iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** V **Month of publication:** May 2022

DOI: https://doi.org/10.22214/ijraset.2022.42246

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue V May 2022- Available at www.ijraset.com

The Design of DDS ADPLL using ARM Micro Controller

Mohd Ziauddin Jahangir[1], Chandra Sekhar Paidimarry[2], Mohammed Sikander[3]
[1, 3]*Electronics and Communication Engineering, Chaitanya Bharathi Institute of Technology, 18-8-104/98, P.N. Nagar, Hyderabad, 500075, Telangana, India, Tel.: 09885717845*
[1, 2]*Electronics and Communication, Osmania University College of Engineering, Osmania University, Hyderabad, 500007, Telangana, India*

Abstract: Abstract In this article, a full custom design and implementation of a sine wave All Digital Phase Lock Loop (ADPLL) system based on ARM microcontroller is described. These ADPLL implementations are also referred as Software - Direct Digital Synthesis ADPLL (DDS-ADPLL). In the literature, the work related to software DDS ADPLLs is to achieve a specific transient response or dynamic behavior. However, there are few literature that addresses the issues encountered while implementing the Sine wave ADPLL on hardware. This work identifies the problem one may face while implementing the DDS-ADPLLs on microcontroller and proposes methods / solutions to mitigate those issues. Moreover, in order to minimize jitter it is proposed to use timer interrupts and integer datatype. The design procedure of this work uses only integer datatype for implementation.
Keywords: DDS ADPLL; ADPLLs; All Digital PLL; Phase Locked Loop; LPC2148ADPLL

I. INTRODUCTION

THE ADPLLs, and specially a sine wave ADPLLs (also called as DDS-ADPLLs) are finding their ways into new applica- tions every day due to their low cost implementation and robustness they provide. Applications which require low frequency signal synchronization are extensively using DDS ADPLLs which are either implemented using Field Programmable Gate Array (FPGA) / Microcontrollers. However, designing these ADPLLs have always been a challenging task. If implemented using microcontrollers it will suffer with a disadvantage of lower sampling frequency. As it is known that microcontrollers are clocked at a rate of few tens/ hundreds of MHz. If the firmware that is being run to provide PLL functionality is longer in size, then it will lower the iteration rate. Hence, in this article a complete design methodology for implementation on microcontroller is described in detail.

As most of the microcontrollers by nature consist of integer ALU and not floating-point ALU, this work aims to describe the design procedure using only integer data types. No floating-point data type is used in this work, the result of this will be a faster implementation and a possibility to increase the frequency range of operation of ADPLL.

The text has been organized in the following order. First we introduce PLLs, then a brief classification of various types of PLLs is presented, then the proposed ADPLL topology for implementation is described. The hardware implementation, and important design - issues in the hardware are described. Proposed algorithm for the design of Sine Wave ADPLL on microcontrollers is presented. At last, the results obtained after implementing the proposed ADPLL topology on LPC2148 microcontroller are presented.

II. CLASSIFICATION OF PLLS PLL IS A BASIC BUILDING BLOCK OF MANY ICS AND SYSTEMS.

It has applications which include phase locking, frequency synthesis, clock synthesis, carrier generation, etc.

A PLL is made up of three blocks i.e. a Phase Detector (PD), Low Pass Filter (LPF) and a Voltage Controlled Oscillator (VCO) as shown in the Figure 1. The dynamic behavior of the above PLL exhibits a second order transfer function for a first order LPF and simple phase detector. It has been extensively studied in the literature, as in [1] [2][3].

Many implementations of the PLL are possible. For example, it is possible to implement PD, LPF and VCO as analog components or all the three can also be implemented in digital domain. It is also possible to implement some of them in analog and the remaining in the digital domain. [2, 4, 5] . It is also possible to implement some of them in analog and the remaining in the digital domain [6–9] . Hence, considering all combinations a lot of different varieties of PLLs are possible. Based on their implementation these are either called PLL / Digital PLL (DPLL)/ All Digital PLL (ADPLL). An ADPLL is the one in which all components are implemented in Digital Domain. i.e. PD, LPF and VCO. In digital implementation a VCO needs to be replaced by a Numerically Controlled Oscillator (NCO). [10] [11](in digital implementation a VCO needs to be replaced by an NCO).

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue V May 2022- Available at www.ijraset.com

Further, these ADPLL can be implemented in various sub-blocks. Different architectures also exist for ADPLL working on different type of signals. For example, some ADPLLs work with square wave [5] while other may work with sine wave [4, 12]
In this work A sine wave ADPLL is implemented on a microcontroller.

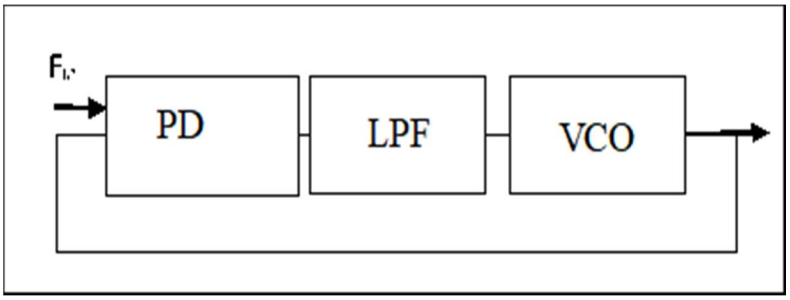


Fig. 1 Block diagram of conventional PLL

III. CONSIDERATIONS FOR SELECTION OF ARCHITECTURE FOR DDS-ADPLL

The architecture used in this work is shown in the Figure 2 . Though Many options exists for Phase Detector and LPF, the architecture implemented in this work tries to be simple by using an EX-OR Phase-Detector [13] and digital PI controller [1] as LPF. As it is an ADPLL the most common type of DDS based NCO [14]is used as VCO. It is informed at this point that a PFD [1] [3] [13] can be used instead of a simple EX-OR phase detector in order to get very large lock range. The subsequent discussions are also valid for an ADPLL with PFD.

IV. ARCHITECTURE OF A DDS-ADPLL

A. Phase Detector

The ADPLL topology selected an EX-OR gate is used as phase detector. EX-OR gate can be used as phase detector only for Square Wave Signal (i.e. with uni-polar signals that vary between GND and VDD). But as the intention of this work is to design a sine wave ADPLL. Therefore, it is first required to convert the Sine wave to Square wave as shown in, Figure2.
It is again to be noted that other way of mitigating this problem is to replace EX_OR phase detector with a sine wave phase detector like CORDIC Phase Detector or multiplying phase detector [14].

B. LPF

The next stage of ADPLL is an LPF. There are many different filters that can be used, The dynamic response of the system by-large will depend on the type of filter. In Analog implementation of these filters usually first order filters are used [13] [8].

