Efficient Frequency Domain Zadoff-Chu Generator with Application to LTE and LTE-A Systems

Felipe A. P de Figueiredo, Fabiano S. Mathilde, Fabbryccio A. C. M. Cardoso,
Rafael M. Vilela and João Paulo Miranda
DRC - Convergent Networks Department
CPqD - Research and Development Center
Campinas, SP - Brazil
{felipep, fabianom, fcardoso, rvilela, jmiranda}@cpqd.com.br

***Abstract*—This paper presents a configurable and optimized hardware architecture for computing Zadoff-Chu (ZC) complex sequences in the Frequency Domain (FD). It is a hardware-efficient and accurate architecture for computing ZC sequences in real-time. The architecture is mainly based on the CORDIC algorithm for computing complex exponentials using only shift and add operations. Due to transformations applied to the Zadoff-Chu equation it is possible to eliminate the use of multipliers with non-constant terms. This hardware architecture is employed by the Physical Random Access Channel (PRACH) in LTE and LTE-A systems during the reception and detection of random access preambles. Its main advantage is that it eliminates the need for storing a large number of long complex ZC sequences.**

I. Introduction

In Long Term Evolution (LTE) and Long Term Evolution-Advanced (LTE-A) communications systems, the PRACH is a common uplink channel used by mobile users within a cell to establish initial access to a base station, along with uplink synchronization to compensate for round-trip delays to the base station. The mechanism is based on the mobile User Equipment (UE) transmitting a randomly-chosen preamble to a base station (e.g., eNodeB) over a dedicated time-frequency resource on the PRACH. A pool of known preambles is allocated to a base station within a cell.

A PRACH receiver, or searcher, in the base station attempts to detect a transmitted preamble by first extracting the PRACH signal from a received wide-band Orthogonal Frequency-Division Multiplexing (OFDM) signal, then performing matched filtering across the pool of preambles allocated to the base station. The matched filtering is performed as a cross-correlation of the extracted PRACH signal with each of the known preambles dedicated to the base station. The cross correlations provide a final metric that is compared to a threshold, from which the presence of a preamble can be detected and the mobile user's timing offset relative to the base station can be estimated [1].

An important requirement is that the system must be capable of supporting a large number of users per cell with quasi-instantaneous access to the radio resources, and sustain a good detection probability, while maintaining a low false alarm rate.

Hence, preambles must be constructed using sequences that possess good periodic correlation properties. One candidate sequence is the well-known Zadoff-Chu (ZC) sequence [2], which belongs to a class of sequences called Constant Amplitude Zero Auto-Correlation (CAZAC) sequences. These sequences are currently employed in the emerging LTE and LTE-A Physical layer (PHY) standards to construct PRACH preambles [3]. ZC sequences are complex exponential codes whose discrete auto-correlations are zero for all non-zero lags, with no restrictions on code lengths. A disadvantage, however, is that ZC sequences are difficult to generate in real-time due to the nature of their construction.

Known implementations typically resort to pre-computing these sequences off-line, quantizing them to the required precision, and storing them in RAM or ROM memory. For example, in LTE systems with 3-sector cells where a pool of 64 preambles of length 839 are allocated to each sector, assuming Format-0 preambles, a memory storage of 4.9 Mbits is needed to store these complex-valued sequences, assuming 16-bit quantization. Accordingly, in one aspect it would be desirable to reduce the need for storing ZC sequences, such as through efficient generation of ZC sequences in real-time.

This paper presents an algorithm that is able to compute ZC sequences on the fly with high accuracy using a low-complexity hardware architecture which eliminates the need for large memory storage, saving on valuable chip area and reducing power consumption.

The remainder of the paper is organized as follows. Section II introduces ZC sequences. In Section III, an efficient algorithm for computing ZC sequences is proposed and a corresponding hardware architecture is presented. Section IV gives important details on the implementation of the proposed algorithm. In Section V simulation and implementation results are presented. Section VI provides some concluding remarks.

