UNIVERSITY OF CALIFORNIA

Los Angeles

An Improved Quadrature Direct Digital Frequency Synthesizer in an FPGA

A thesis submitted in partial satisfaction

of the requirements for the degree Master of Science

in Electrical Engineering

by

Matthew Ryan Bergeron

2013

ABSTRACT OF THE THESIS

An Improved Quadrature Direct Digital Frequency Synthesizer in an FPGA

by

Matthew Ryan Bergeron

Master of Science in Electrical Engineering

University of California, Los Angeles, 2013

Professor Alan N. Willson, Jr., Chair

The architecture and design of a high-speed quadrature direct digital frequency synthesizer (DDFS) is presented. The architecture is based on a novel multiplier-based angle-rotation algorithm that does not distort the magnitude of the sine and cosine outputs. This algorithm maps well into the DSP slices present in modern FPGAs. The design has a 32-bit frequency control word, 16-bit outputs, and a tuning resolution of 0.23 Hz at 1 GHz. Implemented in a Xilinx Virtex-7 FPGA, the design dissipates 54.9 mW of power, a performance previously attainable only in ASIC designs.

The thesis of Matthew Ryan Bergeron is approved.

Babak Daneshrad

Rajeev Jain

Alan N. Willson, Jr., Committee Chair

University of California, Los Angeles

2013

To my children

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank Professor Alan N. Willson, Jr. for giving me the opportunity to do this research and thesis. Without his support, guidance, and patience none of this would be possible.

Thanks to Art Torosyan for discussion and guidance on general DDFS topics. I am also appreciative of Sean Huang and Yen-Liang Shue, with whom I collaborated in 2003 while working on my first prototype DDFS IC design.

Thanks to my mom for continuing to push me to complete this work. Finally, thanks to my wife Pauline and my children, Allison and James, who are my chief inspiration.

Chapter 1

Introduction

Digital communications are an essential part of modern life. Since their inception, direct digital frequency synthesizers (DDFS) have been used in digital communication systems. DDFS have several characteristics that make them advantageous over analog PLLs, including fast continuous-phase frequency switching, fine frequency resolution, the ability to operate over a large frequency range, and high spectral purity.

Most DDFS are based on a structure first proposed by Tierney, Rader, and Gold [1] in 1971, as illustrated in Fig. 1.1. This structure consists of two primary components: a phase accumulator and a sine/cosine mapping function (SCMF). The phase accumulator consists of an overflowing M-bit adder and register. The frequency control word (f_{cw}) controls the rate of overflow and hence the output frequency. The M-bit output of the phase accumulator is truncated to W bits to generate ϕ, which is the input to the SCMF. Early designs of the SCMF consisted entirely of lookup tables to map ϕ to the outputs X and Y, where $X = \cos \pi\phi$ and $Y = \sin \pi\phi$. The lookup table size scales exponentially with

M (and with the desired spectral purity), therefore much research has been done over the last 40 years to reduce the size of the lookup table.

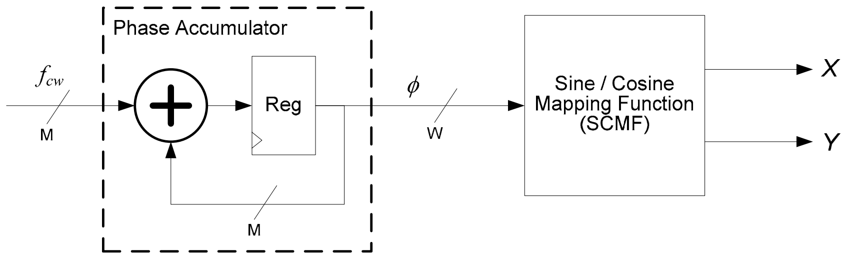


Fig. 1.1. General structure of a DDFS.

1.1 Motivation

FPGAs are becoming increasingly popular for the implementation of digital circuits, despite the significant clock speed degradation and power overhead vs. an ASIC design in the same process. Due to cost prohibitive NRE, many companies and research institutions design ASICs in a process several generations old—making the case for FPGAs more compelling.

With the advent of smart phones and tablets, an increasing amount of digital communication is transitioning from wireline to wireless. The myriad of constantly evolving wireless standards are pushing wireless base stations from fixed implementations to re-programmable ones—a technique often referred to as software defined radio. One method of implementing a re-programmable base station is to use FPGAs (rather than fixed ASICs) for some or all parts of the design. The usage of FPGAs enables field re-programmability on existing deployed hardware as wireless standards evolve.

The increasing amount of wireless data leads to a demand for more wireless capacity at each cell site, and the increased capacity drives the need for smaller and more power efficient base stations. Despite the usage of DDFS as part of the digital radio in wireless base stations, little work has been done to optimize the implementation and power of DDFS in FPGAs. Such digital radio implementations tend to use FPGA vendor provided DDFS cores, which are implemented with large lookup tables in power hungry RAM blocks and wide complex multipliers in order to achieve sufficient spectral purity.

In this thesis we present an optimal implementation of a DDFS in an FPGA, with the primary design goals being high data rates and reduced power consumption.

1.2 Thesis Overview

Chapter 2 of this thesis describes the architecture of the presented DDFS, including the angle rotation algorithm. In Chapter 3, a block diagram is presented and the detailed circuit implementation of each functional block is described. This chapter also explains the hardware optimizations and reductions achieved. The results of the implementation are presented in Chapter 4. Finally, Capture 5 summarizes the thesis.

Chapter 2

Architecture

2.1 Coarse-Fine DDFS

Over the years, many variations of DDFS based on a coarse-fine architecture have been proposed. In such an architecture, the phase angle ϕ is decomposed into a coarse angle ϕ_M and a fine angle ϕ_L. This allows the DDFS to be implemented in two stages, with the coarse stage often implemented in a lookup table and the fine stage implemented through angle rotation, linear interpolation, or polynomial interpolation. The Givens rotation allows us to split the DDFS into these two stages:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\phi_L & -\sin\phi_L \\ \sin\phi_L & \cos\phi_L \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \end{bmatrix} \tag{2.1}$$

where (X_M, Y_M) are the cosine and sine lookup table outputs.