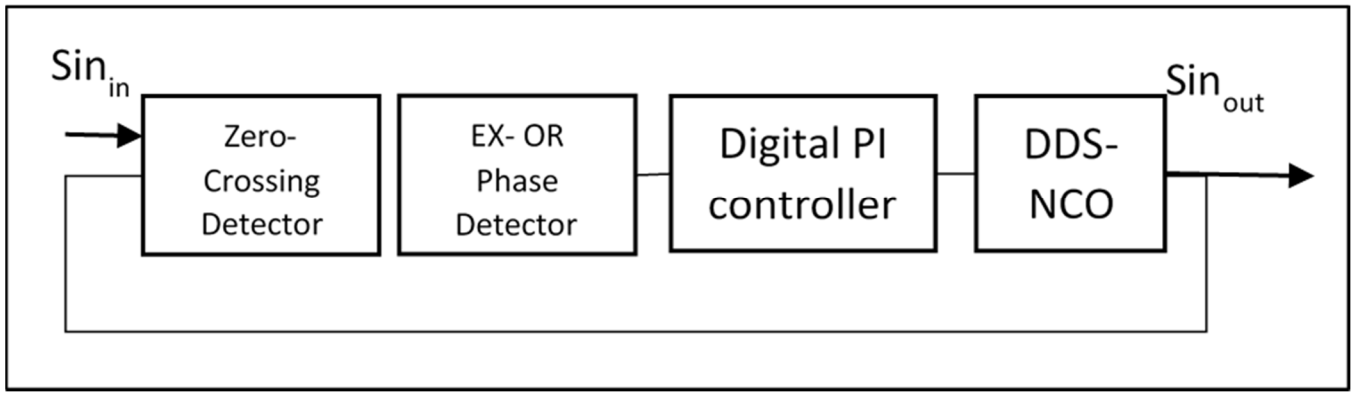


Fig. 2 DDS-ADPLL architecture that can work with Sine Waves

On the other hand, digital implementation requires IIR filters. LPF proposed here does not use fractions. Instead, it is of integer type.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue V May 2022- Available at www.ijraset.com

V. CONSIDERATION FOR THE HARDWARE DESIGN

Now we consider the hardware implementation of the architecture shown in Figure 2 . As it will be seen in the following sections, the design of sine wave ADPLL is dictated by the architecture and the underlying hardware. Therefore, in this section we highlight the considerations for hardware design. As the Microcontrollers are digital ICs and require I/O to be in digital format i.e. 0 volt and VDD, it is not possible to directly connect sine wave input or to obtain sine wave output from the digital pins of microcontrollers. Therefore, in this scenario we need to interface ADC and DAC to the microcontroller to connect input and output respectively to the microcontroller. Which means some extra hardware needs to be interfaced in order to obtain a fully functional hardware which supports ADPLL implementation as shown in theFigure 3

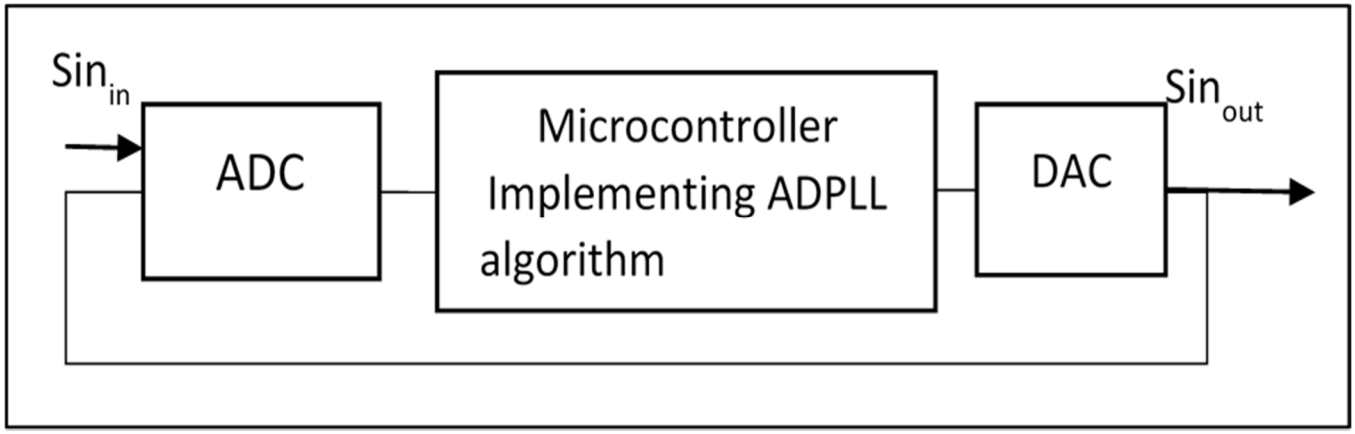


Fig. 3 Hardware required forMicrocontroller based ADPLL

The Figure 3is too simplistic, in practice the ADC/DAC comes with their own interfacing like SPI, I2C etc, which further complicates the implementation. Further, the hardware needs to also contain signal conditioning units at input and output. The need for single conditioning circuits arises because the ADC and DAC will be mostly uni polar in microcontrollers. The Figure 4represents the total hardware required for implementing the ADPLL on microcontroller.

A. Development of algorithm for implementing the DDS-ADPLL architecture

In this section we convert the above architecture shown inFigure 2 to an appropriate algorithm that can be executed on a microcontroller.

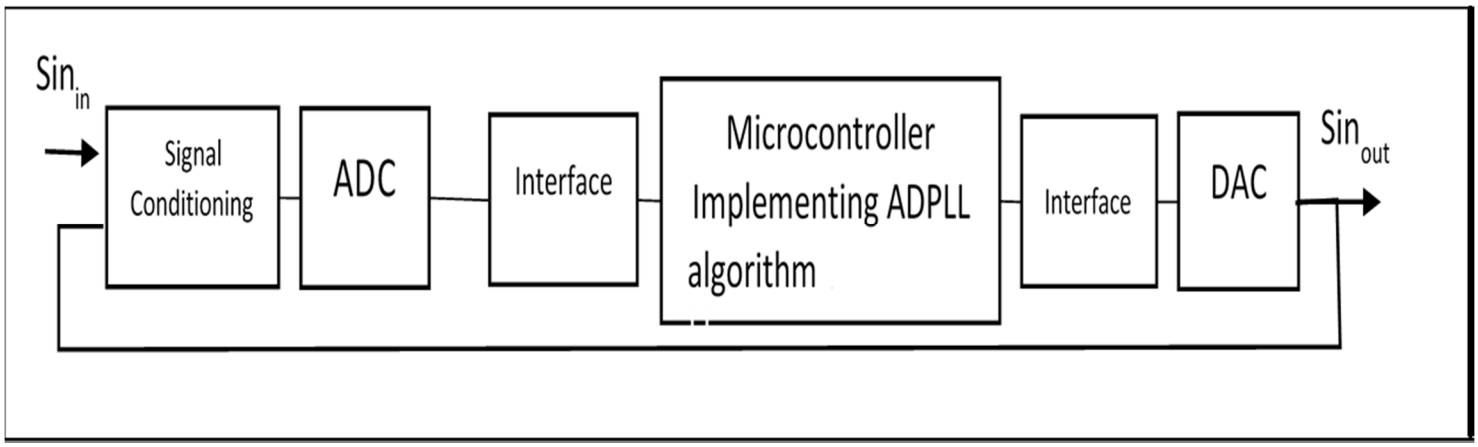


Fig. 4 Complete Hardware required for ADPLL implementation microcontroller including signal conditioning unit.

