IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 11, Issue 6, Ver. IV (Nov.-Dec .2016), PP 05-13
www.iosrjournals.org

Numerically Controlled Oscillator for Software Radio Applications

Ishmael Zibani[1], Kelebaone Tsamaase[2], Kagiso Motshidisi[3], Pran Mahindroo[4]
[1,2,3,4]*(Electrical Department, University of Botswana, Botswana)*

Abstract: *A Digital Down Converter (DDC) performs the critical frequency translation needed to recover the information from a digitized modulated signal. In terms of system performance, the critical component in digital down conversion is the Numerically Controlled Oscillator (NCO). The NCO synthesizes a range of frequencies from a fixed time base. It generates sine values from a ROM lookup table (LUT). The size of the LUT is one of the main concerns as it has to store enough sine samples in order to generate sine values of varying frequency. The size of the LUT will therefore limit the bandwidth of the sine output. A progression of state approach for the NCO is proposed. As we advance through the states, outputs are generated to represent sine and cosine values. The state machine is clocked with a varying clock signal in order to generate various frequency outputs. It is known that only a quarter of the sine/cosine function need be stored, and the rest of the sinusoid can be generated by applying appropriate sign convention. In this study, it is demonstrated how this can be achieved.*
Keywords: *DDC, NCO, ROM-LUT, progression of state.*

I. Introduction

In Software Radio, much of the signal processing is done in software using a reconfigurable hardware platform. The DDC performs the frequency translation to recover the original signal. [1], [3], [10]. Inside the DDC, the digitized modulated input from the Analog to Digital Converter, ADC is mixed with locally generated sinusoid to shift the spectrum of the signal. The mixed signal has to be filtered to isolate the portion of the spectrum containing the signal of interest. The filter typically has to be a narrow-band filter with a fairly high rejection of unwanted spectrum. This translates to an expensive filter if it is done at the input sample rate. Instead, a multi-rate approach can be used, in which the signal is first decimated to a much lower sample rate using a less computationally intensive filter. Then the signal is cleaned up with a second more complex filter working at the decimated sample rate [2], [5], [8], [9]. See Figure 1.

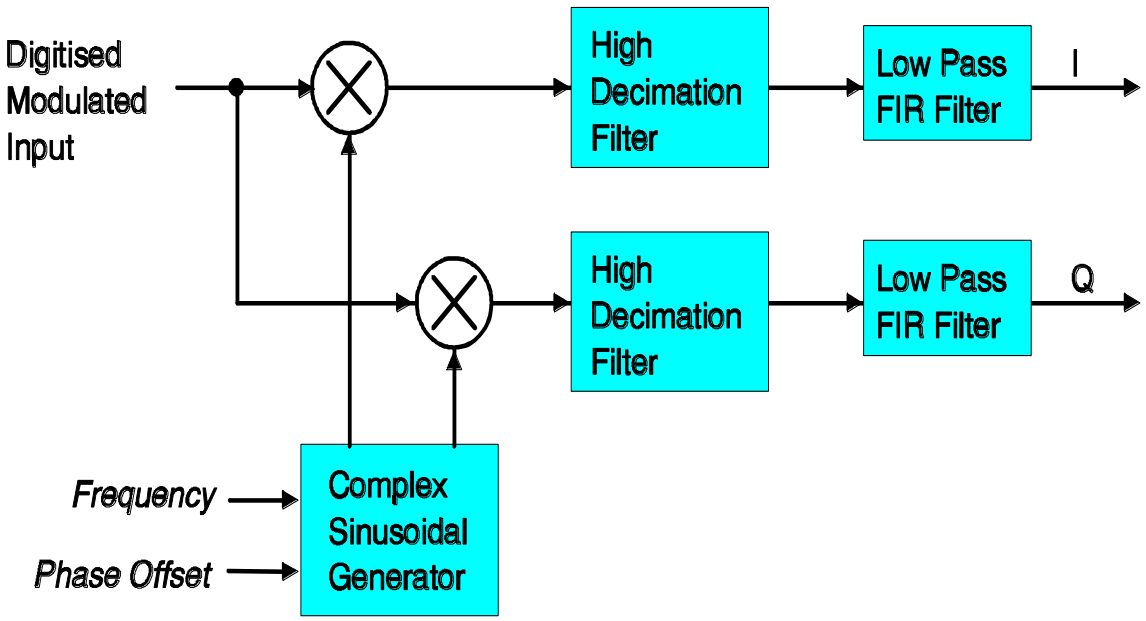

Figure 1. Block diagram of a Digital Down Converter

The rest of the paper is divided as follows: In section II, a brief discussion on basic operation of a NCO is provided. The motivation for the proposal is given in section III. In sections IV and V, a detailed outline and design of the proposed system is given. Section VI through to VII, we present a design example and discuss the results, ending with a conclusion in section VIII.

II. Numerically Controlled Oscillator

The NCO, sometimes called a local oscillator generates digital samples of two sine waves precisely offset by 90 degrees in phase creating sine and cosine signals [8], [10], [11], (See Figure 2). It uses a digital phase accumulator (adder) and sine/cosine look-up tables. The ADC clock is fed into the local oscillator. The digital samples out of the local oscillator are generated at a sampling rate exactly equal to the ADC sample clock frequency, fs. Since the data rates from these two mixer input sources are both at the ADC sampling rate, fs, the complex mixer output samples at fs. The sine and cosine input from the local oscillator create in-phase and quadrature (I and Q) output that are important for maintaining phase information contained in the input signal. (see figure 1). The decimating low pass filter accepts input samples from the mixer output at the full ADC sampling frequency, fs. It utilizes digital signal processing to implement a FIR (finite impulse response) filter transfer function. The filter passes all signals from 0 Hz up to a programmable cutoff frequency or bandwidth, and rejects all signals above that cutoff frequency. The digital filter is a complex filter which processes both I and Q signals from the mixer [2], [4], [7].

