External capacitors may also be required, further increasing the gate delay. An alternative substrate-biasing method is sub-dynamic threshold CMOS (sub-DTMOS), where the bulk of each transistor connects not to V_{DD} or GND, but to the gate of that transistor [41], [45] – [46]. With this configuration, the V_{TH} is increased when the transistor is off, minimizing $I_{Leakage}$, and when the transistor is off, V_{TH} is lowered, increasing the drain-to-source conductance. Such a design exhibits very low power dissipation due to the decreased leakage currents, but higher speed than a traditionally biased low-voltage circuit, thanks to the increased conductance. It also happens that the PDP is actually lower at higher temperatures. The most significant drawback of sub-DTMOS is that the transistors must be constructed on an expensive silicon-on-insulator (SOI), multiple-well, or even custom DTMOS process [45], whereas ABB requires only two-distinct wells and may be implemented on a standard less-expensive process. Exploring, and subsequently minimizing the effects of process variations is highly important, as the DDFS most likely will not be operated in controlled environments.

CHAPTER 5

CONCLUSIONS

A three-level abstraction approach has been presented to suggest methods for optimizing the DDFS for use in portable low-power wireless communication systems. The most popular applications do not require a high system clock frequency, but demand high accuracy and ultra-low power consumption, and consequently a lower PDP. Reduced energy usage becomes imperative as the power supplies become smaller (due to the demand for a smaller device footprint), or use alternate low-energy sources, such as MEMS and solar cells.

The overall accuracy is solely based upon the phase-to-sine approximation circuit. The traditional implementations utilize a large ROM lookup table, which cannot practically provide a high-SFDR and low MAE due to the large amount of ROM required. The approximation of the sine curve through segmented linear interpolation offers very high accuracy with a maximum reported SFDR of 96 dBc (for 64 segments). While these solutions eliminate the need for a multiplier circuit, the required number of multiplexers and the multiple-input adder occupy very significant on-chip area. The segmented parabolic approximation reduces the number of segments by attempting to approximate the MAE of a segmented linear approximation (for the same number of segments) and cancel the result. This solution yields an SFDR of 84 dBc using 16-segments and only 400-bits of ROM, at a ROM complexity cost of approximately 5 bits/dBc. A 16-segment linear approximation yields only a 66 dBc SFDR, and achieving an 84 dBc SFDR requires 32 segments. A complex MAC circuit is required to realize the segmented parabolic approximation, but it is expected that the hardware complexity is lower than the 32-segment linear approximation, which requires six 10 – 16-bit 32-to-1 multiplexers, a 6-input 10 – 16-bit adder, and 448 bits of ROM. The speed and hardware complexity of the MAC may be further reduced using Wallace-tree multiplication with 3|2 counters and eliminating the logic gates in the non-critical path (lower 9 sum bits) of the 21-bit adder.

The most widely used arithmetic component in the DDFS is the adder, as three are required by the segmented parabolic implementation. While parallel-prefix adders consume more power than serial adders, the propagation delay is shorter, as it is logarithmically bounded, rather than linearly. A Han-Carlson adder offers the best compromise between speed and power, and a Kogge-Stone is most appropriate when a minimal delay is required. When using a PPA in the phase accumulator (and even MAC), the hardware complexity may be reduced since only certain upper sum bits must arrive at a high speed.

If achieving a high system clock frequency is imperative (in order to achieve a high f_{out}), then super-pipelining the DDFS should be considered. The phase accumulator may be modeled mathematically to determine the optimal number of pipeline stages. When minimal power consumption is imperative, the PA should not be pipelined, but if a compromise between speed and power is required, then pipelining the PA into $N/2$ stages provides the best results. The approximation circuit must also be pipelined to attain the maximum clock frequency. There are no data hazards associated with pipelining the DDFS, but the total system propagation delay is increased, reducing the overall frequency switching speed. Low-power communication systems require a high switching speed and low power consumption, and therefore, pipelining should be deployed conservatively.

Finally, the speed, power, and energy are affected by the logic gates that realize the adders. Among the four gate types presented, the dynamic gate provides the lowest PDP (having minimal delay and power consumption) but suffers from problems induced by clock skew (which becomes more problematic for very high clock frequencies), and may actually have a higher power consumption than any of the asynchronous gates if the effects of the clock network are included. The pseudo-NMOS gate provides the shortest delay, but with the highest power consumption and reduced noise margins. The effects of process scaling have been investigated, and while improved delay and power consumption are offered with smaller processes, the additional monetary expense may not be worth the additional performance gains, especially for inherently low-speed applications. Even with a large and mature CMOS process, reducing the supply voltage to sub-threshold levels decreases the PDP 100-fold. For high-budget applications where ultra-low energy dissipation is extremely imperative, lowering V_{DD} and migrating to a smaller CMOS process causes a 1000-fold PDP decrease (within relation to a large process, normal V_{DD} gate). Based on the four low-voltage gates, the CMOS gate offers not only a relatively small PDP, but is theoretically the most robust, which is important for portable applications. The dynamic gate should be avoided due to the increased potential for charge leakage because of the low-clock frequency. Additional simulations and design modifications must be performed to investigate the effects of process, temperature, and power supply variations.

