Chapter

An Efficient FPGA-Based Frequency Shifter for LTE/LTE-A Systems

Felipe A.P. de Figueiredo and Fabbryccio A.C.M. Cardoso

Abstract

The Physical Random Access Channel plays an important role in LTE and LTE-A systems. Through this channel, the user equipment aligns its uplink transmissions to the eNodeB's uplink and gains access to the network. One of the initial operations executed by the receiver at eNodeB side is the translation of the channel's signal back to base-band. This operation is a necessary step for preamble detection and can be executed through a time-domain frequency-shift operation. Therefore, in this paper, we present the hardware architecture and design details of an optimised and configurable FPGA-based time-domain frequency shifter. The proposed architecture is based on a customised Numerically Controlled Oscillator that is employed for creating complex exponential samples using only plain logical resources. The main advantage of the proposed architecture is that it completely removes the necessity of saving in memory a huge number of long complex exponentials by making use of a Look-Up Table and exploiting the quarter-wave symmetry of the basis waveform. The results demonstrate that the proposed architecture provides high Spurious Free Dynamic Range signals employing only a minimal number of FPGA resources. Additionally, the proposed architecture presents spur-suppression ranging from 62.13 to 153.58 dB without employing any correction.

Keywords: LTE, LTE-A, 4G, PRACH, NCO, time-domain frequency shift, FPGA

1. Introduction

Long Term Evolution (LTE) technology is the next big step forward in cellular services. It is a 3GPP-defined standard that is able to provide uplink speeds of up to 50 megabits per second (Mbps) and downlink speeds of up to 100 Mbps. This new technology delivers several technical benefits to cellular networks. Its bandwidth can be scaled from 1.25 MHz up to 20 MHz [1–4].

In order to make LTE a true fourth generation (4G) technology, it was enhanced to meet the IMT Advanced requirements issued by the International Telecommunication Union (ITU). The necessary improvements are specified in 3GPP Release 10 and also known as LTE Advanced (LTE-A). The LTE-A technology increases the peak data rates to 1 Gbit/s in the downlink and to 500 Mbit/s in the uplink. LTE-A has several new features such as MIMO extensions (up to 4 × 4 for UL and up to

8 × 8 for DL), carrier aggregation, improvement of the performance at cell edge by supporting enhanced intercell interference coordination (eICIC) and relay nodes (RN) and uplink access enhancements such as simultaneous data and control information (physical uplink shared channel (PUSCH) and physical uplink control channel (PUCCH)) transmissions and clustered single-carrier frequency-division multiple access (SC-FDMA) [3].

In LTE and LTE-A, uplink physical random access channel (PRACH) is used for initial access requests from the user equipment (UE) to the evolved base station (eNodeB) and to obtain time synchronisation [3, 4]. In case of a need to access the network, a UE requests access by transmitting a random access (RA) preamble through PRACH [5]. The RA preamble is then detected by the PRACH receiver at eNodeB side, which estimates both the ID of the transmitted preamble and the propagation delay between UE and eNodeB. Then, the UE is time-synchronised according to a time alignment (TA) value (derived from the propagation delay estimate) transmitted from the eNodeB before the uplink transmission [6].

PRACH transmission opportunity is set by higher layers [7] and determines the frequency-domain location of the random access preamble within the physical resource blocks (RB). In this way, at eNodeB side, a fundamental operation before any attempt to detect random access preambles takes place is the extraction of relevant preamble signals through a time-domain frequency shift operation. This operation translates the PRACH signal from the frequency-domain location set by higher layers back to baseband so that preamble detection can be totally carried out in baseband [4].

This paper is an extension of a previous conference paper [8]. Differently from [8], where we provided only some very superficial aspects of the proposed algorithm and architecture, the current paper presents a meticulous analysis on its design and implementation aspects. Therefore, the main contributions of the current paper are (i) the design of a low computational complexity time-domain frequency shifter algorithm and hardware architecture to be employed in the PRACH receiver at eNodeB side; (ii) a thorough analysis of design and implementation details; (iii) discussion of the computational complexity of the proposed architecture in terms of FPGA resource utilisation and speed; and (iv) careful analysis of the implementation results considering spur suppression, i.e. spurious-free dynamic range (SFDR), signal-to-noise ratio (SNR), probabilities of correct and error detection and average error between time-domain frequency shift operations carried out by a floating-point model, referred here as Golden Model (GM), and by the fixed-point FPGA implementation of the proposed architecture.

This paper contributes with a method and architecture optimised and tested for reduced complexity on a Xilinx Virtex-6 LX240T FPGA device. Results show that the architecture presents spur suppression better than 62 dB and when it is employed in the PRACH receiver, the probability of correct detection achieved by the receiver is greater than 99% at a SNR of −21 dB.

The remainder of the paper is organised as follows. In Section 2 we offer some background on the physical random access channel and its features. Section 3 presents an efficient algorithm for a time-domain frequency shifter. Section 4 gives important practical considerations on the implementation of the proposed algorithm as well as detailed description of the units composing the main - architecture. Test methodology, simulation and implementation results are then presented in Section 5. Finally, Section 6 provides some concluding remarks.

2. Physical random access channel

The PRACH is the physical channel that initiates the communication exchange with the eNodeB. Based on the sequences sent through this channel, the eNodeB is able to compute the time it takes for the signal to travel from the user equipment (UE) to it, identifying and correcting this time delay before establishing a data packet connection. In order to establish a connection with the eNodeB, the UE starts the random access procedure by transmitting the random access sequence (also known as preamble) through the PRACH. The PRACH preamble is made up of a cyclic prefix and a preamble part as presented in Table 5.7.1-1 of [7]. This preamble is orthogonal to other uplink user data to allow the eNodeB do differentiate each UE. The subcarrier spacing for the PRACH is 1.25 KHz for formats 0 to 3 and 7.5 KHz for format 4. See example in **Figure 1**. Formats 0 to 3 are used for frame structure type 1, i.e. frequency division duplexing (FDD), and Format 4 is used for the frame structure type 2, i.e. time division duplexing (TDD) only [3]. As will be discussed later, the PRACH can be positioned at different frequency locations, i.e. RBs, depending on a parameter configured by higher layers. **Figure 1** shows an example of a possible PRACH's frequency-domain location.

Prime-length Zadoff-Chu (ZC) sequences are employed as random access preambles in LTE and LTE-A systems due to their constant amplitude zero autocorrelation waveform (CAZAC) properties, i.e. all samples of a ZC sequence are located on the unit circle (unitary magnitude), and their autocorrelation values are equal to zero for all time-lags different from zero [9, 10]. These properties turn ZC sequences into very useful preambles for channel estimation, time synchronisation and improved performance of the detection of PRACH preambles [4]. ZC sequences transmitted through the PRACH channel present the form defined by Eq. (1) [7]:

$$z_u(n) = \exp\left(\frac{-j\pi un(n+1)}{N_{\mathrm{ZC}}}\right), 0 \le n \le N_{\mathrm{ZC}} - 1, \tag{1}$$

where u is a positive integer known as ZC sequence index, n is the time index and N_{ZC} is the length of the ZC sequence, which for FDD systems is equal to 839 [7]. Random access preambles with zero correlation zones are defined from the uth root ZC sequence.

This sequence length, N_{ZC}, corresponds to approximately 69.92 physical uplink shared channel (PUSCH) subcarriers in each SC-FDMA symbol and offers a band

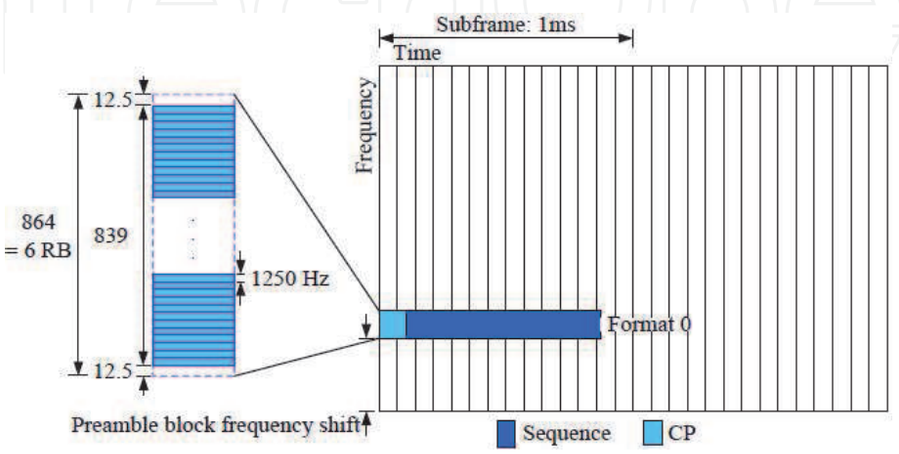

Figure 1.
Example of physical random access channel (PRACH) format 0.

protection of 72 − 69.92 = 2.08 PUSCH subcarriers, which corresponds to approximately one PUSCH subcarrier protection on each side of the preamble [7].

PUSCH subcarriers are spaced 15 KHz apart from each other.

The PRACH occupies a bandwidth of 1.08 MHz that is equivalent to six resource blocks (RB). Differently from other uplink channels, PRACH uses a subcarrier spacing of 1250 Hz for preamble formats 0 to 3 [7]. The ZC sequence is specifically positioned at the centre of the 1.08 MHz bandwidth, i.e. at the centre of the block of 864 available PRACH subcarriers, so that there is a guard band of 15.625 KHz on each side of the preamble, which corresponds to 12.5 null PRACH subcarriers. These guard bands are added to PRACH preamble edges in order to minimise interference from PUSCH. **Figure 1** depicts the PRACH preamble mapping according to what was just exposed.