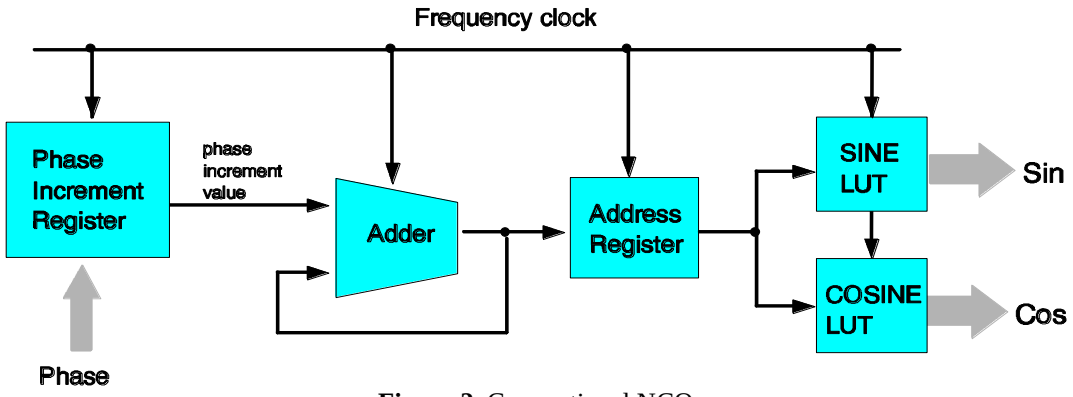

Figure 2. Conventional NCO

III. Motivation For The Proposed NCO

The conventional NCO uses a ROM LUT to store the required sine/cosine samples. [1], [5], [12]. The ROM has a fixed structure and therefore the designer cannot take advantage of redundancy in a system they are designing. Furthermore, the sine/cosine function follows a predetermined sequence, therefore random addressing is not required. In a ROM, the addressing is performed by an address decoder which consists of multiplexers (Figure 3). To store *n* words of length *w*, the ROM will undoubtedly have *n* x *m* memory cells. It will also consist of *n* x *s*, *m* multiplexers, where $s = \frac{\log_2 n}{\log_2 2} = \frac{\ln n}{\ln 2}$. With the proposed method, only *s* memory cells are required, together with some combinational logic to form the required output.

A simple example is given in Figure 3. One quarter of cosine and sine samples are to be stored. The sample rate is 30^0 and the word length is 4bits. The multiplexer (Figure 3a) consists of 7 gates. So to produce sine samples of word length 4, 16 memory cells and 7x4 logic gates are required. Therefore to produce both sine and cosine samples, 56 gates and 32 flipflops are required.

Using the proposed method, we will use 2 flipflops and 7 combinational logic. Using the ROM LUT approach, the number of memory cells will grow exponentially as compared to the proposed method. The decoder will also become very complex.

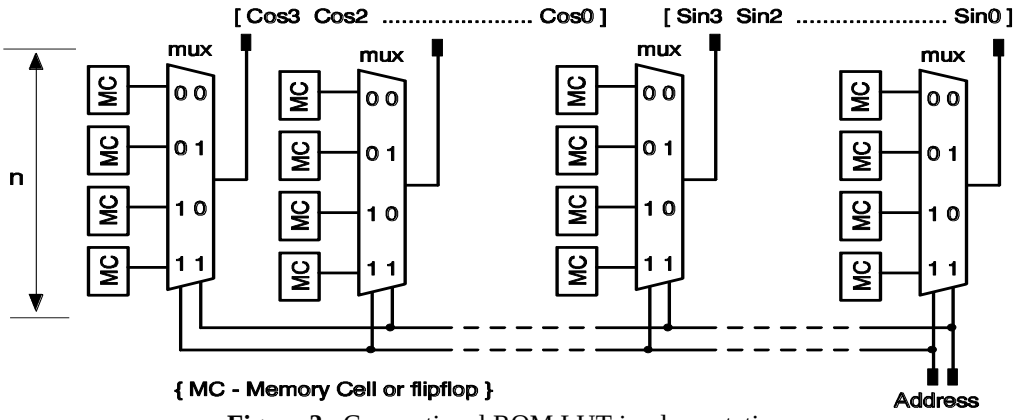

Figure 3a Conventional ROM LUT implementation

Present State		Next State		J K Controls				Outputs								
				$J_A K_A$		$J_B K_B$		Sin Output				Cos Output				Pulse
Angle	State Assignment	Input N		Input N		Input N										
	AB	0	1	0	1	0	1	Sin0	Sin1	Sin2	Sin3	Cos0	Cos1	Cos2	Cos3	
0^0	(S0) 00	(S1) 01	(S1) 01	0X	0X	1X	1X	0	0	0	0	1	1	1	1	1
30^0	(S1) 01	(S2) 11	(S0) 00	1X	0X	X0	X1	1	0	0	0	1	1	0	1	0
60^0	(S2) 11	(S3) 10	(S1) 01	X0	X1	X1	X0	1	1	0	1	1	0	0	0	0
90^0	(S3) 10	(S2) 11	(S2) 11	X0	X0	1X	1X	1	1	1	1	0	0	0	0	1