II. Fundamentals of Zadoff-Chu Sequences

The ZC sequences employed in the PRACH channel have the following form [3]:

$$z_u(n) = \exp\left(\frac{-j\pi un(n+1)}{N_{\text{ZC}}}\right), \quad 0 \leq n \leq N_{\text{ZC}} - 1 \quad (1)$$

where N_{ZC} is the length of the ZC sequence and u is a positive integer called root ZC index. Random access preambles with zero correlation zones are defined from the u-th root ZC sequence.

A ZC sequence is a complex-valued mathematical sequence which possess constant amplitude. These sequences exhibit the useful property that cyclically shifted versions of themselves are orthogonal to one another [2], [4]. Hence ZC sequences are typically termed CAZAC sequences. These properties are essential in a variety of engineering applications

such as establishing timing synchronization between a mobile terminal and a base station, performing channel estimation and reducing Peak-to-Average Power Ratio (PAPR) [1].

In [5] it was noticed that there exists a duality between ZC sequences in Time Domain and ZC sequences in FD. Let Z_u represent the N-point Discrete Fourier transform (DFT) of (1):

$$Z_u[k] = \sum_{n=0}^{N-1} z_u[n] \exp\left(\frac{-j2\pi nk}{N}\right), k = 0, 1, \cdots, N-1 \quad (2)$$

It should be noticed that z_u and Z_u are periodic sequences with period N. Then, as showed in [5], z_u and Z_u are related by the following equation:

$$Z_u[k] \;=\; Z_u[0] \,.\, z_u^*[u'k] \;=\; Z_u[0] \,.\, \exp\left(\frac{j\pi uu'k(u'k+1)}{N_{\text{ZC}}}\right), \quad k \;=\; 0,\, 1,\, \cdots, N-1 \quad (3)$$

where u' is the multiplicative inverse of u modulo N (*i.e.*, $u' \,.\, u$ = 1 mod N). Hence, the DFT of a ZC sequence is also a ZC sequence.

III. Efficient Algorithm for Computing ZC Sequences in Frequency Domain

In this section, we present an efficient algorithm for computing ZC sequences in FD using the CORDIC algorithm [6].

A. The CORDIC Algorithm

The CORDIC algorithm is a known hardware-efficient iterative algorithm for evaluating trigonometric and other transcendental functions using only shift and add operations; *i.e.*, determining trigonometric functions without multiplication operations [6], [7]. The algorithm is derived from the general Givens rotation transform, and can perform the rotation of a two-dimensional vector (x,y) in linear, circular and hyperbolic coordinates. In the present application, it is assumed that the CORDIC algorithm is in rotation mode and the rotation angle is restricted to be within the range $|\theta| \leq \pi/2$. It is noted that other rotation angles outside this range can be easily converted to be within this range. Accordingly, a circular CORDIC rotation with accuracy of B fractional bits is expressed by the group of equations as follows [6]:

$$x_i \;= x_{i-1} - y_{i-1} \,.\, d_{i-1} \,.\, 2^{-(i-1)} \quad (4)$$

$$y_i \;= y_{i-1} + x_{i-1} \,.\, d_{i-1} \,.\, 2^{-(i-1)} \quad (5)$$

$$z_i \;= z_{i-1} - d_{i-1} \,.\, \tan^{-1}(2^{-(i-1)}) \quad (6)$$

for $i = 1, 2, \cdots, B$, where x_i and y_i are the vector coordinates at the ith iteration, and z_i is the residual angle relative to the x-axis at the ith iteration. Addition or subtraction of the ith rotation angle $\tan^{-1}(2^{-(i-1)})$ is selected based on a decision variable d_i, where d_i=-1 if $z_i < 0$, and +1 otherwise. If the initial inputs are set as $x_0 = K$, $y_0 = 0$, $Z_0 = \theta$, where $|\theta| \leq \pi/2$ and $K = \prod_{i=0}^{B-1} \frac{1}{\sqrt{1+2^{-2i}}}$, then the final outputs after B iterations will converge to the cosine and sine functions $x_{B+1} = \cos(\theta)$ and $y_{B+1} = \sin(\theta)$. The scaling constant K is fixed and can be pre-computed off-line for a given precision B. The arc-tangent values for 2^{-i}, for $i = 1, 2, \cdots, B$, may be stored in a look-up table or equivalent storage as $K = \lim_{B\to\infty} K(B) = 0.607252935008881256 1694$, which allows further reduction of the algorithm's complexity.

