Design and Implementation of FPGA based Linear All Digital Phase-Locked Loop

Abhishek Das, Suraj Dash and A.K.Sahoo
Dept. of Electronics and Communication Engineering
National Institute of Technology, Rourkela-769 008
e-mail: i.abhishek.das@gmail.com, surajdash@gmail.com, ajitsahoo@nitrkl.ac.in

B.Chitti Babu, *Member, IEEE*
Department of Electrical Engineering
National Institute of Technology, Rourkela-769 008
e-mail: bcbabunitrkl@ieee.org

Abstract— **This paper presents a linear all-digital phase locked loop based on FPGA. In this ADPLL the phase detection system is realized by generating an analytic signal using a compact implementation of Hilbert transform and then simply computing the instantaneous phase using CORDIC algorithm in vectoring mode of operation. A 16-bit pipelined CORDIC algorithm is employed in order to obtain the phase information of the signal. All the components used in this phase detection system are realized as digital discrete time components. This design does not involve any class of multipliers thus reducing the complexity of the design. The loop filter of the ADPLL has been designed using PI controller which has a low pass behavior and is used to discard the higher order harmonics of the error signal. The CORDIC algorithm in its rotation mode of operation is used to compute sinusoidal values for the DDS. The ADPLL model has been implemented using Xilinx ISE 12.3 and ModelSim PE Student Edition 10.1a.**

Keywords— Phase detector, Hilbert Transform, CORDIC algorithm, Phase locked loop, Signal processing, FPGA Implementation.

I. INTRODUCTION

The steady improvement of components for digital signal processing applications, more applications pertaining to processing signals are experiencing a shift from the analog to the digital domain. The digital domain compared to the analog domain provide manifold benefits like easy calibration, higher accuracy, better predictability and the probability to increase the complexity without the need for tedious adjustments or calibrations[1]. Thus the digital domain certainly provides a better edge over analog domain which attracts more research and experimentation in this field of study.

The main motivation has been derived from the requirement of a flexible offset local oscillator in a heavy ion particle accelerator. This offset local oscillator is needed in a fast phase detection system involving closed loop RF controls for the accelerator SIS18 at GSI [2]-[3]. The need for an FPGA based ADPLL was required basically because the microprocessors do not have enough processing power at such high frequencies even though with integrated analog to digital converters (ADC's) and digital to analog converters (DAC's). Thus the system becomes redundant in this case. The design of an application specific integrated circuit (ASIC) is not suitable for this purpose because the quantity involved is less and thus the fabrication process would be very expensive. The FPGA is an ideal solution to the problem because it combines the speed and computational power of ASICs as well as the flexibility of microprocessors. One of the key components in the design of an ADPLL system is the phase detector. In the present scenario most VLSI implementation uses the phase frequency detectors (PFD's) and the time to digital converters (TDC's) as their phase detectors [4]. The PFD's work on the principle of producing pulse-width modulated (PWM) signals which give the measure of the phase difference between the signals. But in order to extract the phase difference these PWM signals need to be passed through a low pass filter, which causes the pulse-width to change. Thus this method becomes redundant as far as digital filtering is concerned. The TDC is a common phase detector block for fully digital PLL. It is generally made up of chain of tapped constant delays. Although the two methods listed above have a large frequency range and also provide a fine resolution, they are not suitable for our purpose basically because it is very difficult to transfer their logic into FPGA i.e. they are not easily FPGA realizable.

This paper presents a linear all-digital phase locked loop based on FPGA. In this ADPLL the phase detection system is realized by generating an analytic signal using a compact implementation of Hilbert transform and then simply computing the instantaneous phase using CORDIC algorithm in vectoring mode of operation. A 16-bit pipelined CORDIC algorithm is employed in order to obtain the phase information of the signal [5]. All the components used in this phase detection system are realized as digital discrete time components. . The ADPLL model has been implemented using Xilinx ISE 12.3 and ModelSim PE Student Edition 10.1a.

The organization of the paper is given as follows: In section II, the proposed design for the phase detection based on ADPLL system is discussed. Section III gives the implementation of the proposed phase detection system in detail. Then, the results from simulations are presented in section IV, followed by a conclusion in section V.