The algorithm shown below is the ADPLL implementation corresponding to the architecture shown inFigure 2

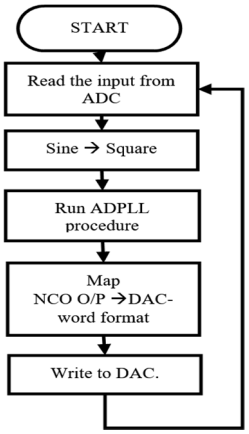


Chart 1: Sine Wave ADPLLalgorithm for microcontrollers.

The algorithm is quite simple it begins by reading ADC Input sample, converting them into square wave, running the ADPLL procedure and sending the value to the output DAC.

However, there are two important considerations that should be taken into account while implementing this algorithm. These considerations are described in detail in

The ADPLL procedure which is called in this flow chart needs a special attention and hence is described below in further detail.

VI. ALGORITHM OF SINE-WAVE ADPLL

A. Converting Previous Sine O/P to Equivalent Square

One of the important point to remember while implementing ADPLL is to implement the Negative Feed-Back. The generated output has to be fed Back to compare with the input. However, in digital implementations, the current input needs to be compared with previous output but not the current output, as it is not available concurrently.

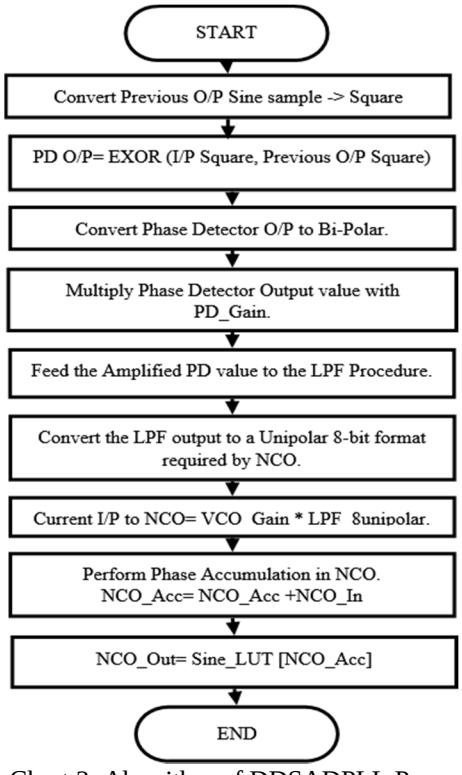


Chart 2: Algorithm of DDSADPLL Procedure

B. Phase-Detection

The Next step of ADPLL algorithm is to perform a phase detection. As per the architecture shown inFigure 2 , this should be done using an XOR gate.(Though different Phase Detectors can be used based on requirement). Therefore, the input value and the previous value of Square converted output of the NCO are XOR.

C. Conversion of Phase-Detector Output to Bipolar

This step is an extremely important step in the implementation of ADPLL. In this step, the output of XOR Phase-Detector is converted to bipolar. i.e., the value which was in the range of 0 to +1, will now be either -1 or +1. This conversion is especially important as this value is fed to the next stage which is the LPF. LPF usually have integration functionality. If a uni-polar value is fed to it, the output of LPF will rise to Infinity with the time and the system saturates. If we want the system to stabilize or converge then the integration should result in a non-infinite value. i.e. a finite value which can only be possible by having both positive and negative numbers. This is illustrated in theFigures 5 and 6. Figure 5 (a) shows a uni polar wave and the result of its integration is shown in Figure 6 (a). Similarly, the bipolar wave and the result of its integration is shown inFigure 5 (b) and Figure 6 (b) respectively. As, is visible inFigure 6 (a) the result is unbounded, whereas the Figure 6 (b) result is confined within bounded limits. Therefore, converting uni polar output of PD to Bi-Polar is by large the most important step in the implementation of a working stable ADPLL.

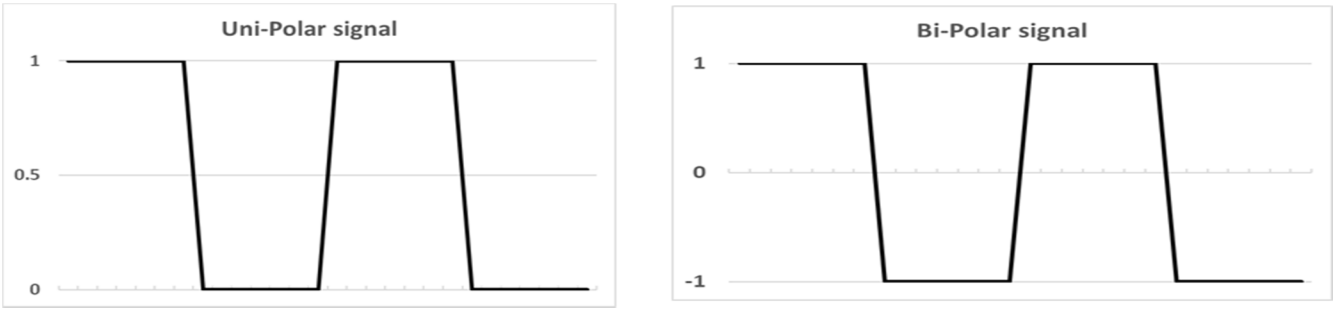


Fig. 5 Difference between uni-polar and Bi-Polar signal

Step-3:

The filter is implemented as a converging, iterative discrete function represented in [1] . The magnitude spectrum of above equation is that of a low pass filter. The discrete time domain function of the filter implemented in this work is represented inEquation (1) . Though it is mandatory to use floating-point data type for filtering operation. The implementation of our work would require just integer data types for implementing the filtering operation. However, this may lead to some degrees of error which is tolerable because the limitations imposed by sampling frequency dominate the quantization error.

