ROM-Less Fully Digital Synthesizer Design, Based on Novel phase to Sinusoidal Interface

Hesam Pashaei, and Dr. Khalil Monfaredi
Engineering Faculty, Department of Electrical and Electronic Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran
khmonfaredi@azaruniv.ac.ir

***Abstract*—An 8-bit fully digital synthesizer is proposed in this paper. Utilizing a phase to sinusoidal code interface is the outstanding difference of this structure versus its other counterparts. In order to achieve higher operation speed an 8 bit pipeline accumulator is used in proposed structure. The phase to sinusoidal interface along with linear DAC makes the Direct Digital Synthesizer (DDS) needless of conventionally used ROM look up table memory. Meanwhile the ASIC sine weighted DAC which requires specifically handled dynamic element matching (DEM) procedure and sophisticated layout techniques to enhance the resolution in accordance with the DDS specifications is eliminated. The intrinsic thermometric nature of the utilized phase to sinusoidal converter makes the system needless of binary to thermometer block required for segmented DACs. While simplifying the DDS overall configuration, the proposed structure makes it also possible to use the sinusoidal digital codes to be compared with sampled form of original signal and hence can eliminate analog comparison in the front end stage. To evaluate the performance of the proposed circuit, it is initially simulated at system-level by Matlab and then it is implemented at logical gate level using Xilinx ISE environment and finally is practically realized by Xilinx SPARTAN 3 FPGA and its performance is evaluated experimentally.**

***Index Terms*— Phase Accumulator (PA); Digital To Analog Converter (DAC); Direct Digital Synthesizer (DDS); ROM-less DDS; Phase To Sinusoidal Interface.**

I. INTRODUCTION

Recently, Direct Digital Synthesizers (DDS) achieved an increasing popularity and more attention due to their intrinsic digital nature which introduces a distinguished effect on enhancing the circuit features like supply voltage reduction, power consumption reduction, increasing operational frequency, and reduced chip area in comparison with its analog counterpart, i.e., PLL Synthesizers, thanks to the astonishing recent advances in digital technology. DDS blocks have more frequency resolution compared to PPLs. DDS blocks have also some other advantages such as higher switching frequency response, wider frequency tuning range, and the capability to establish multi-functional modulations with higher flexibility [1]. Therefore, DDS can be an appropriate alternative for PLL blocks in various communication systems, radars, and measurement devices.

Meanwhile, due to the ROM memory required in DDS structures design and implementation, the output frequency of DDS blocks are generally lower than their competitors like PLL and direct analog (DA). Meanwhile complex DDS blocks with frequencies ranging from 8 to 32 GHz have emerged thanks to recent developments in semiconductor technologies such as SiGe, BICMOS, InP, HBT [2-4]. These technologies, however, are expensive and noneconomic and hence are not preferred except for some special cases.

The architecture proposed at [2] is accomplished with a memory-based structure, while, the DDS presented at [3] is a ROM-Less structure utilizing a triangle to sine converter which is implemented by the expensive technology of bipolar SiGe. The DDS proposed at [4] is also implemented by Inp DHBT special technology utilizing a sine weighted DAC.

Unfortunately, the Sine weighted DAC is a custom ASIC current steering DAC, which requires sophisticated Dynamic Element Matching (DEM) and complex Layout techniques to maintain DDS block's increasing resolution demands.

Designing a phase to sine interface that can be utilized along with traditionally available highly optimized Current Steering DACs, makes the proposed DDS block needless to ROM memory, which is a common block for conventional structures. Requiring no special ASIC DAC and eliminating large ROM memory makes the circuit become more efficient and versatile. An 8 bits pipelined configuration was also used for implementing accumulator which significantly enhances the speed of the circuit.

The article is arranged as below: Section II studies ROM-Less DDS blocks along with DDS blocks based on ROM. Two topologies pros and cons are covered in this section. Proposed DDS structures and their blocks are discussed in section III. Section IV explains the results obtained for proposed structure and finally, section V concludes the paper.

II. ROM-LESS VERSUS CONVENTIONAL ROM-BASED DDS AND THEIR PRINCIPLE OF OPERATION

Conventional DDS blocks normally include a phase accumulator, a ROM lookup table (LUT) and a linear DAC. The phase accumulator computes the appropriate phase angle for the output sine wave by accumulating the input frequency control word (FCW) on each clock cycle. The phase accumulator's bits are connected to memory address bus and are here called "binary address code". If the size of the accumulator is supposed to be N bits as is shown at Fig. 1, the maximum phase value will be $2\pi(2^N-1)/2^N$ and ROM Memory locations are 2^N. For conventional DDS blocks fetching table is a ROM memory, and DDS precision is directly related to data bits called here "binary data" stored in each memory location. Utilizing the ROM block makes these structures suffer from larger circuits, which makes higher power consumption and larger occupied chip area as their main drawbacks. The incorporated linear DAC is

always a segmented one, which has thus binary and unary parts. As a result a binary to thermometer converter is needed to handle the unary part of the DAC [5]. This binary to thermometer interface converts the binary data fetched from memory look up table to its equivalent thermometer code which is later applied to the DAC unary section producing appropriate analog signal at DAC output.

In order to save power consumption and reduce the complexity of the sinusoidal LUT, the N-bit output of the accumulator, i.e. binary address code, may be truncated to P bits before addressing the ROM diminishing the ROM number of locations. In this case the number of points on the sine wave is reduced and the resulted sine wave contains larger spurs and harmonics.

The ROM LUT performs a phase-to-amplitude conversion (PAC) of the output sinusoidal wave. Once the amplitude information, i.e. binary data is obtained, it may be further truncated to D bits that correspond to the number of input bits of the linear DAC. The linear DACs are increasingly used and are significantly optimized and thus they are less sensitive to the element mismatches thanks to deliberately accomplished dynamic element matching and sophisticated layout techniques developed during years for these types of DACs [5].