The critical component in a coarse-fine DDFS is most often the fine stage. The underlying architecture of the DDFS presented in this thesis is based on a novel angle-rotation algorithm that does not introduce magnitude distortion into the output.

2.2 Angle Rotation Algorithm

Madisetti [2] used concepts from CORDIC [3] to design an angle rotation algorithm that does not distort the magnitude of the phasor defined by the point (*X, Y*) as this point is rotated around the unit circle. With the cosine factored out of (2.1) the rotation is split into multiple sub-rotations:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = K \begin{bmatrix} 1 & -\tan r_2\phi_2 \\ \tan r_2\phi_2 & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & -\tan r_{N+1}\phi_{N+1} \\ \tan r_{N+1}\phi_{N+1} & 1 \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \end{bmatrix} \qquad (2.2)$$

where $K = \cos\phi_2 \cos\phi_3 \ldots \cos\phi_{N+1}$ is a scale factor as a result of the factoring and $r_k \in \{-1, 1\}$. The bits of r are obtained by recoding the b binary bits of ϕ along with an initial fixed rotation ϕ_F:

$$\phi = \sum_{k=1}^{N} b_k 2^{-k} = \phi_F + \sum_{k=2}^{N+1} r_k 2^{-k} . \qquad (2.3)$$

The fixed rotation is given by $\phi_F = 2^{-1} - 2^{-(N+1)}$ and $r_k = 2b_{k-1} - 1$.

For sufficiently large k, $\tan 2^{-k} \approx 2^{-k}$, and for a finite signal word length (*SWL*) they are exactly equal. Madisetti also shows that the sub-rotation stages can be merged for $k \geq (SWL - 1)/2$. Our decomposition of ϕ into ϕ_M and ϕ_L is done such that the merge condition is satisfied, with the MSB of ϕ_L at bit position $2^{-(M+1)}$. As a result, the sub-rotations can be recombined into the following single rotation:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = K \begin{bmatrix} 1 & -\sum_{k=M+2}^{N+1} r_k 2^{-k} \\ \sum_{k=M+2}^{N+1} r_k 2^{-k} & 1 \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \end{bmatrix} \tag{2.4}$$

where $K = \prod_{k=M+2}^{N+1} \cos 2^{-k}$ is the scale factor. The initial fixed rotation now becomes $\phi_F = 2^{-(M+1)} - 2^{-(N+1)}$ and it is incorporated into the coarse stage ROM output (X_M, Y_M).

2.3 Two's Complement Recoding

The summation term in (2.4) represents a binary number with the signed-binary digits r_k. We can recode r_k into the bits of a two's complement binary number $t_k \in \{0, 1\}$ by expressing the values of r_k in terms of the b_k:

$$t = \sum_{k=M+2}^{N+1} r_k 2^{-k}$$

$$t = \sum_{k=M+2}^{N+1} (2b_{k-1} - 1) 2^{-k}$$

$$t = \sum_{k=M+2}^{N+1} b_{k-1} 2^{-(k-1)} - \sum_{k=M+2}^{N+1} 2^{-k}$$

$$t = b_{M+1} 2^{-(M+1)} + \sum_{k=M+2}^{N} b_k 2^{-k} - \left(2^{-(M+1)} - 2^{-(N+1)}\right)$$

$$t = (b_{M+1} - 1) 2^{-(M+1)} + \sum_{k=M+2}^{N} b_k 2^{-k} + 2^{-(N+1)}$$

$$t = -\overline{b_{M+1}} 2^{-(M+1)} + \sum_{k=M+2}^{N} b_k 2^{-k} + 2^{-(N+1)} \tag{2.5}$$

where $\overline{b_k}$ represents the complement of the b_k bit. Then (2.5) can be implemented as a trivial bit manipulation, where t_{M+1} is the sign bit:

$$t_{M+1} = \overline{b_{M+1}} \qquad t_{M+2:N} = b_{M+2:N} \qquad t_{N+1} = 1\,. \tag{2.6}$$

The recoding is illustrated in the following example.

Example 1. Take the 4-bit representation of the angle $\phi = 0.375$ radians and recode. For simplicity, let $M = 0$.

weight	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
b_k	0	1	1	0	
r_k		–1	1	1	–1
t_k	–1	1	1	0	1

$t = –0.09375$ and $\phi_F = 2^{-1} – 2^{-5} = 0.46875$, thus t represents ϕ as expected.

Using the recoding shown in (2.5), and incorporating K into (X_M, Y_M), we can rewrite (2.4) as:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} 1 & -t \\ t & 1 \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \end{bmatrix} \tag{2.7}$$

which reduces our fine rotation stage to a simple multiply-add operation. Since K is a constant that is independent of t, the rotation introduces no magnitude error.

2.4 Coarse Stage

The coarse stage in this DDFS design is implemented by a lookup table. The normalized angle ϕ must be converted to radians for proper operation. Rather than employing a dedicated circuit for the required $\pi/4$ multiplication, this factor is simply incorporated into the lookup table [4]. The incorporation of this π/4 term requires two additional values to be stored in the lookup table. This is because the value of X is calculated according to (2.7) as $X = X_M - tY_M$, where the value t is a function of the fine angle ϕ_L, which also needs to be converted to radians. We must factor π/4 out of t and into Y_R, as shown in (2.8). The scale factor now becomes $K = \prod_{k=M+2}^{N+1} \cos\left(\frac{\pi}{4} 2^{-k}\right)$.