BIBLIOGRAPHY

[1] B.P. Lathi, *Modern Digital and Analog Communication Systems*, Oxford University Press, 1998, Chp. 9, pp. 406–416.

[2] *IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications,* IEEE Standard 802.11b, 1999.

[3] B. Miller and C. Bisdikian, *Bluetooth Revealed. The Insider's Guide to an Open Specification for Global Wireless Communications*, Prentice Hall PTR, 2002, Chp. 6, pp. 85–93.

[4] D. Johns and K. Martin, *Analog Integrated Circuit Design*, John Wiley & Sons, 1997, Chp. 16, pages 648–693.

[5] J. Tierney, C.M. Rader, and B. Gold, "A digital frequency synthesizer," *IEEE Transactions on Audio Electroacoustics*., vol. 19, January 1971, pages 48–57.

[6] K. Palomäki, "A digital sinusoidal synthesizer based on feedback," MSc thesis, Department of Information Technology, Tampere University of Technology, Tampere, Finland, Nov. 1999.

[7] D. Betowski and V. Beiu, "Considerations for phase accumulator design for direct digital frequency synthesizers," *Proc. Intl. Conf. Neural Networks & Signal Processing (ICNNSP'03)*, 2003, pp. 176–179.

[8] P.-S. Wu"Towards ROM-less DDFSs: On digital circuits for accurate sine approximation," MSc thesis, School of EE&CS, Washington State Univ., Pullman, WA, Aug. 2003.

[9] J. R. Stuart and J. G. Stuart, "Revolutionary next generation satellite communications architectures and systems," *Proc. IEEE Aerospace Conf.,* 1997, pp. 535–545.

[10] S. Janson, H. Helvajian, S. Amimoto, G. Smit, D. Mayer, and S. Feuerstein, "Mircotechnology for space systems," *Proc. IEEE Aerospace Conf.*, 1998, pp. 409–418.

[11] B. Atwood, B. Warneke, and K. Pister, "Preliminary circuits for smart dust," *Proc. Southwest Symp. Mixed-Signal Design*, Feb. 2000, pp. 87–92.

[12] B. Warneke and K. Pister, "An ultra-low energy microcontroller for smart dust wireless sensor networks*," Proc. Int. Solid-Sate Circ. Conf. (ISSCC'04)*, Feb. 2004.

[13] J. R. Tuttle, “Traditional and emerging technologies and applications in the radio frequency identification (RFID) industry,” *Proc. IEEE Radio Freq. IC Symp.*, Jun. 1997, pp. 5–8.

[14] M. May, “Keeping a closer eye on athletes. New devices help track winners and losers at games,” *San Francisco Chronicle*, Aug. 23, 2004, Available: http://www.sfgate.com/cgi-bin/article.cgi?file=/chronicle/archive/2004/08/23/BUGCL8BT4C1.DTL

[15] S. C. Q. Chen and V. Thomas, “Optimization of inductive RFID technology,” *Proc. 2001 IEEE Int. Symp. Electronics and Environment*, May 2001, pp. 82–87.

[16] J. Vanka and K. Halonen, *Direct Digital Synthesizers – Theory, Design, and Applications*, Kluwer Academic Publishers, 2001.

[17] D. A. Sunderland, R. A. Strauch, S. S. Wharfield, H. T. Preston, and C. R. Cole, “CMOS/SOS Frequency Synthesizer LSI Circuit for Spread Spectrum Communications,” *IEEE J. Solid-State Circ.,* vol. 19, no. 4, Aug. 1984, pp. 497-505.

[18] A. M. Sodagar and G. R. Lahiji, “Parabolic approximation: A new method for phase-to-amplitude conversion in sine-output direct digital frequency synthesizers,” *Proc. Int. Symp. Circ. & Sys. (ISCAS’00)*, 2000, vol. 1, pp. 515–518.

[19] H. T. Nicholas, III, H. Samueli, and B. Kim, “The optimization of direct digital frequency synthesizer performance in the presence of finite word length effects,” *Proc. Intl. Freq. Control Symp*. FCS’88, 1988, pp. 357–363.

[20] A. Yamagishi, M. Ishikawa, T. Tsukahara, A. Date, “A 2-V, 2-GHz low-power direct digital frequency synthesizer chip-set for wireless communication,” *IEEE J. Solid-State Circ.*, vol. 33, 1998, pp. 210–217.

[21] J.M.P. Langlois, and D. Al-Khalili, “ROM size reduction with low processing cost for direct digital frequency synthesis,” *Proc. PACRIM’01*, Aug. 2001, vol. 1, pp. 287–290.

[22] R. C. Meitzler and W. P. Millard, “A direct digital frequency synthesizer prototype for space applications,” *Proc. NASA Symp. VLSI Design*, 2003. Available: http://www.cambr.uidaho.edu/symposiums/symp11/R._Meitzler_dds_nasa.pdf

[23] P.-S. Wu and V. Beiu, “On accurate piecewise linear ROM-less direct digital frequency synthesizers,” *Proc. Intl. Conf. Neural Networks & Signal Proc. (ICNNSP’03)*, 2003, vol. 2, pp. 1595–1599.