In other words, for so-called linear DACs no ASIC design is required. The digital amplitude codes are then fed into aforementioned linear DAC that generates an analog replica of the synthesized waveform. A low-pass filter (LPF) usually follows the DAC to remove the unwanted frequency components. The input clock frequency and FCW determine the frequency step size of the DDS as,

$$\Delta F = \frac{f_{CLK}}{2^N} \tag{1}$$

and the output frequency of the DDS is given by

$$f_{out} = FCW \times \frac{f_{CLK}}{2^N} \tag{2}$$

Where f_{CLK} is the DDS clock frequency, FCW is the input frequency control word, and N is the size of the phase accumulator. According to the Nyquist theorem, at least two samples per clock cycle are required to reconstruct a sinusoidal wave without aliasing. Thus, the largest value of the FCW is 2^N-1 and hence the maximum output frequency of the DDS is limited to less than $f_{CLK}/2$. However, in a practical implementation of the deglitch LPF, the output frequency of the DDS is usually constrained to be less than $f_{CLK}/3$. The ROM size of the conventional DDS increases exponentially with increasing number of phase bits used to address the LUT. Increasing the ROM size, results generally in higher power consumption and larger area in ROM-based DDS designs. Numerous attempts have been made to compress or eliminate the ROM LUT utilized during phase-to-amplitude conversion. Langlois has been published a comprehensive review of the phase-to-amplitude conversion techniques, including angular decomposition, angular rotation, sine amplitude LUT compression, polynomial approximation and phase-to-sine amplitude conversion (PSAC)-DAC combinations [6]. All of the phase-to-amplitude conversion methods with the exception of PSAC-DAC involve either a large ROM or a complex architecture, which operate at relatively low speeds.

The block diagram of the ROM-based DDS, with N-bit phase and (P-1)-bit amplitude resolution is shown at Fig. 1. The major parts of the ROM-based DDS are an N-bit pipeline phase accumulator and a current-steering linear DAC. Since the output frequency cannot exceed the Nyquist rate, the most-significant-bit (MSB) of the accumulator is tied to zero. The N-bit FCW (including MSB=0) is fed into the accumulator which controls the output frequency of the synthesized waveform. Two MSB bits of the accumulator are used to determine the quadrant of the sine wave. The remaining bits are used to control the ROM and generate the amplitude for a quarter phase (0~ $\pi/2$) sine wave. In order to achieve fine step size, a large phase accumulator is desired. However, the phase accumulator output is normally truncated to save die area and power.

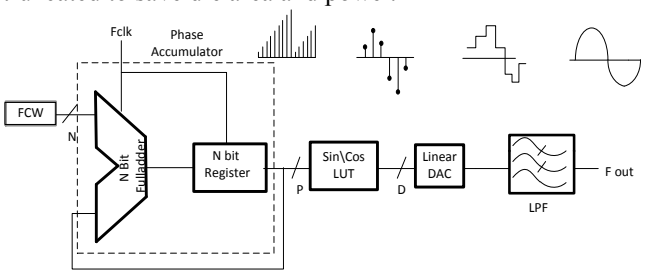

Figure 1: ROM-Based DDS block diagram.

To overcome the speed and power performance limits of the ROM-based high resolution DDS, a ROM-less DDS with sine-weighted DAC (identified as PSAC-DAC by Langlois) has been developed [6]. The conceptual block diagram of this ROM-less DDS employing a sine-weighted DAC is shown at Fig. 2. The ROM-less DDS replaces the ROM and linear DAC with a sine-weighted DAC. It eliminates the sine LUT, which is the speed and area bottleneck for high-speed DDS implementations [7-8]. On the other hand, it is a design challenge to achieve high resolution in the sine-weighted DACs due to nonlinear segmentation process required. This type of DDS adopts a ROM-less architecture, which combines both the sine/cosine mapping and digital-to-analog conversion together in a sine-weighted DAC [9].

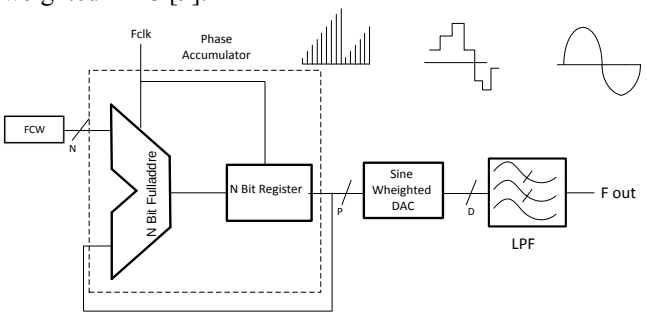

Figure 2: ROM-Less DDS block diagram.

III. PROPOSED DSS STRUCTURE

A. 8-Bits Pipeline Accumulator

Conventional phase accumulator is shown at Fig. 3 which suffers from extra delay which strictly reduces the maximum obtainable frequency of DDS.

