Designs of Angle-Rotation in Digital Frequency Synthesizer/Mixer Using Multi-Stage Architectures

Shen-Fu Hsiao, Cheng-Han Lee, Yen-Chun Cheng
Department of Computer Science and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan

Andrew Lee
Department of Electrical Engineering/Computer Science
University of California, Berkeley
U.S.A.

Abstract—The key operation of the quadrature digital frequency synthesizer/mixer (QDFS/M) in many communication systems is the rotation of an input vector in the complex plane. In this paper, we propose three multi-stage architectures for the design of the angle rotation in QDFS/M, targeting different precision requirements. The rotation is decomposed into three sub-rotations, each implemented by ROM-based multiplication-addition, linear approximation, or CORDIC operations. The major design consideration is to choose the proper architecture for a particular precision so that we can reduce the size of the required look-up table (LUT) and the bit widths of the corresponding arithmetic operations. Implementation results of various bit accuracies show that the best design choice depends on the desired precision.

Index Terms: angle rotation, direct digital frequency synthesizer, digital frequency mixer, CORDIC, quadrature modulation.

I. INTRODUCTION

Frequency synthesis is an important operation in many digital communication systems. Due to the rapid advancement of silicon processing technology and the integrated circuit design, digital implementations of frequency synthesis and mixing have become popular in digital signal processor design for communication systems. Direct digital frequency synthesizer (DDFS) generates the digital values of sine and cosine functions. In fact, the quadrature DDFS is one of the key components in various modulation methods, such as quadrature phase shift keying (QPSK) and quadrature amplitude modulation (QAM). The major arithmetic operation involved in digital frequency synthesizer and the corresponding quadrature digital frequency mixer (QDFM) can be expressed as

\[ x' + jy' = (x + jy) \times e^{j\theta} \]  

where \( x, y \) are the two input signals and \( x', y' \) are the two outputs. The phase \( \theta \) is directly related to the input frequency control word [1]. A typical QDFM consists of a phase accumulator, followed by an angle rotator with the angle restricted in a range, usually between \(-\pi/4 \) and \(\pi/4 \). Since the angle rotator is the key component, we focus on the hardware design of the rotator in this paper.

A straightforward implementation of the QDFM is to generate the cosine and sine function values via look-up tables (LUT), followed by a complex multiplier that is composed of four real multipliers and two real adders. Another possible implementation is to use CORDIC algorithm [2]. Indeed, Eqn. (1) can be written as a two-dimensional (2-D) vector rotation:

\[
\begin{bmatrix}
    x' \\
    y'
\end{bmatrix} =
\begin{bmatrix}
    \cos \theta & -\sin \theta \\
    \sin \theta & \cos \theta
\end{bmatrix}
\begin{bmatrix}
    x \\
    y
\end{bmatrix}.
\]

and CORDIC is well known to be effective in performing such plane rotations using pure shift-and-add operations. However, due to the iterative operations, CORDIC is inherently slow, requiring many iterations (in folded architecture), or many cascaded computation stages (in unfolded architecture).

Recently, some ROM-based CORDIC-like implementation approaches have been presented to realize the rotation in QDFM [3-8]. The major concept is to decompose the rotation angle in the plane rotation of Eqn. (2) into two parts \( \theta = \theta_m + \theta_l \) where the rotation with the large angle \( \theta_m \) is realized in the coarse stage and the rotation with the remaining small angle \( \theta_l \) is executed in the fine stage:

\[
\begin{bmatrix}
    X \\
    Y
\end{bmatrix} = \begin{bmatrix}
    \cos \theta_l & -\sin \theta_l \\
    \sin \theta_l & \cos \theta_l
\end{bmatrix}
\begin{bmatrix}
    \cos \theta_m & -\sin \theta_m \\
    \sin \theta_m & \cos \theta_m
\end{bmatrix}
\begin{bmatrix}
    X_0 \\
    Y_0
\end{bmatrix}.
\]

In Eqn. (3), the multiplication of the scaling factor \( \cos(\theta_m) \) extracted from the coarse stage is performed in the final scaling stage.