B. Efficient ZC-Algorithm

From (3), it can be noticed that based on Euler's formula, it is possible to represent ZC sequences as trigonometric functions:

$$Z_u[k] \;=\; Z_u[0].\cos\left(\frac{2\pi uu'}{N_{\text{ZC}}}\left(\frac{k(u'k+1)}{2}\right)\right) - j\sin\left(\frac{2\pi uu'}{N_{\text{ZC}}}\left(\frac{k(u'k+1)}{2}\right)\right) \quad (7)$$

Therefore, the CORDIC algorithm can be efficiently employed to evaluate the complex exponential portion in (3) using the sine and cosine functions. However, the argument (phase), $\theta[k] = \left(\frac{j\pi uu'k(u'k+1)}{N_{\text{ZC}}}\right)$, $|\theta[k]| \leq \pi/2$, of the sine and cosine functions needs to be computed first. In order to avoid the use of multipliers with non-constant terms when evaluating the arguments for a given k, the arguments are computed recursively as the elements of the ZC sequence are traversed [8].

To evaluate $\theta[k]$ recursively for $k = 0, 1, \cdots, N_{\text{ZC}} - 1$, a portion of the $\theta[k]$ equation can be defined as:

$$\alpha[k] \;=\; \left(uu'\frac{k(u'k+1)}{N_{\text{ZC}}}\right) \;\text{mod} N_{\text{ZC}} \quad (8)$$

The equation above can be expressed in terms of its previous values, $\alpha[k-1]$ according to the following relationships:

$$\alpha[k] \;=\; \begin{cases} 0, & k = 0 \\ (\alpha[k-1] \;+\; \beta[k]) \;\text{mod}\; N_{\text{ZC}}, & k \geq 1 \end{cases} \quad (9)$$

where

$$\beta[k] \;=\; \left(u(u')^2k \;-\; u\frac{u'(u'-1)}{2}\right) \;\text{mod}\; N_{\text{ZC}} \quad (10)$$

In turn, $\beta[k]$ can be recursively expressed as

$$\beta[k] \;=\; \begin{cases} -u\frac{u'(u'-1)}{2}, & k = 0 \\ (\beta[k-1] + u(u')^2) \;\text{mod}\; N_{\text{ZC}}, & k \geq 1 \end{cases} \quad (11)$$

It is important to realize that the terms $u(u')^2$ and $u\frac{u'(u'-1)}{2}$ are constants for a given u', and therefore can be computed off-line. Hence,

$$\theta[k] \;=\; \frac{2\pi}{N_{\text{ZC}}}\alpha[k] \quad (12)$$

Finally, the value $\theta[k]$ of (12) may then be used in the CORDIC equations (4), (5), (6) to efficiently compute (7). In an aspect, before the value $\theta[k]$ is fed to (4), (5) and (6), this value may be translated into the range of $[-\pi/2, \pi/2]$, or equivalently, $\alpha[k]$ translated into the range $[-N/4, N/4]$. In the latter case, the sign of sine and cosine functions would require adjustment, accordingly. The result of the CORDIC equations would then need to be multiplied by the complex scaling constant $Z_u[0]$ to derive the desired FD ZC sequence $Z_u[k]$, however, we have found a way to eliminate the complex multiplication by incorporating the complex scaling constant to the CORDIC initialization values, x_init and y_init. As will be shown in the next section, this improvement decreases the

complexity of the algorithm in terms of resource utilization.