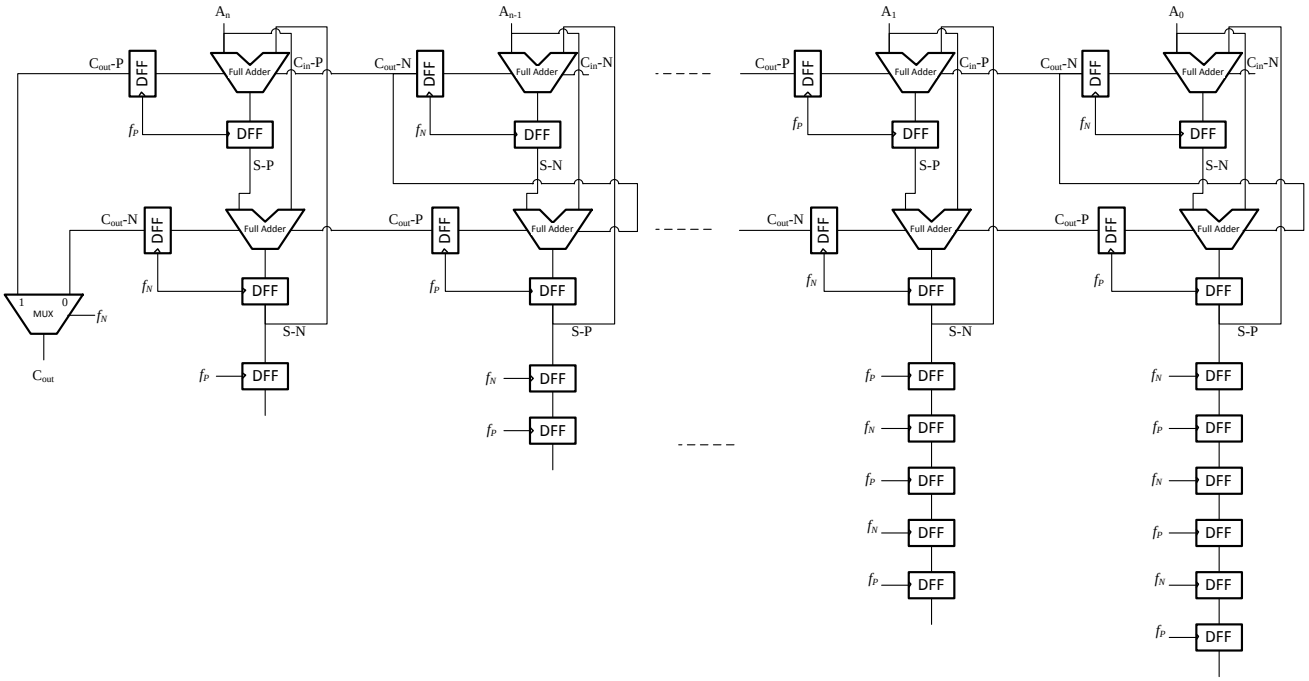

Figure 4: N bit Pipeline Phase Accumulator block diagram.

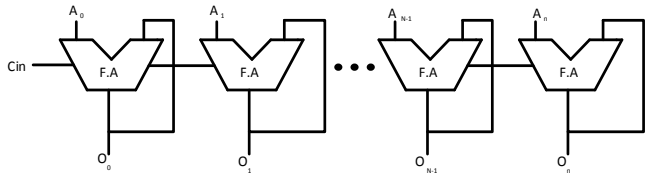

Figure 3: Simple Phase Accumulator block diagram.

To achieve the maximum operating speed with a fixed FCW, an 8-bit pipeline accumulator is used in proposed structure. In spite of relatively large hardware associated with pipeline accumulators, they can achieve higher speeds. This is due to the fact that in spite of a relatively long transient time at the beginning of DDS operation, the remaining data are achieved just in one clock period. The overall delay of the pipeline accumulator is equal with propagation delays sum of a full adder (FA) and a D flip-flop (DFF). Figure 4 shows the architecture of the N-bit pipeline phase accumulator, which consists of N pipelined levels. Each level has a total of N DFF delay stages which are distributed before and after of Full Adders according to their location at the system, i.e. the bit each Full Adder is associated with. One more DFF is used for each level to synchronize the data signal and eliminate any unexpected timing mismatch. Note that an accumulator requires at least one delay stage even if no pipelined stages are utilized.

B. Implementation of the Phase to Sine code Converter

As mentioned earlier, the incorporated linear DAC is always a segmented one containing binary and unary parts which requires a binary to thermometer converter to handle the unary part of the DAC [5]. This is to eliminate or diminish non-uniform output signals initiated by process mismatch especially for mid-code transition, i.e. code change from $(7)_{10}=(0111)_2$ to $(8)_{10}=(1000)_2$, in which all bits change their state. Using binary to thermometer code the number of ones increase uniformly and thus $(7)_{10}=(01111111)_{th}$ changes to $(8)_{10}=(11111111)_{th}$. For thermometer code all bits have identical weights of 1 rather than 1, 2, 4, 8, 16 for binary codes. Hence for code transition from 7 to 8 which for binary scheme requires change of all binary bits with binary weighted values, i.e. 1, 2, and 4 → OFF and 8 → ON, for thermometric structures just one extra block is made ON, which is very efficient.

In conventional structures the binary address is used to fetch out the binary data from a specified memory location. This binary data is a binary coded sine value which was calculated previously and stored at ROM. This binary coded sine data is further processed by a binary to thermometer to manage the segmented DAC.

In the proposed structure due to elimination of ROM no data is available and the only information is the phase accumulator bits which were previously utilized as address bits. Hence the idea is to convert phase accumulator binary bits directly into its binary weighted sinusoidal equivalent and fed these bits to a linear DAC. The produced code has an intrinsic thermometric nature and thus interestingly prohibits the conventionally required binary to thermometer converter.

In order to design the phase to sine interface with intrinsic thermometric nature, the thermometer coefficients are initially calculated applying 6 LSB bits of the phase accumulator, namely x_0, x_1, x_2, x_3, x_4, and x_5, to the relations given by equation (3). This produces the first-quarter of the sine signal which can be further extended to full period signal taking 2 MSB bits of the phase accumulator into account. Also note that $\bar{x}_i$ is complement of the x_i.

$$
\begin{aligned}
I_1 &= I\times[\cos\left(\left(n-1\right)\frac{\pi}{n}\right)-\cos\left(\pi\right)] \\
I_2 &= I\times[\cos\left(\frac{\pi}{n}\right)-\cos\left(\frac{2\pi}{n}\right)] \\
I_3 &= I\times[\cos\left(\frac{2\pi}{n}\right)-\cos\left(\frac{3\pi}{n}\right)] \\
&\ldots \\
I_n &= I\times[\cos\left(\left(n-1\right)\frac{\pi}{n}\right)-\cos\left(\pi\right)]
\end{aligned}
\qquad (3)
$$

Where I_i, $1\le i\le n$ are the sine weighted currents which must be delivered at DAC output and n is the number of current

steering DAC current branches. The M-file provided at Table 1 can be used to calculate the abovementioned coefficients. The values calculated for Y is as below:

20	41	41	41	41	40	40	40	40	40	40
39	39	39	38	38	38	37	37	36	36	35
35	34	34	33	33	32	31	31	30	30	29
28	27	27	26	25	24	23	23	22	21	20
19	18	17	17	16	15	14	13	12	11	10
9	8	7	6	5	4	3	2	1.		