Due to the limited bit-width of the LUT for \( \tan(\theta_m) \), the stored tangent value in the coarse stage will not be exactly the same as \( \tan(\theta_m) \). Let \( \tan(\theta_m) \) denote the actual stored value. The remaining angle can be expressed as

\[ \theta_l = \theta - \theta_m = (\theta_m + \theta_l) - \theta_m = \theta_l + (\theta_m - \theta_m), \]

and \( \theta_m - \theta_m \) can also be stored in a ROM. If the angle decomposition is such that \( |\theta| < 2^{-N/3} \) with \( N \) representing the number of fractional bits in the final accuracy, we can exploit, in the fine stage, the approximations

\[ \tan \theta_l \approx \theta_l, \quad \sin \theta_l \approx 1, \quad \cos \theta_l = 1 - \theta_l^2 / 2, \]

which simplify the computation in the fine rotation stage. Thus, the complete rotation with \( \theta = \theta_m + \theta_l \) can be expressed as:

\[
\begin{bmatrix}
    X \\
    Y
\end{bmatrix} = \begin{bmatrix}
    \cos \theta_m & -\sin \theta_m \\
    \sin \theta_m & \cos \theta_m
\end{bmatrix}
\begin{bmatrix}
    \tan \theta_m \\
    1
\end{bmatrix}
\begin{bmatrix}
    X_0 \\
    Y_0
\end{bmatrix}.
\]
where $|\theta| < 2^{-N/3}$. Fig. 1 shows the three-stage architecture where the rotation is divided into the coarse stage and the fine stage, plus a scaling stage [3].

There are some variations for the above three-stage rotation architecture. In [4-5], the coarse stage and scaling stage are merged into one stage performing the plane rotation of Eqn. (2) with rotation angle $\theta_\infty$. In [6-7], the coarse rotation stage is realized using a sequence of CORDIC micro-rotations with the direction control signs generated concurrently from a LUT. In [8], the so-called quad line approximation (QLA) method is proposed to reduce the ROM size in the coarse stage.

Although previous methods achieve good approximation results in low-precision situations (usually less than 16-bit accuracy), the ROM might become too costly in high bit accuracy requirement, and thus the best architectural design might depend on the precision requirement. In this paper, we will present several multi-stage architecture alternatives.

II. PROPOSED MULTI-STAGE ARCHITECTURES

The basic design philosophy of this paper is to partition the angle rotation into three parts where the rotation of each decomposed angle is realized using either CORDIC or ROM-based linear approximation. The differences of these architectures lie in the boundary of the angle partition.

(A) Type-I Architecture

In this Type-I architecture, the boundaries of the angle decomposition are $N/3$ and $N/2$ fractional bits. If a rotation angle is smaller than $2^{-N/3}$, the approximation of Eqn. (5) can be employed to reduce hardware implementation cost. If the remaining rotation angle is smaller than $2^{-N/2}$, the trigonometric functions can be further simplified into

$$\tan \theta_i \approx \theta_i, \quad \sin \theta_i \approx \theta_i, \quad \cos \theta_i \approx 1$$

(7)

with accuracy of $N$ fractional bits. Thus we can decompose the rotation angle into three parts $\theta = \theta_\rho + \theta_q + \theta_\sigma$ so that after the first rotation of $\theta_\rho$, the remaining angle $|\theta_q + \theta_\sigma| < 2^{-N/3}$, and after the second rotation of $\theta_q$, the remaining angle $|\theta_\sigma| < 2^{-N/2}$.