The M MSBs of ϕ compose the coarse angle ϕ_M. The values in the lookup table are given by the following equations:

$$X_M = K\cos\left(\frac{\pi}{4}(\phi_M + \phi_F)\right)$$

$$Y_M = K\sin\left(\frac{\pi}{4}(\phi_M + \phi_F)\right)$$

$$X_R = \frac{\pi}{4} X_M$$

$$Y_R = \frac{\pi}{4} Y_M \,. \tag{2.8}$$

Fortunately, because the MSB of ϕ_L is in the $2^{-(M+1)}$ bit position, the Y_R multiplier term is added only into the LSBs of X. A finite signal word length in the datapath necessitates fewer bits for X_R and Y_R, as the output of the multiply-add is truncated in the fine stage to generate the output. As a result, the lookup table does not double in size due to the omission of the π/4 multiplication circuit as one might suspect.

Chapter 3

Circuit Implementation

3.1 Overview

The block diagram of the designed DDFS is shown in Fig. 3.1. A 32-bit frequency control word f_{cw} controls the rate at which a 32-bit phase accumulator overflows. The output of the phase accumulator ϕ is then truncated to 20 bits. The truncated phase accumulator output represents an angle within the interval [0, 2π). Since sine and cosine exhibit quarter-wave symmetry, the upper two MSBs of ϕ are removed to map the angle into [0, π/2). Additionally, since the values of cosine (or sine) from π/4 to π/2 are the same as mirror-image values of sine (or cosine) from zero to π/4, we remove another MSB and use this bit to conditionally mirror within the first octant of the unit circle.

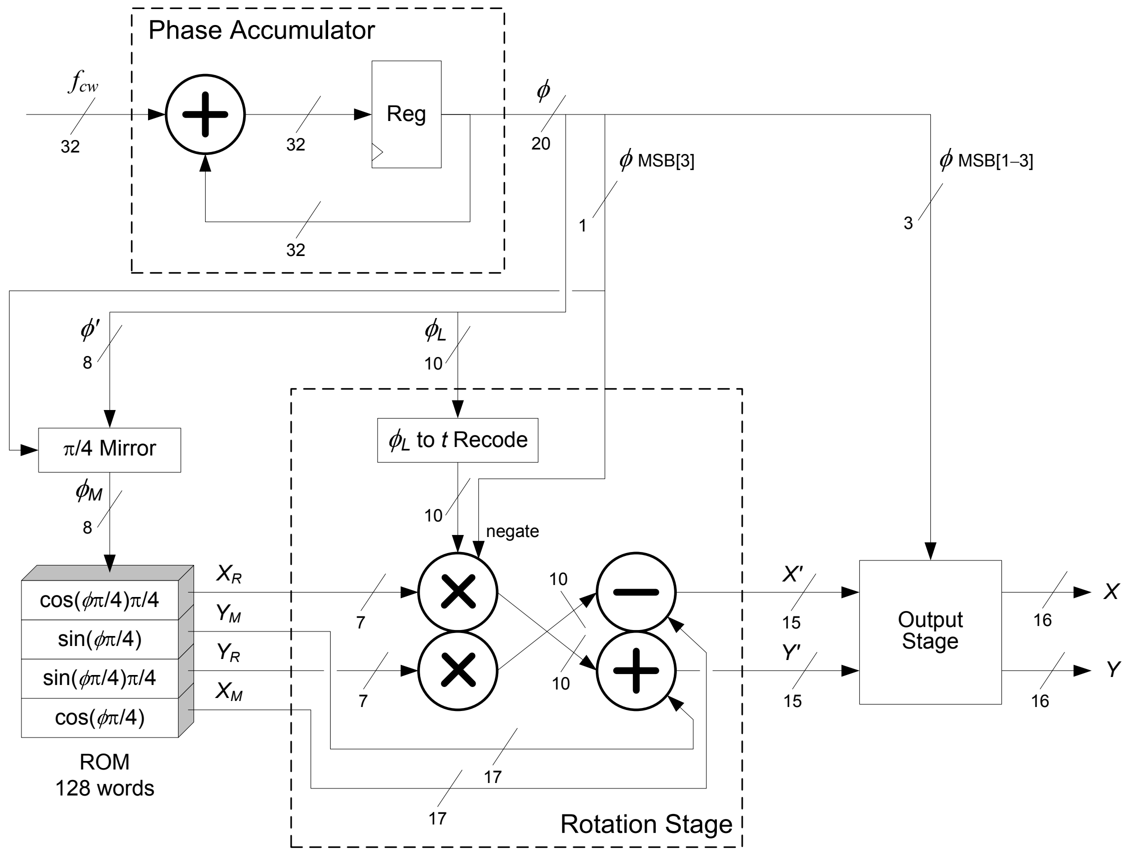


Fig. 3.1. Block diagram of designed DDFS.

The 18-bit mapped, normalized, angle $\phi' \in [0, \pi/2)$, is used to control the datapath. The most significant bit of ϕ' controls the π/4 mirror. The next seven bits of ϕ' generate ϕ_M, which is used to address a 128-word ROM that contains the coarse stage. The least significant ten bits of ϕ' (denoted ϕ_L) control rotation in the fine stage.

The outputs of the fine stage are mapped back to their proper octants through conditional swapping and/or negation, in the output stage, from the first octant of the unit circle. The resulting output of the DDFS is two 16-bit two's complement values $X = \cos \pi\phi$ and $Y = \sin \pi\phi$.

The overall operation of the designed DDFS can be summarized below:

1) Phase Accumulator – generates a normalized phase $\phi \in [0, 2)$.

2) Radian Converter – maps normalized ϕ to ϕ''.

   a) Quadrant mapping – maps ϕ to $\phi' \in [0, 1/2)$.

   b) π/4 Mirror – mirrors the values of ϕ' within [1/4, 1/2) to $\phi'' \in [0, 1/4)$.

3) ROM

   a) Radian converter – converts $\phi'' \in [0, 1/4)$ to $\theta \in [0, \pi/4)$.

   b) Provides the coarse values X_M and Y_M via table lookup.