It is interestingly found that the higher values like 41, 40, 39 … are repeated frequently, implying the thermometric nature of the calculations.

Considering that the largest number obtained by equation (3) for n=64 is 1661, the number of bits for sine weighted digital codes is achieved to be 11. Converting the decimal values into the binary digital codes (Listed at table 2), the Boolean Functions can then be calculated for all bits of digital sine signal, establishing eleven six-variable Karnaugh Map.

Executing m file provided at Table 1, the result of these calculations for n=64 which determine decimal signal steps to reconstruct the sine wave are saved at workspace under variable 'z' and are shown at Table 2.

As mentioned before, these 6 bits are related to the least significant bits of the phase accumulator which are used to calculate 11 bits related to the first-quarter of sine signal. Applying two bits of MSB and MSB-1, it is possible to make a full period sine signal. This system uses a Quarter Sine Wave Technique to reproduce the desired Sine signal, hence producing a sine signal in the range of 0 to $\pi/2$ is adequate to build the full signal. The relations associated with the binary coded Sine signal functions are given by equation (4).

$\mathbf{O_0} = (\bar{x}_5\bar{x}_4\bar{x}_3\bar{x}_2x_0) + (\bar{x}_5x_3x_2x_1) + (\bar{x}_5x_3x_2x_0) + (x_5\bar{x}_4\bar{x}_3x_1\bar{x}_0) + (x_5\bar{x}_4x_2\bar{x}_1x_0) + (x_5\bar{x}_4x_3\bar{x}_1) + (x_4x_1x_0) + (\bar{x}_5\bar{x}_2x_1x_0) + (\bar{x}_4x_3x_2x_0) + (x_5x_4\bar{x}_1\bar{x}_0) + (\bar{x}_5x_4\bar{x}_3\bar{x}_0) + (\bar{x}_5x_4\bar{x}_2\bar{x}_0)$

$\mathbf{O_1} = (\bar{x}_4\bar{x}_3\bar{x}_2x_1) + (\bar{x}_4\bar{x}_2x_1x_0) + (\bar{x}_5\bar{x}_4x_3x_2\bar{x}_0) + (\bar{x}_5x_4\bar{x}_2\bar{x}_1\bar{x}_0) + (\bar{x}_5x_4\bar{x}_3x_2x_1\bar{x}_0) + (\bar{x}_5x_4x_3x_2x_0) + (x_5\bar{x}_4x_3\bar{x}_2\bar{x}_0) + (x_5x_3x_2\bar{x}_1x_0) + (x_5\bar{x}_2x_1) + (x_5x_4x_2\bar{x}_1)$

$\mathbf{O_2} = (\bar{x}_5\bar{x}_4\bar{x}_3\bar{x}_2) + (\bar{x}_5\bar{x}_4x_3x_2\bar{x}_1) + (x_4\bar{x}_3\bar{x}_1x_0) + (\bar{x}_5x_4\bar{x}_3x_2x_1\bar{x}_0) + (\bar{x}_5x_4\bar{x}_2x_0) + (x_5x_3x_2x_1) + (\bar{x}_4\bar{x}_2x_1x_0) + (\bar{x}_3\bar{x}_2\bar{x}_1\bar{x}_0) + (x_4x_3\bar{x}_2x_1\bar{x}_0) + (x_4x_3x_2\bar{x}_1\bar{x}_0) + (\bar{x}_5x_4x_3x_1x_0) + (\bar{x}_3\bar{x}_2x_1x_0) + (x_5\bar{x}_4x_3\bar{x}_2\bar{x}_1)$

$\mathbf{O_3} = (\bar{x}_5\bar{x}_4\bar{x}_3\bar{x}_2x_0) + (\bar{x}_5\bar{x}_4\bar{x}_3x_2\bar{x}_0) + (\bar{x}_5x_4\bar{x}_2\bar{x}_0) + (x_4\bar{x}_3x_1x_0) + (x_5\bar{x}_4\bar{x}_2x_1\bar{x}_0) + (x_5\bar{x}_4\bar{x}_3x_2x_0) + (\bar{x}_4x_3\bar{x}_2\bar{x}_0) + (\bar{x}_4x_3\bar{x}_2x_1) + (x_5x_4\bar{x}_3x_2\bar{x}_1\bar{x}_0) + (x_4x_3\bar{x}_1x_0) + (\bar{x}_5\bar{x}_4x_3x_2x_0) + (\bar{x}_5\bar{x}_2x_1x_0) + (\bar{x}_5x_4x_3\bar{x}_0) + (x_4x_3x_1\bar{x}_0) + (x_5x_4x_3x_2x_1)$

$\mathbf{O_4} = (\bar{x}_4\bar{x}_3x_2x_1x_0) + (\bar{x}_5\bar{x}_4\bar{x}_1\bar{x}_0) + (\bar{x}_5x_4\bar{x}_3\bar{x}_2x_1) + (x_5\bar{x}_4\bar{x}_2x_1\bar{x}_0) + (x_4\bar{x}_3\bar{x}_2\bar{x}_1x_0) + (x_5x_4x_3x_1x_0) + (\bar{x}_5\bar{x}_4\bar{x}_3\bar{x}_2\bar{x}_1) + (\bar{x}_4x_3x_2\bar{x}_1x_0) + (x_5\bar{x}_4x_3\bar{x}_2x_1) + (x_5\bar{x}_3x_2\bar{x}_1\bar{x}_0) + (x_5x_2x_1x_0) + (x_5x_3x_2x_0) + (x_5x_4x_2\bar{x}_0)$