The optimized FD ZC computation algorithm is presented in Algorithm 1. It computes the elements of a ZC sequence recursively in FD. Note that the modulo operations in the algorithm can be easily implemented using only one subtraction operation (with N_{ZC}) for every value of k. The procedure input values B, m, $KZ_u[0]$, β_0 and $angles$ are stored in Look-Up Tables (LUT), where B is the desired number of iterations, m is FD root sequence, u, $KZ_u[0]$ is the multiplication of constants K and $Z_u[0]$, β_0 is given by $u\frac{u'(u'-1)}{2}$ in (11) and $angles$ is the $\tan^{-1}$ (arc-tangent) of 2^{-i} in (6). The input B is the number of iterations the CORDIC must execute and u is the ZC physical root index to be generated.

The algorithm presented here is an improvement made to the one presented in [8]. Instead of following the approach adopted in there which presents an architecture capable of generating ZC sequences both in time and FD, our work focus only on the FD generation. The reason for this is that the PRACH receiver at eNodeB side only needs ZC sequences in the FD to receive and detect random accesses.

By focusing only on a FD architecture for the ZC generation we were able to reduce the number of operations needed and thus simplifying the algorithm proposed in [8]. For example, the algorithm in [8] uses a complex multiplier at the end of the CORDIC to multiply its output by the constant $Z_u[0]$, however, it is possible to incorporate this complex multiplication into the CORDIC's initialization which in turn makes the complex multiplier circuit useless and therefore decreasing the complexity of the algorithm, i.e., the number of used resources. This and all the other simplifications are presented in Algorithm 1.

The contributions provided by the present work are namely a less resource-intensive FD ZC sequence generator architecture, important details and results of implementation of the proposed architecture on an FPGA.

Algorithm 1 Efficient Frequency Domain ZC Algorithm

procedure FDZCGEN($B, u, m, KZ_u[0], \beta_0, angles$)
  for $n \leftarrow 0$ *to* $N_{ZC}-1$ **do**
    $s \leftarrow 1$
    if $n == 0$ **then**
      $alpha(n+1) \leftarrow 0$
      $beta(n+1) \leftarrow \beta_0(u)$
    else
      $v \leftarrow (beta(n) + m(u))$
      if $v > N_{ZC}-1$ **then**
        $v \leftarrow v - N_{ZC}$
      end if
      $beta(n+1) \leftarrow v$
      $v \leftarrow (alpha(n) + beta(n+1))$
      if $v < -\text{floor}(N_{ZC}/4)$ **then**
        $v \leftarrow v + N_{ZC}$
      else if $v > \text{floor}(3 * N_{ZC}/4)$ **then**
        $v \leftarrow v - N_{ZC}$
      end if
      $alpha(n+1) \leftarrow v$
    end if
    $r \leftarrow alpha(n+1)$
    if $alpha(n+1) > \text{floor}(N_{ZC}/4)$ **then**
      $r \leftarrow r - N_{ZC}/2$
      $s \leftarrow -1$
    end if
    $theta(n+1) \leftarrow 2 * (pi/N_{ZC}) * r$
    $x(1) \leftarrow real(KZ_u[0](u))$
    $y(1) \leftarrow imag(KZ_u[0](u))$
    $z(1) \leftarrow theta(n+1)$
    for $i \leftarrow 0$ *to* $B-1$ **do**
      $sigma \leftarrow sgn(z(i+1))$
      $factor \leftarrow sigma * (2^{-i})$
      $x(i+2) \leftarrow x(i+1) - factor * y(i+1)$
      $y(i+2) \leftarrow y(i+1) + factor * x(i+1)$
      $z(i+2) \leftarrow z(i+1) - sigma * angles(i+1)$
    end for
    $f(n+1) \leftarrow s * (x(B+1) + j * y(B+1))$
  end for
  return f
end procedure

IV. Implementation Details

Figure 1 shows the hardware architecture of the FD ZC generator. It employs fixed point operations for all the computations. The architecture employs only one real multiplier which performs multiplication by a constant.