Figure 3b. Truth table for a 4bit sine-cosine generator

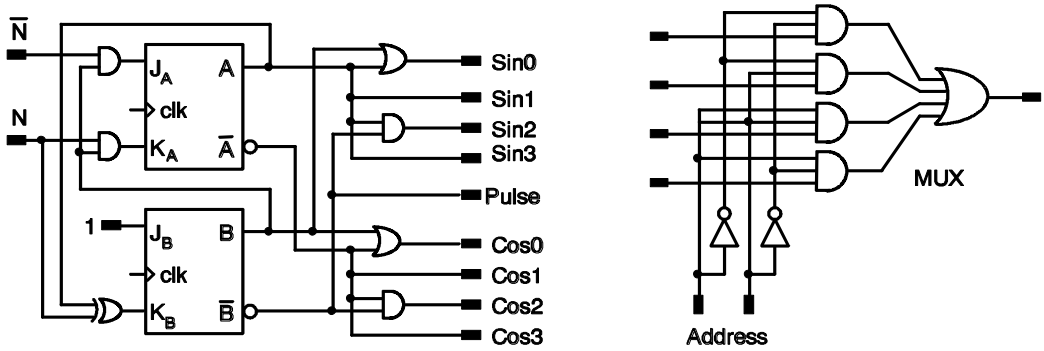

Figure 3c. Logic Diagram for Figure 3b and Multiplexer

IV. The Proposed NCO

The block diagram of the proposed NCO is given in Figure 4. Here, a ¼ Sin_Cos Function is used instead of a Sin/Cos LUT. The three components driving a ROM LUT (i.e., phase increment register, adder, and address register) are replaced by the frequency synthesizer. The additional logic is used to generate appropriate sin_cos values in the other three quadrants. Compare figures 2 and 4.

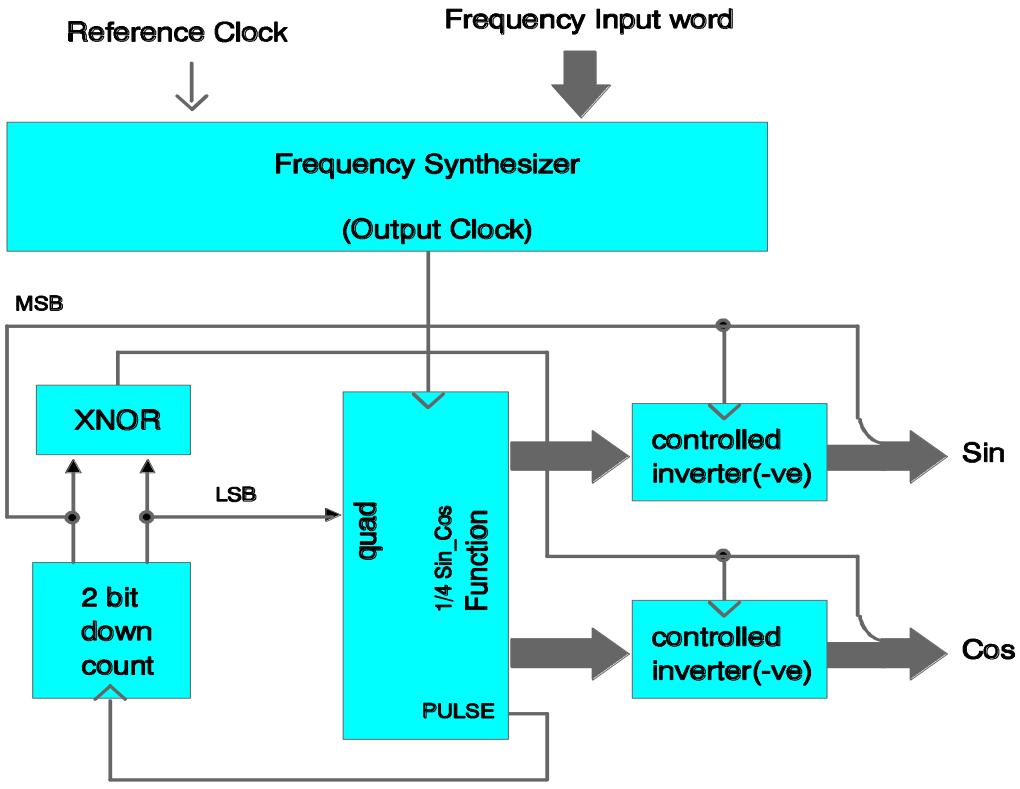

Figure 4. The proposed NCO

V. Design Of Sin-Cos Using Progression Of States

The ¼ Sin_Cos function can be implemented using a state machine, with appropriate outputs at each state. Each time we go through a state cycle, we cover the one quadrant (shaded in Figure 5). The output *PULSE* (Figure 4) signifies the beginning of a new state cycle, hence a new quadrant. With proper sign convention, the same samples are reused in other quadrants.