The PRACH sequence, which for formats 0 and 1 is 800 us long, is created by cyclically shifting a root ZC sequence of prime-length N_{ZC}, defined as in Eq. (1). Random access preambles with zero correlation zones of length $N_{CS} - 1$ are generated by applying cyclic shifts to the uth root ZC sequence, according to Eq. (2):

$$x_{u,v}(n) = x_u((n + Cv) \bmod N_{ZC}), \tag{2}$$

where v is the sequence index and Cv is the cyclic shift applied to the root ZC sequence and calculated as $Cv = vN_{CS}$ for unrestricted sets [7]. The parameter N_{CS} gives the fixed length of the cyclic shift. All the possible values for these parameters are defined in [7].

2.1 PRACH receiver

In the literature there are two approaches for PRACH receivers, the full frequency domain and the hybrid time/frequency domain [4, 11]. Although the full frequency-domain approach provides the optimal detection performance, this approach uses considerably large size discrete Fourier transform (DFT), to be more

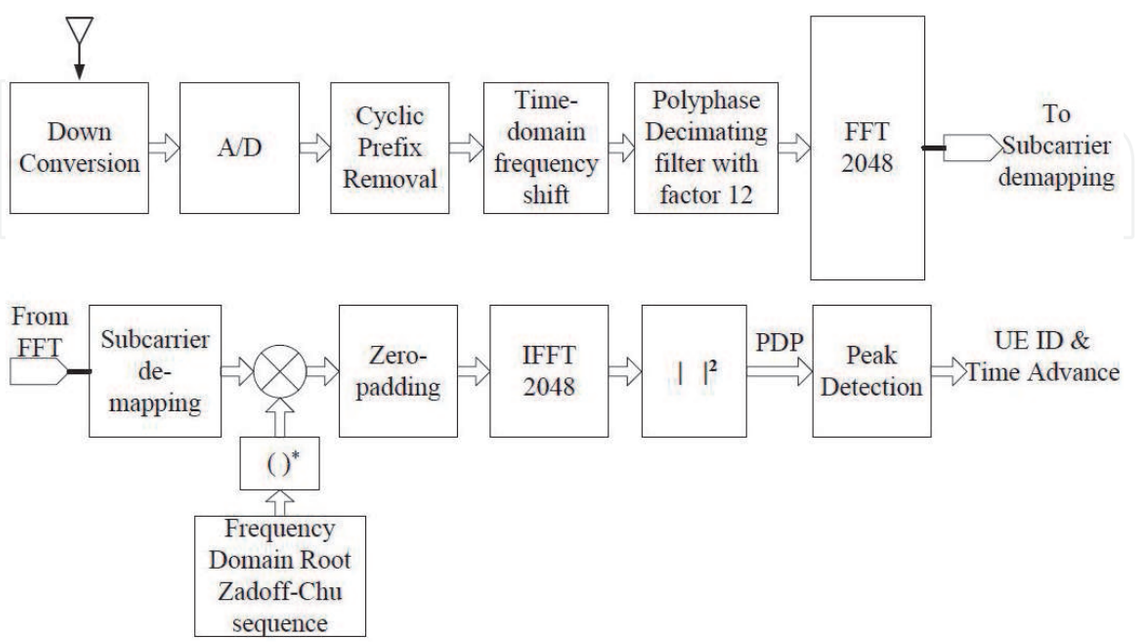

Figure 2.
Architecture of a hybrid time-/frequency-domain PRACH receiver.

precise a 24576-point DFT. On the other hand, the hybrid time-/frequency-domain approach uses FFT/IFFT blocks of the same size, i.e. 2048-point FFT when the decimation factor adopted is 12. Thus, the hybrid time/frequency domain substantially reduces the complexity of the hardware implementation. Therefore, in order to reduce the implementation complexity of the PRACH receiver, we adopt the hybrid time-/frequency-domain approach, which results in more practical implementations [4]. **Figure 2** depicts the PRACH receiver architecture implemented and being used in our L1 solution.

The received signal, i.e. possible random access preamble signal, is first preprocessed in time domain and then transformed into the frequency domain by an FFT block, multiplied by a Fourier transformed root Zadoff-Chu (ZC) sequence, and then the resulting sequence is searched for peaks above a predefined threshold which is calculated to produce a given probability of false alarm. **Figure 2** depicts the main components of the PRACH receiver at eNodeB side (for further details refer to [12]). The first block in **Figure 2** is the cyclic prefix remover, which discards all samples from the CP part of the preamble. Next, the PRACH pass-band signal is shifted to baseband by multiplying it with a complex exponential. In the sequence, the baseband signal is fed into a decimator block, which decimates the signal by a factor of 12; now instead of 24,576 samples in the case of format 0, we have only 2048 samples. The FFT block is responsible for transforming the SC-FDMA symbols from time domain into frequency domain. Next, the subcarrier demapping module extracts the RACH preamble sequence from the correct FFT bins. Then, the output of subcarrier demapping module is multiplied by the locally stored root ZC preamble, and then, the result of the multiplication is fed into the zero-padding module. Finally, the IFFT block is used to produce the cross-correlation between the root ZC sequence and the received preamble signal. All samples coming out of the IFFT block have their square modulus calculated producing what is known as power delay profile (PDP) samples. Finally, the preamble detection block employs the PDP samples to estimate the noise power, set a detection threshold and then decide whether a preamble is present or not. As an output of the detection process, this block reports to the MAC layer all detected preambles and its respective time advance (TA) estimates. For further information on this receiver architecture and detection algorithm, refer to [4, 12].

2.2 Preamble format

The PRACH preamble, illustrated in **Figure 3**, consists of three parts: a cyclic prefix (CP) with length T_{CP}, which is added to the preamble in order to effectively eliminate intersymbol interference (ISI) and a signature or sequence part of length

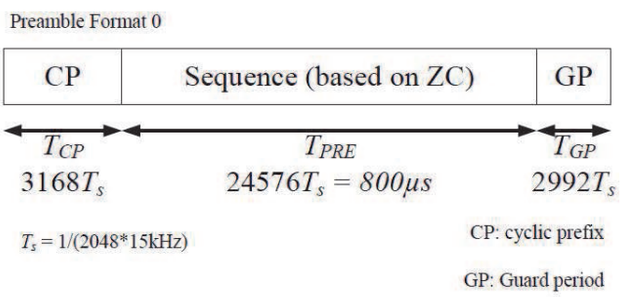

Figure 3.
Random access preamble format 0.

Preamble format	T_{CP}	T_{PRE}	T_{GP}
0	$3168.T_s$	$24576.T_s$	2976
1	$21024.T_s$	$24576.T_s$	15,840
2	$6240.T_s$	$2.24576.T_s$	6048
3	$21024.T_s$	$2.24576.T_s$	21,984

Table 1.
Random access preamble formats.

T_{PRE} and of a guard period T_{GP} which is an unused portion of time at the end of the preamble used for absorbing the propagation delay. The standard defines four different preamble formats for FDD operation [7]. Parameters T_{PRE}, T_{CP} and T_{GP} are set according to the chosen preamble format.

Figure 3 shows the parameter values for format 0, and the values for all formats are listed in **Table 1** where T_s is known as the standard time unit which is used throughout the LTE specification documents. It is defined as $T_s = 1/(15{,}000 \times 2048)$ seconds, which corresponds to a sampling rate of 30.72 MHz.

2.3 PRACH preamble signal

The PRACH preamble signal $s(t)$ can be defined as follows [7]:

$$s(t) = \beta_{PRACH} \sum_{k=0}^{N_{ZC}-1} \sum_{n=0}^{N_{ZC}-1} x_{u,v}(n).\exp\left[-\frac{j2\pi nk}{N_{ZC}}\right] .\exp\left[j2\pi[k+\varphi+K(k_0+1/2)]\Delta f_{RA}(t-T_{CP})\right], \quad (3)$$

where $0 \le t < T_{PRE} + T_{CP}$, β_{PRACH} is an amplitude scaling factor and $k_0 = n_{PRB}^{RA} N_{SC}^{RB} - N_{RB}^{UL} N_{SC}^{RB}/2$. The location in the frequency domain is controlled by the parameter n_{PRB}^{RA} also known as $n_{PRB_{offset}}^{RA}$ (it is the input *frequency_offset_i* of the time-domain frequency shifter module) expressed as a resource block number configure by higher layers and fulfilling $0 \le n_{PRB}^{RA} \le N_{RB}^{UL} - 6$; this inequality is only valid for formats 0, 1, 2 and 3, i.e. FDD. The factor $K = \Delta f/\Delta f_{RA}$ accounts for the ratio of subcarrier spacing between the PUSCH and PRACH, and it is equal to 12 as Δf = 15 KHz and Δf_{RA} = 1250 Hz. The variable φ (equal to 7 for LTE FDD) defines a fixed offset determining the frequency-domain location of the random access preamble within the resource blocks. N_{RB}^{UL} is the uplink system bandwidth (in RBs), and N_{SC}^{BB} is the number of subcarriers per RB, i.e. 12.

By noticing that the inner summation is the DFT of $x_{u,v}(n)$ of length N_{ZC}, we can rewrite Eq. (3) in the following way:

$$s(t) = \beta_{PRACH} \sum_{k=0}^{N_{ZC}-1} X_{u,v}(k).\exp\left[j2\pi k\Delta f_{RA}(t-T_{CP})\right] .\exp\left[j2\pi(\varphi+K(k_0+1/2))\Delta f_{RA}(t-T_{CP})\right]. \quad (4)$$

Again, by noticing that the first part of the summation in Eq. (4) is a time shift applied to the DFT of $x_{u,v}(n)$, we can rewrite that first part of the equation as follows by replacing t by Δt, which is referred in [7] as the standard time unit T_s, i.e. the sampling rate:

$$s(t) = \beta_{PRACH} \sum_{k=0}^{N_{ZC}-1} X_{u,v}(k) \cdot \exp\left[j2\pi k \Delta f_{RA} \Delta t (n - N_{CP})\right], \tag{5}$$

where N_{CP} is the number of samples corresponding to the CP interval as shown in **Figure 3** and $\Delta f_{RA}\Delta t = 1/N_{PRE}$ (where for formats 0 ad 1, $N_{PRE} = 24{,}576$).