$$y(n) = \alpha_0 \times y(n-1) + \beta_0 \times x(n) \quad (1)$$

The values for converging response for bandwidth values as shown inEquation (2) :

$$y(n) = 0.833 \times y(n-1) + 0.167 \times x(n) \quad (2)$$

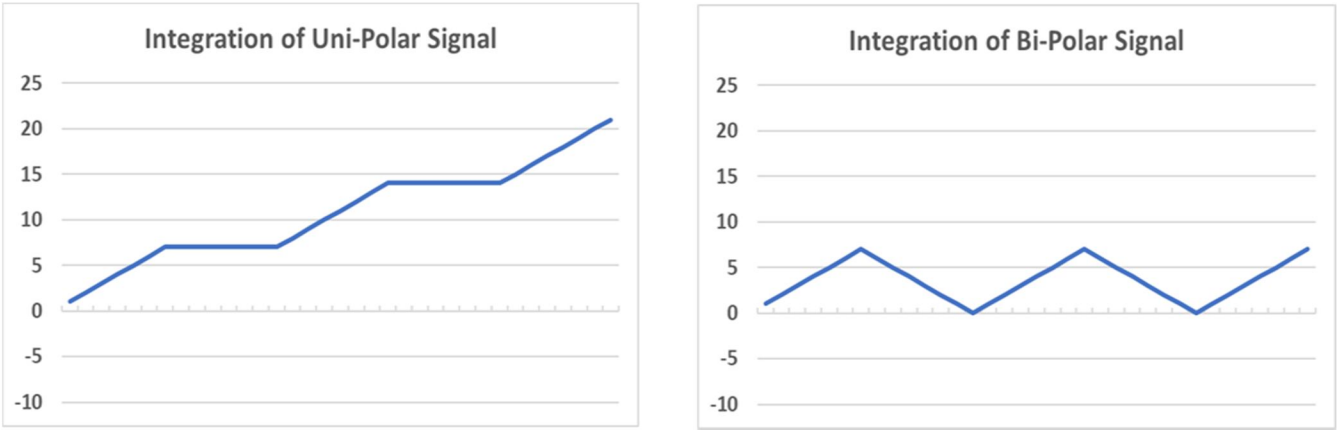


Fig. 6 Result of integration of uni-polar and Bi-polar signals.

D. Architecture of an NCO

An NCO shown inFigure 7 is a digital version of a voltage-controlled oscillator. It uses numerical arithmetic to generate output of variable frequency. The frequency of the output depends on the NCO input word.

As can be seen in the Figure 7 above an NCO consist of a Phase-Accumulator and a DDS Look-up Table, plus some logic to adjust widths of various variables. The width of Input to Phase Accumulator, Phase Accumulator's actual width and width of output from Phase accumulator are all different and are treated as parameters of design, the choice of which is heavily affected by the depth of Sine LUT, sampling frequency etc. The details of this shall be shortly published by the authors of thistext.

VII. IMPORTANT CONSIDERATIONS FOR SOFTWARE DESIGN

There are two very important issues or bottlenecks that arises in the implementation of ADPLL architecture described in section-6 on a micro controller. They are,

- Choice of Data-types of variables.
- Synchronizing the algorithm to sampling frequency.

These considerations are in addition to that of dynamic response. As dynamic responses is extensively studied in [1, 3, 13] , We try to focus on above two issues only.

A. Data- Types

Major deviation in the implementation of ADPLL on micro-controller arises from the fact that datatype that can be used on micro-controllers are limited. Unlike FPGA / ASIC implementation where we can have any custom datatype like Fixed-Point e.g. 2.13, etc., in Microcontrollers we have to use the data types supported by compilers which will be like char, int, float, etc. At max, we can specify whether we want a signed version of above datatype or an unsigned version of them. But, apart from the options mentioned above, no other choice exist. Char data types provide us with a 8-bit data word, a short int with 16-bit and a long-int with 32bit data-word. And all these are non-fractional datatype. If we require fractional datatype we can use the data-types like float and double, But, it is to be noted here that most of the microcontrollers have an integer ALU, they are not meant to perform floating-point arithmetic. But the compilers like ARM -C supports it. Which means if any floating-point arithmetic is instruction is written in the program then the compiler replaces it with a very lengthy machine code. This is obvious as the ALU of microcontroller is not meant for fractional arithmetic and compiler is trying hard to implement the floating-point algorithm.

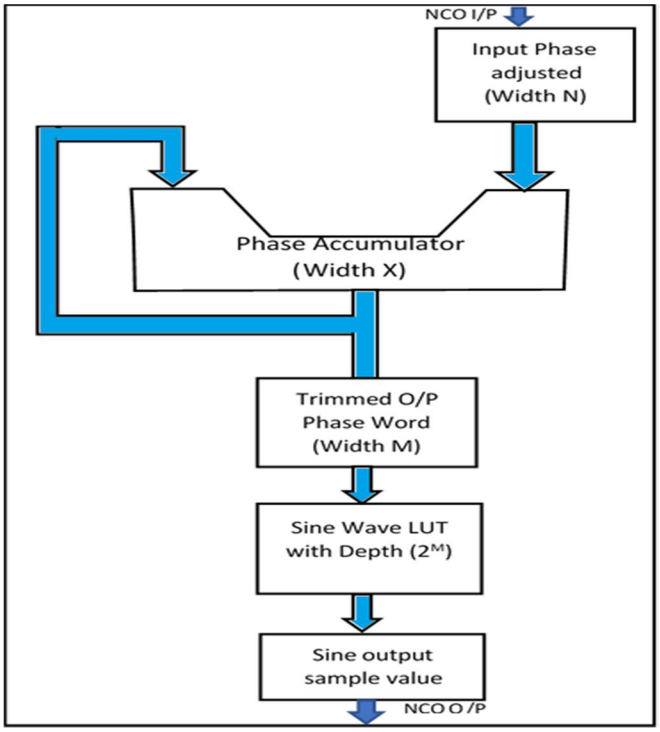


Fig. 7 Architecture of Numerically controlled Oscillator

The reason why we can't tolerate this delay is because of the application involved i.e. ADPLL. The dynamic performance of an ADPLL is related to the delay experienced in the loop. In ADPLL to maintain a constant phase it is required that the Process is able to compute and take corrective measure in a very small interval. But if this processing becomes longer, it not only affects the frequency range of ADPLL but also introduces a considerable amount of jitter in the output.

The delay experienced by each sample to produce a corresponding result is $t_{Delay.}$ which can be given by the equation below.

$$tDelay = tsampling + tproces\sin g$$

One more reason why the consideration of Data-type matters is because, the architecture that is selected in this work requires different-data widths at different scenarios. For an example in order to store a Look-up table of sine it is required to limit to finite number of samples.

VIII. DATA TYPE CONSIDERATION FOR PHASE DETECTOR:

The algorithm begins by reading the value from ADC. And as required by the architecture shown in Figure 2,the read input value needs to be converted to a square wave. To do this a threshold voltage is f $\frac{VDD}{2}$ xed. Any input value greater than this threshold value is treated as logic one and any value less than this is treated as logic zero. In other words, we implement a zero-crossing detector in software. In this implementation as we are using the inbuilt ADC of LPC 2148. Which is 10-bit ADC, with reference voltages as 0V and 3.3V. The threshold voltage would be 1.65V. The 10-bit value corresponding to it will be 10'b10_0000_0000. Any value great then this binary number is treated as logic one and any value less than or equal to this binary number is treated as logic zero.