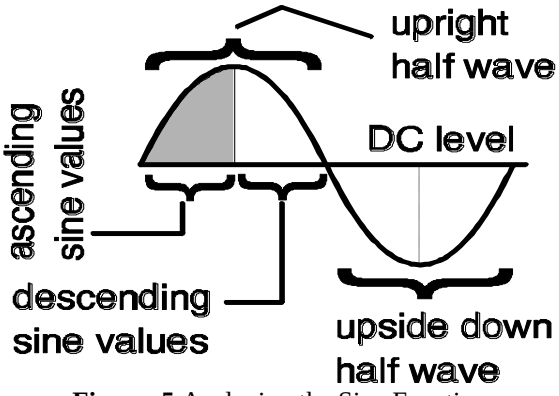

Figure. 5 Analyzing the Sine Function

Figure 6b shows the flow chart for the Sine Function. The first 2 quadrants are defined by counting up and down the states. A reflection along the dc line will define the last 2 quadrants. See Figure 6a.

Suppose we are at the beginning of the first quadrant. dc will be (externally) set to 1 to signify that we are in the 'positive' quadrants. We start counting at the beginning of the quadrant and continue counting up as long as we are in the first (odd quadrant), until we reach the end of that quadrant. Then we count down and continue like so as long as we are in the second (even) quadrant, until we reach the beginning of the next quadrant. At this point, dc is set to 0, so that repeating the above procedure will produce sine values for the third and fourth quadrants. Then the whole cycle repeats again. A similar argument goes with the cosine function.

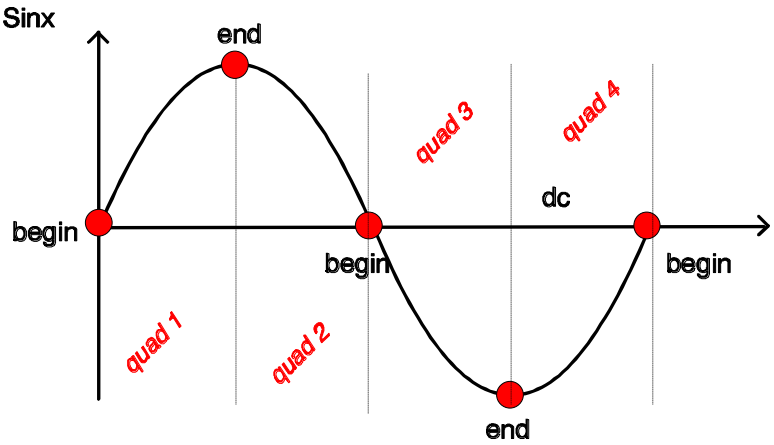

Figure. 6a. The Sine Function

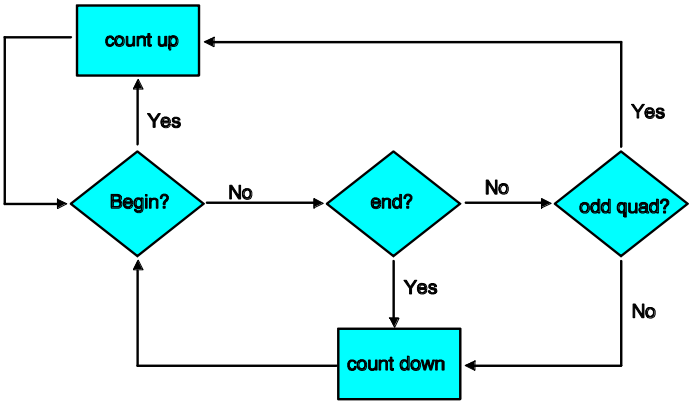

Figure 6b. Flow Chart.

Figure 7a shows the behavior of sine and cosine functions in all four quadrants. If we assume absolute values for sine and cosine, we can see that when sine is increasing, cosine is decreasing but with different increments. Therefore inverting sine bits won't generate cosine bits. So sine and cosine values are generated as previously shown in Figure 3.

The output of the 2 bit down counter (figure 4) identifies the current quadrant. dc(sin) = 1 in the first 2 quadrants. Therefore dc(sin) = $MSB.LSB + MSB.\overline{LSB} = MSB$. dc(cos) =1 in the first and last quadrant. Therefore, dc(cos) $= MSB.LSB + \overline{MSB}.\overline{LSB} = \overline{MSB \oplus LSB}$.

For argument sake, let's assume that we store sine samples for the whole 360^0. Top of Figure 8 shows some of the original samples. In the ROM LUT approach, a change of frequency is achieved in altering the phase angle (also see Figure 2). For example, to increase the frequency of the output sine wave, the phase angle is increased so that some of the samples are 'dropped'. That is we read samples a,c,e,f,etc instead of a,b,c,d,e,f,etc. (shown middle Figure 8). To achieve the same frequency with the proposed method, the same samples are generated much faster. For example, instead of generating samples at t0,t1,t2,t3,etc, we generate at t0, $t0 + \frac{t0+t1}{2}$, t1, $t1 + \frac{t1+t2}{2}$, t2, etc, (shown in bottom of Figure 8).

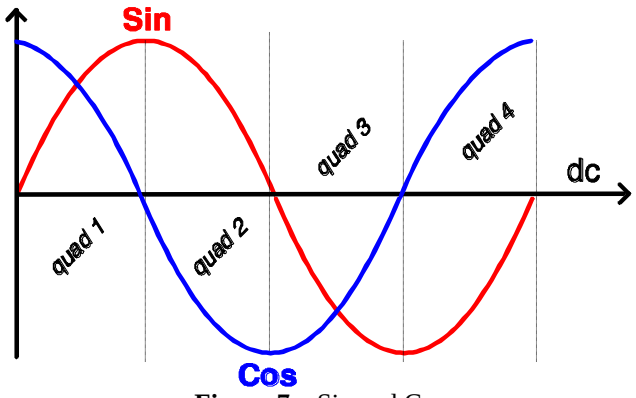

Figure 7a. Sin and Cos

Quadrant	MSB	LSB	dc (cos)	dc (sin)
first	1	1	1	1
second	1	0	0	1
third	0	1	0	0
fourth	0	0	1	0