Then rewriting Eq. (5), we have

$$\begin{aligned} s(t) &= \beta_{PRACH} \sum_{k=0}^{N_{ZC}-1} X_{u,v}(k) . \exp\left[\frac{j2\pi k(n - N_{CP})}{N_{PRE}}\right] \\ &= \beta_{PRACH} . x'_{u,v}(n - N_{CP}). \end{aligned} \tag{6}$$

Therefore as it can be easily seen, the result of the above equation is nothing more than the application of the DFT's time-shift theorem. It is also easy to see that this equation is the IDFT of $X_{u,v}(k)$ with length N_{PRE}. With that in mind, Eq. (4) can be rewritten as

$$\begin{aligned} s(t) = \beta_{PRACH} . x'_{u,v}(n - N_{CP}) \\ . \exp\left[\frac{j2\pi(\varphi + K(k_0 + 1/2))(n - N_{CP})}{N_{PRE}}\right]. \end{aligned} \tag{7}$$

Eq. (4) can be reorganised in the following way:

$$\begin{aligned} s(t) = \beta^{PRACH} . \exp\left[\frac{-j2\pi N_{CP}(\varphi + K(k_0 + 1/2))}{N_{PRE}}\right] \\ . \left\{ x'_{u,v}(n - N_{CP}) . \exp\left[\frac{j2\pi n(\varphi + K(k_0 + 1/2))}{N_{PRE}}\right] \right\}. \end{aligned} \tag{8}$$

The part of Eq. (8) between curly braces represents a circular frequency shift u, v applied to $x'_{u,v}$ $(n - N_{CP})$, i.e.

$$x'_{u,v}(n - N_{CP}) . \exp\left[\frac{j2\pi nm}{N_{PRE}}\right] \overset{DFT}{\leftrightarrow} x_{u,v}(K - m), \tag{9}$$

where m is the frequency shift applied to the PRACH signal before it is transmitted and it is given by the following equation

$$m = \varphi + K(k_0 + 1/2). \tag{10}$$

Once we are only dealing with FDD, Eq. (10) can be further simplified as

$$m = 13 + 144 n_{PRB}^{RA} - 72 N_{RB}^{UL}. \tag{11}$$

Therefore, at the PRACH receiver side, after removing CP and GP, the preamble is still shifted in frequency domain by an offset factor given by m. For further processing, it is necessary to convert the shifted preamble into baseband. This conversion is performed by the time-domain frequency shift module (see **Figure 2**), which multiplies the received preamble by the conjugate of the complex exponential term given in Eq. (9).

3. Efficient algorithm of a time-domain frequency shifter

In this section, we present an efficient algorithm used to apply frequency-domain shifts to random access preamble signals in time domain (i.e. without the need to convert them to the frequency domain) through the use of a customised numerically controlled oscillator (NCO) and a complex multiplier. We also discuss the advantages presented by the proposed algorithm.

3.1 Numerically controlled oscillator

Numerically controlled oscillators (NCO) are important components in many digital communication systems. They are generally employed in quadrature synthesisers, which are used for constructing digital down- and upconverters and demodulators and here for time-domain frequency shifters. A very common method for creating digital complex or real valued sinusoid signals uses a lookup table (LUT) approach [13]. In this approach, a LUT saves into memory digital samples of a sinusoid signal.

A digital integrator is then employed to compute the correct phase arguments, which are mapped by the LUT to the desired output sinusoid samples. The integrator computes a phase slope that is mapped to a sinusoid (possibly complex) by the LUT. This value is presented to the address port of the LUT that performs the mapping from phase space to time [14].

A LUT usually saves into memory uniformly spaced samples of sine and cosine waveforms. This set of samples comprises a single cycle of a prototype complex sinusoid waveform with length $N = 2^{B_{\Theta(n)}}$ and consists of specific values of the argument $\Theta(n)$ of sinusoid waveform, as defined by Eq. (12).

$$\Theta(n) = n\frac{2\pi}{N}, \tag{12}$$

where n is the index of the time sample and $B_{\Theta(n)}$ is the number of bits employed in the phase accumulator which is calculated as shown in Eq. (13):

$$B_{\Theta(n)} = \log_2\left\lceil\frac{f_{clk}}{\Delta f}\right\rceil, \tag{13}$$

where $\lceil\cdot\rceil$ denotes the ceiling operator, f_{clk} is the system clock frequency and Δf is the frequency resolution of the NCO. The frequency resolution, Δf, of the NCO is a function of f_{clk} and $B_{\Theta(n)}$. Then Δf can be determined using the following equation:

$$\Delta f = \frac{f_{clk}}{2^{B_{\Theta(n)}}} = \frac{f_{clk}}{N}. \tag{14}$$

The output frequency, f_{out}, of the NCO waveform is a function of f_{clk}, $B_{\Theta(n)}$ and the phase increment value $\Delta\theta$. That is, $f_{out} = f(f_{clk}, B_{\Theta(n)}, \Delta\theta)$ which is given in Hertz and is defined in Eq. (15). The phase increment, $\Delta\theta$, is an unsigned value which defines the NCO output frequency:

$$f_{out} = \frac{f_{clk}\Delta\theta}{2^{B_{\Theta(n)}}} = \frac{f_{clk}\Delta\theta}{N}. \tag{15}$$

The accuracy of a signal sequence formed by reading samples of a sinusoid signal from a LUT is influenced by both the amplitude and the phase of the quantization

process. The width and length of the LUT memory directly impact the resolution of both the signal's amplitude and phase angle. These resolution limits correspond to time base jitter and amplitude quantization of the signal, respectively. Additionally, these resolution limits add a white broadband noise floor and spectral modulation lines to the spectrum of the generated signal sequence [15].

Quarter-wave symmetry in the basis waveform can be exploited to construct an NCO that uses shortened tables. We will discuss this approach next.

3.2 Iterative time-domain frequency shift algorithm

The optimised algorithm representing the time-domain frequency shift operation is presented in Algorithm 1. It depicts the data processing executed by each one of the units in **Figure 4**.

At first, during eNodeB's initialisation, the parameters *offset* and *bandwidth* (*bw*) are sent by higher layers to the PHY which in turn feeds them into the time-domain frequency shifter module so that the discrete frequency shift calculator unit is able to calculate the actual frequency shift to be applied to the received PRACH signal. Whenever a subframe in which random access preamble transmissions are allowed happens (it is set according to Table 5.7.1-2 in [7]) and after CP is removed, the customised NCO unit generates a complex exponential signal with frequency set earlier by the discrete frequency shift calculator unit and multiplies it sample by sample with the incoming PRACH complex signal samples; note that it is a complex multiplication once both are complex signals.

The procedure inputs are *re_ad*, *im_ad*, *offset*, *bw* and *cos table* where *re_ad* and *im_ad* are the already CP removed quadrature samples coming from the analog to digital converter (ADC), *offset* and *bw* are the configuration parameters coming from higher layers and used to calculate the frequency shift necessary to translate the pass-band preamble signal back to baseband and *cos_table* is the LUT containing the samples of a sinusoid used to generate the complex exponential signal. The angle mapper is the main part of the customised NCO algorithm shown in Algorithm 1 once it maps *theta* into a value of a 1/4-length cosine table.

In the light of what was presented in the previous section, we now discuss Algorithm 1. The first part of the algorithm is responsible for calculating the discrete frequency of the complex exponential signal that the NCO must generate in order to

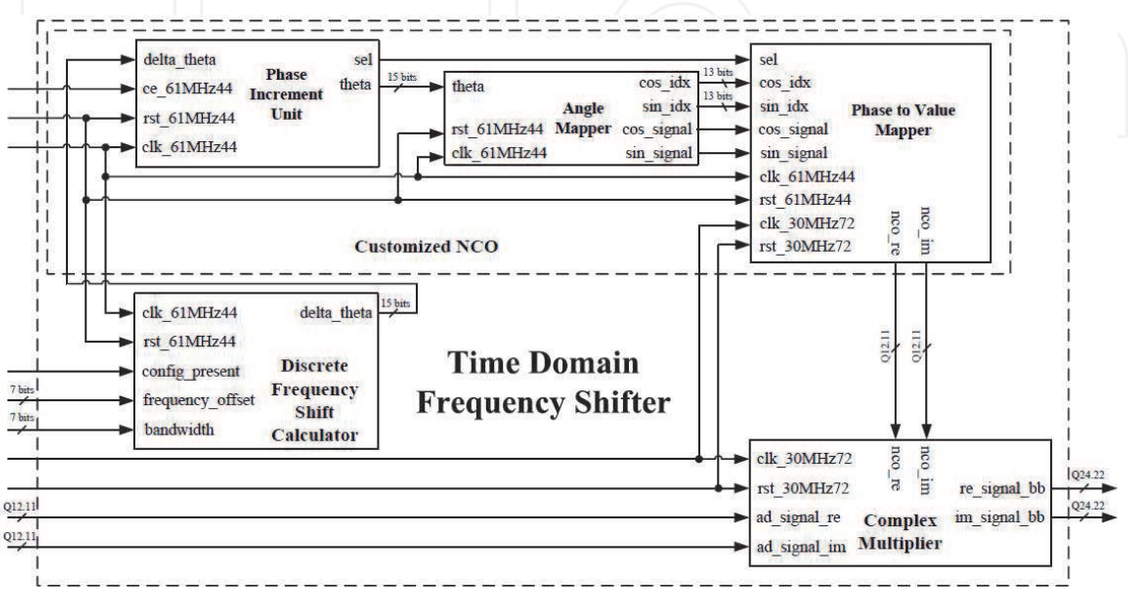

Figure 4.
Blocks composing the time-domain frequency shifter module.

shift the received pass-band preamble signal to baseband. The discrete frequency shift, m, is calculated as shown in Eq. (11). By remembering that $\Delta f_{RA}\Delta t = 1/N_{PRE}$, we can then rewrite the exponential part of Eq. (9) as

$$\exp\left[j2\pi\left(m\Delta f_{RA}\right)t\right]. \tag{16}$$

By analysing the equation above, it is noticeable that all frequency shifts are integer multiples of Δf_{RA}, and in this way we state that the output frequency of the NCO must be $f_{out} = m\Delta f_{RA}$. Before proceeding we must define the values for some parameters presented in the previous section. The frequency resolution $\Delta f = \Delta f_{RA} =$ 1250 Hz, the system clock frequency $f_{clk} = 1/T_s = 30.72$ MHz, the length, $N = 2^{B_{\Theta(n)}}$, of the single cycle of the basis complex waveform is made equal to 24,576 samples. The cycle length can be expressed as $N = f_{clk}/\Delta f_{RA}$.

