UNIVERSAL MODULATOR USING CORDIC ALGORITHM FOR COMMUNICATION APPLICATION

S. Arvind Kumar¹, V. Priyanka Brahmaiah², L. Dharma Teja²
¹ECE Department, MLR Institute of Technology & Management, Hyderabad, A.P., India
²ECE Department, VNR Vignana Jyothi Institute of Engg. & Tech., Hyderabad, A.P., India

ABSTRACT
The modern communication systems and software radio based applications demands fully digital receivers, consisting of only an antenna and a fully programmable circuit with digital modulators and demodulators. A basic communication system’s transmitter modulates the amplitude, phase or frequency proportional to the signal being transmitted. The revolutions which are coming up in mobile phones are mainly because of usage of SDR (software defined radio) principles. In all communication algorithms we need the carrier. The traditional techniques of analog implementation of carrier and reading its digital value in processor have several limitations. Instead it is possible to generate the carrier on the microcontroller directly by using few algorithms. This paper involves implementation of CORDIC (Co-ordinate Rotation Digital Computer) algorithm on ARM controller for generating high carrier frequency. An efficient solution (that doesn’t require large tables/memory) for realizing a carrier source is CORDIC algorithm. Finally the generation of SIN and COS output values will are tested for a given input angle (θ) value. The universal modulator will be designed around the CORDIC algorithms which can generate all most all digital modulation schemes such as ASK, PSK, FSK, QPSK. The GNU tool chain will be used to build the application.

KEYWORDS: CORDIC, ASK, PSK, FSK, ARM7.

I. INTRODUCTION TO CORDIC
Carrier signal plays a key role in all communication algorithms. The traditional techniques of analog implementation of carrier and reading its digital value in processor have several limitations. Instead, it is possible to generate the carrier on the microcontroller directly by using an algorithm called CORDIC [2]. CORDIC means “Coordinate Rotation Digital Computer” algorithm which avoids the use of function generator which generates a carrier or different wave forms like sine wave, triangular wave, square wave etc. By using CORDIC algorithm, the digital modulation techniques can also be implemented like ASK, FSK, PSK, and QPSK. A carrier generator is generated by using CORDIC algorithm for digital modulation techniques by avoiding hardware complexity at very less power. A constraint low power can be obtained by using ARM7 processor. So by dumping the code into ARM7 processor, generation of a carrier waveform for digital modulation techniques takes place depending on the input given by the user on the control word which is written in the code.
Compared to other approaches, CORDIC is a clear winner when a hardware multiplier is unavailable, e.g. in a microcontroller, or when there is a need to save the gates required to implement one, e.g. in an FPGA. On the other hand, when a hardware multiplier is available, e.g. in a DSP microprocessor, table-lookup methods and good old-fashioned power series are generally faster than CORDIC [1-4]. CORDIC is an iterative algorithm for calculating trigonometric functions including sine, cosine, magnitude and phase. It is particularly suited to hardware implementations because it does not require any multiplies. It calculates the trigonometric functions of sine, cosine, magnitude and phase (arctangent) to any desired precision. CORDIC algorithm will work by revolving around the idea of
"rotating" the phase of a complex number, by multiplying it by a succession of constant values. However, the "multiplies" can all be powers of 2, so in binary arithmetic they can be done using just shifts and adds; no actual "multiplier" is needed.

The CORDIC algorithm has been introduced and the key ideas are described in the preceding section. The techniques involved in this algorithm are specified by steps. The next section includes details of the ARM7 processor with Microcontroller. Finally the development and implementation of code has been done.