II. ALL DIGITAL PLL

An all-digital phase-locked loop (ADPLL) is one of the many types of phase locked loop (PLL) in which all the components that are being used are digital in nature. Thus these types of PLL have the advantage of being realizable in an FPGA. The use of digitized components gives it flexibility and the parameters of the components can be changed on the field itself. Also, the use of digitized components provides immunity from factors such as temperature dependency and parasitic capacitances which are otherwise very much prominent in analog devices. The digital nature of the components also helps in noise immunity thus improving the functionality of the circuit.

The structure of an all-digital phase locked loop is shown in Fig.1. The phase detection system for the ADPLL consists of the Hilbert Transform and the CORDIC algorithm as the phase detection system [1], the PI controller as the loop filter and the CORDIC algorithm is the rotation mode as the digitally controlled oscillator (DCO). The advantage of the phase detection system comprising of Hilbert filter and the CORDIC algorithm in vectoring mode is that it not only has a good working range and a better resolution but is also FPGA realizable.

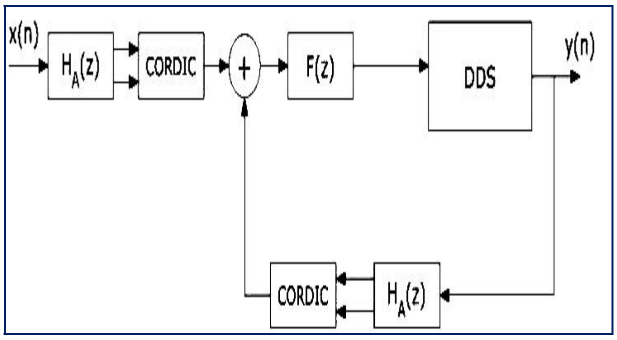

Fig.1. Structure of an ADPLL

A. Hilbert Transform

The concept of analytic signal is used in order to obtain the phase information of the signal. The analytic signal has the property that it has no negative frequency components. Any real sinusoid $A\cos(\omega t+\phi)$ can be concerted to a positive frequency complex sinusoid $Ae^{j(\omega t+\phi)}$ by simply generating a phase quadrature component $A\sin(\omega t+\phi)$ to serve as the imaginay part. For complicated signals consisting of sum of many sinusoids , a filter is used which shifts each component by a quarter cycle. This type of filters are called Hilbert Transform Filters. This type of filter ideally has a magnitude of 1 and introduces a phase shift of $-\pi/2$ at each positive frequency and $+\pi/2$ at each negative frequency [8].

The analytic signal is generated by applying hilbert transform to the real signal and using it as the imaginary part to extend the original signal. Let $H\{x\}$ denote the hilbert transform of an input signal x. Thus for an input signal x(n), the analytic signal can be represented by,

$$\underline{x}(n)=x(n)+jH\{x(n)\} \tag{1}$$

where $\underline{x}(n)$ represents the analytic signal. This signal does not have any spectral components in the range $-\pi<\Omega<0$. Using the above, the filter operation can then be formulated as,

$$H_A(e^{j\Omega})=1+jH_H(e^{j\Omega}) \tag{2}$$

The phase information is then obtained by simply using the arctangent through a CORDIC algorithm given by the representation,

$$\varphi(t)=\arctan\frac{\operatorname{Im}(\underline{x}(n))}{\operatorname{Re}(\underline{x}(n))} \tag{3}$$

B. CORDIC Algorithm

The CORDIC (COordinate Rotation DIgital Computer) algorithm [5] is a method used to compute elementary trigonometric functions by means of planar rotation and vectoring. The coordinate system of the input parameter is rotated through constant angles until the angle is reduced to zero.

To calculate the rotation, trigonometric functions (cos and sin) are involved. This rotation angle θ in fact can be executed in steps by means of iterative process. By selecting the angle steps θ_n such that the tangent of a step ($\tan\theta_n$) is a power of 2, the multiplication operation can be eliminated and substituted by shift operation.

Practically, the iteration does not go up to infinity. In the hardware implementation, the resolution of the result is increased by one bit for each iteration, and hence look-up table of N entries is required to store the pre-computed arctan values where N is the no. of iterations to be performed.