Algorithm 1. Time-domain frequency shifter algorithm

```
1: procedure TDFreqShifter(re_ad, im_ad, offset, bw, cos_table)
2:
3:    ▷ ********* Discrete Frequency Shift Calculator *********
4:    m = 13 + 144 * offset − 72 * bw;
5:    ▷ —- Frequency Control Word (FCW) —-
6:    delta_theta = m;
7:    if m < 0 then
8:       delta_theta = N + m;
9:    end if
10:
11:   ▷ ************* customised NCO Algorithm ************
12:   theta = 0;
13:   for i ← 0 to N − 1 do
14:      ▷ —- Angle Mapper —-
15:      cos_signal = 1;
16:      sin_signal = 1;
17:      if theta > 3 * N/4 then
18:         cos_idx = (N − theta);
19:      else
20:         if theta > N/2 then
21:            cos_idx = (theta − (N/2));
22:            cos_signal = −1;
23:         else
24:            if theta > N/4 then
25:               cos_idx = ((N/2) − theta);
26:               cos_signal = −1;
27:               sin_signal = −1;
28:            else
29:               cos_idx = theta;
30:               sin_signal = −1;
31:            end if
32:         end if
33:      end if
34:
35:      sin_idx = (N/4) – cos_idx;
36:
37:      if cos_idx == N/4 then
```

```
38:            cos_idx = ((N/4) − 1);
39:          end if
40:          if sin_idx == N/4 then
41:            sin_idx = ((N/4) − 1);
42:          end if
43:
44:          ▷ —- Phase to Value Mapping (LUT) —-
45:          re_nco(i) = cos_signal ∗ cos_table(cos_idx);
46:          im_nco(i) = sin_signal ∗ cos_table(sin_idx);
47:
48:          ▷ —- Phase Increment (Phase Accumulator) —-
49:          theta = (theta + delta theta);
50:          if theta >= N then
51:            theta = (theta − N);
52:          end if
53:
54:          ▷********** Complex Multiplier **********
55:          re_bb(i) = re_ad(i) ∗ re_nco(i) – im_ad(i) ∗ im_nco(i);
56:          im_bb(i) = re_ad(i) ∗ im_nco(i) + im_ad(i) ∗ re_nco(i);
57:      end for
58:      return re_bb, im_bb
59: end procedure
```

Then, using the aforementioned definitions and rewriting Eq. (15) letting $\Delta\theta$ in evidence, we have

$$\Delta\theta = \frac{f_{out}N}{f_{clk}} = \frac{m\Delta f_{RA}N}{f_{clk}} = m\frac{m\Delta f_{RA}}{f_{clk}}\frac{f_{clk}}{\Delta f_{RA}} = m. \tag{17}$$

In case m is negative, it is necessary to calculate its module in relation to N_{PRE} before feeding it into the customised NCO. Note in Algorithm 1 that the module operation is simply done by adding N_{PRE} to the negative value of m.

In a traditional NCO algorithm, i.e. one that adopts full-period waveforms, there would be two main parts, namely, phase accumulator and LUT. In its simplest form, there would be two LUTs storing samples of a cosine and a sine wave. However, this approach generally results very large tables, which sometimes are impractical. Therefore, for a practical implementation with reduced tables, the proposed algorithm employs only one LUT exploiting quarter-wave symmetry in the basis waveform and the constant phase offset ($pi/2$) between sine and cosine signals. In this approach we use one LUT with $N/4$ samples. However, when exploiting quarter-wave symmetry, the mapping from phase space to time is not direct as in the traditional NCO algorithm.

In order to exploit quarter-wave symmetry, an algorithm is needed to map the angle values (phase space), θ, output by the phase accumulator into valid positions of a shortened LUT containing the samples of a cosine signal. This task is performed by the angle mapper part of Algorithm 1. The angle mapper maps angle values in the second, third and fourth quadrants into the first one and tracks the signals that must be applied to cosine and sine values. As can be seen in Algorithm 1, the indices for generating the cosine signal are calculated first, and then a $N/4$-phase offset, which is equivalent to a $pi/2$ offset, is applied to it in order to generate the sine indexes. In order to store values ranging from 1 to 0, i.e. the first quadrant of a cosine signal, $(N/4) + 1$ samples would be necessary where the last one is zero.

The zero value can be mapped to the value stored at the $N/4$-th position with minimal degradation on SFDR performance. Therefore, as can be seen in the algorithm, when either sine or cosine indexes are equal to $N/4$, their values are changed to $(N/4) - 1$, which is the closest value to zero.

As the LUT only stores samples from the first quadrant of a cosine signal, i.e. only positive values, the phase to value mapper part of the algorithm must apply the correct signals (provided by the angle mapper) to the LUT's output.

The phase increment (also referred as phase accumulator) part of the algorithm acts as an integrator. It calculates at each iteration of the algorithm a new phase value, θ, by using the phase increment $\Delta\theta$ value provided by the discrete frequency shift calculator part. Once the angle mapper algorithm can only map values ranging from 0 to $N - 1$ into the range 0 to $(N/4) - 1$, it is necessary to apply a module operation in case the resulting phase value is equal or greater than N.

The module operation is easily performed by subtracting N from the phase value. The last part of the algorithm multiplies sample by sample the generated complex exponential signal by the ADC signal. That complex multiplication operation then translates the pass-band PRACH signal into baseband.

3.3 Advantage of the proposed algorithm

The main advantage of the proposed algorithm is the memory savings attained by the use of a 1/4-length cosine table instead of storing in RAM each one of the $2N$ possible samples of a complex exponential signal. **Table 2** shows the memory utilisation for some data widths if we were to store all the 2N samples (sine and cosine waves) corresponding to one complete period of the basis complex exponential waveform necessary to translate the received PRACH signal into baseband. In Xilinx FPGAs, a block RAM (BRAM) is a dedicated, i.e. they cannot be used for anything, but RAM, a two-port memory containing several kilobits of RAM. A FPGA contains a limited number of these blocks. The configuration logical blocks (CLB) in most of Xilinx FPGA contain a small RAM. They are called distributed (LUT) RAM because they are distributed throughout the FPGA once they are part of a CLB. This kind of RAM can normally store only a dozen bits. A reasonable rule of thumb when designing with FPGAs is that if you need a lot of RAM, as is the case here, you should use BRAMs; otherwise, the FPGA resources will be eaten up implementing the RAM in distributed RAM. It is important to say that a Virtex-6 FPGA device has only 416 36 kb BRAMs which makes it a very precious resource when implementing a large project as an L1 PHY, and in this way its usage must be taken into account during planning and development. One alternative to decrease the number of occupied BRAMs is the exploitation of quarter-wave symmetry in the basis waveform. This alternative results in a customised NCO that employs a shortened LUT, as can be seen in **Table 3**.

The fourth column in **Table 3** shows the reduction of used BRAMs when exploiting quarter-wave symmetry. As can be noticed, this approach results in a

Data width	Size in kb	No. of 36 kb BRAMs
8	384	11
12	576	16
16	768	22
24	1152	32

Table 2.
Memory utilisation when storing the full period of the complex exponential.

Data width	Size in kb	No. of 36 kb BRAMs	Reduction in %
8	48	2	81.8
12	72	2	87.5
16	96	3	86.4
24	144	4	87.5

Table 3.
Memory utilisation when employing quarter-wave symmetry.

more area efficient implementation because the memory requirements are minimised, i.e. fewer FPGA BRAMs are required. Therefore this approach saves on valuable chip area and also reduces power consumption.

4. Implementation details

This section presents some discussions on implementation details of the proposed architecture. It is suitable for implementation on devices that employ hardware description language (HDL) as part of its design process such as field-programmable gate arrays (FPGA) and application-specific integrated circuits (ASIC). **Figure 4** shows the hardware architecture of the time-domain frequency shifter module. The architecture employs only one LUT and exploits quarter-wave symmetry for shortened tables.

The proposed architecture works with two different system clocks. The first clock, of 30.72 MHz, is used by the ADC unit and therefore dictates the rate the complex multiplication is performed. The complex multiplier and phase to value mapper modules are the only two modules running at this clock rate. The second clock rate employed in the system is 61.44 MHz and is used so that two samples of the complex exponential waveform can be read from the same LUT memory during the period of one sample arriving from the ADC module. This dual system clock scheme drastically reduces the amount of RAM memory necessary for the system to be implemented. All modules composing the proposed architecture run at this clock rate, the complex multiplier module being the only exception.

The two input parameters, *bandwidth* and *frequency_offset*, are fed into the module only when the eNodeB is being initialised. They can be considered static parameters of eNodeB. The input signal *config_present* is asserted by higher layers during the initialisation process to inform when the input parameter values are valid.