Figure 7b. Truth Table for dc

The frequency synthesizer outputs different clock waveforms in response to different frequency input words. It basically consists of a frequency input register and a binary counter. See Figure 9. Note that the quad input inverts the sine in the first quadrant along the y axis to obtain the sine in the second quadrant. the half sine wave thus obtained is reflected along the x axis (dc) to obtain third and fourth quadrants, hence the controlled inverters operated by the dc. (See Figures 7 and 9). The shape of the output clock, clk_O is a pulse. Since it is used for clocking purposes, there is no need to add a square-wave shaper. The frequency of clk_O, f_{out}, is given by, $f_{ref}/(w_v + 1)$, where f_{ref} is the frequency of the reference clock and w_v is the value of the frequency input word. For example, if $w_v = 0$, then $f_{out} = f_{ref}$.

The bandwidth or frequency span of f_{out}, Δf_{out} is given by $f_{ref}\{1-(w_{\max}+1)^{-1}\}$, where $w_{\max}$ is the maximum value of the input frequency word. Finally, the bandwidth of the sine (or cosine) output is given by $\Delta f_{\sin} = \Delta f_{out}/\{(s+1)+3s\}$, where s is the number of non-zero samples in the first quadrant. $s_{\max} = 2^n$, where n is the word length for ¼ sine or ¼ cosine.

Again note that the dc also forms part of the sin and cos output. However, they are generated outside the ¼ sin_cos function to minimize its complexity.