$\mathbf{O_5} = (\bar{x}_5\bar{x}_4\bar{x}_2x_1\bar{x}_0) + (\bar{x}_5x_2\bar{x}_1\bar{x}_0) + (\bar{x}_5x_4x_2\bar{x}_0) + (x_4x_3\bar{x}_0) + (x_5\bar{x}_4\bar{x}_2\bar{x}_1\bar{x}_0) + (x_5\bar{x}_2x_1x_0) + (\bar{x}_4\bar{x}_3x_2\bar{x}_1x_0) + (x_5x_3x_2x_1) + (x_5x_4\bar{x}_2x_1) + (x_4x_2\bar{x}_1\bar{x}_0) + (x_5x_4x_3) + (\bar{x}_5\bar{x}_3\bar{x}_2\bar{x}_1x_0) + (\bar{x}_5\bar{x}_4x_2x_1x_0) + (\bar{x}_5x_4\bar{x}_3\bar{x}_2x_0)$

$\mathbf{O_6} = (\bar{x}_5\bar{x}_3x_2\bar{x}_1x_0) + (\bar{x}_4x_3\bar{x}_1\bar{x}_0) + (\bar{x}_5\bar{x}_4x_3x_1x_0) + (\bar{x}_5x_4\bar{x}_3\bar{x}_2x_1) + (x_4x_3\bar{x}_1x_0) + (x_5\bar{x}_4\bar{x}_3\bar{x}_2x_0) + (x_5x_3x_2) + (x_5x_4x_3) + (\bar{x}_4\bar{x}_3\bar{x}_2x_1\bar{x}_0) + (x_3x_2x_1\bar{x}_0) + (\bar{x}_5x_4x_1\bar{x}_0) + (x_5x_4x_2x_0) + (x_5x_2x_1)$

$\mathbf{O_7} = (\bar{x}_5\bar{x}_4\bar{x}_3x_2\bar{x}_1) + (\bar{x}_5x_4x_2x_1x_0) + (x_5\bar{x}_4\bar{x}_3\bar{x}_2) + (\bar{x}_4x_3x_2\bar{x}_1\bar{x}_0) + (\bar{x}_4x_3\bar{x}_2x_0) + (\bar{x}_4x_3\bar{x}_2x_1) + (\bar{x}_5x_4\bar{x}_2\bar{x}_1) + (\bar{x}_5x_4\bar{x}_2\bar{x}_0) + (x_5\bar{x}_4x_3x_2) + (\bar{x}_5\bar{x}_3\bar{x}_2x_1x_0)$

$\mathbf{O_8} = (\bar{x}_4x_3\bar{x}_2) + (\bar{x}_4x_3\bar{x}_1\bar{x}_0) + (\bar{x}_5x_4\bar{x}_3x_2) + (\bar{x}_5x_3\bar{x}_2\bar{x}_1) + (\bar{x}_5x_3\bar{x}_2\bar{x}_0) + (x_5\bar{x}_4x_2) + (\bar{x}_5\bar{x}_3x_2x_1)$

$\mathbf{O_9} = (\bar{x}_5\bar{x}_4x_3x_2x_0) + (\bar{x}_5\bar{x}_4x_3x_2x_1) + (x_4\bar{x}_3) + (x_4\bar{x}_2\bar{x}_1) + (x_4\bar{x}_2\bar{x}_0) + (x_5x_4)$

$\mathbf{O_{10}} = (x_4x_3x_1x_0) + (x_4x_3x_2) + (x_5)$ (4)

Where $\mathbf{O_i}$, $0 \le i \le 10$ are the binary coded Sine signal functions which rebuilt sine function desired amplitude for each set of x_i parameters.

The main problem of the relations given by equation (4) is that they require a lot of gates for implementation. In order to overcome this problem and by doing some simplifications, the simplified relations are acquired as given by (5).

Table 1
Matlab code to calculate digital coefficients of the phase to sine signal convertor

```
n=input('n=');
A=2*n;
A(1)=(pi/(2*A));
TETA(1)=A(1);
for k=2:1:n
   TETA(k)=(2*A(1))+TETA(k-1);
end
b(1)=sin(TETA(1));
for B=2:1:n
   b(B)=sin(TETA(B))-sin(TETA(B-1));
end
for y=1:1:n
   a(y)=(b(y)/b(n));
   Y(y)=round(a(y));
end
   z(1)=Y(1);
   step(1)=1;
   for i=1:1:n-1
      z(i+1)=Y(i+1)+z(i);
      step(i+1)=i+1;
   end
    plot(step,z);
```

$\mathbf{O_0} = (k_1k_{25}) + \bar{x}_5\ (k_2+k_3+k_7) + (k_4k_{26}) + (k_5k_{31}) + (\bar{x}_1k_6) + (k_{27}) + (\bar{x}_4k_3) + (k_{32}k_{36}) + \bar{x}_0\ (k_{11}+k_{16})$

$\mathbf{O_1} = \bar{x}_4\ (k_7+k_8) + (k_9k_{38}) + (k_{10}k_{30}) + (k_{11}k_{12}) + (k_3k_{30}) + (k_6k_{39}) + (k_5k_{40}) + (x_5\,k_{41}) + (\bar{x}_1\,k_{29})$

$\mathbf{O_2} = (\bar{x}_2\ k_1) + (k_9k_{37}) + x_0\ (\bar{x}_1\ k_{35} + k_{16} + \bar{x}_5k_{15}) + (k_{11}k_{12}) + (x_5k_2) + (\bar{x}_4k_7) + \bar{x}_3\ (k_7+k_{10}) + (k_{13}k_{42}) + (k_{14}k_{42}) + (k_6k_{43})$

$\mathbf{O_3} = k_1\ (k_{25}+k_{38}) + \bar{x}_0\ (k_{15}+k_{16}+k_{17}) + \bar{x}_3\ (k_{27}+\ k_{14}k_{32}) + (k_{13}k_{31}) + (x_0x_2k_4) + (x_1k_{17}) + (k_{19}) + (k_3k_{24}) + \bar{x}_5\ (k_7+k_{20}) + (k_2k_{32})$