IX. DATA TYPE CONSIDERATION FOR I/P ADC SAMPLES:

One important design considerations that needs to be taken is about the data type that should be used to store the input ADC sample values. As there are no 10-bit data types supported by compiler, we would have to take a 16 bit word to store this values which corresponds to a Shot Int data type in C language. As the output of ADC is of unsigned, the data type used to capture this value should also be of unsigned type.

X. DATA TYPE CONSIDERATION FOR SINE- SQUARE CONVERTER:

When the sine wave sample values are converted into square wave, the output should be a binary value, in other word we require a Boolean data type. But it was found that the ARM C compilers didn't support Boolean data type. Therefore, an unsigned CHAR data type is used to store the square waves sample. As, it is the minimum possible word-size supported by the compiler.

XI. DATA TYPE CONSIDERATIONS FOR NCO:

The temporary variable used while iterating the DDS values should be of unsigned-char type. This makes sure that the counter when overflows, will again point to the starting element of the DDS array which is 256 word deep.

XII. DATA TYPE CONSIDERATIONS FOR LOOK UP TABLE:

The width of Sine samples which are stored in the memory has a limit. This is because ultimately, the wave has to be generated using the internal-DAC of the microcontroller, which is only 10-Bit wide. But, there doesn't exist a 10-bit data-type in C-Compiler. Therefore, suitable data type need to be chosen for these variables, so that they map well to hardware and produce an optimum binary executable.

XIII. DATA TYPE FOR CONSIDERATIONS FOR DAC:

The square wave sample that is obtained , is fed to the ADPLL procedure for further processing. The details of a ADPLL procedure are described in the sections-VI.

The output from the ADPLL will be sine wave sample values which are taken from the lookup table. These values should be converted to the format required by the DAC. The DAC of LPC 2148 requires a 10 bit input word. Therefore, the sine wave sample values generated by the NCO of ADPLL needs to be right shifted 6 times in order to obtain the format required. The next step would be to write the output sample value into the DAC procedure.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue V May 2022- Available at www.ijraset.com

A. Synchronizing the Algorithm to Sampling Frequency:

The next issue encountered in the implementation of ADPLL is to get the exact sampling frequency. The architecture shown in Figure 2will be able to track input frequency with less jitters only when the sampling frequency is maintained constant.

If the above algorithm is written in a super-loop i.e. a while(1) loop, then the sampling period of each cycle will not be the same. As the loop may get delayed waiting for the ADC to complete conversion or at DAC.

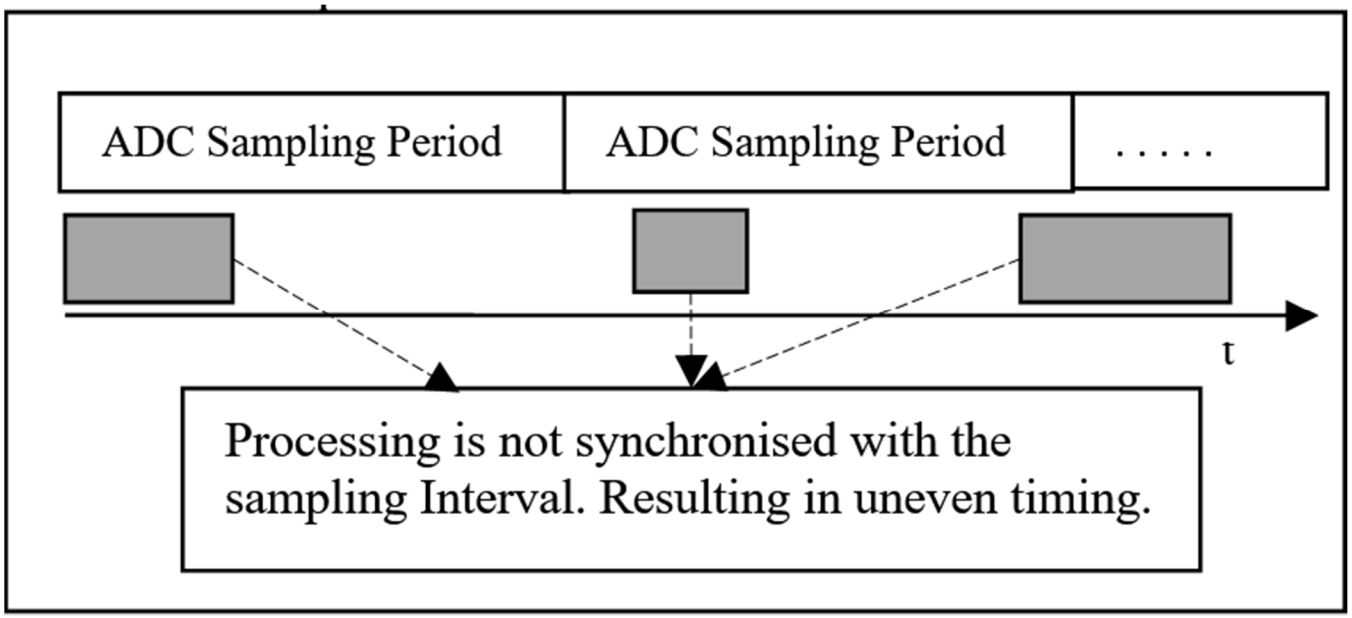


Fig. 8 Processing notsynchronized, resulting in uneven processing times

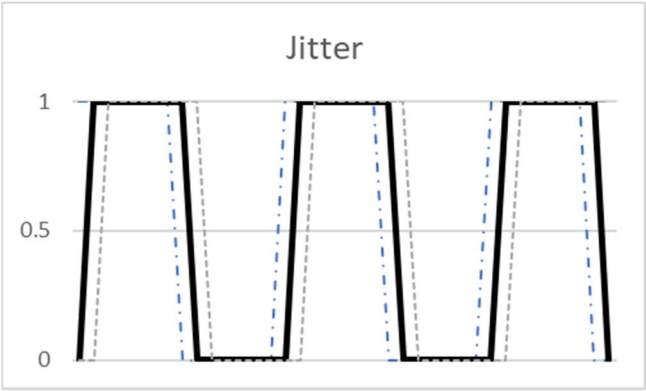


Fig. 9 sample square waveform demonstrating Jitter

The method of synchronizing each operation to the sampling frequency can be done by implementing the above repetitive algorithm not in the loop but as an ISR for Timer Interrupt. The Timer regularly interrupts the the microprocessor at the specified time interval and whenever the Interrupt is generated the algorithm is run. Therefore, ISR provides the best known mechanism of implementing the sampling frequency. Therefore, setting the time interval of generating interrupt is analogous to setting the Sampling time Period. The best value of timer interrupt would be that of the ADC sampling Interval. But, it need not be always, it can be a value that is integer multiple of ADC sampling value. For an example, It is possible to make timer frequency double that of ADC sampling frequency,. In such case the I/P value may not change but the O/P gets calculated twice and perform the loop corrections more rapidly.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue V May 2022- Available at www.ijraset.com

The Figure 10 shows the proposed method of synchronizing the sampling frequency with the processing with the help of a timer ISR.