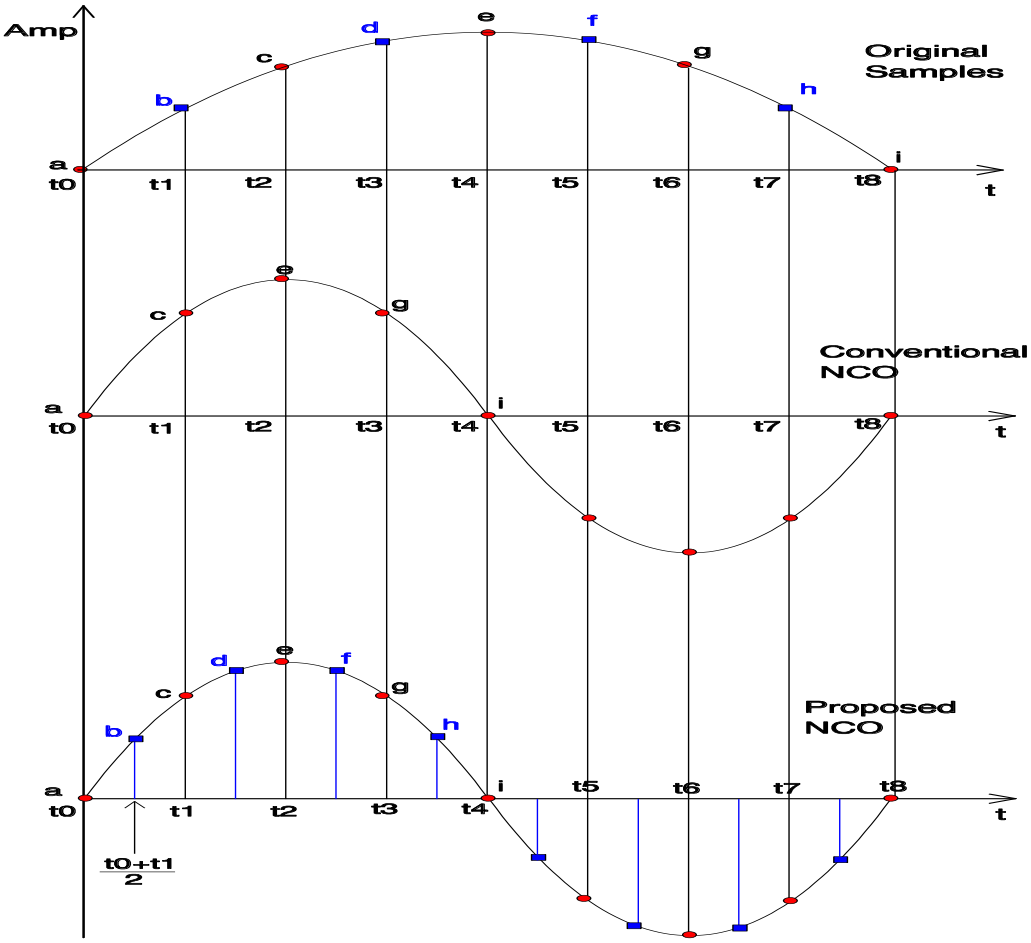

Figure 8 Generating Sine samples

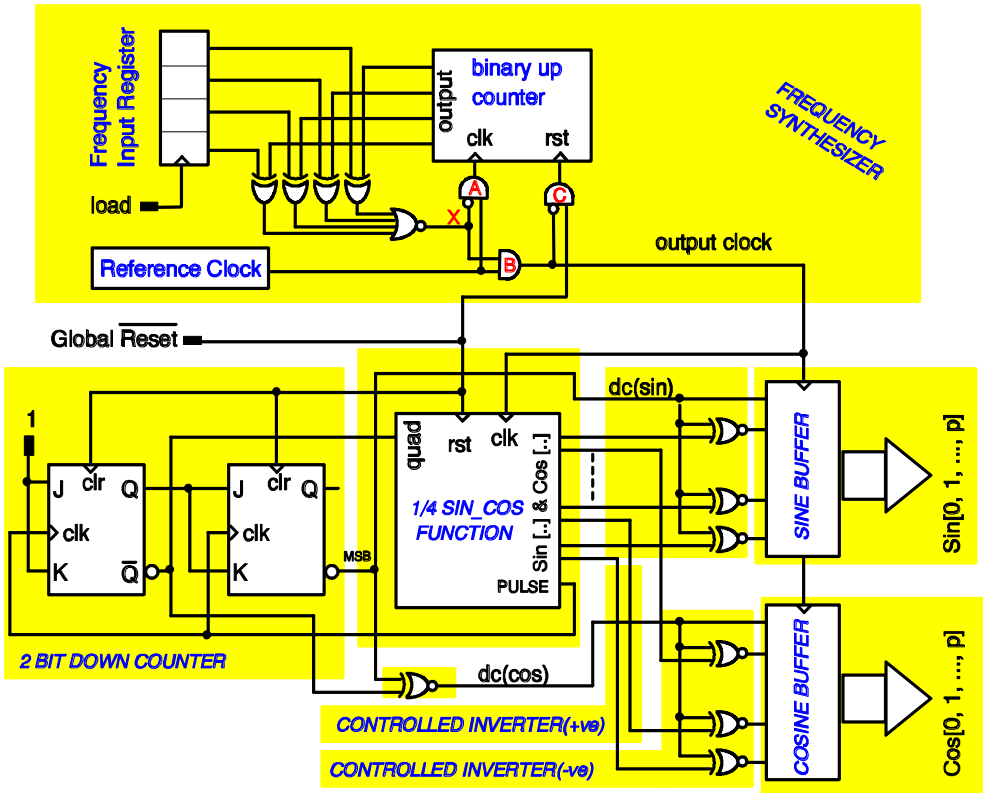

Figure 9 Logic diagram of the proposed NCO

VI. Altera Design Example

For software radio applications, the word length *n+1*, of each complete sine/cosine sample is 16bits [6]. For the following example, n+1=16 was used. As pointed out earlier, we design for a quarter, i.e. 15bit, so n=15 (word length for the quarter). The 16th bit is supplied externally and it is also used to differentiate between positive and negative quadrants. For n=15, we have 2^{15} samples (including the all zeros sample). If we take the maximum value of the sine function to be 1, then the least significant bit, LSB will have a weight of $\frac{1}{2^{15}-1}$. This corresponds to sample rate/angle resolution of $\{\frac{90}{(2^{15}-1)}\}^0$. For our demonstration purposes, this resolution was found to be too high. An angle resolution of 1^0 was used in this example.

For $0^0<\theta<90^0$, $0<\text{Sin}\theta<1$. So we need to multiply $\sin\theta$ with a suitable multiplying factor m_f, so that we can represent sine values in binary form. For word length n, $m_f = 2^n - 1$. In our example, the binary equivalent of $\sin 89^0$ is $(2^{15}-1)\text{Sin}89$ = 111111111111010. All other entries were computed in the same way. (See Figure 10).

In the final stage, the MSB is supplied externally from the two bit down counter and added to make a 16 bit word. (See Figure 9).

Present State	Next State input		θ	Outputs		
	$\overline{quad}$	quad		$(2^{15}-1)\sin\theta$ $(2^{15}-1)\cos\theta$	Sinθ (15 bit binary representation) Cosθ (15 bit binary representation)	Pulse
S0	S1	S1	0^0	0	0000000000000000	1
				32767	1111111111111111	
S1	S0	S2	1^0	572	0000010001111100	0
				32762	1111111111111010	
S2	S1	S3	2^0	1144	0000100011110000	0
				32747	1111111111110101	
S3	S2	S4	3^0	1715	0000110101100011	0
				32722	1111111110100010	
S88	S89	S87	88^0	32747	1111111111101011	0
				1144	0000100011110000	
S89	S90	S88	89^0	32762	1111111111111010	0
				572	0000010001111100	
S90	S89	S89	90^0	32767	1111111111111111	1
				0	0000000000000000	

Figure 10. State Table for ¼ Sin_Cos Function

VII. Discussion Of The Results

Figure 10 show the sine and cosine output waveforms. The results from Figure 10 were plotted using Microsoft excel to obtain the graphs in Figure 11. The positive sine and cosine were generated using ¼ sine and ¼ cosine respectively. The dc value was used to 'raise' the sine and cosine so that they appear above the x-axis. The results obtained were expected.

The Delay Matrix (Fig. 13) gives us the critical speed paths that limit the design performance. From the Delay Matrix, the longest delay is 25.8nS. If we choose to use 26nS instead, then we get a bandwidth of 38.46MHz. From $f_{\sin}(\max) = f_{out}(\max)/\{(s+1)+3s\}$, $f_{out}(\max) = f_{ref}$, so that $f_{\sin}(\max) = f_{ref}/\{(s+1)+3s\}$ = 38.46MHz /{(91)+3x90)} = 0.1MHz.

After the design has been verified, it can be migrated to an ALTERA's HardCopy device. This involves removing the reprogrammability of the original FPGA device, resulting in increased speed, reduced complexity, area and power consumption.