$\mathbf{O_4} = (k_{18}k_{34}) + (k_{24}k_{36}) + (k_8k_{30}) + (k_{13}k_{31}) + (x_4k_{21}) + (k_1k_{43}) + (k_5k_{33}) + (k_4k_{41}) + x_5\ (k_3+\bar{x}_3k_{14}+\ x_0k_{15} + k_{18}) + (\bar{x}_0k_{29})$

$\mathbf{O_5} = k_{24}\ (k_{13}+\ k_{18}) + \bar{x}_5\ (k_{14}+k_{21}) + (k_{30}k_{38}) + (k_{20}) + (k_{10}k_{31}) + x_5\ (k_2+k_7) + (k_5k_{34}) + (k_{32}k_{41}) + (x_4k_{14}) + (k_{22}) + (k_{11}k_{25})$

$\mathbf{O_6} = (k_5k_{45}) + (k_{28}) + x_0\ (x_1k_9+\ k_{29}) + k_{30}\ (k_8+\ k_{26}) + (k_{19}) + (k_4k_{25}) + (x_2k_{40}) + (k_{22}) + \bar{x}_0\ (\bar{x}_4k_8+\ k_2) + (x_5k_{44})$

$\mathbf{O_7} = (k_1k_{37}) + (k_{18}k_{30}) + (\bar{x}_2k_4) + (k_{14}k_{33}) + k_{17}\ (x_0+x_1) + k_{16}\ (\bar{x}_0+\bar{x}_1) + (x_2k_6) + (k_7k_{45})$

$\mathbf{O_8} = (k_{17}) + (k_{28}) + (x_2k_{11}) + k_{23}\ (\bar{x}_0+\bar{x}_1) + (x_2k_{31}) + (k_{44}k_{45})$

$\mathbf{O_9} = (k_3k_{24}) + (k_2k_{24}) + (k_{35}) + (x_4k_{43}) + (x_4k_{39}) + (k_{32})$

$\mathbf{O_{10}} = (x_0k_{15}) + (x_2k_{42}) + (x_5)$ (5)

where k_1 to k_{45} are the commonly used expressions given as following:

$k_1=\bar{x}_5\bar{x}_4\bar{x}_3$	$k_{10}=\bar{x}_2\bar{x}_1\bar{x}_0$	$k_{19}=x_4x_3\bar{x}_1x_0$	$k_{28}=\bar{x}_4x_3\bar{x}_1\bar{x}_0$	$k_{37}=x_2\bar{x}_1$	
$k_2=x_3x_2x_1$	$k_{11}=\bar{x}_5x_4\bar{x}_3$	$k_{20}=x_4x_3\bar{x}_0$	$k_{29}=x_5x_4x_2$	$k_{38}=x_2\bar{x}_0$	
$k_3=x_3x_2x_0$	$k_{12}=x_2x_1\bar{x}_0$	$k_{21}=\bar{x}_3\bar{x}_2\bar{x}_1x_0$	$k_{30}=\bar{x}_5x_4$	$k_{39}=\bar{x}_2\bar{x}_0$	
$k_4=x_5\bar{x}_4\bar{x}_3$	$k_{13}=\bar{x}_2x_1\bar{x}_0$	$k_{22}=x_5x_4x_3$	$k_{31}=x_5\bar{x}_4$	$k_{40}=x_5x_3$	
$k_5=x_2\bar{x}_1x_0$	$k_{14}=x_2\bar{x}_1\bar{x}_0$	$k_{23}=\bar{x}_5x_3\bar{x}_2$	$k_{32}=x_5x_4$	$k_{41}=\bar{x}_2x_1$	
$k_6=x_5\bar{x}_4x_3$	$k_{15}=x_4x_3x_1$	$k_{24}=\bar{x}_5\bar{x}_4$	$k_{33}=\bar{x}_4x_3$	$k_{42}=x_4x_3$	
$k_7=\bar{x}_2x_1x_0$	$k_{16}=\bar{x}_5x_4\bar{x}_2$	$k_{25}=\bar{x}_2x_0$	$k_{34}=\bar{x}_4\bar{x}_3$	$k_{43}=\bar{x}_2\bar{x}_1$	
$k_8=\bar{x}_3\bar{x}_2x_1$	$k_{17}=\bar{x}_4x_3\bar{x}_2$	$k_{26}=x_1\bar{x}_0$	$k_{35}=x_4\bar{x}_3$	$k_{44}=x_2x_1$	
$k_9=\bar{x}_5\bar{x}_4x_3$	$k_{18}=x_2x_1x_0$	$k_{27}=x_4x_1x_0$	$k_{36}=\bar{x}_1\bar{x}_0$	$k_{45}=\bar{x}_5\bar{x}_3$	