4) Rotation Stage – calculate sin θ and cos θ.

   a) Recoding – converts the positive value ϕ_L into the signed value t.

   b) Multiplier-Adder – conditional negation – mirrors the values of ϕ' within [1/4, 1/2) to [0, 1/4].

   c) Multiplier-Adder – implements the fine rotation stage.

5) Output Stage – maps sine and cosine $\theta \in [0, \pi/4]$ to [0, 2π).

3.2 Phase Accumulator

The structure of the phase accumulator is shown in Fig. 3.2. The frequency control word f_{cw} adjusts the tuning frequency f_0 of the DDFS. Mathematically $f_0 = \frac{f_{cw}}{2^M} F_{clk}$, where F_{clk} is the clock frequency of the system and M is the number of bits in the phase accumulator. The tuning resolution, defined as the change in output frequency as a result of changing f_{cw} by one, is given by $\frac{F_{clk}}{2^M}$. For the implemented design F_{clk} = 1 GHz and M = 32, yielding a tuning resolution of 0.23 Hz.

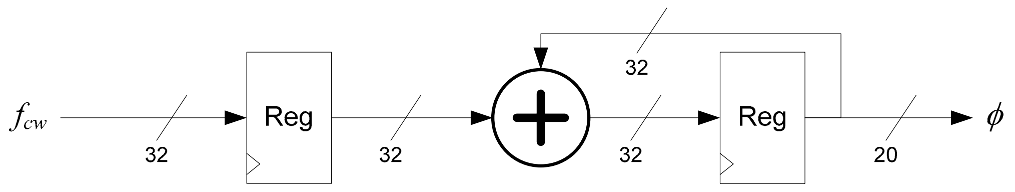


Fig. 3.2. Implementation of phase accumulator.

At every clock cycle the output of the phase accumulator represents the normalized angle $\phi = \frac{f_{cw}}{2^M} n \in$ [0, 2). The output of the phase accumulator is truncated to 20 bits. This choice creates phase truncation spurs that are well matched to the SCMF spurs. See Chapter 4 for more discussion on spurious performance of the DDFS.

Implementation of the registers and the adder is done with a DSP48E1 primitive in the Xilinx Virtex-7 FPGA. This DSP slice is used to reduce the number of CLB (configurable logic block) slices, to reduce loading on the clock network, and to guarantee timing at 1 GHz. f_{cw} is supplied to the C input of the DSP slice and ϕ is taken from the P output. The 32-bit phase accumulator is mapped to bits 47:16 of the DSP in order to minimize power consumption—otherwise the MSBs of the DSP would continue to count on a carry out from the MSB of ϕ.

Parameter	**Value**
USE_MULT	FALSE
AREG	0
BREG	0
CREG	1
PREG	1
ALUMODE	0000
OPMODE	000 11 10

Table 3.1. Phase accumulator DSP parameters.

3.3 Radian Converter

A "radian converter" converts the normalized output of the phase accumulator ϕ into a value ϕ'' in the first octant of the unit circle. To do this mapping, the upper two MSBs of ϕ (MSB[1] and MSB[2]) are removed. This effectively truncates the phase from [0, 2) to [0, 1/2) and hence remaps the angle to the first quadrant, as shown in Fig. 3.3(a). The resulting angle ϕ' is passed to the π/4 mirror to generate the value $\phi'' \in [0, \pi/4)$.

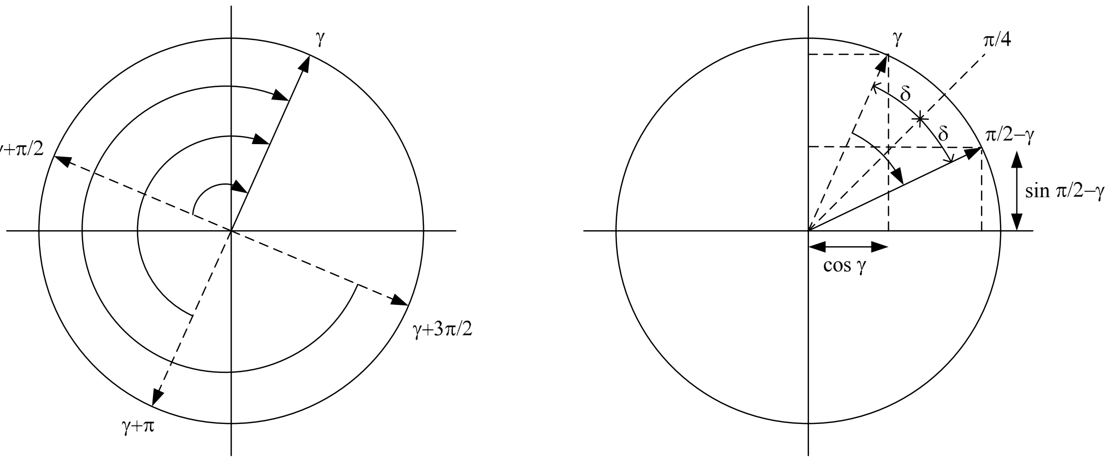


Fig. 3.3. (a) Mapping of angles to first quadrant. (b) π/4 mirroring.

3.3.1 π/4 Mirror

Symmetry exists within the first quadrant of the unit circle that allows the sine and cosine of any angle greater than π/4 to be obtained from an angle equally less than π/4, and vice versa, as shown in Fig. 3.3(b). The cosine of an angle offset by δ from π/4, denoted γ, can be found by calculating the sine of an angle that is equally offset in the opposite direction, denoted (π/2 – γ). This means that cos γ = sin (π/2 – γ) and sin γ = cos (π/2 – γ). MSB[3] = 1 indicates that ϕ' represents the range [π/4, π/2), and is used in this design as a condition for mirroring.