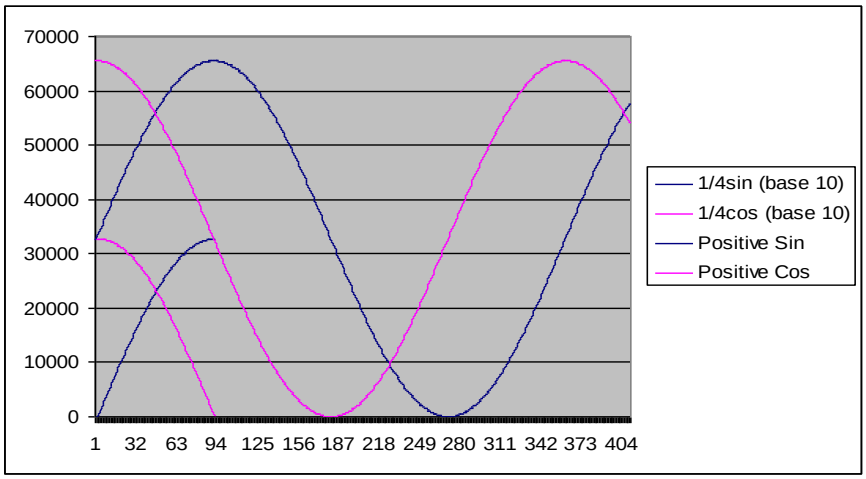

Figure 11. Graphs for sine and cosine functions

For example, to calculate the error in the sine output, will be the difference between the true sine value, $\sin_{true}$ and the 16bit binary value (reconverted to decimal for easy manipulation). The decimal equivalent of the sine binary values, $\sin_{dec}$, are also shown in Figure 10. We can calculate average % error, $\%err_{ave}$, using either sine or cosine values. Thus we have, $\%err_{ave} = \{\sum_{1^0}^{89^0} |\sin_{true} - \sin_{dec}|\}/89$. ($0^0$ and 90^0 have been left out since they give an error of zero). Using values from the complete table of Figure 10 and true sine values, $\%err_{ave}$ came to be 0.00218003%.

Angle resolution is related to the sampling rate. A high sampling rate will mean a good angle resolution but slower output speed. So during design, the angular resolution and speed trade offs must be understood. By looking at the slowest sine/cosine output required, the number of desired samples per quarter can be set. The number of (non zero) samples to be used is $\min(desired, 2^n - 1)$, otherwise duplicate samples will be stored. For this example, 1^0 sample rate was chosen for this example. This gives an accuracy of 1 part in 90 or 1.1%.

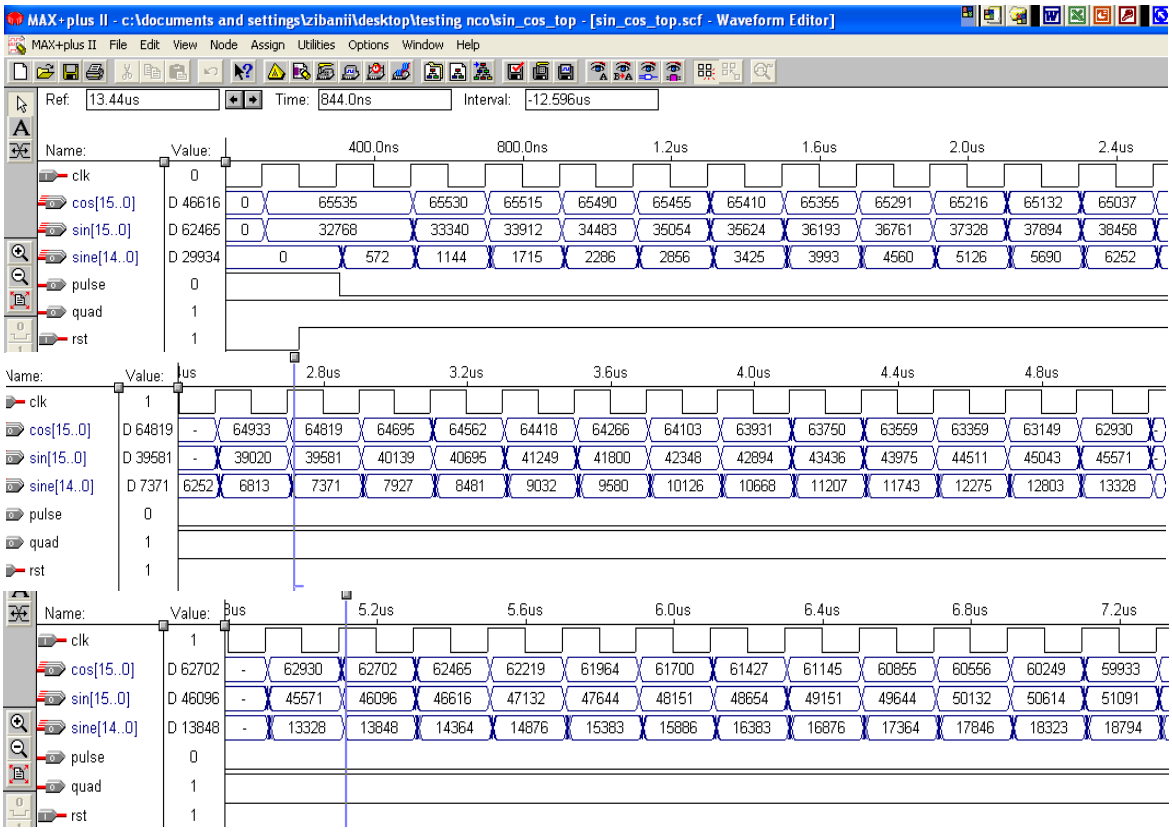

Figure 12 Sine and Cosine output waveforms.