Driving Z to zero, the CORDIC performs:

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}=\begin{bmatrix} P(X_i\cos Z_i - Y_i\sin Z_i) \\ P(Y_i\cos Z_i + X_i\sin Z_i) \\ 0 \end{bmatrix} \tag{4}$$

With initial values $X_i=1/P$, $Y_i=0$, and $Z_i=\theta$ we will have at the end of the iteration:

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}=\begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} \tag{5}$$

To suit our demand, we need to drive Y to zero instead of Z, and with $X_i=X$, $Y_i=Y$, and $Z_i=0$ we obtain

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}=\begin{bmatrix} P\sqrt{X^2+Y^2} \\ 0 \\ \arctan\frac{Y_i}{X_i} \end{bmatrix} \tag{6}$$

C. Loop Filter

The basic functionality of a loop filter is to stabilize the complete PLL. It does that by suppressing the unwanted higher frequency signals and allowing the low frequency signals through it such that only the required signals gets passed and not its harmonics which may cause erroneous results as the DDS will try generating the signals of the higher frequencies as well. This ADPLL makes use of a PI controller [6] as loop filter *F(z)* which satisfies the required properties. The transfer function for a PI controller is,

$$u(s) = (K_P + K_I \frac{1}{s})e(s) \tag{7}$$

This is discretized using bilinear transformation done by substituting,

$$s \leftarrow \frac{1}{T_S}(1 - z^{-1}) \tag{8}$$

This results in the IIR filter

$$F_{PI}(z) = \frac{b_0 + b_1 z^{-1}}{1 - z^{-1}} \tag{9}$$

Where,

$$b_0 = K_P + K_I T_S$$

$$b_{11} = -K_P$$

D. Direct Digital Synthesizer

This ADPLL model was synthesized using two different DDSs - CORDIC algorithm in the rotation mode and an LUT based sine waveform generator was also used. Both these methods find application but in different requirements. Since there is no strict memory constraint in case of using an FPGA a LUT based DDS can be used which produces faster result with very less delay. But when this ADPLL has to be fabricated onto a chip reducing memory and thus the cost of the hardware becomes a target which can be augmented using the CORDIC based DDS which stores very less values compared to the LUT based DDS. In addition to that, use of a CORDIC based DDS facilitates easy increase in the resolution of output since only the iteration number needs to be changed. Whereas in case of an LUT based DDS increasing the resolution becomes very tedious due to the fact that the complete LUT needs to be replaced by another LUT that stores values of the sine waveform in higher precision. Hence the use of the DDS is very application specific.

- *LUT based DDS:* In this method the values of the first quadrant of sine waveform is stored in an LUT. The values of the remaining quadrants can be found out by either inverting the result or inverting the argument and adding π/2 or doing both depending on the value of the phase accumulator. The values of the sine wave stored in a LUT were created using MATLAB.
- *CORDIC algorithm based DDS:* **T**his method used the CORDIC algorithm in the rotation mode which has already been discussed in section II-B.

III. SYSTEM IMPLEMENTATION

The Hilbert filter implementation involves the generation of an analytic signal by using Hilbert transform. The main advantage that an analytic signal provides is that there are no negative frequency components. The Hilbert transformation uses a multiplier less frequency sampling filter (FSF) as the Hilbert filter. Such type of filters can be taken as a generalization of comb filters which have the following transfer function,

$$F_{zN}(z) = 1 - z^{-N} \tag{10}$$

Such type of filters are characterized by the property of producing equidistant zeros on the unit circle at multiples of Ω=360˚/N. In order to cancel the zeros on the desired frequencies at the unit circle IIR filters are used to introduce poles at those frequencies. The zero cancelling IIR filters are represented by,

$$C_{\Omega}(z) = \frac{z^{-N}}{1 + a_1 z^{-1} + \ldots + a_N z^{-N}} \tag{11}$$

The different coefficients at different pole locations for the IIR filter has been listed in table I, which have been directly used from [4].