1.1. CORDIC algorithm key ideas

\[ z = \alpha (1) + \alpha (2) + \ldots + \alpha (m) \quad \text{--- (1)} \]

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad \text{--- (2)}
\]

\[
x' = x \cos \varphi - y \sin \varphi \quad \text{--- (3)}
\]

\[
y' = y \cos \varphi + x \sin \varphi \quad \text{--- (3)}
\]

Above equations can be rewritten as

\[
x' = \cos \varphi \left[ X - Y \tan \varphi \right] \quad \text{--- (4)}
\]

\[
y' = \cos \varphi \left[ Y + x \tan \varphi \right] \quad \text{--- (5)}
\]

Where \( \tan \varphi = 2^{-i} \)

\[ X_{i+1} = k_i \left[ X_i - Y_i \cdot d_i \cdot 2^{-i} \right] \quad \text{--- (6)}
\]

\[ Y_{i+1} = k_i \left[ Y_i + X_i \cdot d_i \cdot 2^{-i} \right] \quad \text{--- (6)}
\]

Where \( k_i = \cos \left( \tan^{-1} 2^{-i} \right) = 1 / \sqrt{1 + 2^{-2i}} \) & \( d_i = \pm 1 \)

Therefore

\[ X_{i+1} = x_i - y_i \cdot d_i \cdot 2^i \quad \text{--- (4)}
\]

\[ Y_{i+1} = Y_i + x_i \cdot d_i \cdot 2^i \quad \text{--- (5)}
\]

\[ Z_{i+1} = Z_i - d_i \cdot \tan^{-1} \left( 2^i \right) \quad \text{--- (6)}
\]

Where \( d_i = -1 \) if \( z_i < 0, +1 \) otherwise \( d_i \) means direction, \( k_i \) = scaling factor

Removing the scale constant from the iterative equations yields a shift add algorithm for vector rotation. The product of the Ki’s can be applied elsewhere in the system or treated as a part of the system processing gain. The product approaches 0.6073 as the number of iteration goes to infinity.

\[ A_n = \pi n \sqrt{1 + 2^{-2i}} \quad \text{--- (7)} \]

The angle of a composite rotation is uniquely defined by the sequence of the directions of the elementary rotations. That sequence can be represented by a decision vector. The set of all possible
decision vectors is an angular measurement system based on binary arc tangents. Conversions between the angular systems and any other can be accomplished using a lookup. A better conversion method uses an additional adder-subtractor that accumulates the elementary rotation angles at each iteration. The elementary angles can be expressed in any convenient angular unit. Those angular values are supplied by a small lookup table or are hard wired depending on the application.

The angle accumulator adds a third difference equation to the CORDIC algorithm:

\[ Z_{i+1} = Z_i - d_i \tan(2^{-i}) \]

Obviously, in cases where the angle is useful in the arc tangent base, this extra element is not needed.

### 1.2. CORDIC Technique

Given a complex value \( C = I + jQ \)

We will create a rotated value \( C' = I' + jQ' \)

By multiplying by a rotation value: \( R = I_r + jQ_r \)

How multiplier less:

1. Recall that when you multiply a pair of complex numbers, their phases (angles) add and their magnitudes multiply. Similarly, when you multiply one complex number by the conjugate of the other, the phase of the conjugate done is subtracted (though the magnitudes still multiply).

2. Therefore:

   \[
   \text{To add R’s phase to } C \quad C' = C \cdot R \quad I_{c'} = I_c \cdot I_r - Q_c \cdot Q_r \\
   \text{To subtract phase from } C \quad C' = C \cdot R^* \quad I_{c'} = I_c \cdot I_r + Q_c \cdot Q_r
   \]

3. To rotate by +90 degrees, multiply by \( R = 0 + j1 \). Similarly, to rotate by -90 degrees, multiply by \( R = 0 - j1 \). If you go through the Algebra above, then effect is:

   \[
   \text{To add 90 degrees Multiply by } R = 0 + j1 \quad I_{c'} = -Q_c \\
   \text{To subtract 90 degrees Multiply by } R = 0 - j1 \quad I_{c'} = Q_c
   \]

   \[
   \text{(negate } Q, \text{ then swap)} \\
   \text{(negate } I, \text{ then swap)}
   \]