The mirroring operation is represented mathematically as $\phi'' = 1/2 - \phi'$. Because ϕ' represents a fixed point number within the interval [0, 1/2) with no sign bit, subtracting the number from 1/2 is equivalent to two's complement negation. Instead of performing two's complement negation, which requires a carry ripple, we perform a ones' complement negation. Hence the resulting value of ϕ'' can be one LSB less than it should be. Normally this one LSB error results in a significant phase truncation spur. To avoid this spur, we will compensate for this missing LSB in the multiplier—which is an important new technique, related to a method mentioned in [4]. Fig. 3.4 shows the implementation of the π/4 mirror. When MSB[3] = 0, ϕ' is passed through unmodified, otherwise it is complemented.

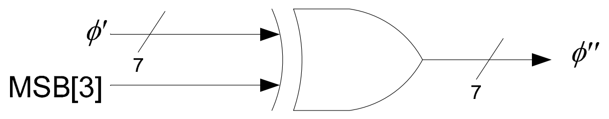


Fig. 3.4. Implementation of π/4 mirror.

3.4 ROM

The ROM serves three purposes. The first is to implement the coarse stage of our DDFS. In this design the ROM has 128 words.

The second purpose of the ROM is to act as a radian converter, converting $\phi'' \in [0, 1/4)$ into $\theta \in [0, \pi/4)$. Many rotation based DDFS designs use a dedicated π/4 multiplier circuit for this purpose. Instead, we include the π/4 factor within the pre-computed ROM data. In doing so, instead of requiring only two ROM outputs, cos θ_M and sin θ_M, we now require four outputs: X_M, Y_M, X_R, and Y_R. The values stored at the ROM locations are given by (2.8). Fortunately, the second set of outputs require only seven bits, as opposed to the entire 17-bit SWL. This requires more FPGA fabric LUTs to implement the ROM, but the difference is actually smaller than implementing a π/4 multiplier in the fabric. Alternatively, we could implement the π/4 multiplier in a DSP slice, but that would result in a greater power consumption than simply making the ROM larger. The other advantage of including the π/4 factor in the ROM is a reduced latency.

The final purpose of the ROM is to compensate for finite signal word-length errors and truncation in the datapath. The values for the ROM are optimized to minimize error at the X and Y outputs of the DDFS relative to ideal rounded values. This allows for a datapath with an SWL that is only two bits larger than the output precision—without the use of rounding anywhere. The ROM optimization algorithm is as follows:

1) Calculate ideal ROM values using floating point arithmetic and round to the SWL.
2) Simulate the entire DDFS design with the signature f_{cw} and record the $X_{initial}$ and $Y_{initial}$ output sequences.
3) For each X ROM location, identify the $X_{initial}$ values in the output sequence that correspond to that location. Modify the ROM location by subtracting the mean of the differences between each $X_{initial}$ and X_{ideal} value, followed by re-rounding to the SWL.
4) Apply the same technique to the Y ROM.

The implemented ROM values are omitted from this thesis for brevity.

The native LUT in the Virtex-7 FPGA has six inputs. Each output bit of the 128-word ROM is decomposed into two six-input LUTs, a MUXF7, and an output register. All four of these components can be packed into a single logic slice, which is crucial for 1-GHz operation.

3.5 Rotation Stage

3.5.1 Recoding

Naively providing the input angle ϕ_L into a multiplier does result in an angle rotation, but the magnitude of the phasor defined by the DDFS output (X, Y) is not maintained, resulting in an error that degrades the spectral purity of the DDFS. The key to using a single multiplier for angle rotation is to maintain the magnitude of the phasor by using the angle rotation algorithm discussed in Chapter 2. We do this by recoding the bits of ϕ_L into the signed two's complement number t, as shown in (2.6):

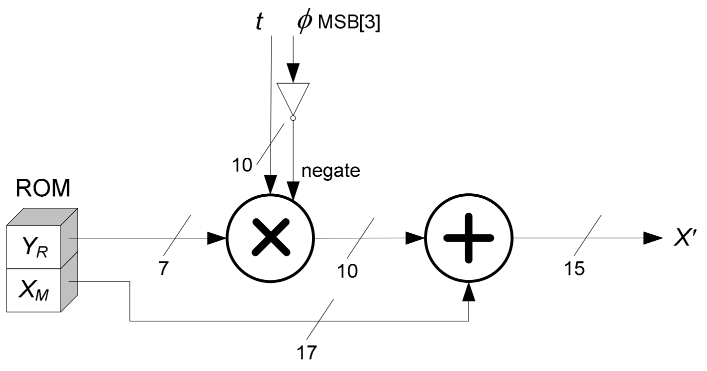
$$t_0 = \overline{\phi_L[0]} \qquad t_{1:N} = \phi_L[1:N] \qquad t_{N+1} = 1$$

where the MSB of ϕ_L is in the $k = M + 1$ position of the b_k bits of ϕ. The t_{N+1} bit can be set to 0 and compensated in the ROM without significant degradation in the spectral performance. The recoding is thus reduced to a single inverter on the MSB of the angle.

3.5.2 Multiplier-Adder

The multiplier-adder is the core of the angle rotation stage. Its structure is shown in Fig. 3.5. This block serves two functions. The primary function is the fine angle rotation (2.7) using the signed two's complement value t (which is a recoded version of ϕ_L).

The secondary function of the multiplier is to act as the π/4 mirror for the LSBs of ϕ'. Recall that a physical π/4 mirror was only placed before the ROM, and not before the multiplier. Using the conditional negation feature of the multiplier allows us to implement the mirror for free, with no logic cost. In addition, by performing a two's complement negation of t, we compensate for the ones' complement negation's missing LSB in the ROM π/4 mirror. This is because, for the minimum value represented by a two's complement n-bit number, the negative of that number overflows into an additional bit. On first glance this does not have the behavior we want—to virtually add an LSB into the ROM π/4 mirror. The crucial detail is that, due to recoding, the sign bit of t is the inverse of the MSB of ϕ_L. This means that for $\phi' = 1/4$ (representing the radian angle π/4), the value of t is –1/4. Negating t in the multiplier based on ϕ MSB[3] causes it to rotate by 1/4 instead of overflowing to 0. This virtual bit emulates adding an LSB in the ROM π/4 mirror.