As illustrated in figure 1, the architecture includes an angle unit which includes the alpha and beta units. These units are responsible for computing $\alpha[k]$ based on the ZC root index input, u. The values output by the angle unit are already constrained to the range $[-\pi/2, \pi/2]$. The $\alpha[k]$ values are input to the translator unit which converts the $\alpha[k]$ values into $\theta[k]$ values by multiplying $\alpha[k]$ by $2\pi/N_{ZC}$. This unit employs the only multiplier used in the whole architecture. Therefore, the result of the translator unit is the argument value, $\theta[k]$, given by (12). The value $\theta[k]$ is input to the CORDIC calculation unit for determination of (7). If the $\theta[k]$ values input to the CORDIC unit are outside the range $[-\pi/2, \pi/2]$ (or $[-N_{ZC}/4, N_{ZC}/4]$) its outputs x and y must have their signs changed.

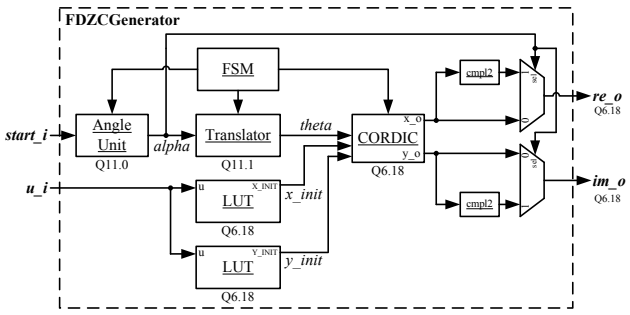

Figure 1: Architecture of the Frequency Domain ZC Generator.

Figures 2 and 3 depict the hardware architectures for units alpha and beta corresponding to implementations of (9) and (11) respectively. Together these two units make up the angle unit, responsible for recursively generating the $\alpha[k]$ values.

The physical root sequence number, u, input to the Beta unit is converted through a LUT to the equivalent FD root sequence number, u'. This LUT stores all possible conversions between u and u'. The second LUT present in the Beta unit architecture stores the inital β values, *i.e.*, α values for $k = 0$, for all possible physical root sequence numbers, u. This LUT stores the values defined by $u\frac{u'(u'-1)}{2}$ shown in (11). The translator unit translates the $\alpha[k]$ values into $\theta[k]$

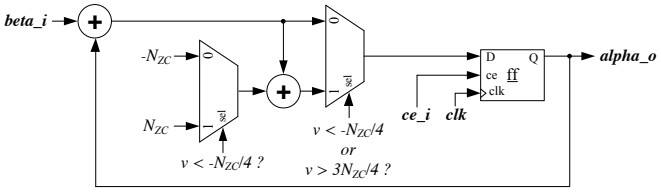

Figure 2: Architecture of the Alpha unit.

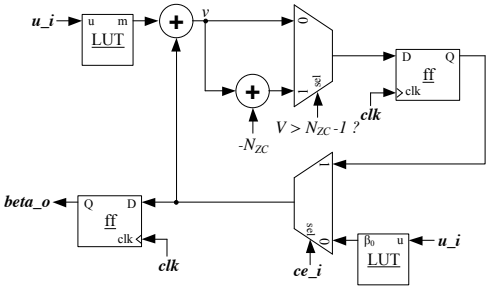

Figure 3: Architecture of the Beta unit.

values within the range $[-N_{\text{ZC}}/4, N_{\text{ZC}}/4]$, according to (12).

Figure 4 shows the hardware architecture of the CORDIC unit which corresponds to implementation of (4), (5) and (6). This implementation of the CORDIC algorithm adopts the rotation mode which explains the reason why the angle values, θ, fed into it must be within the range $[-\pi/2, \pi/2]$, [6], [7]. The architecture presented for this unit is of a bit-parallel and iterative CORDIC, which presents a good trade-off between speed and resource utilization [9]. The number of iterations the CORDIC unit takes to compute the sine and cosine values of an angle can be changed according to the value set to the parameter B. The higher the number of iterations the more precise the output values, x and y. Each iteration of the CORDIC unit takes one clock cycle to complete, therefore, it takes B clock cycles to output a valid value.