4. To rotate by phases of less than 90 degrees, we will be multiplying by numbers of the form "R = 1 +/− jK". \( K \) will be decreasing powers of two, starting with \( 2^0 = 1.0 \). Therefore, \( K = 1.0, 0.5, 0.25, \text{ etc.} \) (We use the symbol "L" to designate the power of two itself: 0, -1,-2, etc.) Since the phase of a complex number "I + jQ" is at an \( (Q/I) \), the phase of "I + jK" is at an(K). Likewise, the phase of "I - jK" = at an(-K) = -at an(K). To add phases we use "R = 1 + jK"; to subtract phases we use "R = 1 - jK". Since the real part of this, \( I_r \), is equal to 1, we can simplify our table of equations to add and subtract phases for the special case of CORDIC multiplications to:

   \[
   \text{To add a phase Multiply by } R = 1 + jK \quad I_{c'} = I_c - K \cdot Q_c = I_c(2^L) \cdot Q_c \\
   \text{To add a phase Multiply by } R = 1 + jK \quad I_{c'} = I_c + K \cdot Q_c = I_c + (2^L) \cdot Q_c
   \]
5. Each rotation has a magnitude greater than 1.0. That isn't desirable, but it's the price we pay for using rotations of the form "1 + jK". The "CORDIC Gain" column in the table is simply a "cumulative magnitude" calculated by multiplying the current magnitude by the previous magnitude. Notice that it converges to about 1.647; however, the actual CORDIC Gain depends on how many iterations we do. (It doesn't depend on whether we add or subtract phases, because the magnitudes multiply either way).[6,7]

1.3. CORDIC Basic Issues

- Since we're using powers of two for the K values, we can just shift and add our binary numbers. That's why the CORDIC algorithm doesn't need any multiplies.
- The sum of the phases in the table up to L = 3 exceeds 92 degrees, so we can rotate a complex number by +/-90 degrees as long as we do four or more "R = 1 +/- jK" rotations. Put that together with the ability to rotate +/-90 degrees using "R = 0 +/- j1", and you can rotate a full +/-180 degrees.
- You can see that starting with a phase of 45 degrees, the phase of each successive R multiplier is a little over half of the phase of the previous R. That's the key to understanding CORDIC: we will be doing a "binary search" on phase by adding or subtracting successively smaller phases to reach some "target" phase.
- Each rotation has a magnitude greater than 1.0. That isn't desirable, but it's the price we pay for using rotations of the form "1 + jK". The "CORDIC Gain" column in the table is simply a "cumulative magnitude" calculated by multiplying the current magnitude by the previous magnitude. Notice that it converges to about 1.647; however, the actual CORDIC Gain depends on how many iterations we do. (It doesn't depend on whether we add or subtract phases, because the magnitudes multiply either way.)

<table>
<thead>
<tr>
<th>L</th>
<th>K = 2^L</th>
<th>R = 1 + jK</th>
<th>Phase of R in degrees</th>
<th>Magnitude of R</th>
<th>CORDIC Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0</td>
<td>1 + j1.0</td>
<td>45.00000</td>
<td>1.41421356</td>
<td>1.414213562</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>1 + j0.5</td>
<td>26.56505</td>
<td>1.11803399</td>
<td>1.581138830</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
<td>1 + j0.25</td>
<td>14.03624</td>
<td>1.03077641</td>
<td>1.629800601</td>
</tr>
<tr>
<td>3</td>
<td>0.125</td>
<td>1 + j0.125</td>
<td>7.12502</td>
<td>1.00778222</td>
<td>1.642484066</td>
</tr>
<tr>
<td>4</td>
<td>0.0625</td>
<td>1 + j0.0625</td>
<td>3.57633</td>
<td>1.00195122</td>
<td>1.645688916</td>
</tr>
<tr>
<td>5</td>
<td>0.03125</td>
<td>1 + j0.03125</td>
<td>1.78991</td>
<td>1.00048816</td>
<td>1.646492279</td>
</tr>
<tr>
<td>6</td>
<td>0.015625</td>
<td>1 + j0.015625</td>
<td>0.89517</td>
<td>1.00012206</td>
<td>1.646693254</td>
</tr>
<tr>
<td>7</td>
<td>0.007813</td>
<td>1 + j0.007813</td>
<td>0.44761</td>
<td>1.00003052</td>
<td>1.646743507</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Modes of operation:
Basic mode: I and Q carrier signals for a chosen frequency value.
Modulator mode: producing ASK, FSK and PSK signals.