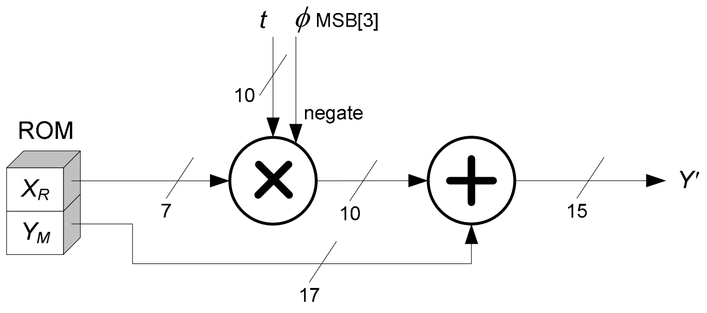


Fig. 3.5. Implementation of multiplier-adder.

Each multiplier-adder combination is implemented in a DSP slice in the FPGA. The hard multiplier in the DSP slice is much more efficient than a multiplier implemented in the FPGA fabric. For X', Y_R is supplied to the A input of the DSP slice, t is driven into the B input, X_M is supplied to the C input, and X' is taken from the P output. For Y', X_R is supplied to the A input, t is supplied to the B input, Y_M is supplied to the C input, and Y' is taken from the P output. Input bits A[6:0], B[9:0], and C[23:7] are used. Because t is a signed number, it must be sign-extended into B[17:10]. X' and Y' are taken from P[23:9].

Parameter	**Value**
USE_MULT	TRUE
AREG	1
BREG	2
CREG	1
PREG	1
ALUMODE	00 {negate, negate}
OPMODE	010 01 01

Table 3.2. Multiplier-adder DSP parameters.

Notice that BREG in Table 3.2 is one larger than AREG. This extra delay is used to delay *t* to balance the pipeline. These pipeline registers were implemented in the DSP slice rather than in CLB registers to reduce power consumption.

3.6 Output Stage

The output stage is responsible for mapping the *X′* and *Y′* outputs in the first octant of the unit circle [0, π/4] back into its proper octant within [0, 2π). This can be achieved by optionally swapping and negating *X′* and *Y′*, as shown in Table 3.3 below.

ϕ MSB[1–3]	**$\pi\phi$**	***SwapXY***	***NegateX***	***NegateY***
000	$0 \le \pi\phi < \pi/4$	0	0	0
001	$\pi/4 \le \pi\phi < \pi/2$	1	0	0
010	$\pi/2 \le \pi\phi < 3\pi/4$	1	1	0
011	$3\pi/4 \le \pi\phi < \pi$	0	1	0
100	$\pi \le \pi\phi < 5\pi/4$	0	1	1
101	$5\pi/4 \le \pi\phi < 3\pi/2$	1	1	1
110	$3\pi/2 \le \pi\phi < 7\pi/4$	1	0	1
111	$7\pi/4 \le \pi\phi < 2\pi$	0	0	1

Table 3.3. Swap and negation control conditions.

The three MSBs of ϕ are used to generate the controls *SwapXY*, *NegateX*, and *NegateY*. These controls are generated using two XORs as shown in Fig. 3.6 below.

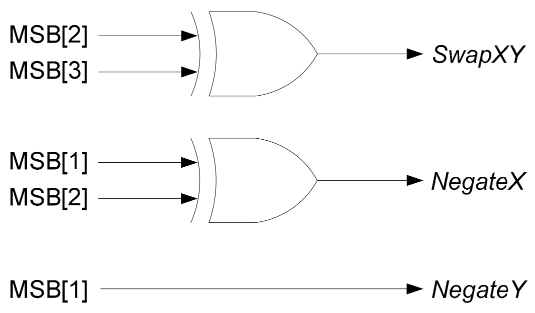


Fig. 3.6. Generation of output stage control signals.

The swap and negation circuit is implemented as shown in Fig. 3.7. *X′* and *Y′* are swapped with two 2:1 multiplexors, followed by a subtractor which optionally negates the two. Notice the growth from 15 bits to 16 bits after the swap stage to allow for the sign bit after negation.

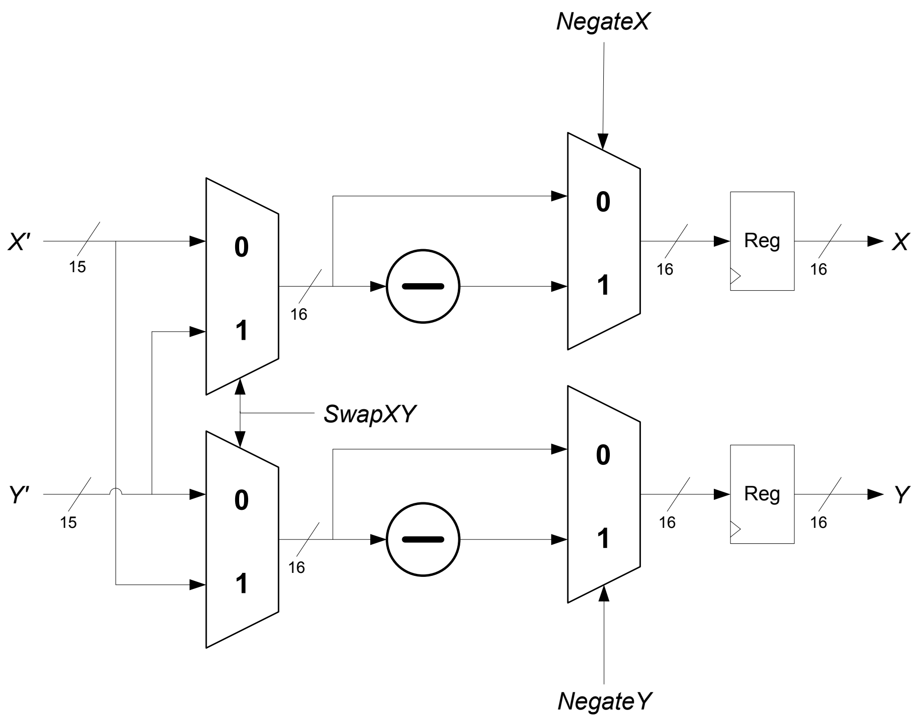


Fig. 3.7. Implementation of output stage.