Thus the analytic filter is represented by,

$$\underline{H}_A(z) = \left\{ \frac{1}{4} Z_{-90}(z) Z_{-30/-150}(z) P(z) \right\}^n \tag{12}$$

Where, n is the order of the filter.

$$Z_{-90}(z) = 1 - jz^{-1} \tag{13}$$

$$Z_{-30/-150}(z) = 1 + jz^{-1} + z^{-2} \tag{14}$$

$$P(z) = \frac{z^{-2}}{1 - 0.5z^{-2}} \tag{15}$$

There is a trade-off between the brick wall characteristics of the band pass filter and the amount of damping at the negative frequencies, which depends on the order of the filter. More the order of the filter more is the damping effect of the negative frequencies. Thus an order of two is chosen for the Hilbert filter which optimizes the result so as to have both sufficient damping of the negative frequencies as well as an adequately sharp transition band.

TABLE I: COEFFICIENTS OF IIR FILTER EQUATION WITH CORRESPONDING ANGLES AT POLE LOCATIONS

$C_{\Omega}(z)$	a_1	a_2	a_3	Ω_1	Ω_2
$C_0(z)$	-1			0°	
$C_{180}(z)$	1			180°	
$C_{\pm 60}(z)$	-1	1		±60°	
$C_{\pm 90}(z)$	0	1		±90°	
$C_{\pm 120}(z)$	1	1		±120°	
$C_{0/180}(z)$	0	-1		0°/180°	
$C_{+90}(z)$	-j			+90°	
$C_{-90}(z)$	j			-90°	
$C_{+30/+150}(z)$	-j	-1		+30°	+150°
$C_{-30/-150}(z)$	j	-1		-30°	-150°
$C_{0/+90/+180}(z)$	-j	-1	j	0°/180°	+90°

The analytic signal thus generated for the input signal and the DCO output using the Hilbert transformer is used to get their corresponding phase values with the help of CORDIC algorithm. Here a CORDIC algorithm with a variable precision is presented. The phase value of the two signals thus found is subtracted and the difference is used to drive the loop filter of the phase locked loop. The PI controller has been coded in VHDL as per the equations described in the previous section. Also the CORDIC algorithm in rotation mode uses a variable precision method which allows the user to change the precision values in-field thus proving to be of slight advantage compared to the LUT based DDS.

The implementation of CORDIC algorithm [8] has been divided into four stages which are as follows:

- Sampling
- Pre-processing
- CORDIC core
- Post-processing.

IV. RESULTS AND DISCUSSION

The hardware implementation of the ADPLL in this project has been done on a Field-Programmable Gate Array (FPGA). It is called 'field-programmable' due to the fact that the circuit can be configured according to the designer's requirement even after manufacturing. The FPGA implementation is done using a Hardware Description Language (HDL) which in this case is VHDL. The ADPLL in this project was implemented using Xilinx ISE 12.3 and ModelSim PE Student Edition 10.1a. An FPGA platform has been chosen for the hardware implementation mainly due to the reasons that conventional microprocessors do not have enough computational power to support the ADPLL design and the Application Specific Integrated Circuit (ASIC) although meets the computational requirement would prove to be very costly when it comes to manufacturing of lesser numbers of the ADPLL. Thus an FPGA is an ideal choice due to the fact that it has the computational requirement that is present in an ASIC, and also it is in expensive and easily configurable.

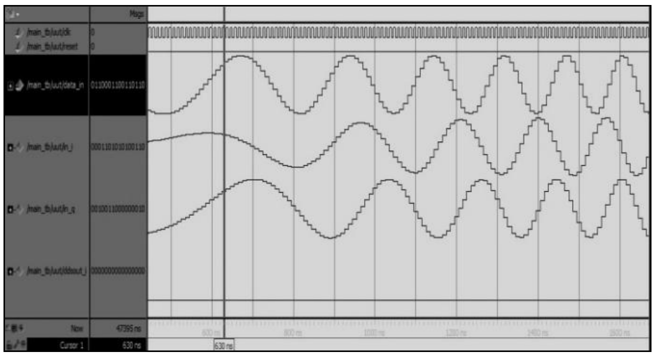

Fig. 2. Output of Hilbert Filter using ModelSim

The VHDL code for the Hilbert filter that damps the frequency components at -90°, -30° and -150° was simulated and the code was run with a sine waveform of varying frequency as can be seen in the Fig. 2. In this code the negative frequency components are damped using frequency sampling filters and poles are added at frequency of 0°, 90° and 180°.The output of the Hilbert filter was thus found to be matching the theoretical behavior as the in-phase and quadrature-phase components are shifted exactly by 90°.After successful simulation the code was dumped onto a Xilinx Spartan 3E board to check the device utilization against the available devices. Table II shows the device utilization table for the Hilbert filter using xc3s500e-4fg320 as the target device in Xilinx ISE 12.3 environment.

TABLE II: DEVICE UTILIZATION SUMMARY OF HILBERT TRANSFORM

Logic Utilization	Used	Available	Utilization
Number of Slices	325	4656	6%
Number of Slice Flip-Flops	587	9312	6%
Number of 4-input LUTs	352	9312	3%
Number of bonded IOBs	50	232	21%
Number of GCLKs	1	24	4%

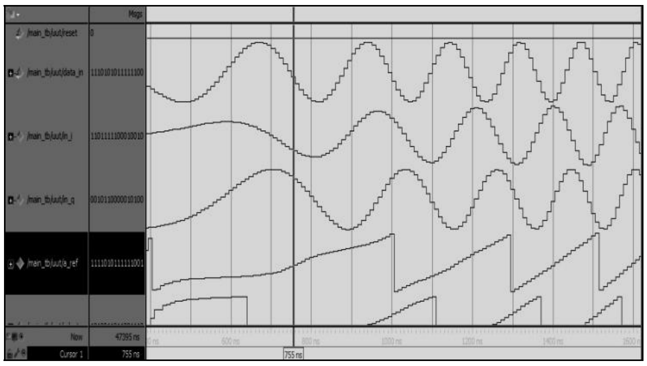

Fig. 3. Output of CORDIC algorithm using ModelSim

A 16 stage pipelined structure was implemented for CORDIC algorithm that produces 16-bit output of the instantaneous phase value of the signals. As is expected of the behavior of the code the phase value changes linearly from $-\pi$ to $+\pi$ for the sine wave is shown in Fig. 3.The ADPLL model uses two sets of phase detection system – one for the input signal and the other for the output of the DDS. The results i.e. the phase values of both these systems are then subtracted to get the phase difference of the two signals [10]. This result is then fed to the loop filter for processing. Following to the simulation of the CORDIC algorithm its device utilization was checked and is given in Table III.

TABLE III: DEVICE UTILIZATION SUMMARY OF CORDIC ALGORITHM

Logic Utilization	Used	Available	Utilization
Number of Slices	592	4656	12%
Number of Slice Flip-Flops	1027	9312	11%
Number of 4-input LUTs	1161	9312	12%
Number of bonded IOBs	66	232	28%
Number of GCLKs	1	24	4%

A PI controller used as a loop filter which feeds the phase difference of the two signals to produce a 16-bit output. This output as shown in Fig. 4 is then used to drive the DDS. The loop filter was also simulated using MATLAB Simulink® and from the results obtained as shown in Fig. 5, it can be seen that the settling time for the loop filter is around 1.5 seconds and also the overshoot of the loop filter is very less. This indicates that the system is stable and the loop filter is efficiently able to discard all higher order harmonics as per the specific requirement.

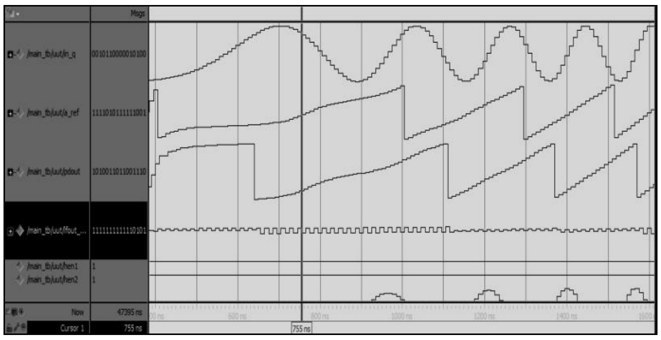

Fig. 4. Output of Loop Filter using ModelSim

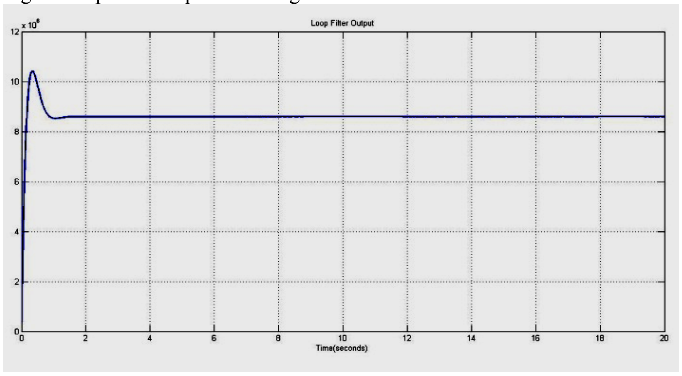

Fig. 5. Time response of loop filter

The phase error plot over time has been shown in Fig. 6. The plot shows that although the variance of the phase error is very high at the start, the phase error reduces to zero at a very short time thus indicating that the efficiency of the ADPLL design is high. As can be seen from the plot, the phase error reduces to zero almost within 1.5 seconds. This also sheds some light on the stability of the system indicating that the system is indeed very efficient in nature.

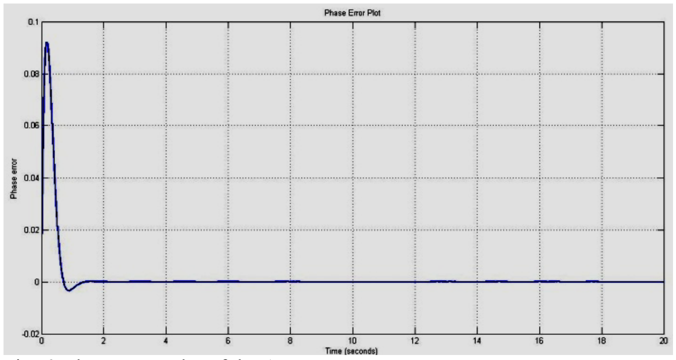

Fig. 6. Phase Error plot of the ADPLL.

A CORDIC algorithm in its rotation mode is used as a DDS which when provided by the phase produces the corresponding sine value. Shown in Fig. 7 is a simulation result for the DDS working at 50 kHz frequency. Table IV shows the device utilization of the DDS tabulated after synthesizing the DDS in Xilinx ISE 12.3.

TABLE IV: DEVICE UTILIZATION SUMMARY OF DDS

Logic Utilization	Used	Available	Utilization
Number of Slices	376	4656	8%
Number of Slice Flip-Flops	711	9312	7%
Number of 4-input LUTs	727	9312	7%
Number of bonded IOBs	50	232	21%
Number of GCLKs	1	24	4%

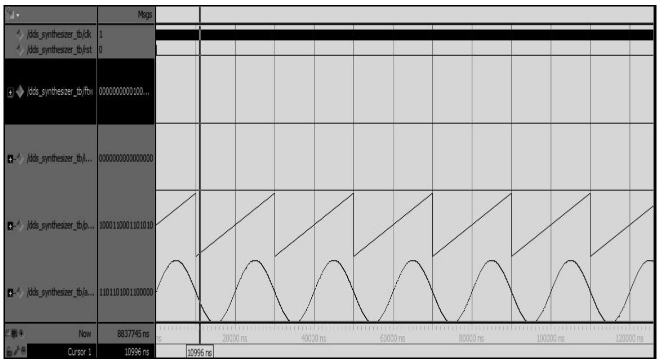

Fig. 7. Output of DDS using ModelSim

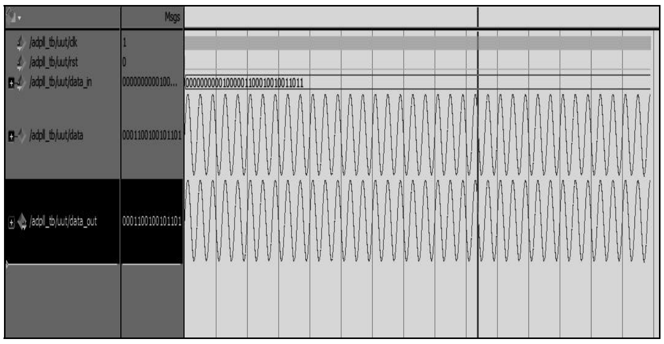

Fig. 8. Output of ADPLL using ModelSim

After successfully simulating individual modules of the ADPLL these components were integrated in one VHDL file and simulated together. The result is shown in Fig. 8 where the output gets locked to the input almost instantaneously and thus follows it continuously. The behavior of the ADPLL was found to be as expected from the theoretical explanations.