The proposed architecture has to be informed through the *ce_61MHz44* signal when the sequence section of the RACH preamble starts so that it can be multiplied by the local complex exponential, which is generated by the customised NCO module. The signal *ce_61MHz44* is set by the PRACH receiver module after it removes (i.e. discards) the CP portion of the RACH preamble and has to stay in high state level for the whole duration of the RACH sequence portion, e.g. in the case of format 0, the signal *ce_61MHz44* must remain in high state for 24,576 30.72 MHz clock cycles, i.e. $2 \times 24{,}576$, when considering the 61.44 MHz system clock rate. It is important to highlight that some latency is expected once all modules have registered outputs, and therefore, this latency has to be taken into consideration by the PRACH receiver module when setting the enable signal of the chip.

Another important characteristic of the proposed architecture is that it only employs a total of four multipliers that are used by the complex multiplier module. Moreover, the proposed architecture only uses plain add and bit-shift operations to

compute the value of trigonometric functions such as complex exponential sequences, which turns it into a highly efficient hardware architecture in terms of logical resource consumption.

Additionally, the proposed frequency shift architecture can have its inputs and outputs entirely configured, i.e. the width of input and output signals can be set to one of the following choices: 8, 12, 16 and 24 bits. In the case of our actual implementation, we employ an ADC with an output of 12 bits for each one of the quadrature components, i.e. in-phase (I) and quadrature (Q) components, and it has a fixed-point representation (Q-format) of Q12.11, i.e. 1 bit for the integer part and 11 bits for the fractional part. The I and Q components computed by the customised NCO module present the same fixed-point representation of the input of the frequency shifter module. In relation to this particular point, after the complex multiplication between the NCO and the ADC quadrature samples, which requires the multiplication and subsequent addition of samples, the fixed-point representation of the modules' output is equal to Q25.22. Since the maximum possible value generated by the complex multiplication operation is 2, the integer part only needs 2 bits instead of 3 bits. Therefore, depending on the selected width configuration, the fixed-point representation of the complex signal output by the module can be configured to Q8.6, Q12.10, Q16.14 and Q24.22.

4.1 Discrete frequency shift calculator unit

Figure 5 depicts the proposed architecture for the discrete frequency shift calculator module. This module is employed to compute the frequency shift, m, that must be applied to the received PRACH signal sequence in order to translate it into a baseband signal, i.e. a signal centred around 0 Hz. Therefore, in order to compute such frequency shifts, the module implements Eq. (11). All multiplications involved here are executed by bit-shifting the input values N_{RB}^{UL} and n_{PRB}^{RA} by the constant values −72 and 144, respectively, and then adding the result to the constant value 13. Before sending the value of m to the customised NCO module, it is necessary to verify whether the resulting value is negative or not; if it is negative, then the constant value N has to be summed to the result value, which turns m into a positive value. It is done due to the fact that the phase increment module, which composes the customised NCO module, only expects positive input values. As defined by Eq. (17), the discrete frequency shift, m, is equal to $\Delta\theta$, which is the necessary input value for the customised NCO module to operate properly.

4.2 Customised numerically controlled oscillator

The customised numerically controlled module is composed of three blocks, namely, phase increment, angle mapper and phase to value mapper, as can be seen

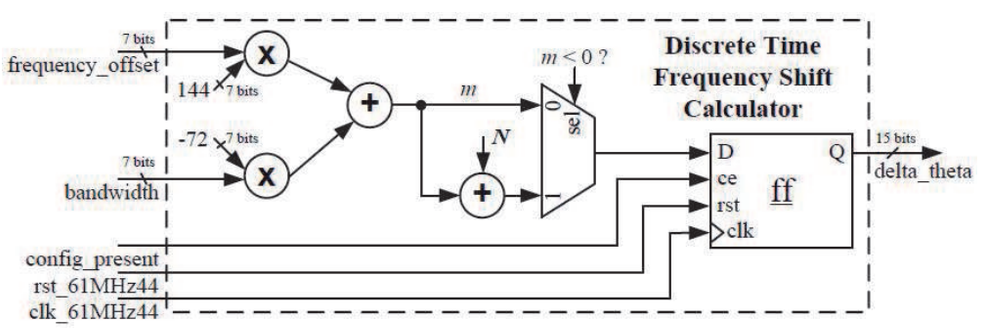

Figure 5.
Architecture of the discrete frequency shift calculator unit.

in **Figure 4**. The first two modules run at a system clock of 61.44 MHz, and the last one runs at 30.72 and 61.44 MHz since it is the module in charge of reading both quadrature components, I and Q, from the LUT memory inside one period of the 30.72 MHz system clock rate.

Figure 6 depicts the proposed architecture of the phase increment module. This module is the implementation of a digital integrator, which computes the phase argument, θ, sent to the angle mapper module. At each iteration, $\Delta\theta$ is added to θ, which starts from a value equal to 0. If θ results in a value that is greater than $N + 1$, then the constant value $-N$ is added to it so that its value remains less than N and, therefore, it can be correctly mapped into a valid phase argument value. The procedure we have just described is nothing but the direct implementation of the module operation, $\mathrm{mod}(\theta, N)$. This module only produces a valid θ value when the selection signal, *sel*, is set to high level. The *sel* signal is produced by a system clock divisor that divides the 61.44 MHz system clock by 2, i.e. the module produces a valid output value at a clock rate of 30.72 MHz. The *sel* signal is also generated by the module in order to feed the phase to value mapper module. As Eq. (13) is equal to N and it is not equal to a power of 2, the data width of the phase increment module, $B_{\Theta(n)}$, is ceiled, then resulting in a value with 15 bits.

Figure 7 shows the proposed architecture for the angle mapper module. This module is in charge of translating the phase argument value, θ, which can vary in the range between 0 and $2 * \pi$ (i.e. from 0 to N) into a phase argument value inside the first quadrant of the circle, i.e. a value in the range between 0 and $pi/2$ (i.e. from 0 and $N/4$). The output value of this module is the index of cosine waveform, which is employed as an address value to access one of the $N/4$ values saved in the LUT memory. In order to compute the index of the sine waveform, a constant phase offset value equal to $pi/2$, i.e. $N/4$, has to be applied to the cosine index value. Since the value corresponding to $\cos(\pi/2)$, i.e. 0, is not saved into the LUT memory, whenever either cosine or sine index values are equal to $N/4$, then their index values are modified to $(N/4) - 1$, which is the closest index value to 0.

The module is also responsible for keeping track of the signals (i.e. +/−) that must be applied to the I and Q values at the output of the phase to value mapper module. These signals translate the phase argument value, which are represented by the sine and cosine index values, back to its original quadrant of the disc. At the input of this module, the phase argument, θ, presents a databus width of 15 bits so that it is able to access N samples stored in the LUT memory, i.e. all the four

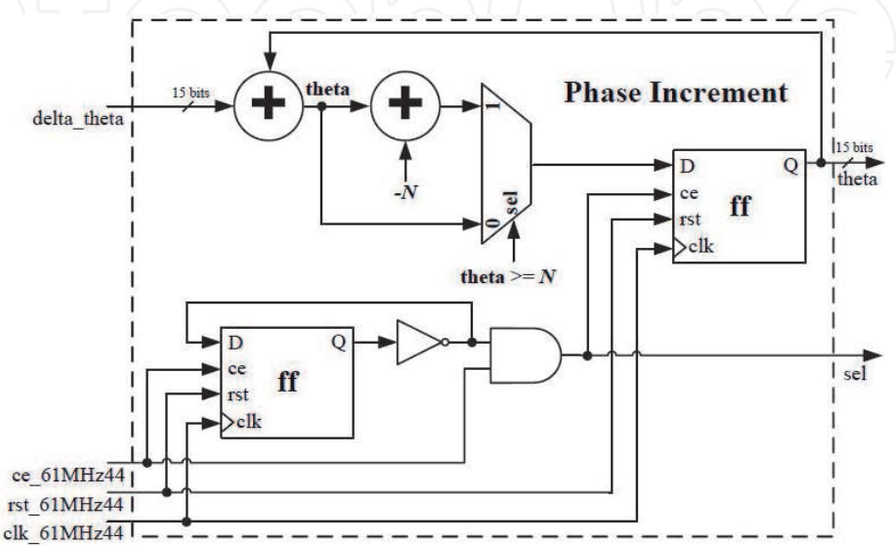

Figure 6.
Architecture of the phase increment unit.

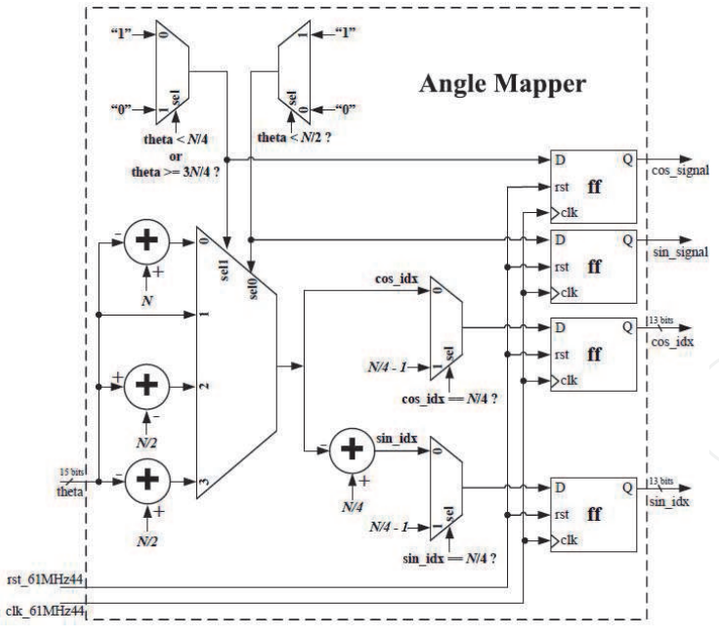

Figure 7.
Architecture of the angle mapper unit.