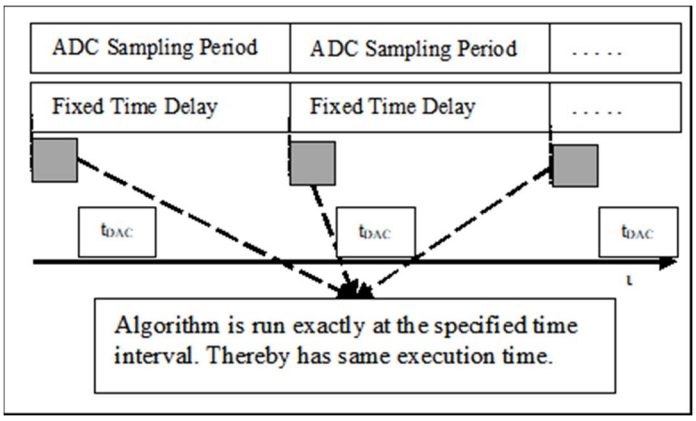


Fig. 10 The processing is synchronized with the Sampling Frequency

The method to write ADPLL program using the timer ISR is shown in the Figure 11 .

```
Void Main()
{
   Initialize IO();
   Initialize Timer();
   Initialize ADC();
   Initialize DAC();
   Initialize Timer  Interrupt ();
   While(1)
   {
      Sleep();
   }
}
ISR for Timer Interrupt()
{
   Sine wave ADPLL algorithm
        Is implemented here.
}
```

Fig. 11 Algorithm implemented in embedded-C. The ADPLL algorithm is synchronised with the help of Timer ISR

The ARM LPC2148 microcontroller is chosen for this implementation because of two reasons. First is it has inbuilt ADC and DAC and second is it has a reasonable good operating frequency from 50MHz which can be scaled upto 200MHz. The Timer peripheral can operate at a maximum frequency of 50MHz/4 which can again be scaled to 200MHz/4.

XIV. RESULTS

The Procedure described above was used to design a sample ADPLL on LPC2148 microcontroller. LPC2148 is an ARM based microcontroller with buit in ADC, DAC and Timers i.e. all those hardware mentioned in section IV to implement ADPLL.
The ADC and DAC are 10 bit wide.
A 256 word look up table was used with an 8-bit input NCO-Word. The output of NCO was also set to 8-bit in order to match with the depth of LUT.
The input signal is applied using a function generator with peak to peak amplitude 3 volt and a DC offset of 1.65V. The output was observed on a 20MHz DSO.
The Input was swept from 100Hz to 2KHz manually in order to find the range of locking.

A. Experimental Setup

Fig. 12 Experimental Set-up: A linked setup with host connected to LPC2148 evaluation board which in-turn is connected to function generator for providing input and a DSO to view Input and output time domain waveform.

TheFigure 12above shows the experimental set-up made for this work. LPC2148 evaluation Board used here is linked using a USB-Com port to a Host PC running Windows OS and installed with Flash Magic ROM burner software. The entire design is done using Keil microvision-5 IDE in the VLSI-Lab of Dept. of ECE, CBIT. The image also shows the DSO displaying the Input and Output waveform. A function generator was used to provide the input.
The Figure 13 is the screen shot of DSO showing the time domain input and output signals. Blue colored signal is the Input applied to the above development board though a function generator, The input signal applied, has the DC-Offset in order to make the signal uni-polar. The maximum amplitude of the signal is restricted below 3.3 volt i.e. reference voltage of the ADC. The sampled signal is the output generated from the microcontroller based on ADPLL.

The ADPLL output was found to follow the input signal from 580Hz to 750 Hz. The frequency based phase offset – which is the characteristic of any Ex-OR phase detector based PLL [13] was found to exist and it was found to vary with the input frequency. TheFigure 14 represent the X-Y mode graph of the ADPLL @ a frequency of 678 Hz.
Figure 15 andFigure 16shows the time domain input, output and X-Y mode graph of the ADPLL @ a frequency of 622 Hz.

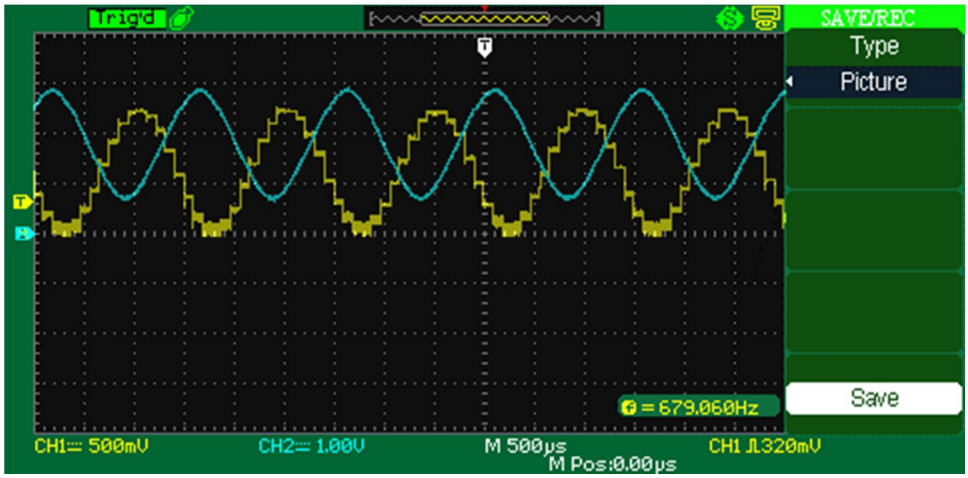


Fig. 13 ime Domain Input and Output of Sine Wave ADPLL implemented on LPC2148 @ frequency 678Hz

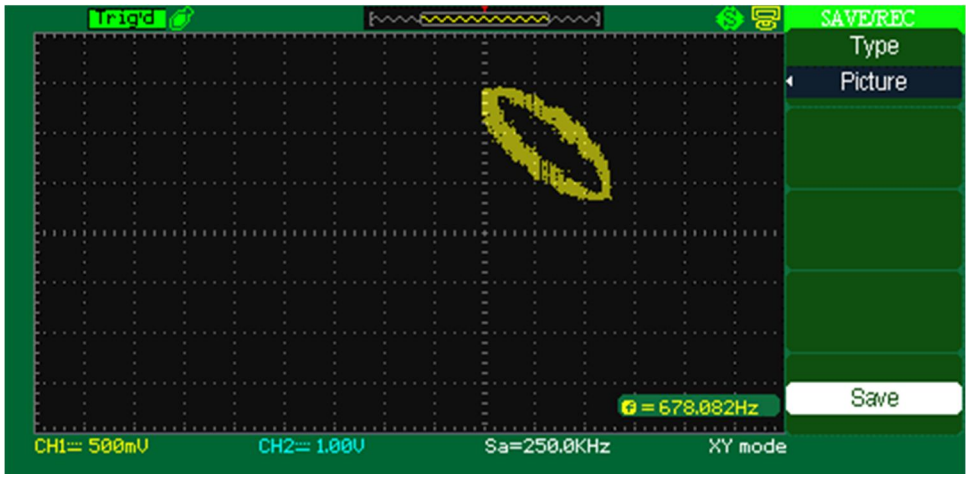


Fig. 14 X-Y mode graph of Sine Wave ADPLL implemented on LPC2148 @ frequency 678Hz.