As may be seen in figure 4 the CORDIC unit includes a LUT of pre-calculated values for the arc-tangent. The input $shift$ yields an output of the arc-tangent of 2^{-i}. Assuming that the initialization values $x[1] = \text{real}(K * Z_u[0])$, $y[1] = \text{imag}(K * Z_u[0])$ and $z[1] = \theta[k]$ (with $\theta[k]$ being output from the translator unit) where real(.) and imag(.) are functions which extract the real and imaginary parts of a complex value respectively. The three adder units in the CORDIC architecture are configured to perform either addition or subtraction operations described in (4), (5) and (6) depending on the positive or negative sign of the decision variable d_i (this is also achieved mathematically with the use of the sign function ($\text{sgn}(z_i)$) used in Algorithm 1) which depends on whether z_i is less than 0 or not (hence the most significant bit of z_i selects the sign for the calculations).

From (3) it is possible to see that for a given root sequence, u the output of the CORDIC unit would need to be multiplied by a complex scaling constant $Z_u[0]$. This kind of multiplication demands a complex multiplier circuit which requires 4 real multipliers and 2 adders. The two inputs, x_init_i and y_init_i shown in figure 4 were added to the CORDIC circuit in order to initialize the CORDIC's $x[1]$ and $y[1]$ signals so that the complex multiplication by the $Z_u[0]$ constant is incorporated by the CORDIC circuit initialization. This further improves the algorithm efficiency by decreasing its complexity. These initialization values depend on the physical root sequence number, u, input to the unit and are stored in LUTs shown in figure 1.

Additionally, the units named $shifter$ in figure 4 are configured to perform signed left-shifting of the received input bits to the right by $shift$ positions. This is mathematically equivalent to dividing the number by 2^i as given in (4), (5) and (6).

The angle unit (composed of the alpha and beta units) can generate a valid $\alpha[k]$ value at every clock cycle rising, however, as the CORDIC unit takes B clock cycles to output valid values for x and y it needs to wait for the CORDIC before generating a new valid value. The Finite State Machine (FSM) shown in figure 1 controls the interaction among all the units making up the generator. The FSM is also responsible for signaling both the validity of the output values, x and y and when the whole sequence is done. Through the use of a FSM circuit the whole generator circuit needs only one clock signal, making it easier to implement the generator in cheap Field-Programmable Gate Array (FPGA) devices.

V. Simulation and Implementation Results

In order to assess the efficiency of the FD ZC generator proposed in this work some simulations were carried out. The first one compares floating-point precision Matlab generated ZC sequences with fixed-point precision ZC sequences generated by the circuit under test. The proposed architecture was developed in VHSIC Hardware Description Language (VHDL) and a corresponding bit-accurate Matlab model was developed for verification. The full design was targeted to a Xilinx Virtex 6 xc6vlx240t FPGA. Figure 5 shows the average error between ideal ZC elements (computed with floating point precision) and ZC sequence elements generated using the proposed architecture (computed with fixed point precision) for various values of B. Where B indicates the number of iterations the CORDIC module executes in order to find the cosine and sine values of the input angle. PRACH Format-0 preamble sequences of length N_{ZC} = 839 were considered.

Table I presents the average, minimum and maximum error values when all 838 possible ZC sequences are taken into account. The errors introduced by the CORDIC algorithm are due to the combination of quantization and approximation errors. Table I also shows the number of clock cicles the ZC generator takes to complete the generation of the 839 points that makes up a ZC sequence. The timing column gives the amount of time, in micro-seconds, to complete the generation when the system clock is set to 245.76 MHz, *i.e.*, 8*30.72 MHz which is 8 times faster than the LTE base sampling rate. As will be exemplified next, this system clock allows for the generator to conclude its task earlier leaving plenty of time for time-consuming tasks, such as peak detection, to complete.