V. CONCLUSIONS

In this paper, a detailed analysis of phase detection system involving the Hilbert transform and the CORDIC algorithm has been discussed. The design of the ADPLL in this project does not involve any class of multipliers which implies that the complexity of the circuit is reduced. Also compared to conventional PLL models of phase detection like the PFD's and TDC's this method of phase detection provides the advantage of easy calibration, programmability, and higher phase accuracy with minimum phase error. This conclusion is derived from the fact that the CORDIC algorithm used in this project is of 16-bit in nature which implicitly provides a phase accuracy of arctan $\left(\frac{1}{2^{16}}\right)$ degrees, which comes to a precision of 0.0008 degrees. The CORDIC algorithm in its vectoring mode of operation which has been used as the DDS system for the ADPLL provides a definite advantage over the LUT based method of readjustment of precision on the field itself. Also the LUT based method would require more memory since it had to store all possible sine values of a quadrant for a particular precision. But since the CORDIC algorithm computes the sine value at each sample the memory requirement is not an issue in this case. Also the current trend in digital signal processing applications shows a shift towards ultra-short waveband and above too, and this ADPLL can still be modified and improvised upon in the near future to cater to the needs of future demands for such designs.

REFERENCES

[1] Martin Kumm, Harald Klingbeil, and Peter Zipf, Member, "An FPGA-Based Linear All-Digital Phase-Locked Loop", IEEE Trans. on Circuits Syst-I, Regular paper, vol. 57, no. 9, Sept. 2010. Pp.2487-2497.

[2] H. Klingbeil, "A fast DSP-based phase-detector for closed-loop RF control in synchrotrons," IEEE Trans. Instrum. Meas., vol. 54, no. 3, pp. 1209–1213, Jun. 2005.

[3] H. Klingbeil, B. Zipfel, M. Kumm, and P. Moritz, "A digital beam-phase control system for heavy-ion synchrotrons," IEEE Trans. Nucl. Sci., vol. 54, no. 6, pp. 2604–2610, Dec. 2007.

[4] M.Kumm and M. S. Sanjari, "Digital Hilbert transformers for FPGA-based phase-locked loops," *International Conference on Field Programmable Logic and Applications, 2008. FPL 2008,* pp. 251–256, 2008.

[5] Abhishek Das, Suraj Dash, B.Chitti Babu and A.K.Sahoo, "Cordic Algorithm based Novel Phase Detection System for All Digital Phase Locked Loop", Journal of Computational Intelligence and Electronics Systems, vol.1, no.1, pp.01-08, Dec. 2012.

[6] Lima, J.; Menotti, R.; Cardoso, J.M.P.; Marques, E.; , "A Methodology to Design FPGA-based PID Controllers," IEEE International Conference on Systems, Man and Cybernetics, 2006. SMC '06., vol.3, pp.2577-2583, . 2006.

[7] U. Meyer-Baese, "Digital Signal Processing with Field Programmable Gate Arrays", 3rd Edition. Berlin: Springer, 2007.

[8] J. E. Volder, "The CORDIC trigonometric computing technique," IRE Trans. Electron. Comput., vol. 8, no. 3, pp. 330–334, Sep. 1959.

[9] A. Guntoro and M. Glesner, "Resolving longitudinal amplitude and phase information of two continuous data streams for high-speed and real-time processing", Journal of Advances in Radio Sciences , pp 133-137, 2009.

[10] Abhiskek Das, Suraj Dash, B.Chitti Babu, A.K.Sahoo, " A Novel Phase Detection System for Linear All-Digital Phase Locked Loop", In *Proc, IEEE Students' Conference on Engineering & Systems (SCES 2012),* MNNIT, Allahabad, pp.1-6, 2012.