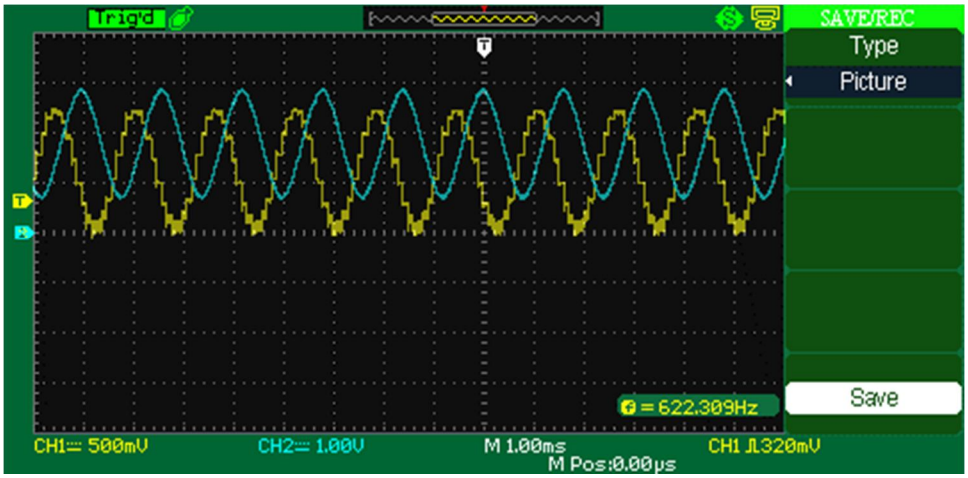


Fig. 15 Time Domain output of sine wave ADPLL implmented on LPC2148 @ 622 Hz frequency

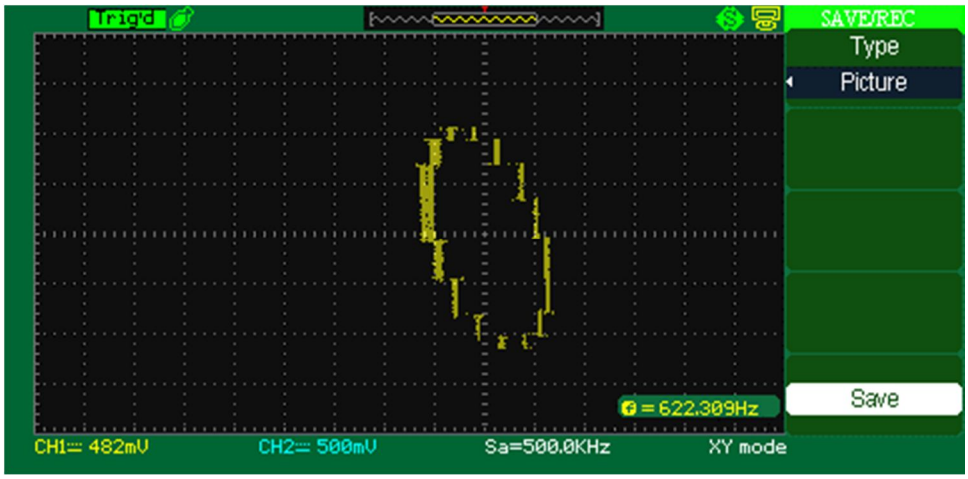


Fig. 16 Y mode graph of Input and Output of Sine wave ADPLL implmented on LPC2148 @ 622 Hz frequency

XV. CONCLUSION

In this work complete design procedure and design cosiderations while designing hardware and software for microcontroller based DDS-ADPLL are highlighted and solutions are proposed to mitigate them. The complete hardware assembly required, the importance related to data-types and synchronization were highlighted and solutions were provided. Following above procedure and consideration a Software DDS-ADPLL was implemented on a LPC2148 (ARM) micro-controller. The designed ADPLL demonstrated the frequency locking from 580Hz to 750Hz. The sampling frequency was set by the timer interrupt.

REFERENCES

[1] V. Kratyuk, P. K. Hanumolu, U. Moon, and K. Mayaram, "A Design Procedure for All-Digital Phase-Locked Loops Based on a Charge-Pump Phase- Locked-Loop Analogy," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 54, pp. 247–251,2007.
[2] A. Patil and R. Saini, "Design and implementation of FPGA based linear all digital phase-locked loop," in 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE), 2014, pp. 1–1.
[3] N. D. Dalt, "Linearized Analysis of a Digital Bang-Bang PLL and Its Validity Limits Applied to Jitter Transfer and Jitter Generation," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 55, pp. 3663–3675, 2008.
[4] A. Singhal, A. Goen, and T. Mohapatra, "FPGA implementation and power efficient CORDIC based ADPLL for signal processing and application," in 7th International Conference on Communication Systems and Network Technologies (CSNT), 2017, pp. 325–329.
[5] C. Tzeng, S. Huang, and P. Chao, "Parameterized All-Digital PLL Architecture and its Compiler to Support Easy Process Migration," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 22, pp. 621–630, 2014.
[6] P. Kumar, V. Kumar, and R. Pratap, "Design and implementation of phase detector on FPGA," in 6th International Conference on Computer Applications In Electrical Engineering-Recent Advances (CERA), 2017, pp. 108–110.
[7] M. Huang and C. Hung, "Full-custom all-digital phase locked loop for clock generation," in VLSI Design, Automation and Test(VLSI-DAT), 2015, pp. 1–4.
[8] D. Kim and S. Cho, "A Hybrid PLL Using Low-Power GRO-TDC for Reduced In-Band Phase Noise," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 66, pp. 232–236, 2019.
[9] L. Xiu, W. Li, J. Meiners, and R. Padakanti, "A novel all-digital PLL with software adaptive filter," IEEE Journal of Solid-State Circuits, vol. 39, no. 3, pp. 476–483, 2004.
[10] S. S. Yunhua, L. Shimin, J. Yue, and Lijiu, "Implementation of a 6.5 MHz 34-B NCO," Proceedings of 4th International Conference on Solid-State and IC Technology, pp. 205–207, 1995.
[11] S. Kadam, D. Sasidaran, A. Awawdeh, L. Johnson, and M. Soderstrand, "Comparison of various numerically controlled oscillators," The 2002 45th Midwest Symposium on Circuits and Systems, 2002.
[12] R. B. Staszewski, D. Leipold, and P. T. Balsara, "Direct frequency modulation of an ADPLL for bluetooth/GSM with injection pulling elimination," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 52, pp. 339–343, 2005.
[13] B. Razavi, C. Analog, and I. Design.
[14] P. P. Sotiriadis, "All Digital Frequency Synthesis based on Pulse Direct Digital Synthesizer with spurs free output and improved noise floor," in IEEE International Frequency Control Symposium (FCS), 2014, pp. 1–5.