Each bit of *X* and *Y* in the output stage is packed into a single CLB slice in the FPGA. This is possible because the Virtex-7 slice contains a six-input LUT, a carry chain, and a register. *X′*, *Y′*, *SwapXY*, and *NegateX* (or *NegateY*) are only four inputs, which easily fit into a single LUT. This packing is once again critical for achieving our high-speed operation. DSP slices could be used to implement the output stage, but they were found to consume more power than the CLB-based circuit.

Chapter 4

Results

The DDFS implementation presented in Chapter 3 was targeted to a Xilinx Virtex-7 FPGA and characterized. The specifications and results of the implementation are summarized in Table 4.1 below.

Parameter	**Value**
Target	Xilinx Virtex-7
Clock Frequency	1 GHz
Output Resolution	16-bit
Frequency Control Resolution	32-bit
Tuning Resolution	0.23 Hz
Phase Modulation	20-bit
Tuning Latency	8 clock cycles
Output SFDR	120.19 dBc
Output SNR	96.53 dB
Average Power	54.9 mW
LUT Count	139

Table 4.1. Design summary.

4.1 Verification

Functional verification of the implemented design was through simulation. The *X* and *Y* outputs of the DDFS were validated against a bit-accurate MATLAB model of the design.

The design was programmed into a Virtex-7 FPGA on a VC707 evaluation board from Xilinx. Verification was through built in self test (BIST), a circuit which inputs a pseudo-random sequence into the f_{cw} input of the DDFS and monitors the outputs. A signature analyzer generates a unique signature value for each clock cycle based on the captured DDFS outputs. The pseudo-random sequence generated by the BIST is identical for every test and, if the DDFS is functioning correctly, the output of the signature analyzer will be identical as well. The BIST test runs for a predetermined number of clock cycles and then stops the signature analyzer, storing the unique signature value.

4.2 Speed

The speed of the implemented design was verified with static timing analysis in the Xilinx Vivado software suite. It was found to run at 741.84 MHz, the maximum speed specified for DSP slice operation in the FPGA device. BIST was used to verify the design in hardware. The clock speed was incrementally increased while checking for passing BIST results. The design was functional at 1 GHz, which is above the maximum specified clock tree speed in the FPGA device.

4.3 Spectral Performance

The DDFS was analyzed using the algorithm described in [5]. The worst case SFDR (in the presence of phase truncation) is 116.49 dBc, while for large L the SFDR is 120.19 dBc. Fig. 4.1 below shows the SFDR for all L, the rightmost non-zero bit of f_{cw}.

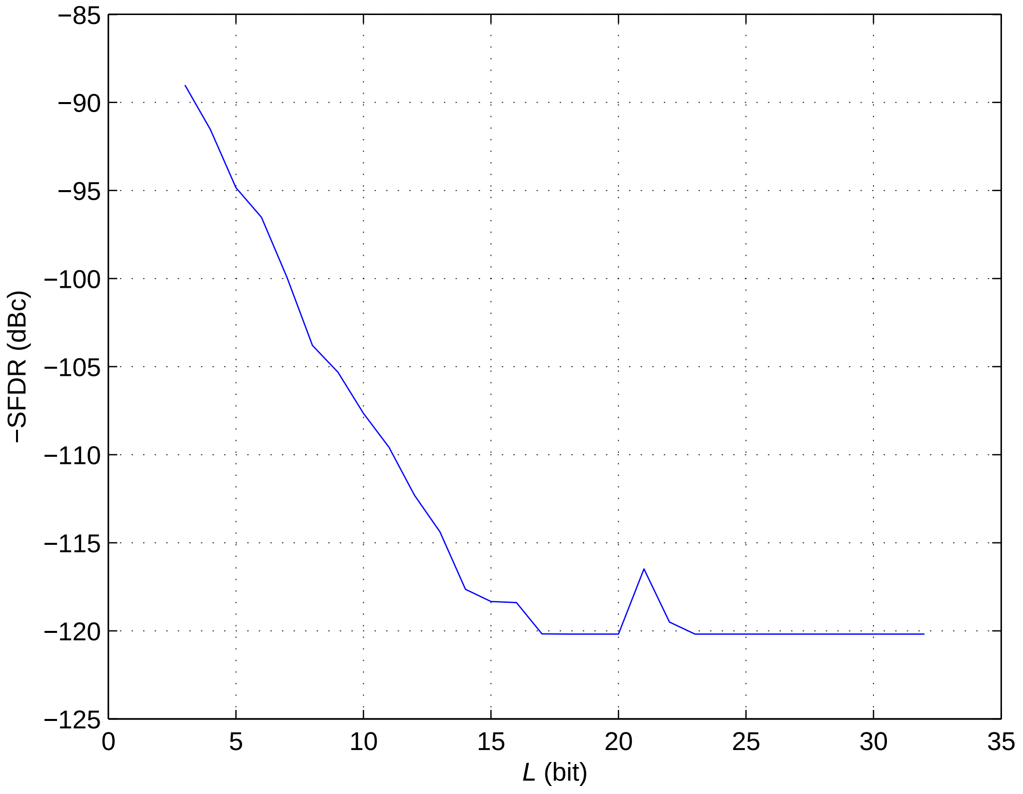


Fig. 4.1. Output SFDR vs. *L*.