Thus, the rotation of Eqn. (3) becomes

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos \theta_\rho \times \cos \theta_q & \cos \theta_\sigma \\ \cos \theta_\rho \times \cos \theta_q & \cos \theta_\sigma \end{bmatrix} \begin{bmatrix} \tan \theta_p \\ \tan \theta_p \end{bmatrix} \begin{bmatrix} \tan \theta_\rho \\ \tan \theta_\rho \end{bmatrix} \begin{bmatrix} 1 - \theta_q \\ 1 - \theta_q \end{bmatrix} X_0$$

(8)

It is observed that when $|\theta| < 2^{-N/3}$, the value of $\cos \theta_q$ is close to 1. Thus, instead of storing the value of $\cos \theta_q$ in the LUT and performing the subsequent full-width multiplication, we store the value of $1 - \cos \theta_q$, whose magnitude is less than $2^{-2N/3}$, so that the bit widths of the ROM and the subsequent multiplier are reduced. The scaling operation of $\cos \theta_\rho \times \cos \theta_q$ can be expressed as

$$\cos \theta_\rho \times \cos \theta_q = \cos \theta_\rho - \cos \theta_\rho \cdot [1 - \cos \theta_q].$$

(9)

Fig. 2 shows the proposed Type-I four-stage architecture. Another variation of Type-I architecture is to approximate the scaling of $\cos \theta_\rho \times \cos \theta_q$ by

$$\cos \theta_\rho \times \cos \theta_q = \cos \theta_\rho \times [1 - \theta_q^2/2]$$

(10)
because $|\theta_q| < 2^{-N/3}$. This implementation calls for a squarer, a subtractor, and a multiplier.

**B) Type-II Architecture**

As the bit accuracy $N$ increases, the LUT in stage 1 of Fig. 2 might become too costly. Thus, we propose another 4-stage architecture where the decomposition of the rotation angle $\theta = \theta_p + \theta_q$ is chosen such that $|\theta_p + \theta_q| < 2^{-N/4}$ after the first rotation with $\theta_p$, and $|\theta_q| < 2^{-N/2}$ after the second rotation with $\theta_q$. The complete rotation can be expressed as

$$X = \begin{bmatrix} X_0 \\ Y_0 \\ \vdots \\ X_4 \\ Y_4 \\ \vdots \\ X_N \\ Y_N \\ \vdots \end{bmatrix} = \begin{bmatrix} \cos \theta_p \times \cos \theta_q \\ \sin \theta_p \times \cos \theta_q \\ \vdots \\ \cos \theta_p \times \sin \theta_q \\ \sin \theta_p \times \sin \theta_q \\ \vdots \\ \cos \theta_p \times \sin \theta_q \\ \sin \theta_p \times \sin \theta_q \\ \vdots \end{bmatrix}. \quad (11)$$

The corresponding architecture is shown in Fig. 3. The major difference between the Type-II and Type-I architectures is that the stage-1 rotation in the Type-II architecture is only accurate up to $N/4$ (instead of $N/3$) fractional bits and thus the size of the LUT is only $2^{-N/4}$ entries instead of $2^{-N/3}$ entries. However, the approximation of Eqn. (5) is no longer valid after the first rotation, and thus the arithmetic operations in stage 2 become more complex. This reflects the design trade-off between the LUT in stage-1 and the computation complexity in stage-2.

**C) Type-III Architecture**

In this architecture, the first stage of Fig. 2 is replaced by unfolded CORDIC iterations in order to eliminate the large LUT ROM especially when the bit accuracy $N$ is large. CORDIC replaces a plane rotation by a sequence of shift-and-add operations, expressed as

$$\begin{bmatrix} X_i \\ Y_i \\ \vdots \\ X_N \\ Y_N \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ -\theta_p & 1 & \ldots & \ldots & \ldots & -\theta_p \\ 1 & \tan \theta_p & \ldots & \ldots & \ldots & 1 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & \ldots & \ldots & \ldots & \ldots & \ldots \\ -\theta_q & \ldots & \ldots & \ldots & \ldots & \ldots \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ \vdots \\ X_4 \\ Y_4 \\ \vdots \end{bmatrix}. \quad (12)$$

$$Z_i = Z_0 - \theta_p = Z_0 - \sum_i \sigma_i \times \tan(2^i), \quad \sigma_i \in \{\pm 1\}$$

The number of CORDIC iterations is chosen such that the remaining angle $|Z_i| < 2^{-N/3}$ and thus the approximation in Eqn. (5) can be employed. In Eqn. (12), the constant CORDIC scaling factor

The number of CORDIC iterations can be chosen such that the remaining angle $|Z_i| < 2^{-N/3}$ and thus the approximation in Eqn. (5) can be employed. In Eqn. (12), the constant CORDIC scaling factor

$$K = \prod_{i=1}^{\infty} \frac{1}{\sqrt{1 + 2^{-2i}}}$$

can be merged with the scaling factor in the second stage and stored in a ROM containing $K \times \cos(Z_i^H)$. The remaining rotation angle after stage-1 CORDIC operation can be further decomposed as $Z_i = Z_i^H + Z_i^L$ with $|Z_i^L| < 2^{-N/2}$ so that the approximation of Eqn. (7) can be applied.

Fig. 4 shows the architecture where the CORDIC iterations in stage 1 is realized using the unfolded architecture given in Fig. 5. This Type-III architecture only calls for a LUT of around $2^{3N/6}$ entries in stage-2, plus a small table storing the arctangent values for CORDIC operations in stage-1.

The sequential determination of the control signs and the Z-datapath in the iterative CORDIC operations can be implemented more effectively using a ROM that stores the control signs for the first $N/3$ iterations and the remaining rotation angles. In this variation, the Z-datapath in stage 1 of Type-III architecture can be eliminated, and the control signs can be generated concurrently.

**III. BIT-WIDTH OPTIMIZATION AND COMPARISON**

For the purpose of comparing different QDFM architecture designs, we need to analyze the errors introduced in the architectures in order to determine the bit widths of all the hardware components, including ROM, adders, and multipliers. In general, there are two types of error sources:
quantization error and rounding error. The quantization error is due to the limited bit width of each ROM entry. The rounding error represents the truncation of the output bits after an arithmetic operation.

We can exploit the range of operands to optimize the bit width of the multipliers. Assume $|A| < 2^{-m}$ and $|B| < 2^{-n}$ have $m_1$ and $n_1$ fractional bits respectively, as illustrated in Fig. 6 where the leading sign bits are all zero for positive numbers and are all ones for negative numbers. If we want the product $A \times B$ to achieve an accuracy of $N$ fractional bits, it is easy to show that we only need an $(N - m_1 - n_1 + 1) \times (N - m_1 - n_1 + 1)$ hardware multiplier instead of an $m_1 \times n_1$ multiplier. This property is used throughout our designs to minimize the multiplier area cost.

Using the above property and after performing error analysis to compute the influences of all the error sources on the output, we can determine the proper bit widths for the arithmetic components and the LUT ROMs. Take Type-III design as an example, as shown in Fig. 7 with error sources $\varepsilon_i$ highlighted. Tab. 1 shows the individual error sources and the resultant total errors appeared at the output of the Y path in the Type-III architecture. Observing that

$$[K] < 1, \quad Z_i'' < 2^{-N(T+1)}, \quad Z_i' < 2^{-N(T+1)},$$

we can calculate the total error at the Y output as

$$E_{total} = |E_1 + E_2 + E_3 + E_4 + E_5 + E_6|,$$

$$< |\varepsilon_1| + 4|\varepsilon_2| + |\varepsilon_3| + 4|\varepsilon_4| + |\varepsilon_5| + 4|\varepsilon_6|.$$  

If we decompose the rotation angle properly so that the approximation error sources $\varepsilon_2, \varepsilon_5$ are bounded by

$$|\varepsilon_2| < 2^{-3(N/13)}, \quad |\varepsilon_5| < 2^{-3(N/13)},$$

we can choose the bounds of the error sources as follows:

$$|\varepsilon_1| < 2^{-N(6)}, \quad |\varepsilon_2| < 2^{-3(N/3)}, \quad |\varepsilon_3| < 2^{-3(N/3)}, \quad |\varepsilon_4| < 2^{-3(N/3)}, \quad |\varepsilon_5| < 2^{-3(N/3)}, \quad |\varepsilon_6| < 2^{-3(N/3)}.$$  

so that the maximum magnitude of the total error is smaller than $2^{-N}$, the target precision requirement. To satisfy the bounds of the individual error sources shown in Eqn. (17), we can choose the bit widths for the internal signals as follows:

- **stage 1**: $A_1: b = N + [\log_2([N/3] + 1)] + 3$
  $T_1: ([N/3] + 1) \times b$
- **stage 2**: $A_2, A_3: N + 9$
  $M_1, M_2: ([2N/3] + 6) \times ([N/6] + 1)$
- **stage 3**: $A_4, A_5: N + 7$
  $M_3, M_4: ([N/2] + 6) \times ([N/2] + 2)$
- **scaling stage**: $M_5, M_6: (N + 7) \times (N + 3)$

The error analysis of other architectures and the derivation of the bit widths are similar.

### IV. EXPERIMENTAL RESULTS

**A. Architectural-Level Comparison**

In order to quickly see the impact of precision on the total area cost, we first measure the area cost and delay of individual hardware components with respect to a full adder (FA) based on the estimations listed in Tab. 2 where $A_{FA}$ and $T_{FA}$ denotes respectively the area and delay of a full adder cell. These estimates are based on extensive synthesis experiments across several process technology generation. The $B$-bit fast adder adopts carry look-ahead structure; The $p$-input carry save adder (CSA) of $B$ bits employs $(3, 2)$ counters for multi-operand reduction. The $M \times N$ multiplier with the output in carry-save (CS) form generates the product of two numbers without the carry-propagate final adder. The estimation of ROM is derived based on the traditional PLA structure with AND plane and OR plane.
Tabs. 3 and 4 compare respectively the area and latency of different QDFM architectures under various precision requirements. We observe that in low-precision situations of 8-b and 12-b accuracies, the two-stage architecture leads to smaller area. However, in higher precision cases such as 16-b, 24-b, and 32-b, the proposed three types of architectures have lower area cost. Take 32-bit as an example. The area of the Type-III architecture is only about 1/3 that of the previous two-stage design, at the cost of longer latency, while the Type-II architecture, has best compromise of area and latency.

### B. Comparison Based on Synthesis Results

We have designed various architectures with accuracies of 12, 16, 24 and 32 fractional bits. Synopsys Design Compiler (DC) is employed for logic synthesis based on the UMC 90nm standard cell library. Tab. 5 shows the implementation results. The area is normalized to the two-input NAND gate in the standard cell library. Tab. 5 also includes the product of latency and area to reflect the compromise between area and speed optimization.

It is observed that for 12-bit or 16-bit accuracy, the proposed Type-I architecture has best area latency result compared to the previous design [5] and other architectures because the LUT takes only a small portion of total area. However, for higher bit accuracies such as 24-bit or 32-bit where the ROM area is large (exponentially increased with respect to bits), Type-III design is the better choice as far as area is concerned although the latency (the delay from the input to the final output) is longer than the other designs.

In many communication and DSP application, throughput (number of computed outputs per clock cycle) instead of latency is the more important speed metric. Note that it is possible to increase the throughput of all the QDFM designs by directly pipelining the feed-forward datapath to achieve the desired clock rate. In this situation, Type-III architecture might be good candidate in area and in speed (throughput).

### V. CONCLUSIONS

In this paper we propose three different architectures for the design of quadrature direct digital frequency synthesizer/mixer. Bit width optimization is also done through error analysis. Experimental results show that the best design choice is highly dependent on the precision requirement and the cost function, and that the proposed designs usually have better area or latency than existing designs, especially in high-precision requirements.

### ACKNOWLEDGMENT

This work is supported by Taiwan’s National Science Council (NSC) under grants NSC 99-2220-E-110-005 and NSC 99-2221-E-110-055. Chip Implementation Center (CIC) in Taiwan provides the support of process technology files and related

### REFERENCES