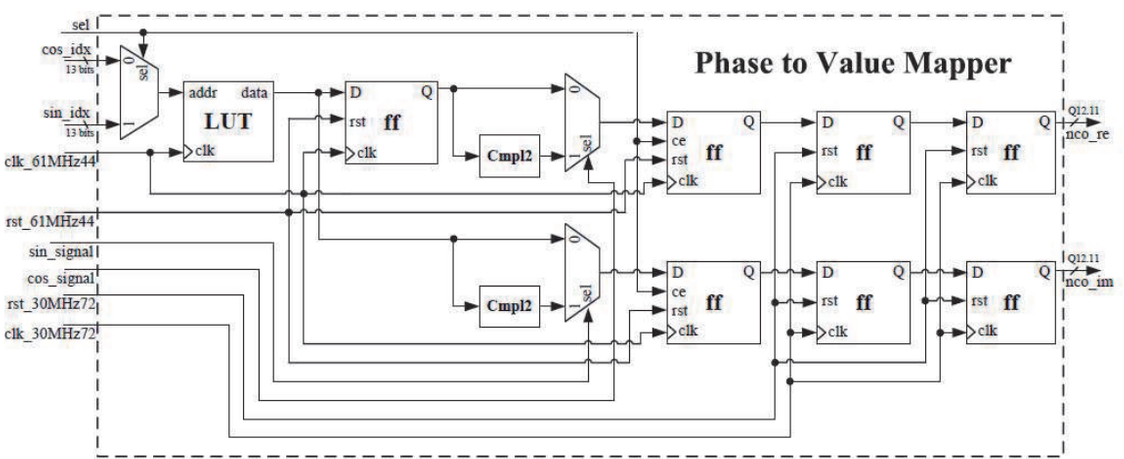

Figure 8.
Architecture of the phase to value mapper unit.

quadrants of the disc. Therefore, since the angle mapper module converts θ to the first quadrant of the disc, its databus width can be decreased to 13 bits, which is the number of bits used to access the $N/4$ samples of the first quadrant (i.e. quarter-wave symmetry) and that are saved in the LUT memory. This module runs at a clock rate of 61.44 MHz and outputs two new phase argument indexes, for the sine and cosine waveforms, at a clock rate of 30.72 MHz once the phase argument θ is sent to the module at that clock rate.

Figure 8 depicts the proposed architecture for the phase to value mapper module, which is in charge of translating values from phase space to time domain. The sine and cosine index values are employed as address values to access the correct positions of the LUT memory. It is the LUT memory that executes the translation from phase space to time domain. The LUT memory stores only 1/4, i.e. $N/4$, samples of the cosine waveform signal employed as the basis waveform signal. The *sel* signal selects whether the sine or cosine index value is employed to access the LUT memory. As it is shown in **Figure 8**, the cosine index value is employed as the address value when the *sel* signal is at low level and the sine index value is used when it is at a high level.

Both sine and cosine index values remain constant for a cycle of the clock rate of 30.72 MHz. Since the LUT memory works at a clock rate of 61.44 MHz and both index values are present at the same time at its input, it is possible to read two samples inside one period of the clock rate of 30.72 MHz. Through the creation of a delayed data path with the use of a register at the output of the LUT memory, it is possible to redirect the I and Q sample components to two distinct data paths and therefore generate the complex exponential sequence that is necessary to translate the received PRACH preamble sequence into baseband. At this stage, the resulting quadrature sequence signal is fully synchronised to the 61.44 MHz clock rate; however, each quadrature pair of values lasts for one period of the 30.72 MHz clock rate. This is explained by the two registers with the chip enable signal inputs set by the *sel* signal that is located at the output of the multiplexers responsible for changing the signal of the quadrature components.

In order to convert the quadrature sample values to their original quadrants, the sine and cosine signals created by the angle mapper module are applied to their respective data paths. When due, the change of signal is easily executed through the application of the complement of two operations to the sample value. Finally, it is necessary to change the clock domain of the complex exponential signal sequence since the ADC module works at a data rate of 30.72×10^6 samples per second. Even though its samples last for the correct period, they are not synchronised to the 30.72 MHz clock rate. The simplest way to execute the clock domain crossing is to use two different registers at the desired clock rate. As we work with complex signals, we employ a pair of dual registers for each one of the quadrature component values. At this stage, the resulting complex exponential sequence signal is totally ready to be multiplied by the received PRACH preamble signal.

4.3 Complex multiplier unit

Figure 9 shows the proposed architecture for the complex multiplier module. A complex multiplier is necessary to multiply the samples coming from both NCO and ADC modules and perform the required frequency shift in time domain, once samples coming from these modules are complex. The complex multiplier module, which is also known as mixer, executes the multiplication of the ADC samples, i.e. the received PRACH sequence signal, by the complex exponential signal created by

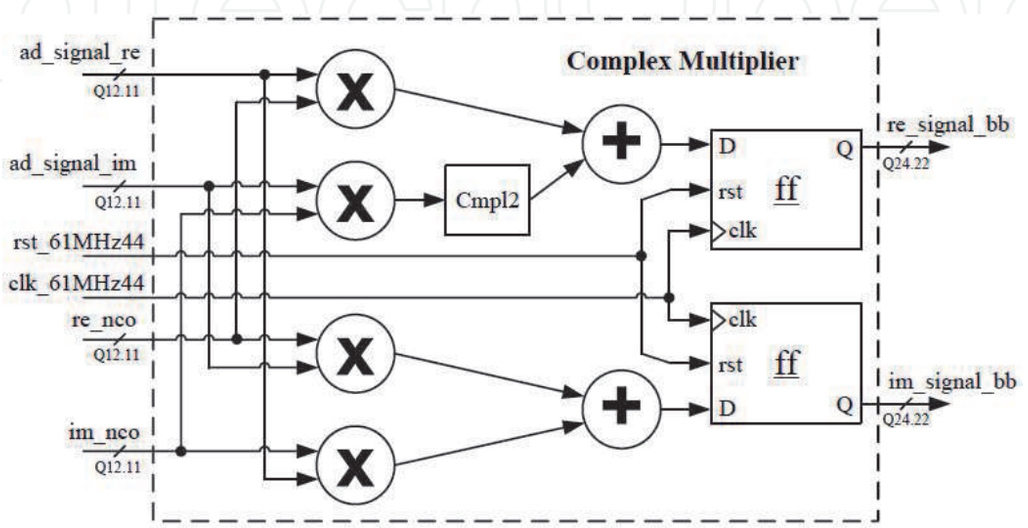

Figure 9.
Architecture of the complex multiplier unit.

the customised NCO module. The multiplication of these two complex values, i.e. $a + jb$ and $c + jd$, results in the complex product defined by Eq. (18):

$$\begin{aligned} real + j * imag &= (a + j * b) * (c + j * d) \\ &= (ac - bd) + j * (ad + bc). \end{aligned} \tag{18}$$

As can be noticed by analysing Eq. (18), the complex multiplication operation needs two additions and four multiplications since a subtraction operation is considered as being an addition in complement of two. The complex multiplier modules works at the clock rate of 30.72 MHz since it must always obey the data rate determined by the ADC module.

5. Implementation and simulation results

In order to assess the efficiency of the customised NCO and time-domain frequency shifter units proposed in this paper, some simulations were carried out. The proposed time-domain frequency shifter architecture was developed in VHSIC hardware description language (VHDL), and a corresponding bit-accurate Matlab model, referred here as Golden Model (GM), was developed for verification. The full design was targeted to a Xilinx Virtex-6 xc6vlx240t FPGA. The results presented next are split into parts: the first one provides the simulation results for the customised NCO architecture implementation, and the second part presents the results regarding the implementation of the time-domain frequency shifter architecture.

5.1 Customised numerically controlled oscillator

This section presents results regarding the customised NCO implementation. The first simulation result, shown in **Figure 10**, compares floating-point precision Matlab-generated complex exponential sequences with fixed-point precision sequences generated by the device under test (DUT) along all possible discrete

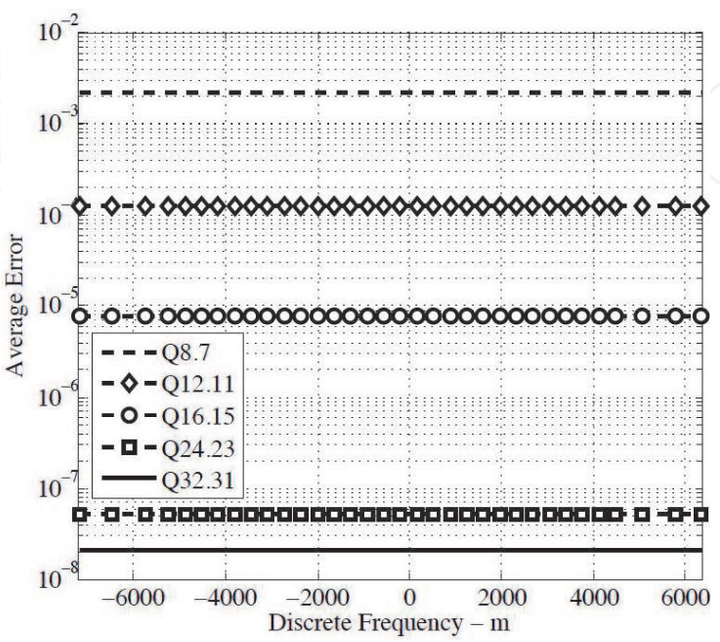

Figure 10.
Average error between GM and DUT implementations of the customised NCO.

frequency shifts, m, and for some Q-formats. PRACH format 0 preambles were considered for this and all other simulation results. **Figure 11** presents the SFDR variation of the implemented NCO unit, i.e. the DUT, along all possible discrete frequency shifts, m, and for some Q-formats. SFDR is the power ratio between the fundamental signal and the strongest spurious signal, i.e. the most prominent harmonic, present at the output of the customised NCO. By analysing this result, it is noticeable that the SFDR attained by the DUT is almost the same for formats Q24.23 and Q32.31. The SFDR values achieved by the DUT for formats Q24.23 and Q32.31 are 153.58 and 154.2 dB, respectively. These high SFDR values are due to the fact that the phase increment bits are not truncated and all output frequencies, f_{out}, are integer multiples of the system clock frequency, f_{clk}, which is the frequency used to sample the basis waveform, therefore eliminating the spectral artefacts resultant from phase jitter [15].