Analyzing the average error values in Table I it is possible to notice that after B = 20 the error stops decreasing and reaches its minimum value. The minimum error a CORDIC implementation can reach is proportional to the data-path width [6]. Once the average error value is almost the same for B ranging from 20 up to 24, B can clearly be set to 20 without any noticeable loss of precision and with a gain in calculation speed.

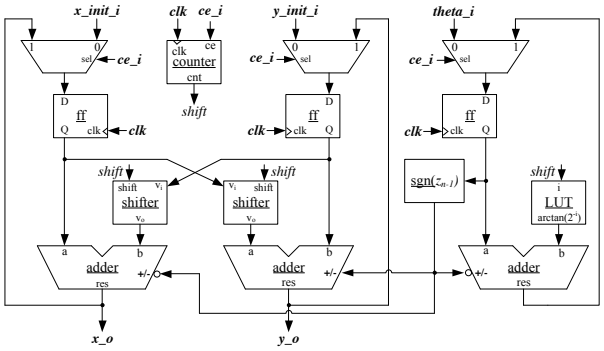

Figure 4: CORDIC architecture in circular rotation mode.

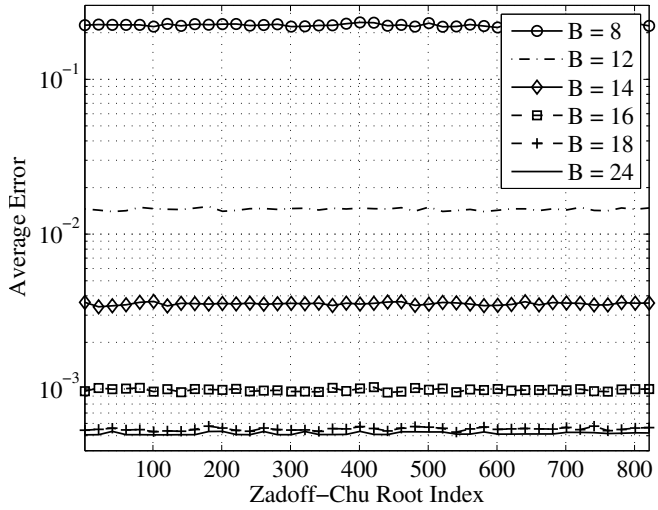

Figure 5: Average error.

As can be perceived, even when B is set to 24 iterations the generator takes less than 100 us to finish the whole sequence generation. For preamble format-0, the PRACH receiver must report all received preambles and respective delays to upper layers in less than 4 ms. This is required because the network is supposed to transmit the Random Access Response (RAR) message to a UE three subframes, each 1 ms long, after the end of the subframe containing the preamble transmission. This results in a total of 4 ms to receive, process, detect and report all preambles [10].

The maximum cell radius for format-0 preambles is of approximately 14 Km and consequently, the maximum cyclic-shift, N_{CS}, that can be applied to a ZC root sequence is 93, which requires the generation of 8 ZC root sequences. In this scenario, if B is set to 24 (the more precise the slower) iterations it would only require 765 us to generate all the 8 ZC sequences, leaving more than 3 ms for the remaining tasks which include filtering, down-conversion, Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) processing and preamble detection.

If we had employed a Time Domain (TD) approach instead of the FD one it would be necessary to translate the TD sequences into the corresponding FD sequences in order to carry out the cross-correlation with the received preambles. That way, the TD approach requires a Discrete Fourier Transform (DFT) block, which would increase the resource utilization and the time taken to report detected preambles to upper layers.

Table II presents information on the resource utilization of the proposed ZC generator. It summarizes the main results obtained from the implementation of the proposed architecture in a FPGA device. The number of registers, occupied slices, LUTs, memory resources and Digital Signal Processor (DSP) resource blocks are presented. The maximum operation frequency reached by the module is 252.143 MHz which allows it to run at 8*30.72 MHz without any problem.