Table2
The results of calculations for the digital code of Sine signal

n	Accumulator Output $X_5X_4X_3X_2X_1X_0$	Decimal Signal Step Binary Weighted Sine Signal with Intrinsic Thermometric Nature	Binary Signal Step	O_{10}	O_9	O_8	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	**00000**	**20**	**00000010100**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**
1	**000001**	**61=20+ 41**	**00000111101**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**
2	**000010**	**102=61+ 41**	**00001100110**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**
3	**000011**	**143=102+ 41**	**00010001111**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**1**
4	**000100**	**184=143+ 41**	**00010111000**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**
5	**000101**	**224=184+ 40**	**00011100000**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**
6	**000110**	**264=224+ 40**	**00100001000**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
7	**000111**	**304=264+ 40**	**00100110000**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**
8	**001000**	**344=304+ 40**	**00101011000**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
9	**001001**	**384=344+ 40**	**00110000000**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
10	**001010**	**424=384+ 40**	**00110101000**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**
11	**001011**	**463=424 +39**	**00111001111**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**1**
12	**001100**	**502=463 +39**	**00111110110**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**1**	**0**
13	**001101**	**541=502 +39**	**01000011101**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**1**
14	**001110**	**579=541+ 38**	**01001000011**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**
15	**001111**	**617=579+ 38**	**01001101001**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**1**
16	**010000**	**655=617+ 38**	**01010001111**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**1**
17	**010001**	**692=655+ 37**	**01010110100**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**
18	**010010**	**729=692+ 37**	**01011011001**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**1**
19	**010011**	**765=729+ 36**	**01011111101**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**
20	**010100**	**801=765+ 36**	**01100100001**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**
21	**010101**	**836=801+ 35**	**01101000100**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**
22	**010110**	**871=836+ 35**	**01101100111**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**1**
23	**010111**	**905=871+ 34**	**01110001001**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**1**
24	**011000**	**939=905+ 34**	**01110101011**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**1**
25	**011001**	**972=939+ 33**	**01111001100**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**
26	**011010**	**1005=972+33**	**01111101101**	**0**	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**1**
27	**011011**	**1037=1005+32**	**10000001101**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**1**
28	**011100**	**1068=1037+31**	**10000101100**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**0**
29	**011101**	**1099=1068+31**	**10001001011**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**1**
30	**011110**	**1129=1099+30**	**10001101001**	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**1**
31	**011111**	**1159=1129+30**	**10010000111**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**
32	**100000**	**1188=1159+29**	**10010100100**	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**0**
34	**100001**	**1216=1188+28**	**10011000000**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
35	**100010**	**1243=1216+27**	**10011011011**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**1**
36	**100011**	**1270=1243+27**	**10011110110**	**1**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**0**
37	**100100**	**1296=1270+26**	**10100010000**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
38	**100101**	**1321=1296+25**	**10100101001**	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**
39	**100110**	**1345=1321+24**	**10101000001**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**
40	**100111**	**1368=1345+23**	**10101011000**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
41	**101000**	**1391=1368+23**	**10101101111**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**1**
42	**101001**	**1413=1391+22**	**10110000101**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**1**
43	**101010**	**1434=1413+21**	**10110011010**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**
44	**101011**	**1454=1434+20**	**10110101110**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**1**	**0**
45	**101100**	**1473=1454+19**	**10111000001**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**1**
46	**101101**	**1491=1473+18**	**10111010011**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**1**
47	**101110**	**1508=1491+17**	**10111100100**	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**0**
48	**101111**	**1525=1508+17**	**10111110101**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**1**
49	**110000**	**1541=1525+16**	**11000000101**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**
50	**110001**	**1556=1541+15**	**11000010100**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**
51	**110010**	**1570=1556+14**	**11000100010**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**
52	**110011**	**1583=1570+13**	**11000101111**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**1**
53	**110100**	**1595=1583+12**	**11000111011**	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**1**
54	**110101**	**1606=1595+11**	**11001000110**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**
55	**110110**	**1616=1606+10**	**11001010000**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**
56	**110111**	**1625=1616+ 9**	**11001011001**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**1**
57	**111000**	**1633=1625+ 8**	**11001100001**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**1**
58	**111001**	**1640=1633+ 7**	**11001101000**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**
59	**111010**	**1646=1640+ 6**	**11001101110**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**
60	**111011**	**1651=1646+ 5**	**11001110011**	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**
61	**111100**	**1655=1651+ 4**	**11001110111**	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**1**
62	**111101**	**1658=1655+ 3**	**11001111010**	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**
63	**111110**	**1660=1658+ 2**	**11001111100**	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**0**	**0**
64	**111111**	**1661=1660+ 1**	**11001111101**	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**0**	**1**

Figure 5 shows the block diagram of the proposed structure.

IV. Simulation and Practical Results

Applying FCW=1 to the input of the circuit simulated in Matlab, four continuous periods of the output waveform are shown at Fig. 6. The role of MSB in producing positive and negative half-period signal is also shown and the final wave acquired from the combination of these two half-waves shows the appropriate functionality of the simulated circuit.

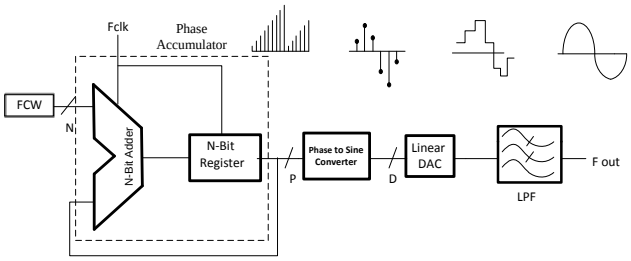

Figure 5: Proposed DDS Structure with Phase to Sine Converter.

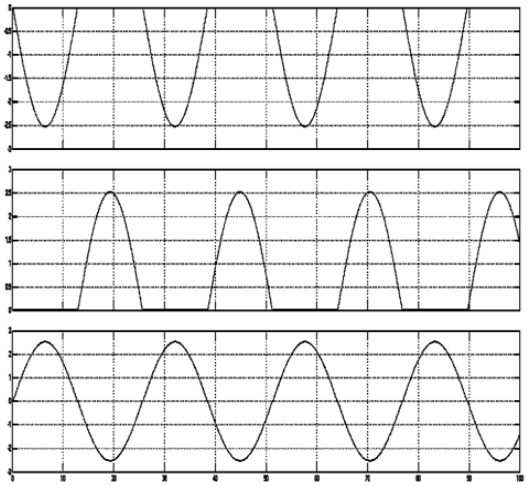

Figure 6: The simulation result for FCW=1.

Meanwhile, considering the frequency controlling word equal to one (FCW=1), which corresponds to the binary code of 00000001, to the input port of the implemented FPGA board, the output waveform measured by oscilloscope is shown at Fig. 7. Figure 8 and 9 show the block diagram and hardware implementation of the suggested circuit.

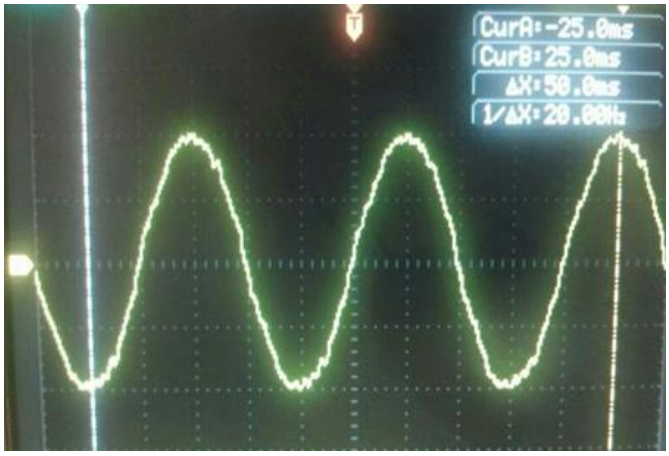

Figure 7: The measurement result for FCW=1.