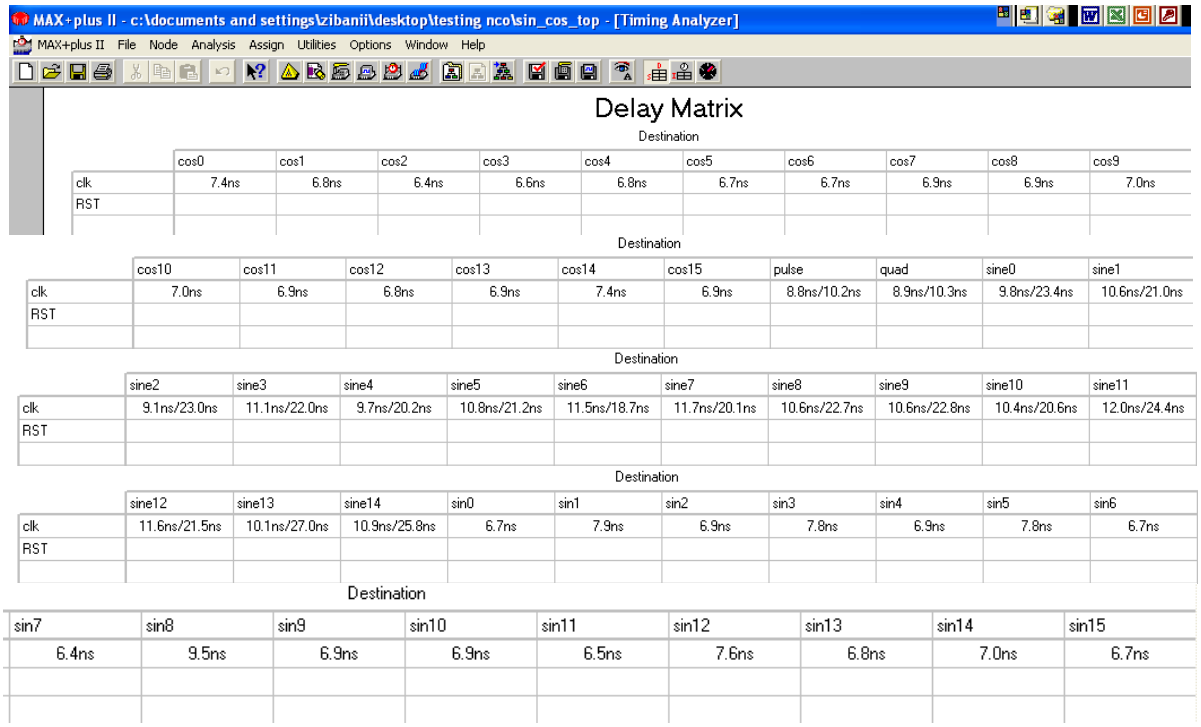

Figure 13. Delay Matrix for Sine and Cosine.

References

[1]. Nagarjun Marappa, *Design Of Digital Down Converter Chain For Software Defined Radio Systems On Fpga*, MSc diss., Western Michigan University, 2015.

[2]. Maruthi G.B, Saleem Malik Prashanth Kumar and Pratap M.S, Implementation of High Performance DUC and DDC for Software Defined Radio Applications, *International Journal of Computer Applications, Vol 110 – No. 6, Jan 2015.*

[3]. Vaidyanathan, P.P., "Multirate digital filters, filter banks, polyphase networks, and Applications : a tutorial, *Proceedings of the IEEE*, Vols. 78, no.1, pp. 56-93, Jan 1990.

[4]. A. A. Abidi, Direct-Conversion Radio Transceivers for Digital Communications, *IEEE Journal of Solid-State Circuits, Vol. 30, No. 12, December 1995, pp 1399-1410.*

[5]. T. Hentschel, M. Henker, and G. P. Fettweis. The Digital Front-End of Software, Radio Terminals. *IEEE Personal Communications, 6(4):40–46, Aug 1999.*

[6]. J. Mitola, ìThe Software Radio Architecture, *IEEE Communications magazine, Vol. 33, No. 5, May 1995, pp. 26-38.*

[7]. Henrik Ohlsson and Lars Wanhammar, "A Digital Down Converter for a Wideband Radar Receiver", Department of Electrical Engineering, Linköping University SE-581 83 Linköping, Sweden.

[8]. "Direct Digital Synthesis (DDS) technology", http://www.hit.bme.hu/people/papay/sci/DDS/start.htm.

[9]. A. R. Panda, D. Mishra and H. K. Ratha, FPGA Implementation of Software Defined Radio-Based Flight Termination System, *IEEE Trans. on, Industrial Informatics, vol. 11, no. 1, (2015). , pp. 74-82*

[10]. NOVA Engineering, "Numerically Controlled Oscillator Megafunction", ALTERA Corporation, San Jose.

[11]. Mitola, J., III, Software radios: Survey, critical evaluation and future directions, *Aerospace and Electronic Systems Magazine, IEEE , Vols. 8, no.4, pp. 25-36, April 1993.*

[12]. R. Kumar, T. M. Nguyen, C. C. Wang and G. W. Goo, Signal processing techniques for wideband communications systems, *MILCOM 1999 - IEEE Military Communications Conference, no. 1, October 1999, pp. 452 - 457*