Table 2: 11-bit Control word format

<table>
<thead>
<tr>
<th>M</th>
<th>M</th>
<th>F</th>
<th>F</th>
<th>F</th>
<th>F</th>
<th>F</th>
<th>F</th>
<th>K</th>
<th>K</th>
</tr>
</thead>
</table>

MM --- Mode of operation
00 – Basic mode
00 – Amplitude shift keying
01 – Period shift keying
01 – QPSK
10 – OQPSK
11 – 8psk.

FFFFFF-7-bit Freq word KK (in Binary modulation Mode)
10– Freq shift keying KK (in M-ary modulation mode)

II. ARM-BASED MICROCONTROLLER LPC2148
2.1 Features of LPC2148 Microcontroller

- 16/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.
- 8 to 40 kB of on-chip static RAM and 32 to 512 kB of on-chip flash program memory.
- 128 bit wide interface/accelerator enables high speed 60 MHz operation.
- In-System/In-Application Programming (ISP/IAP) via on-chip boot-loader software. Single flash sector or full chip erase in 400 ms and programming of 256 bytes in 1 ms.
- Embedded ICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip Real Monitor software and high speed tracing of instruction execution.
- USB 2.0 Full Speed compliant Device Controller with 2 kB of endpoint RAMs.

The LPC2148 microcontrollers are based on a 32 bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combines the microcontroller with embedded high speed flash memory of 512 kB. A 128-bit wide memory interface and unique accelerator architecture enable 32-bit code execution at the maximum clock rate. For critical code size applications, the alternative 16-bit Thumb mode reduces the code by more than 30 % with minimal performance penalty.

Due to their tiny size and low power consumption, LPC2148 microcontrollers are ideal for the applications where miniaturization is a key requirement, such as access control and point-of-sale. A blend of serial communications interfaces ranging from a USB 2.0 Full Speed device, multiple UARTS, SPI, SSP to I2Cs and on-chip SRAM of 8 kB up to 40 kB, make these devices very well suited for communication gateways and protocol converters, soft modems, voice recognition and low end imaging, providing both large buffer size and high processing power. Various 32-bit timers, single or dual 10-bit ADC(s), 10-bit DAC, PWM channels and 45 fast GPIO lines with up to nine edge or level sensitive external interrupt pins make these microcontrollers particularly suitable for industrial control and medical systems.
Figure 2: Block Diagram of LPC2148 Microcontroller [10]

III. CODE DEVELOPMENT AND IMPLEMENTATION

The following steps are followed:

- C Language will be used for developing the application.
- GNU ARM Cross Compiler (armgcc) Tool Will be used for Software Compilation and Generating the Hex file.
- LPC2000 Flash utility Software will be used for downloading the Hex file into microcontroller.

The following user defined options are required to implement the cordic algorithm

- Displaying the options for selecting the command on HyperTerminal
- Processing the given command
- Computing the samples depending on the process command

3.1 User Options for CORDIC algorithm.

- Code has written in C language for CORDIC algorithm for generating carrier wave form and digital modulation techniques like ASK, FSK, PSK, and QPSK. [8,9]
- For implementing the CORDIC algorithm the following control word format is used

K M M F FFFF A

K = 0 normal mode (wave form generation)
K = 1 modulator mode (ASK, FSK, PSK, QPSK)
MM = 00 Basic mode; Sin wave
MM = 01 Square wave mode
MM = 10 triangular wave mode.
MM = 11 arbitrary wave form mode (Gaussian noise is implemented)
In shift keying mode
MM =00 ASK
MM =01 FSK
MM=10 PSK
MM =11 QPSK
FFFFF = the decimal number of frequency in Hz (must be 20000, 10000, 05000, 02500)
A = Amplitude factor must be 1 or 2
The angle is represented with 16 bit fractional part and 16 bit integer part
The value of sin will be represented by 24 bit integer and 8 bit fractional value
(Effectively 2 bit integer part and 8 bit fractional part going to 10 bit DAC)

Output waveforms of cordic based universal modulator
For generating output wave forms of carrier wave and all digital modulation techniques the
User has to enter the command on HyperTerminal which is connected to ARM controller
With the help of UART to standard transmit and receive data lines.[8]
After compiling the hex file of the code is generated. This hex file is dumped in to the ARM controller.