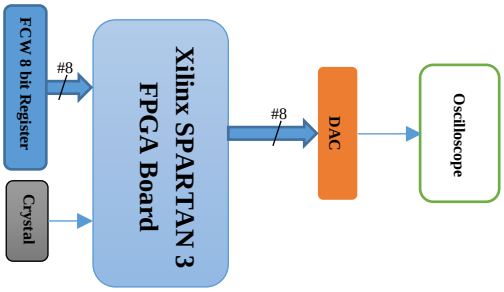

Figure 8: Block diagram of proposed DDS.

As shown at Figure 8, all of the hardware implementations of the DDS system are accomplished within the FPGA and only a Digital to Analog Converter which has analog nature and cannot be implemented inside the FPGA is utilized as a discrete element.

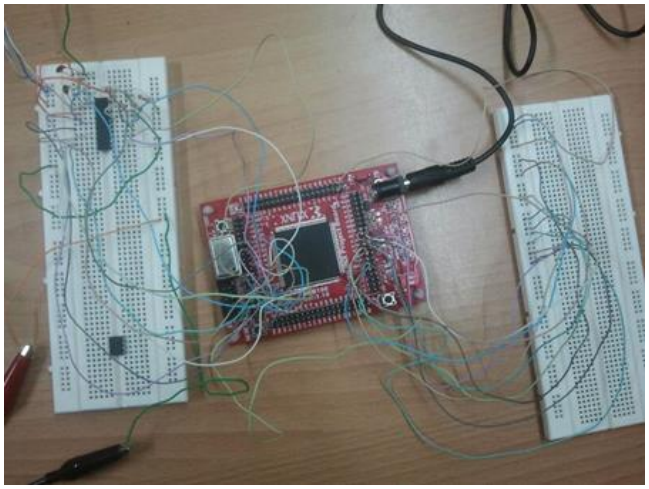

Figure 9: Hardware implementation of proposed DDS.

The measurement results are included at Table 3.

Table 3
Measurement results

Clock Frequncy	5 Ghz
DAC Ampilitude Resolution	8 bits
SFDR(@ Nyquist Frequency)	23.24 dB
Power supply	4.6 Volts
Power consumption	2.4 w

V. Conclusion

The structure of an 8-bit fully digital synthesizer is proposed which is utilizing a thermometer to sinusoidal interface. In the proposed structure an 8 bit pipeline accumulator is used for implementing the accumulator. The thermometer to sinusoidal interface along with linear DAC makes the Direct Digital Synthesizer (DDS) needless of conventionally used ROM.

Meanwhile the ASIC sine weighted DAC which requires specifically handled dynamic element matching (DEM) procedure and sophisticated layout techniques to enhance its resolution in accordance with the DDS specifications is eliminated. Besides simplifying the DDS overall configuration, the proposed structure makes it also possible to use the sinusoidal digital codes to be compared with sampled form of original signal and hence can eliminate analog comparison in the front end stage. To evaluate the performance of the proposed circuit the system is first

simulated in Matlab and then it is implemented in logical gate level at Xilinx ISE environment and finally is practically implemented by Xilinx SPARTAN 3 FPGA and its performance is evaluated experimentally.

REFERENCE

[1] Anjali Pawar, "Direct Digital Synthesizer Based on FPGA", *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, Vol. 4, Issue 7, pp. 6129-6134, July 2015.

[2] Li Wenxing, and Zhang Ye, "The Design of Digital Frequency Synthesizer Based on VHDL", *Journal of Theoretical and Applied Information Technology*,. Vol. 48 No.3, pp. 1680-1684, 2013.

[3] Benjamin Laemmle, Christoph Wagner, Herbert Knapp, Linus Maurer, and Robert Weigel. "A 366mW direct digital synthesizer at 15GHz clock frequency in SiGe Bipolar technology." In *2009 IEEE Radio Frequency Integrated Circuits Symposium*, pp. 415-418. IEEE, 2009.

[4] S. Yanbina,G. Jianb, and C. Ningc, "High Precision Digital Frequency Signal Source Based on FPGA", Physics Procedia vol. 25, pp. 1342 – 1347, 2012.

[5] E. Greenwald et al., "A CMOS Current Steering Neurostimulation Array With Integrated DAC Calibration and Charge Balancing," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 2, pp. 324-335, April 2017. doi: 10.1109/TBCAS.2016.2609854

[6] J. M. P. Langlois and D. Al-Khalili, "Phase to sinusoid amplitude conversion techniques for direct digital frequency synthesis," in *IEE Proc. Circuits Devices Syst.*, Dec. 2004, pp. 519–528.

[7] B. Laemmle, C. Wagner, H. Knapp, L. Maurer, and R. Weigel, "A 366mW direct digital synthesizer at 15GHz clock frequency in SiGe Bipolar technology," in *Radio Frequency Integrated Circuits Symposium, 2009. RFIC 2009. IEEE*, 2009, pp. 415-418.

[8] Arti D. Gaikwad, Govind U. Kharat, Shekhar H. Bodake, "A Review on Design and Implementation of Numerically Controlled Oscillator", *International Journal of Innovative Research in Computer and Communication Engineering*, Vol. 4, Issue 4, pp. 5023-5029, April 2016

[9] X. Geng, X. Yu, F. F. Dai, J. D. Irwin, and R. C. Jaeger, "An 11-bit 8.6 GHz direct digital synthesizer MMIC with 10-bit segmented nonlinear DAC," in *34th Eur. Solid-State Circuits Conf. (ESSCIRC)*, Sep. 2008, pp. 362–365.