Table I: Timing and Error Results

B	# of Cycles	Timing [µs]	Avg Error	Max Error	Min Error
8	10071	40.98	0.223865	0.233311	0.214569
10	11749	47.81	0.056496	0.058809	0.054196
12	13427	54.63	0.014503	0.015298	0.013832
14	15105	61.46	0.003558	0.003747	0.003360
16	16783	68.29	0.000989	0.001042	0.000941
18	18461	75.12	0.000554	0.000590	0.000519
20	20139	81.95	0.000522	0.000540	0.000505
22	21817	88.77	0.000519	0.000535	0.000504
24	23495	95.60	0.000519	0.000535	0.000505

Table II: Resource Utilization

Part number	XC6VLX240T-1ff1156 (Virtex 6)
Number of Slice Registers	257 out of 301440 0%
Number of Slice LUTs	1803 out of 150720 1%
Number of occupied Slices	482 out of 37,680 1%
Number of RAMB36E1/FIFO36E1s	0 out of 416 0%
Number of DSP48E1s	1 out of 768 0%
Maximum Frequency	**252.143 MHz**

From the results of Table II, it can be pointed out that no Block RAM (BRAM) memory is used. The synthesis tool distributes the contents of all LUTs present in the architecture (x_init, y_init, $angles$, m and β_0) across the device's distributed RAM. The unique xilinx's DSP48 resource used is employed by the translator unit to implement the multiplication shown in equation (12).

The implementation in FPGA of the proposed architecture presents a very low occupancy rate which may allow the use of lower cost FPGAs. There are two points to take into account when choosing a low-cost FPGA: 1) the maximum achievable frequency, since low-cost FPGAs tend to have worse timing characteristics and 2) slice count could increase for technologies below Virtex-6, once other families may use 4-bit LUT instead of the 6-bit LUT per slice.

VI. Conclusion

A hardware-efficient algorithm and architecture for computing FD ZC sequences featuring high-accuracy and low complexity characteristics has been presented. The algorithm is optimized to eliminate the use of memory and multipliers. A corresponding hardware architecture has been developed and employed in the PRACH receiver. Implementation and simulation results have demonstrated the efficiency, accuracy and low complexity of the proposed algorithm and architecture.

References

[1] Stefania Sesia, Issam Toufik and Matthew Baker, "*LTE - The UMTS Long Term Evolution: From Theory to Practice*", John Wiley & Sons, 2011.

[2] David C. Chu, "*Polyphase codes with good periodic correlation properties*", IEEE Transactions on Information Theory, pp. 531-532, July 1972.

[3] 3GPP TS 36.211, "*Physical Channels and Modulation (Release 10)*", September 2009.

[4] R. L. Frank, S. A. Zadoff and R. Heimiller, "*Phase shift pulse codes with good periodic correlation properties*", IRE Transactions on Information Theory, Vol 7, pp.254-257, October 1961.

[5] D. V. Sarwate, "*Bounds on crosscorrelation and autocorrelation of sequences*", IEEE Trans. on Inf. Theory, vol. IT-25, pp. 720–724, Nov. 1979.

[6] J. E. Volder, "*The CORDIC trigonometric computing technique*", IRE Transactions on Electronic Computers, vol. 8, pp. 330–334, Sep. 1959.

[7] Y. H. Hu and Z. Wu, "*An efficient CORDIC array structure for the implementation of discrete cosine transform*", IEEE Trans. Signal Proc., vol. 43, pp. 331–336, Jan. 1995.

[8] Mansour, M. M., "*Optimized architecture for computing Zadoff-Chu sequences with application to LTE*", GLOBECOM 2009. IEEE, Nov. 30 2009-Dec. 4, pp. 1-6, 2009.

[9] Ramesh Bhakthavatchalu, Parvathi Nair, Jismi K., and Sinith M.S., "*A Comparison of Pipelined Parallel and Iterative CORDIC Design on FPGA*", 2010 5th International Conference on Industrial and Information Systems, ICIIS 2010, Jul 29 - Aug 01, 2010, India.

[10] 3GPP TS 36.321, "*Medium Access Control (MAC) protocol specification (Release 10)*", March 2009.