Figure 3: Hex file dumped in to ARM controller

After displaying the command prompt user have to enter the command for KMMFFFFFA
For generating the respective wave form on CRO, which is connected to output pin no 27 of ARM controller.
If command format is KMMFFFFFA = 000100002
Then it generates sine wave as output. The command prompt for generating sine wave is shown in fig below

Figure 4 : HyperTerminal prompt window for the command format sine wave

The dumped ARM7 processor is connected to CRO for displaying the carrier and digital modulated signals.
Figure 5: Output waveforms on CRO

IV. APPLICATIONS

- Universal modulator is highly preferred option in instrumentation for signal generation and frequency sweep etc.
- Universal modulator can be efficiently used in several digital modulation schemes such as FSK, PSK, ASK, QPSK, OPQPSK, PI/QPSK, QAM and 8-bitPSK.
- Universal modulator has been used in universal HAM radio/generator. Universal modulator is capable of well controlled, rapid changes in frequency.
- Computing trigonometric functions and converting Cartesian coordinates to polar coordinates and vice-versa.

V. ADVANTAGE AND DISADVANTAGES

- High frequency resolution and accuracy.
- CORDIC is a clear winner when a hardware multiplier is unavailable.
- CORDIC is widely used due to its simplicity and its property of relatively fast convergence.
- Fast switching between frequencies over a large bandwidth.

VI. CONCLUSION & FUTURE WORK

CORDIC algorithms are an efficient method it is applicable to the entire range of angles are presented in this paper. Convergence proofs for the new algorithms are also presented. In some cases, CORDIC evaluates rotational functions more efficiently than MAC units. CORDIC saves more hardware cost.
By the regularity, the CORDIC based architecture is very suitable for implementation with pipelined VLSI array processors. The utility of the CORDIC based architecture lies in its generality and flexibility. By using this method many communication applications can be developed. These applications can be defined user friendly and can be introduced with low cost.

REFERENCES

[5]. A VLSI implementation of Logarithmic and Exponential Functions Using a Novel Parboil Synthesis Methodology Compared to the CORDIC Algorithms IEEE paper 2011.

AUTHORS

S.Arvind Kumar obtained his B.Tech. Degree from JNT University, Hyderabad in 2007, and M.Tech in VLSI & Embedded Systems from JNT University, Hyderabad in 2009. He has started his career as Assistant professor and served for 2 years in DRK Institute of Engg. & Technology. Assistant professor in MLR Institute of Technology & Management, Dundigal, Hyderabad from June 2010 to till date. His areas of research interest Embedded Systems, Signal and Image Processing, Human Computer Interface.

V. Priyanka Brahmaiah obtained her B.Tech. Degree from JNT University, Hyderabad in 2007, and M.Tech in VLSI System design from JNT University, Hyderabad in 2010. She has started her career as Assistant professor and served for 2 years in MLR Institute of Technology & Management, Dundigal, Hyderabad from June 2010 to June 2012. Assistant Professor in the department of ECE in Gokaraju Rangaraju Institute of Engineering & Technology from July 2012 to November 2012, Assistant Professor in the department of ECE in VNR Vignana Jyothi Institute of Engineering Technology from December 2012 to till date. She is a Life member of ISTE and IETE. She presented three research paper in International Journal. Her areas of research interest are Bio-Medical, Signal and Image Processing, Human Computer Interface.

L. Dharma Teja obtained her B.Tech. Degree from JNT University, Hyderabad in 2006, and M.Tech in VLSI System design from JNT University, Hyderabad in 2010. Pursuing PhD in the field of VLSI in JNT University, Hyderabad, registered in 2011. She has started her career as Assistant professor and served for 2 years in DVR & Dr. HS MIC College of Technology, Kanchikacherla, Vijayawada from June 2006 to September 2008, Assistant Professor in the department of ECE in CMR Institute of Technology from May 2010 to May 2011, Assistant Professor in the department of ECE in VNR Vignana Jyothi Institute of Engineering Technology from June 2011 to till date. She is a life member of ISTE and IETE. Her areas of research interest are Fault Tolerance and Digital System Design in the field of VLSI design.