[24] J. M. P. Langlois and D. Al-Khalili, “Hardware optimized direct digital frequency synthesizer architecture with 60-dBc spectral purity,” *Proc. Int. Symp. Circ. & Sys. (ISCAS’02)*, 2002, pp. 361–364.

[25] J. M. P. Langlois and D. Al-Khalili, “Novel approach to the design of direct digital frequency synthesizers based on linear interpolation,” *IEEE Trans. Circ. & Sys. II*, vol. 50, Sep. 2003, pp. 567–578.

[26] D. Betowski, D. Dwyer, and V. Beiu, “A Novel Parabolic Sine Approximation for Direct Digital Frequency Synthesizers,” *Proc. 2004 Int. Conf. VLSI (VLSI’04)*, Jun. 2004, pp. 523–529.

[27] J. Rabaey, A. Chandrakasan, and B. Nikolić, *Digital Integrated Circuits. A Design Perspective (2nd edition)*, Prentice Hall, 2003, Chp. 6, pp. 235–308.

[28] C. S. Wallace, “A suggestion for a fast multiplier,” *IEEE Trans. Electr. Comp.*, vol. EC-13, Feb. 1964, pp. 14–17.

[29] R. Zimmerman, “Binary adder architectures for cell-based VLSI and their synthesis,” PhD thesis, Swiss Federal Institute of Tech., Zurich, Switzerland, 1997.

[30] V. Beiu, “Ultra-fast noise immune CMOS threshold logic gates,” *Proc. MWSCAS’00*, USA, 2000, pp. 1310-1313.

[31] V. Beiu, J. M. Quintana, and M. J. Avedillo, “VLSI implementations of threshold logic: A comprehensive survey,” *IEEE Trans. Neural Networks*, vol. 14, Sep. 2003, pp. 1217–1243.

[32] V. Oklobdzija, V. Stojanovic, D. Markovic, N. Nedovic, *Digital System Clocking: High Performance & Low-Power Aspects*, Wiley-IEEE Press, 2003.

[33] R. Gonzalez, B. Gorgon, and M. Horowitz, “Supply and threshold voltage scaling for low power CMOS,” *IEEE J. Solid-State Circ.*, vol. 32, Aug. 1997, pp. 1210–1216.

[34] R. F. Tinder, *Engineering Digital Design (Revised 2nd edition)*, Academic Press, 2000, Chp. 11, pp. 517–520.

[35] K. Chu and D. Pulfrey, “Design procedures for differential cascode voltage switch circuits,” *IEEE J. Solid-State Circ.*, vol. Sc-21, Dec. 1986, pp. 1082–1087.

[36] F. Lai and W. Hwang, “Design and implementation of differential cascode voltage switch with pass-gate (DCVSPG) logic for high-performance digital systems,” *IEEE J. Solid-State Circ.*, vol. 32, Apr. 1997, pp. 563–573.

[37] *International Technology Roadmap for Semiconductors 2003 Edition*, 2003, Available: http://public.itrs.net/Files/2003ITRS/Home2003.htm

[38] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," *Proc. IEEE CICC*, Jun. 2000, pp. 201–204.

[39] H. Soeleman and K. Roy, “Ultra-low power digital subthreshold logic circuits,” *Proc. Int. Symp. Low Power Electron. Design*, 1999, pp. 94–96.

[40] V. Svilan, M. Matsui, and J. Burr, “Energy-efficient 32×32-bit multiplier in tunable near-zero threshold CMOS,” *Proc. Int. Symp. Low Power Electronics & Design (ISLPED’00)*, Jul. 2000, pp. 268–272.

[41] H. Soeleman, K. Roy, and B. C. Paul, “Robust subthreshold logic for ultra-low power operation,” *IEEE Trans. VLSI Sys.*, vol. 9, Feb. 2001, pp. 90–99.

[42] A. Wang and A. Chandrakasan, “Optimal supply and threshold scaling for subthreshold CMOS circuits,” *Proc. IEEE Comp. Sci. Annual Sym. VLSI (ISVLSI’02)*, 2002.

[43] T. Kobayashi and T. Sakurai, “Self-adjusting threshold-voltage scheme (SATS) for low-voltage high-speed operation,” *Proc. IEEE Custom IC Conf.*, 1994, pp. 271–274.

[44] J. T. Kao, M. Miyazaki, and A. Chandrakasan, “A 175 mV multiply-accumulate unit using an adaptive supply voltage and body bias architecture,” *IEEE J. Solid-State Circ.*, vol. 37, Nov. 2002, pp. 1545–1554.

[45] F. de la Hidalga-W. and M. Deen, “The dynamic threshold voltage MOSFET,” *Proc. 2nd. IEEE Int. Caracas Conf. Devices, Circ., and Sys.*, Mar. 2000, pp. D63-1–D63-7.

[46] A. Drake, K. Nowka, R. Brown, “Evaluation of dynamic-threshold logic for low-power VLSI design in 0.13 μm PD-SOI,” *Proc. IFIP Int. Conf. on VLSI (VLSI-SoC’03)*, Dec. 2003.