As an illustrative example of the performance presented by the customised NCO, **Figure 12** shows its power spectrum for some fixed-point formats with SFDR indication for frequency shift, m, equal to 7187, i.e. 8,983,750 Hz. These results clearly show the cleanliness achieved by the proposed customised NCO even for format Q8.7. The noise floor for format Q24.23 is so small that it is imperceptible.

Figure 13 depicts the SNR variation of the customised NCO along all possible discrete frequency shifts, m, and for some Q-formats. The SNR results are obtained by the ratio between the signal average power and the noise average power.

5.2 Time-domain frequency shifter

This section presents results regarding the implementation of the time-domain frequency shifter architecture. From this point on, we refer to the architecture implementation as the DUT. This time a Matlab floating-point model (GM) of the whole time-domain frequency shifter architecture is used to assess the performance of the circuit.

Table 4 presents information regarding the resource usage of the proposed architecture. It sums up the key results obtained after the implementation of the proposed frequency shifter architecture on a given FPGA chip. The number of

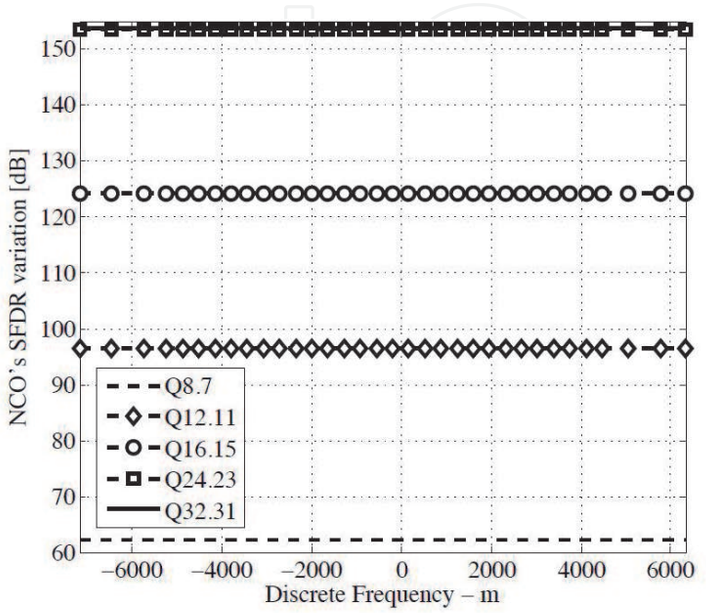

Figure 11.
SFDR variation of the customised NCO.

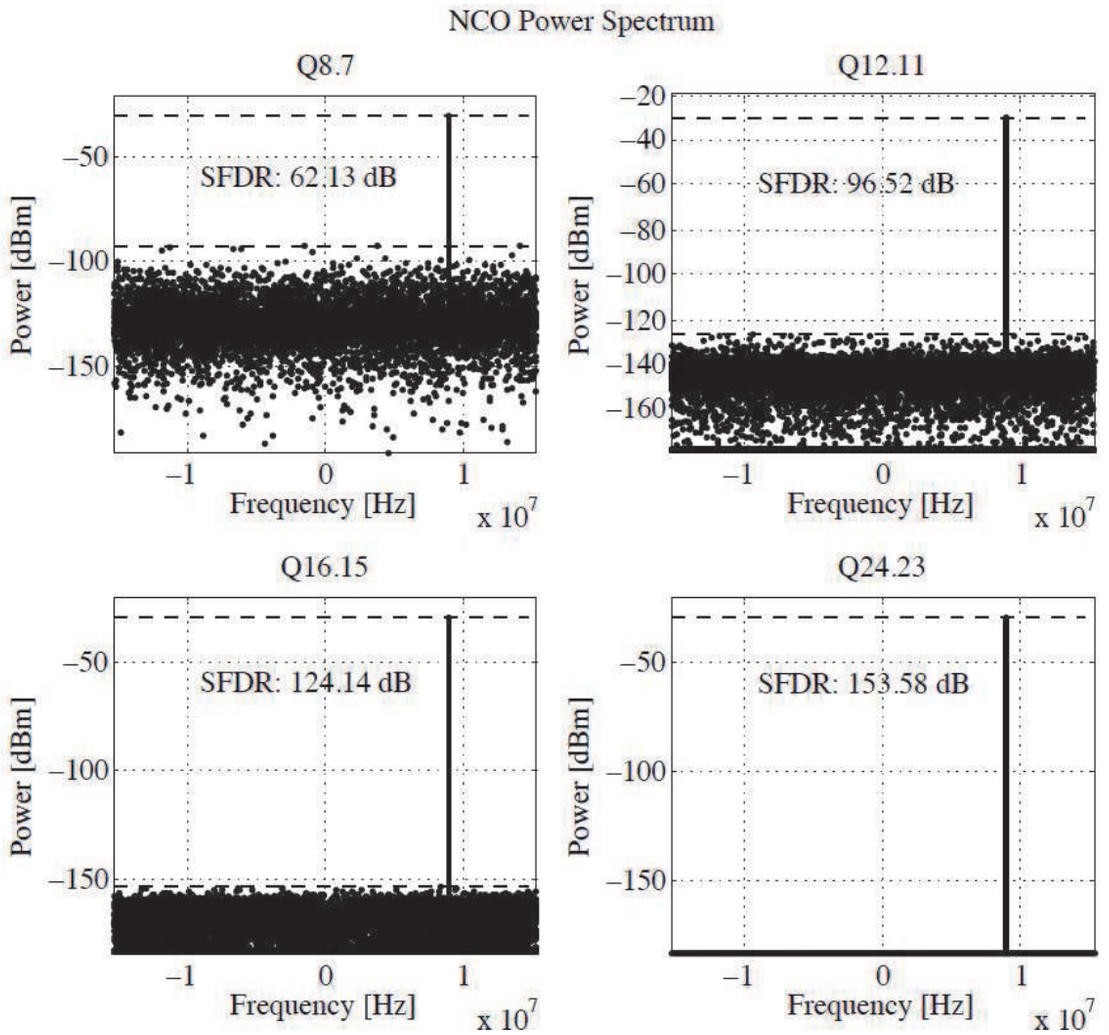

Figure 12.
Customised NCO power spectrum.

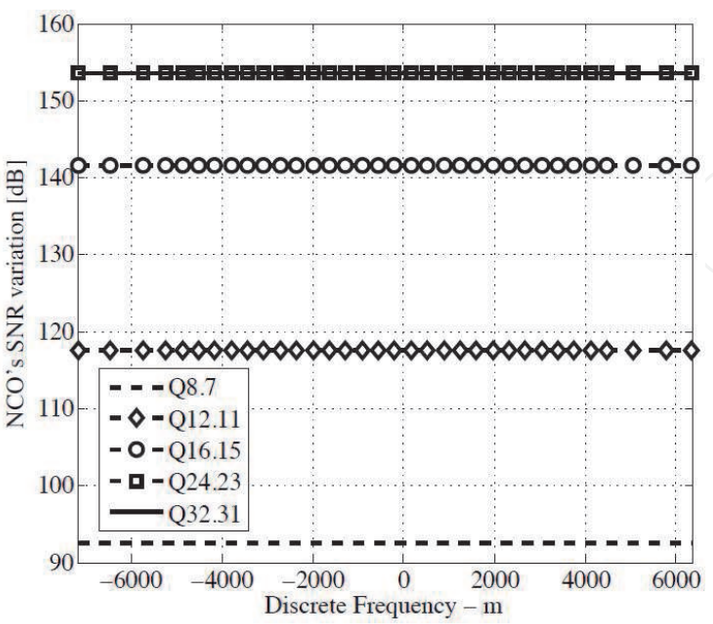

Figure 13.
SNR variation of the customised NCO.

occupied slices, registers, memory resources, LUTs and digital signal processor (DSP) resource blocks is shown in the table. The maximum achievable working frequency that can be reached by the module is equal to 239.981 MHz.

FPGA model number	XC6VLX240T-1ff1156 (Virtex-6)
Amount of slice registers	170 out of 301440 0%
Amount of slice LUTs	215 out of 150720 0%
Amount of occupied slices	84 out of 37,680 0%
Amount of RAMB36E1/FIFO36E1s	3 out of 416 0%
Amount of DSP48E1s	4 out of 768 0%
Maximum achievable frequency	239.981 MHz

Table 4.
Resource usage.

After observing the results presented in **Table 4**, we realise that three block RAM (BRAM) memory resources are employed instead of the two mentioned before. This is explained due to the fact that the synthesis tool maps all the contents of the LUT memory into three BRAM resources since the number of bits employed to address the LUT memory is equal to 13, and therefore, $\text{ceil}((2^{13} * 12)/36K) = 3$ instead of the two BRAMs mentioned earlier. Noticed that each address position of the BRAM resources saves a 12 bit value that is sampled from the basis cosine waveform. The four Xilinx DSP48 resources that are instantiated are used in the complex multiplier module to implement the multiplication operation defined in Eq. (18).

It is important to mention that in Virtex-6 family of FPGAs, one slice consists of eight flip-flops and four LUTs. Block RAMs and FIFO resources are embedded resources of the 36 bit memory resources. A DSP48 resource is an embedded processing unit that corresponds to one multiplier with two 18 bit inputs and one accumulator of 48 bits.

The reuse of DSP48 units is an important and feasible approach that is able to optimise the FPGA area utilisation at the expense of a higher clock rate operation and additional usage of control logical resources. For instance, the four fully parallel multiplications in the complex multiplier module could be serialised, which would save three out of the four DSP48 already being used.