The SNR of the DDFS was also analyzed using the algorithm in [5]. The SNR with no phase truncation ($L = 20$) is 96.53 dB. In addition, there is always no more than one LSB of time-domain error (compared to ideal rounded values).

4.4 Latency

The implemented design has a tuning latency of eight clock cycles. At 1 GHz this equates to 8 ns. The clock cycle latency breakdown is: two cycles in the phase accumulator, one cycle in the π/4 mirror, one cycle in the ROM, three cycles in the multiplier-adder, and one cycle in the output stage.

4.5 Device Utilization

Table 4.2 below shows the overall utilization for the DDFS in the Virtex-7 FPGA device. As shown in Table 4.3, the LUT count could be reduced significantly if the coarse ROM was stored in dedicated block RAMs as opposed to LUTs. This would, however, consume more power.

Resource	Count
Slice LUTs	139
Slice Registers	128
DSP48E1	3

Table 4.2. Device utilization.

Module	LUT Count
Coarse ROM	96
π/4 Mirror	7
Output Stage	32
Miscellaneous	4

Table 4.3. LUT breakdown by module.

4.6 Power

The Virtex-7 485T device on the VC707 board is a rather large device, so it is important to only measure the power associated with the DDFS itself, and not the device static power or power consumed by the BIST or other miscellaneous circuitry. We used a differential power measurement to isolate the DDFS power. First we measure the FPGA core power for a chip built with the BIST circuitry and a single DDFS core. Then we build a second chip with additional DDFS cores and measured its power. The DDFS core power is the difference of the two measurements.

It is also important to note that clock tree power in an FPGA not a strictly linear function of loading. In fact the power per load is inversely proportional to the number of loads. For a large number of loads the clock tree power asymptotically approaches a linear region. For accurate measurements, the second "chip build" described above contains 41 DDFS cores, all constrained to a single clock region.

Power was measured for various values of f_{cw} with the clock frequency set to 1 GHz, as shown in Table 4.4 below. The average power consumption is 54.9 mW, for a normalized power consumption of 54.9 µW/MHz.

f_{cw} (hex)	**Power (mW)**	**Power (µW/MHz)**
0000_1000	17.40	17.40
0000_8000	19.70	19.70
DFFF_8000	87.30	87.30
1111_8000	65.70	65.70
4000_0000	21.05	21.05
0021_4038	37.05	37.05
13CB_D8D3	68.85	68.85
1999_999A	73.30	73.30
23DF_1DE3	78.05	78.05
3BE7_957A	62.90	62.90
5555_5555	73.05	73.05

Table 4.4. Power consumption for different f_{cw}.

Chapter 5

Conclusion

A novel multiplier-based, magnitude-invariant, angle-rotation algorithm was introduced. A quadrature DDFS using a coarse-fine architecture was realized and its design, implementation, and results were described and presented. Our angle rotation algorithm enabled us to implement the fine stage of the DDFS with a single multiplier-adder, minimizing power dissipation. The conditional negation feature of multiplier enables a new technique to compensate for ones' complement negation in the $\pi/4$ mirror.

Implemented in a Virtex-7 485T FPGA, the DDFS dissipates 54.9 mW of power at 1 GHz. This level of performance was previously attainable only in ASIC designs. The measured power compares favorably with recently reported FPGA work [6]–[7], even when adjusting for device differences. The DDFS also compares favorably with an ASIC design [4] implemented on a decade-old process, with the process advantage being cancelled out by the FPGA vs. ASIC power overhead. Table 5.1 presents the comparisons described above.

Design	**Target**	**PA (bits)**	**Out-put (bits)**	**SFDR (dBc)**	**Data Rate (MHz)**	**Power (µW/ MHz)**	**Slice**	**DSP**
This work	Xilinx Virtex-7	32	16	120.2	1000	54.9	46	3
Genovese – PQ1* [6]–[7]	Xilinx Virtex-5	24	19	124.9	138.6	281.4	57	5
Genovese – PQ2* [6]–[7]	Xilinx Virtex-5	24	16	104.1	281.7	159.2	45	2
Willson – Excess Fours [4]	0.18-µm CMOS	32	16	113	260	63.5	–	–

Table 5.1. Comparisons.

Bibliography

[1] J. Tierney, C. Rader, and B. Gold, “A digital frequency synthesizer,” *IEEE Transactions on Audio and Electroacoustics*, vol. AU-19, pp. 48–57, Mar. 1971.

[2] A. Madisetti, A. Y. Kwentus, and A. N. Willson, Jr., “A 100-MHz, 16-b, direct digital frequency synthesizer with a 100-dBc spurious-free dynamic range,” *IEEE Journal of Solid-State Circuits*, vol. 34, pp. 1034–1043, Aug. 1999.

[3] J. E. Volder, “The CORDIC trigonometric computing technique,” *IRE Transactions on Electronic Computers*, vol. EC-8, pp. 330–334, Sept. 1959.

[4] A. Willson, M. Ojha, S. Agarwal, T. Lai, and T.-C. Kuo, “A direct digital frequency synthesizer with minimized tuning latency of 12ns,” in *IEEE ISSCC Digest of Technical Papers*, Feb. 2011, pp. 138–139.

[5] A. Torosyan and A. N. Willson, Jr., “Analysis of the output spectrum for direct digital frequency synthesizers in the presence of phase truncation and finite arithmetic precision,” *The 2nd International Symposium on Image and Signal Processing and Analysis*, Jun. 2001, pp. 458–463.

[6] M. Genovese and E. Napoli, “Direct Digital Frequency Synthesizers implemented on high end FPGA devices,” *9th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, Jun. 2013, pp. 137–140.

[7] M. Genovese, E. Napoli, D. De Caro, N. Petra, and A. G. M. Strollo, “Analysis and comparison of direct digital frequency synthesizers implemented on FPGA”, *Integration, the VLSI journal (2013),* in press.