AUTHOR BIOGRAPHY

Mohd Ziauddin Jahangir Mohd Ziauddin Jahangir The author has completed his bachelor of engineering degree in Electronics and Communications Engineering from Osmania University, Hyderabad, India in the year 2009. He has completed his Master of Engineering with the specialization of Embedded Systems and VLSI Design from Osmania University, Hyderabad, India in the year 2013. He is current pursuing PhD from Osmania University College of Engineering, Hyderabad, India.

He is at present working as an Assistant Professor at the Department of Electronics and Communications Engineering, Chaitanya Bharathi Institute of Technology, Hyderabad, India. He has publications in many IEEE Conferences in India like IEEE- PrimeAsia 2013, IEEE-PrimeAsia 2015, IEEE-INDICON 2015, etc.. His areas of interest include Analog IC Design, Digital Systems Designs and Embedded Systems. He is currently working on the design of various Phase Locked Loops. Mr. Mohd Ziauddin Jahangir has received the Best Paper award at IEEE-INDICON Conference held at Jamia Milia University, New Delhi in December 2015.

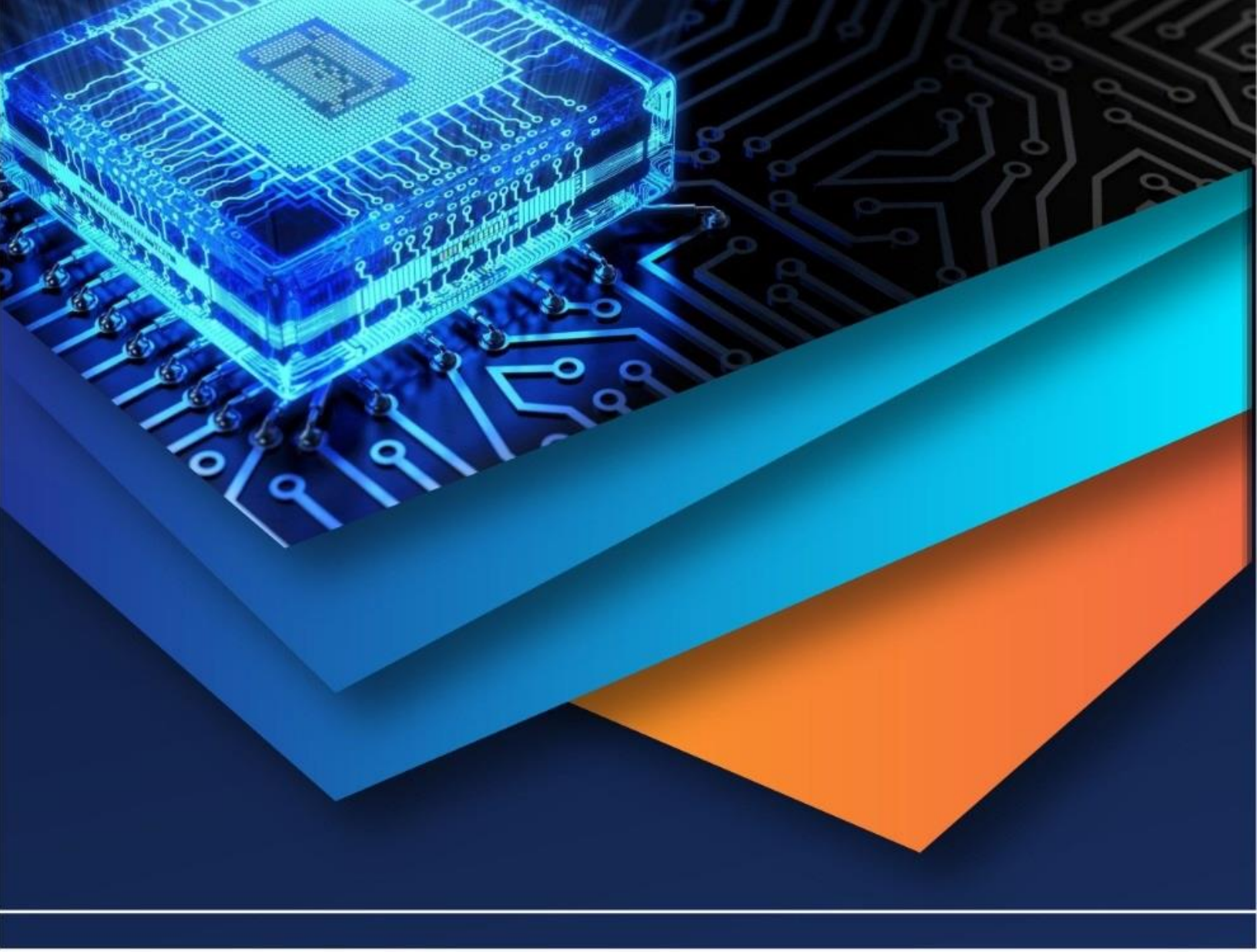
Chandra Sekhar Paidimarry He has completed his Bachelor of Engineering degree in Electronics and Communications Engineering in the year 1991. He has completed his Master of Engineering in the year 1999. He has completed his PhD from Osmania University College of Engineering, Hyderabad, India in 2009. He has completed his Post Doc from Shizuoka University, Japan

He is at presently working as a Professor at the Department of Electronics and Communications Engineering, University College of Engineering, Hyderabad, India. Earlier he has worked as R&D Engineer at DigiSun Electronics, Hyderabad. He has publications in many IEEE Conferences in India.

Dr. P. Chandra Sekhar is currently the Chair of CAS/EDS Joint Chapter, IEEE- Hyderabad Section. Seven students have completed heir PhD under his guidance.

Mohammed Sikander The author has completed his bachelor of engineering degree in Electronics and Communications Engineering from JNTUH, Hyderabad, India in the year 2007. He has completed his master of technology with the specialization of Electronic Design and Technology from NIT-Calicut, India in the year 2011.

He is at present working as an Assistant Professor at the Department of Electronics and Communications Engineering, Chaitanya Bharathi Institute of Technology, Hyderabad, India.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY