FPGA Implementation of a Low Latency and High SFDR Direct Digital Synthesizer for Resource-Efficient Quantum-Enhanced Communication*

N. Fajar R. Annafianto*[,1,2], M.V. Jabir [2], I.A. Burenkov [2], H.F. Ugurdag[1], A. Battou[2] and S.V. Polyakov[2]
[1]Electrical and Electronics Engineering Dept., Ozyegin University, Istanbul, Turkey
[2]National Institute of Standards and Technology, Maryland, USA
*annafianto.annafianto@ozu.edu.tr

Abstract—**A Direct Digital Synthesizer (DDS) generates a sinusoidal signal, which is a significant component of many communication systems using modulation schemes. A CORDIC algorithm offers minimum memory requirements compared to look-up-based methods and low latency. The latency depends on the number of iterations, which is determined by the number of angles in the rotation set. However, it is necessary to maintain high spectral purity to optimize the overall system performance. To optimize the opportunity of quantum measurement, low latency and a high spectral purity sine wave generator is essential. The implementation of this design generates output with 64% latency reduction compared to that of the conventional CORDIC design and 72.2 dB SFDR value.**

Keywords—FPGA, CORDIC, DDS, SFDR, pipeline, latency.

I. Introduction

In most modulation schemes for a digital telecommunication system, a fast and efficient sinusoidal signal generator is needed. Here we report on an FPGA implementation of a versatile Coordinate Rotation Digital Computer (CORDIC) based Direct Digital Synthesizer (DDS). Most commercial lightwave communication systems use standard modulation protocols, such as Phase-Shift Keying (PSK) and Frequency-Shift Keying (FSK), whose implementation is supported by specialized dedicated hardware. There is a need for significant improvement in energy and bandwidth efficiency. Therefore, to gain further improvement a new class of communication systems, namely quantum-measurement enhanced optical communication systems are being actively pursued. In those systems, a classical receiver is replaced with a quantum receiver, while the transmitter remains the same. Properties of quantum measurement are in general different from that of classical measurement. They require more complex modulation schemes than PSK and FSK [1]. Digital synthesis of these signals requires versatile DDS whose development is reported in this work. By design, a DDS generates signals with a nearly arbitrary combination of phase and frequency modulations. Many other applications such as software-defined radio, wireless satellite transceiver, HDTV transmission, radar communication, etc. can take advantage of this low latency and re-configurable sine wave generation [2]. With its high spectral purity and low latency, the DDS accommodates the energy-efficient and rapid response properties needed by quantum measurement instruments in order to surpass efficient of classical measurement and maximize the modulation capabilities.

Many strategies and techniques have been developed to enhance the hardware area and speed efficiency of CORDIC algorithm implementation. CORDIC was initially introduced by Volder in 1959 to calculate trigonometric functions in digital hardware devices [2]. Later, a modified version of a CORDIC algorithm was proposed by S. Walther with the ability to calculate circular, hyperbolic, and linear rotation systems [4]. The motivation to use this algorithm in digital platforms has gained popularity since then. Refinements on the efficiency of implementation have resulted in reduced latency and hardware area usage.

In the next section, we provide background information on several CORDIC techniques that are adopted in this work. In section III, we explain the implementation of the proposed method. In section IV, we summarize and compare the obtained results. Finally, we conclude with an evaluation and discussion of the prospective developments.

II. Background

The demand for higher-throughput communication has always existed and provides the environment for developing new applications. Enhancing the performance of a DDS will lead to a throughput increase. Many communication systems use a modulation scheme that generates sinusoidal signal output. One popular approach is to use a DDS hardware module that takes the Frequency Tuning Word (FTW) or Frequency Control Word (FCW) as input and passes the amplitude of the sinusoidal signal to the output. Figure 1 shows the general block diagram of a DDS which consists of a phase accumulator, a signal processor, Digital-to-Analog Converter (DAC) and a low pass filter. The phase accumulator defines the frequency of the sinusoidal signal as the increment of the output phase that is dictated by the input FCW, hence the smaller the input the lower the resulting signal's frequency and vice versa. The low pass filter removes aliasing of the signal processor's output that contains some noise due to the techniques being employed. Here, we focus on the signal processor that takes the phase as the input and generates a sinusoidal amplitude.

* Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

978-1-7281-9899-6/20/$31.00 ©2020 IEEE

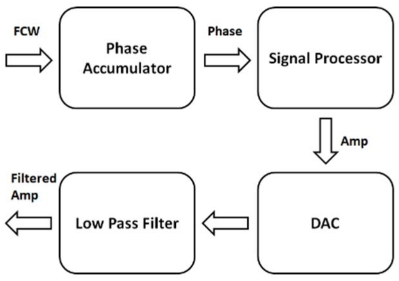

Fig. 1. General block diagram of DDS

CORDIC provides an efficient hardware implementation in terms of area utilization, power consumption and latency. There are three popular approaches to the realization of DDS: (1) Look-Up Tables (LUTs), (2) Polynomial Functions, (namely Tylor series expansion), and (3) a CORDIC algorithm [5].

The LUTs occupy memory, namely, Read-Only Memory (ROM), to store the amplitude of the sinusoidal signal. LUTs are computationally fast, but they require a considerable amount of memory even when compression techniques are used [6]. The memory occupancy is mainly based on the width of FCW input and the width of the output. To get higher spectral purity, the quantization error is minimized by increasing the LUT memory widths of the output amplitude to yield higher precision. The consequence is that memory size grows significantly with the greater spectral purity requirement. In turn, more memory results in higher power consumption, slower operation, and lower stability [7].

The Taylor series expansion has a complicated implementation that uses several multipliers/dividers. To get higher spectral purity, higher-order terms need to be included which in turn increases latency [5].

The CORDIC algorithm calculates sinusoidal amplitude by a set of rotations. The rotational angles in the set are processed in series and the accumulation of the angles approximates the desired angle that corresponds to the necessary output. The number of angles in the set determines the number of iterations, hence the latency. Thus, the correct selection of an angle set is essential. The rotational operation is carried out by adders, logic shifters, and optionally a small amount of memory that makes implementation and integration easier and simpler [3]. For these reasons, CORDIC is better at resource utilization and power consumption.

Spurious Free Dynamic Range (SFDR) defines the spectral purity of the produced signal. The spectral purity of a signal is significant for the overall performance of the system. Since there exists more than one frequency component in a signal, it is necessary to keep the desired frequency's dominance over the spurious components. SFDR is the ratio of the power of the desired signal to the power of the strongest spurious signal. Thus, the higher SFDR the smoother the output obtained, which is preferred and pursued in this work.

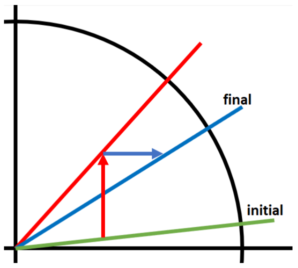

Fig. 2. Rotation mode of CORDIC in DDS with two angular steps

CORDIC has two functional modes: a vector mode and a rotation mode. Our DDS algorithm is based on the rotation mode. In this mode, the initial vector experiences several rotations in cartesian coordinates based on the angle set to reach the desired vector position that corresponds to the destination phase. Figure 2 shows the rotation mode with two phases in the set. In vector mode, the destination angle is estimated by using a set of pre-specified vectors as the reversal of the rotation mode, where it uses the vectors to approximate the target angle [5]. However, CORDIC also comes with some drawbacks. First, it needs scale factor compensation due to numerical operations in the algorithm. Usually, the result is achieved by dividing the scale factor with the output of the series of rotation. To eliminate this requirement, the initial vector is arranged such that it has already been regulated (pre-divided) with the scale factor prior to the calculations. Secondly, accuracy restriction: the number of rotations and the selection of the angles set impacts how close the final angle is to the desired angle [8]. A smaller angle set is beneficial. The following sections describe components in each stage of the hardware architecture and the mathematical operations that they employ.

A. Conventional CORDIC

The first CORDIC algorithm was proposed by removing the burdensome cosine and sine functions multiplications with logic shift operations. Equation (1) computes a rotation of the initial vector (x, y) to the vector (x', y') with the angular distance of θ.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (1)$$

$$\theta = \sum_{i=0}^{b-1} \alpha_i \qquad (2)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \prod_{i=0}^{b-1} cos\alpha_i \begin{bmatrix} 1 & -d_i \tan\alpha_i \\ d_i \tan\alpha_i & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (3)$$

The angle set of α_i converges to θ with a combination of clockwise and/or counterclockwise rotations, see equation (2).

Substituting (2) into (1) and taking $cos\alpha_i$ out of the matrix, we obtain equation (3), where d_i corresponds to the direction of rotation at the respective stage i, given as $d_i = \{-1, 1\} = sign(z_i)$, -1 is for counterclockwise direction and 1 is for clockwise direction. b is the angle set size and the number of iterations. z_i is the remaining phase at iteration i. Our goal is to establish a recurrent formula for rotations that can be conveniently calculated on an FPGA.

$$\alpha_i = \arctan(2^{-i}) \qquad (4)$$

$$\prod_{i=0}^{b-1} cos\alpha_i = K \qquad (5)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = K \begin{bmatrix} 1 & -d_i * 2^{-i} \\ d_i * 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (6)$$

$$z_i = \theta - \sum_{c=i}^{b-1} \alpha_c \quad (7)$$

K in equation (5) is the overall scale factor that can be pre-calculated for the initial vector (x, y). Substituting equation (4) and (5) into equation (3), we obtain equation (6). The division by 2^i can be replaced by an arithmetic shift operator. Thus, iterations are written as:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & -d_i * 2^{-i} \\ d_i * 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (8)$$

The block diagram in Figure 3 shows three stages (i-1), (i), and (i+1) of a conventional CORDIC as the realization of equation (8). The multiplexers have +/- tags that determine the additions or extractions of variables based on the sign of z_i. Note that intersections of lines show the crossing of paths with no connection between them, this is valid for all diagrams.

Equation (4) implies an angle in the angle set for the iteration i. For the sake of simplicity, by selecting 7 iterations ($i = [0, 6]$), we obtain the following angle set: {45, 26.565, 14.036, 7.125, 3.576, 1.789, 0.895}. Note that the conventional CORDIC has 15 iterations. To ensure that z_i is approximately 0 at the end for any destination angle θ, the number of iterations (b in equation (3)) is specified as 15, hence i ranges from 0 to 14. This number is also the accuracy limit of the digital system which uses variables with a bit width of 16 bits, because shifting more than 15 bits results in 0 in such a variable. Thus, these additional variables have no impact on the algorithm, their operations add redundant latency and produce no modification on the final output. However, the range of convergence, that specifies the absolute value of the angle θ, is 99.882. One uses a domain folding technique with 2 blocks that cover [-90, 90) and [90, 270). This way any θ can be reached by locating an appropriate initial vector.

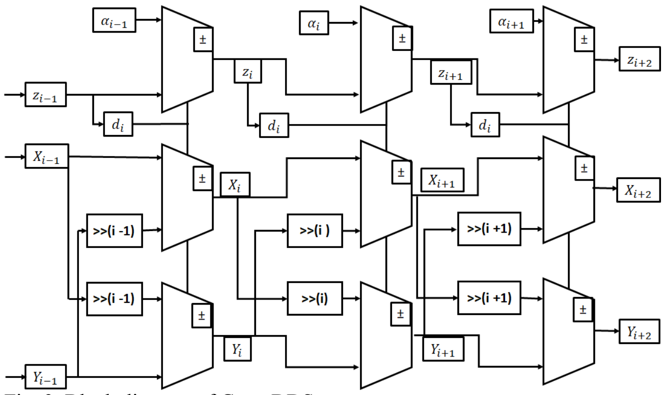

Fig. 3. Block diagram of Conv DDS

B. *Scaling Free CORDIC*

As the name implies, Scaling Free CORDIC pursues an algorithm to avoid multiplication by scale factor prior to the final output for fast performance. Scaling-free CORDIC recognizes one direction of rotation and a halting state, meaning $d_i \in \{0, 1\}$. It rotates counterclockwise only when z_i is greater than the angle at stage i or stays at the current position otherwise. Thus, z_i is always a positive number. This makes the attainable maximum frequency higher as we will see in the result section. The sine and cosine terms can be simplified when any of them is considerably small.

$$\left\lceil \frac{w - log_2 6}{3} \right\rceil \le j \le w - 1 \quad (9)$$

$$\begin{bmatrix} sin\theta \\ cos\theta \end{bmatrix} = \begin{bmatrix} 2^{-i} \\ 1 - 2^{-(2i+1)} \end{bmatrix} \quad (10)$$

The approximation in equation (10) is accurate if the requirement in equation (9) is met [4], where w is the bit width. By substituting (10) and (2) into (1) we obtain:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \prod_{i=0}^{b-1} d_i \begin{bmatrix} 1 - 2^{-(2i+1)} & -2^{-\mathrm{i}} \\ 2^{-\mathrm{i}} & 1 - 2^{-(2i+1)} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (11)$$

Then, the recursive formula is given by:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = d_i \begin{bmatrix} 1 - 2^{-(2i+1)} & -2^{-\mathrm{i}} \\ 2^{-\mathrm{i}} & 1 - 2^{-(2i+1)} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (12)$$

Note that (12) has no scale factor unlike in (6). Figure 4 depicts the block diagram of the Scaling Free CORDIC algorithm at stage i.

The low range of convergence uses a domain folding technique to squeeze the blocks into several extra regions. Due to the condition in (9), and with bit width (w in equation (9)) of 16, where j is i+1, the approximation in (10) only holds for i between 3 to 14, but after the 8th iteration, the logic shifter of (2i+1) results in more than 17 bits shift. Thus, iterations 9 through 14 have no effect. Therefore, iterations 9 through 14 are omitted, to reduce latency and redundant area usage. Hence i goes between 3 to 8. The range of convergence becomes [0, 22.5). The low range of convergence requires extensive use of a domain folding technique. To obtain convergence, 16 domain folding is employed and requires multiplication by a bothersome factor of $1/\sqrt{2}$. Thus, each domain's distance is 20 degrees which is within the range of convergence.

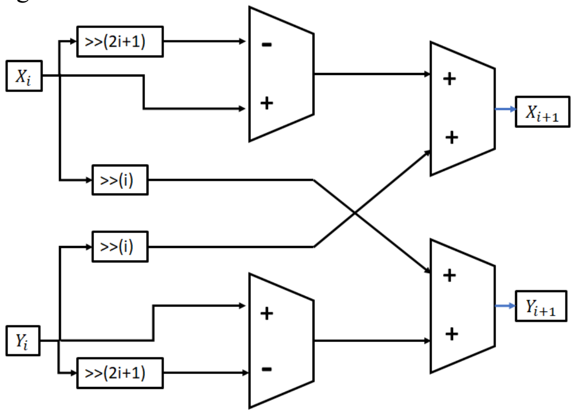

Fig. 4. Block diagram of Scaling-Free CORDIC

This argument reduction technique improves the latency of the method. Particularly, one may jump over several stages by predicting the output at the end of the skipped stages [4]. Typically, more than one computational path is possible, particularly at the early stages. This is because the initial angles are larger than that of the last stages. These computational paths can be pre-computed, and the results can be assigned using multiplexers. Probable combinations of output in the earlier stages are still few and can be estimated by using multiplexers. Hence the first 3 stages are skipped and the output at the end of the 3rd stage is obtained. However, the argument reduction technique requires a variadic scale factor, therefore scale factor multiplication is not avoided completely [4]. Nonetheless, this technique reduces a significant amount of latency (from 12 to 9 iterations as we see in the result section for state-machine based design). The consequence of not reaching the desired angle due to the insufficiency of the range of convergence is to repeat iterations for the rest of the angular gap. The angular gap means the remaining angle to the desired angle that the range of convergence could not cover. Thus, double and even triple latency may occur. Domain folding and the argument reduction techniques are critical in this regard.

The angle set is {36.87/16.26/0, 7.125/0, 1.789, 0.895, S*0.112} where S is an integer in a range from 0 to 8 [9]. The range of convergence is (-57.57, 57.57). Thus, to cover the entire space, quadrant domain folding can be adopted. Domain folding occurs at the first stage. The computation of each phase assumes different strategies.

Friend angles: Any group of angles that have identical magnitude is considered friend angles [9]. For instance, in Cartesian coordinate R = 4 + 3i with the phase of 36.869 and R = 5 with the phase of 0 are friend angle because they have the same magnitude of 5. Thus, all angles in Figure 2 are friend angles since they all have the same magnitude of 1. The identical magnitude is essential for the consistency of the system because different magnitudes impose divergence in power gains and result in different scale factors that make the system even more complicated.

Redundant CORDIC: A Conventional rotator moves a vector in either direction: clockwise or counterclockwise. However, rotation with a large angular gap may require the next angles to cover up the unnecessarily extensive jump in a reverse direction. In those cases, holding the position instead of rotating is advantageous. Thus, the direction of rotation choices are $d_i = \{-1, 0, 1\}$. However, adding one more “direction”, that is 0 or no angular movement, yet regulated with appropriate power gain to attain consistency with the other directions, reduces the maximum frequency of the design.

Nanorotator: Rotation by a sufficiently small angle can be approximated further. Given $R = A + Si$, a rotation is sufficiently small if S<<A, therefore $\alpha = \arctan(S/A) \approx S/A$. The other rotators are the same as previously explained for CORDIC algorithms.

III. Implementation

To ensure a successful implementation, we have chosen the workflow depicted in Figure 5. We implemented the algorithm as a MATLAB script and simulated the code, taking advantage of functions that ultimately are not feasible in the hardware platform, such as floating-point, exponentiation, and numerous other operators. This reduces design effort and completion time. Then, we verify the result and evaluate the performance in the software domain, which gives us insight into the possible performance in the hardware domain. We write the register transfer level (RTL) implementation of the design in Xilinx ISE using Verilog HDL. Then, we designed the testbench to simulate the RTL implementation and confirm its functionality. Since all variables in the hardware are integer, we map the hardware simulation results to that of MATLAB simulation for consistency. The hardware platform we utilized is the Xilinx Virtex-6 ML605 FPGA. Next, we verify the FPGA’s functionality using an Integrated Logic Analyzer (ILA). We store and extract the output values from the ILA signal analyzer, compare the results with that of RTL simulation, and assess the data for evaluation.

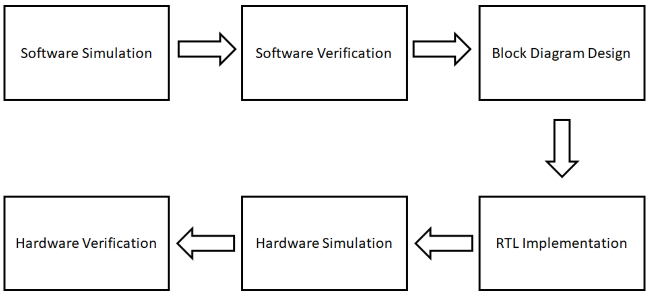

Fig. 5. Workflow

Here, we describe the stages of our implementation:

Stage 1: The domains folding technique, using 8 domains, retains resource efficiency for the system. Additionally, a higher maximum frequency is achieved using one-directional rotations. Folding the coordinate space into several domains leads to a smaller convergence range, but when the number of domains reaches or exceeds 16, complicated operations such as multiplication by $1/\sqrt{2}$ are required. Thus, we use 8 domains to ensure simplicity. The assignment of the initial vector can be done by trivial swapping between imaginary and real parts and negation as seen in Table 1. The angular range of each domain is 45 degrees, which is within the convergence range of the angle set in the counterclockwise direction: it enables one-directional rotation for the next stage.

Table 1. Eight domain folding coordinate assignment

Domain	X	Y
0-45	X	Y
45-90	Y	X
90-135	-Y	X
135-180	-X	Y
180-225	-X	-Y
225-270	-Y	-X
270-315	Y	-X
315-360	X	-Y

Stage 2: The first rotation yields 3 phase options with angles {36.87, 16.26, 0}. All rotation coefficients have the same magnitudes that imply the same power gain/scale factor. We use coefficients, with an angular magnitude of 1.5625. Hence, R = 1.25 + 0.9375i for phase of 36.869 degrees, R = 1.5 + 0.4375i for phase of 16.26 degrees, and R = 1.5625 + 0.0i for phase of 0 degrees. Equation (12) is modified to optimize the hardware implementation for the above three angles such as:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 + 2^{-2} & -1 + 2^{-4} \\ 1 - 2^{-4} & 1 + 2^{-2} \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 2^{-1} + 1 & -2^{-1} + 2^{-4} \\ 2^{-1} - 2^{-4} & 2^{-1} + 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad (14)$$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 2^{-1} + 1 + 2^{-4} \\ 2^{-1} + 1 + 2^{-4} \end{bmatrix}^T \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad (15)$$

Equations (13), (14), and (15) can be implemented with just logic shifter and adder.

Resource sharing eliminates redundancy in resource usage. In Figure 6, we use 6 logic shifters as some operators share the same logic shifter's output. The switching rules for the multiplexers are shown as numbers {0, 1, 2}, where {0, 1, 2} encodes the jump angles {36.87, 16.26, 0}, respectively. The architecture of stage 2 is somewhat complex, which may impact the maximum frequency of the hardware implementation. Thus, having a one-directional rotation approach shortens the longest path of the architecture, and in our case, we have 3 rotational options instead of 5 in the regular mode, which shrinks the area usage and improves the speed of this segment.

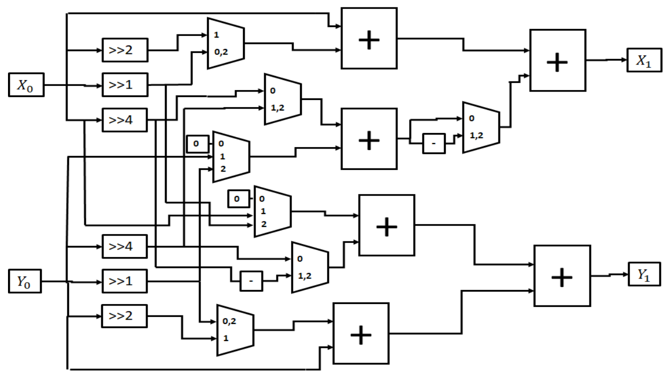

Fig. 6. Stage 2 block diagram

Stage 3: In this stage, we adopt redundant CORDIC to eliminate several rotations and guarantee convergence provided by the remaining angles in the set. The coefficients of this rotator have an angular magnitude of 1.0078125: R = 1 + 0.125i for a phase of 7.125 degrees, and R = 1.0078125 + 0.0i for a phase of 0 degrees. Equation (12) turns into:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & -d * 2^{-3} \\ d * 2^{-3} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad (16)$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 + 2^{-7} \\ 1 + 2^{-7} \end{bmatrix}^T \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad (17)$$

Hardware implementation of equations (16) and (17) requires just 2 logic shifters per coordinate. The direction d in (16) can be {-1, 1}. The rotation by 0 degrees (described by equation (17)) is equivalent to a no-rotation choice. Such redundancy is tolerable because we end up with three jumping options similar to that of the previous stage. No degradation in the maximum frequency of the design results from this architecture in that regard. The coefficients in stages 2 and 3 ensure consistency of the scale factor as the friend angle's condition is fulfilled.

The block diagram for this stage, Figure 7, shows 4 multiplexers where two of them have the tag numbers. Tag 0 indicates the halting condition for no rotation of the current vector. Note that the appropriate power gain is imposed. Tag 1 indicates either clockwise or counterclockwise rotation set by the sign of the active phase z.

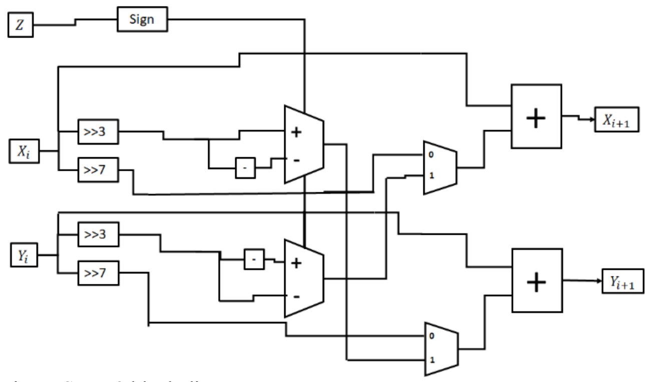

Fig. 7. Stage 3 block diagram

Stage 4: Entering this stage, the residual angle gap's range is 3.58, which is within the range of convergence of the remaining angle in the set. Hence, this stage requires no redundant CORDIC rotation: $d = \{-1, 1\}$. In this stage, we adopt conventional CORDIC architecture at the 5th iteration. The coefficient is R = 1 + 0.03125i for a phase of 1.789 degrees, and the hardware compatible computation is given by equation (18):

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & -d * 2^{-j} \\ d * 2^{-j} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (18)$$

The hardware implementation requires two shifters, see Figure 8. For this stage i = 3 and j = 5.

Stage 5: We reuse conventional CORDIC architecture similar to the previous stage but with R = 1 + 0.015625 for a phase of 0.895 degrees. The hardware compatible computation is also given by equation (18), but here i = 4 and j = 6. The block diagram is identical to that of stage 4, which is shown Figure 8.

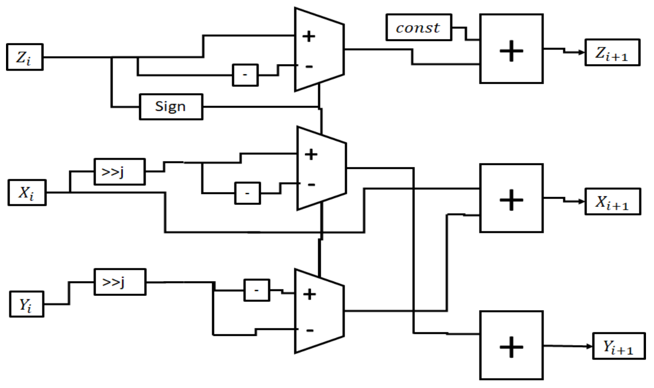

Fig. 8. Stage 4 and 5 block diagrams

Stage 6 (last stage): We are left with a residual angle gap, whose range is 0.875. For this reason, the rotator takes advantage of a nanorotator approximation with a non-constant, adaptive scaling coefficient: R = 1+ (S * 0.001953125i) for variadic phase, where S ∈ [0,8]. Considering the allowed values of S, the range of convergence is (-0.895, 0.895). The hardware compatible version of the coefficient is given in equation (19) and its architecture is shown in Figure 9. The stage 6 implementation requires extra logic: a scale decoder and an attenuation block.

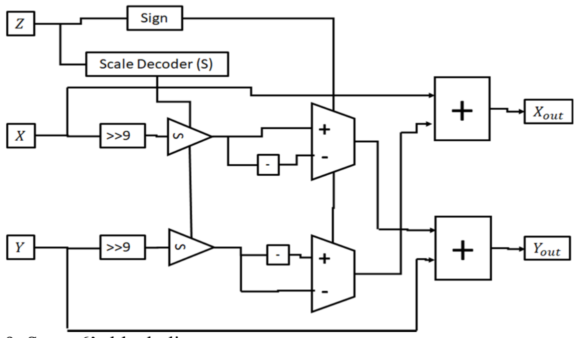

Fig. 9. Stage 6's block diagram

$$\begin{bmatrix} x_5 \\ y_5 \end{bmatrix} = \begin{bmatrix} 1 & -d * 2^{-9} * S \\ d * 2^{-9} * S & 1 \end{bmatrix} \begin{bmatrix} x_4 \\ y_4 \end{bmatrix} \quad (19)$$

The scale decoder block determines the magnitude of the adaptive coefficient of S using the remaining phase, to make the residual of Z as close to 0 as possible. 9 combinations of S are obtained, see Table 2.

Table 2. Remaining angle range for S

Range	S	Range	S
(0, 0.0988]	0	(0. 4998, 0.5987]	5
(0. 0988, 0.1977]	1	(0. 5987, 0.6976]	6
(0. 1977, 0.2966]	2	(0. 6976, 0.7965]	7
(0. 2966, 0.3955]	3	(0.7965, 0.895]	8
(0. 3955, 0.4998]	4		

The attenuation block in Figure 9, depicted as a triangular block with an "S" tag, multiplies the adaptive scale S to the shifted coordinate variables X and Y as defined in equation 19. Here, we use a regular multiplier.

The FPGA implementation integrates all these stages in series to complete the design. Given the combination of coefficients of all stages, the cumulative scale factor is K = 1.5757. Hence, we modify the initial vector in stage 1 by pre-dividing the values by this scale factor, which eliminates the other extra multipliers/dividers.

IV. RESULTS

We evaluate the design's performance by measuring the latency, resource usage, logic operator utilization, SFDR, and maximum frequency. These parameters provide trade-off considerations for a target application with specific requirements. For the sake of comparison, we provide values for three different CORDIC-based DDS implementations with and without a pipeline in the architecture. These are conventional CORDIC, modified Scaling Free CORDIC [3], and our proposed design.

Table 3 shows resource utilization based on the number of LUTs, FFs, and memory for a given target device. Here, the LUTs are Xilinx logic blocks. We use ROM as memory: here its usage is measured in bits. In the table, CORDIC represents the conventional CORDIC (it stores 15 phases for the angle set and assumes 16 bits of variable width). SF-CORDIC stands for the modified Scaling Free CORDIC algorithm. The "P" next to the algorithm's name indicates a pipelined version. The initiation interval of every pipelined algorithm is 1, meaning the module can take inputs every clock cycle with no extra delay.

Table 3. Resource utilization

Algorithm	LUT	FF	ROM
CORDIC	764	84	240
CORDIC P	2506	840	
SF-CORDIC	479	98	96
SF-CORDIC P	835	340	
Proposed	400	140	
Proposed P	498	206	

Table 4 presents the utilization of logic operators. Mult, Add, Comp, Mux, and Shift stand for the multiplier, adder, comparator, multiplexer, and logic shifter. All these logic operators run with variables of 16 bits. The logic shifter accepts the variadic length of shift argument.

Table 4. Logic operator usage

Algorithm	Mult	Add	Register	Comp	Mux	Shift
CORDIC	1	7	5	4	21	2
CORDIC P	1	100	65	19	60	
SF-CORDIC		10	6	14	58	4
SF-CORDIC P		28	63	19	49	
Proposed	2	16	31	20	42	
Proposed P	2	24	68	22	32	

In Table 5, the values of SFDR are specified in dB. We compute the SFDR by using the results obtained from the ILA signal analyzer. Iteration in Table 5 indicates the number of rotations, this number is equal to the number of phases in the set. Latency is specified in the number of clock cycles. It represents the overall delay, in clock cycles, due to iterations and additional strategies such as domain folding and argument reduction techniques.

Table 6 lists the maximum frequency in MHz if implemented on a Xilinx Virtex-6 FPGA. The first column shows the maximum frequency for State-Machine (SM) based DDS and the second one lists that of the pipelined version.

Table 5. SFDR, iteration and overall latency

Algorithm	SFDR	iteration	Latency
CORDIC	92.7394	15	17
SF-CORDIC	56.8218	6	9
Proposed	72.2068	5	6

Table 6. Maximum Frequency

Algorithm	Max frequency	Max Frequency P
CORDIC	180	240
SF-CORDIC	226	354
Proposed	211	251

In terms of resource utilization, the proposed design provides an improvement over the existing designs both in pipelined and SM versions. Conventional CORDIC occupies the largest memory usage due to more angles in the set. Although it uses no memory for the pipelined version, the high number of iterations makes the the increment of the other resources enormous. Memory is no longer needed because every stage is implemented separately and is active simultaneously, hence each stage is pre-assigned with a constant angle. SM based SF-CORDIC has the lowest resources compared to our design, but its pipeline-base is less efficient as it employs more iterations than our design. Our SM based design doesn't need any iteration because each stage employs a different type of rotator, which makes the resource usage close to that of a pipeline-based design. Our design occupies the smallest area and provides an energy consumption advantage.

SM based conventional CORDIC has the least overall logic operators, while our proposed design compares positively to the pipelined SF-CORDIC. The pipelined version of a conventional CORDIC uses significantly more logic operators compared to the SM based one because 15 iterations that run on a set of resources are expanded into 15 identical sets of resources. The same explanation holds for the logic operator usage in the SM based and pipelined version of the proposed design. Here, the synthesis tool optimizes the resource allocation by substituting a shifter for a concatenation operator due to constant bit shifting. Consequently, the number of logic shifters may not be the same as shown in the block diagrams.

Low latency is desired to enhance the throughput and efficiency of the communication system because delay in the system slows down quantum feedback - a bottleneck and great challenge for quantum-enhanced communication systems. With a latency of just 6 clock cycles, the proposed design is superior to the other algorithms. Although the number of iterations of SF-CORDIC is very close to that of the proposed design, there is an extra delay of 3 clock cycles due to a required additional compensation to the rotation.

The SF-CORDIC has the highest maximum frequency due to one-directional rotation. Our design has a moderate maximum frequency for its implementation. The conventional CORDIC achieves the highest SFDR value due to the high number iteration.

Our design achieves moderate SFDR yet the lowest latency, with approximately 20 dB SFDR and 64% latency reductions compared to that of the conventional CORDIC design.

V. CONCLUSION

In conclusion, we report a new memory free low latency DDS architecture. Although, complex value computations could be employed to calculate the trigonometric equations, but those calculations are computationally difficult in hardware and energy inefficient. To avoid calculation-related inefficiencies, the common approach is to use a Look-Up Table (LUT) with phase being the input and amplitude being the output. To generate a desired smooth radio-frequency signal small-step quantization is needed, requiring a larger LUT. The LUT requirement leads to an increase in memory usage and may lead to the reduction of the maximum frequency of the FPGA design which would also limit modulation capabilities.

On the other hand, a CORDIC technique offers low complexity and memory-free trigonometric calculation approach, at the expense of extra latencies to complete the computation. We use a pipelined approach to shorten latency. In this design, the sinusoidal wave amplitude is obtained every cycle, in order to maximize our modulation capabilities. To make a quantum measurement enhanced transceiver, we choose the modulation scheme which includes choosing the number of states M, the frequency, and the initial phase detuning between the adjacent states and other communication parameters. All M states are being prepared in parallel at all times, and the active output state is picked according to the encoding and measurement protocols. Because M could be quite large (up to 16 in our implementation) the low-resource usage DDSs are essential for this purpose. In communication links, sensitivity is often measured as the probability to receive an erroneous symbol with certain energy at the receiver. Classical receivers have a sensitivity limit known as the standard quantum limit (SQL). This limit arises from the inevitable shot noise on the idealized classical receiver scheme - a homodyne measurement followed by a perfect detector with no noise of its own and with the 100% detection-efficiency. The SQL is accessible only through quantum measurement. With the help of the described DDS, we have implemented a quantum-measurement telecommunication testbed and demonstrated that the sensitivity of a telecommunication channel is better than the SQL for many different modulation protocols, including quantum-measurement specific modulation protocols, described elsewhere [10].

We intend to use modulation schemes that require a simultaneous phase and frequency modulation. Our novel design achieves the shortest latencies, maximizes modulation capabilities, and uses the minimal footprint compared to other CORDIC-based DDSs.

ACKNOWLEDGMENT

This work is partially supported by National Science Foundation USA through ECCS 1927674.

REFERENCES

[1] I.A. Burenkov, M.V. Jabir, N.F.R. Annafianto, A. Battou, and S.V. Polyakov, "Experimental Demonstration of Time Resolving Quantum Receiver for Bandwidth and Power Efficient Communications", Proc. of *CLEO Conf. on Laser Science to Photonic Applications*, California, USA, 2020.

[2] P. Saravanan and S. Ramasamy, "Sine/cos generator for direct digital frequency synthesizer using pipelined CORDIC processor," Proc. of *Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pp. 1-6, Tiruchengode, 2013.

[3] R. Xin, X. Zhang, H. Li, Q. Wang, and Z. Li, "An Area Optimized Direct Digital Frequency Synthesizer Based on Improved Hybrid CORDIC Algorithm," Proc. of *Int. Workshop on Signal Design and Its Applications in Communications*, pp. 243-246, Chengdu, China, 2007.

[4] Y. Xue and Z. Ma, "Design and Implementation of an Efficient Modified CORDIC Algorithm," Proc. of *IEEE Int. Conf. on Signal and Image Processing (ICSIP)*, pp. 480-484, Wuxi, China, 2019.

[5] M.M. Anas, R.S. Padiyar, and A.S. Boban, "Implementation of Cordic Algorithm and Design of High Speed Cordic Algorithm," Proc. of *Int. Conf. on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pp. 1278-1281, Chennai, India, 2017.

[6] Y.S. Gener, S. Gören, and H.F. Ugurdag, "Lossless Look-Up Table Compression for Hardware Implementation of Transcendental Functions," Proc. of *IFIP/IEEE Int. Conf. on Very Large Scale Integration (VLSI-SoC)*, pp. 52-57, Cuzco, Peru, 2019.

[7] W. Shuqin, H. Yiding, Z. Kaihong, and Y. Zongguang, "A 200MHz Low-Power Direct Digital Frequency Synthesizer Based on Mixed Structure of Angle Rotation," Proc. of *IEEE Int. Conf. on ASIC*, pp. 1177-1179, Changsha, China, 2009.

[8] K. Maharatna, S. Banerjee, E. Grass, M. Krstic, and A. Troya, "Modified Virtually Scaling-Free Adaptive CORDIC Rotator Algorithm and Architecture," in *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, pp. 1463-1474, 2005.

[9] M. Garrido, P. Källström, M. Kumm, and O. Gustafsson, "CORDIC II: A New Improved CORDIC Algorithm," in *IEEE Tran. on Circuits and Systems II: Express Briefs*, vol. 63, pp. 186-190, 2016.

[10] I.A. Burenkov, O.V. Tikhonova, and S.V. Polyakov, "Quantum Receiver for Large Alphabet Communication," *Optica*, vol. 5, pp. 227-232, 201