After analysing **Table 4**, it is possible to notice that the proposed frequency shifter architecture uses less than 1% of all available Virtex-6 logical resources. Given that the actual implementation of the proposed architecture on FPGA presents a very low occupancy rate, the utilisation of low-cost FPGA models is possible. Therefore, there are two important points that must be taken into consideration when selecting a low-cost FPGA model: (i) the maximum achievable frequency operation, once low-cost FPGA models tend to present worse timing characteristics, and (ii) the number of used slices might increase in the case of families earlier than the Virtex-6 family, since other families may employ LUT memories with 4 bits instead of LUT memories with 6 bits per slice.

In **Figure 14**, the average error between the frequency shifted preambles generated by the GM and DUT is presented. In order to generate a representative result, the average error for a given RACH preamble is averaged over all possible offsets applied to that RACH preamble. In other words, the figure shows the average error over all possible offsets that can be applied to a given RACH preamble. Moreover, the figure presents the average error for several Q-formats when the PRACH bandwidth parameter, N_{RB}^{UL}, is made equal to 25 and 50 RBs, respectively. These bandwidth parameters correspond to bandwidths of 5 and 10 MHz, respectively. Additionally, the PRACH offset parameter, n_{PRB}^{RA}, is set to all possible values.

Figure 15 shows an exploded view of the results presented in **Figure 14**. Each subplot, representing the error for a specific Q-format, depicts the average of the

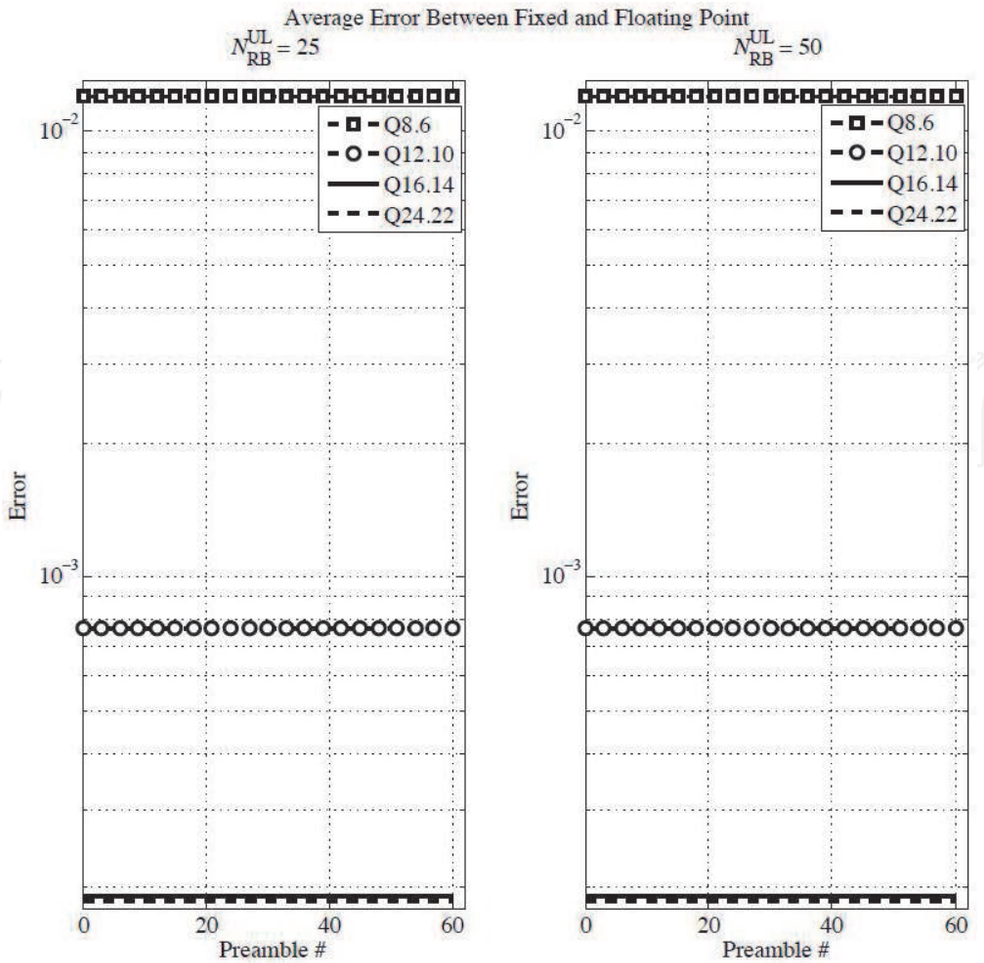

Figure 14.
Average error between GM and DUT.

average error over all possible offsets applied to a given preamble for 25 and 50 RB bandwidths. It is clearly seen that the error has a very small variation, almost constant, along the preambles.

Next we present results regarding the use of the time-domain frequency shifter architecture proposed in this work in the context of the PRACH receiver at eNodeB PHY side. The PRACH receiver architecture adopted in this work is shown in **Figure 2**, and a bit-accurate Matlab model was developed for its verification.

At the receiver side, the eNodeB attempts to detect a transmitted preamble by first extracting the PRACH signal from a received OFDM signal. The extraction involves applying downconversion, analog to digital conversion, CP removal, frequency shift, demapping and decimation to the received PRCAH signal. Next the receiver performs a matched filtering across the pool of preambles allocated to the eNodeB. Matched filtering is performed as a circular cross-correlation between the extracted PRACH signal and each of the known preambles dedicated to the eNodeB.

Figure 2 depicts the preamble detection module, which is the last block in the PRACH receiver processing flow. This module is responsible for detecting the transmission of RACH preambles at the PHY layer. This module employs the detection algorithm proposed in a previous work by the authors of [12]. All samples being received from the IFFT module have their squared modulus computed, then producing what is called as the power delay profile (PDP) samples. This module uses the PDP samples to (i) estimate a noise power value, which is performed by identifying PDP samples that can be regarded as containing only the presence of noise, and (ii) compute a RACH detection threshold. The RACH detection threshold

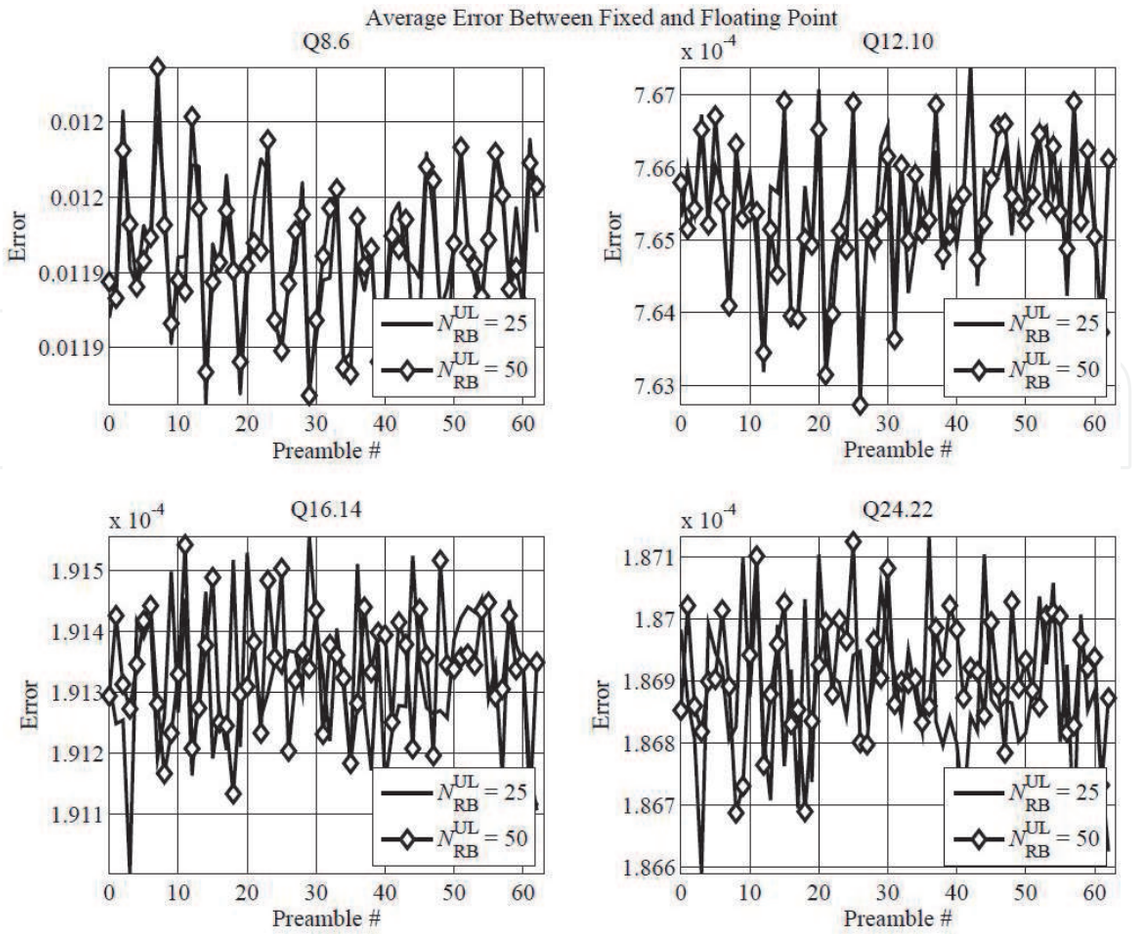

Figure 15.
Exploded view of the average error between GM and DUT.

is calculated based on the noise power estimate that minimises the probability of false alarms. Based on the RACH detection threshold, the PRACH receiver is able to decide whether a RACH preamble is present or not. The RACH detection module reports back to the MAC layer the timing offset estimates and IDs of all detected RACH preambles in a given reception interval. Interested readers are referred to [4, 16] for further details on the PARCH receiver architecture and RACH detection algorithm, respectively.

The bit-accurate PRACH receiver model adopted in this work includes the proposed time-domain frequency shift algorithm. In order to assess the performance of the proposed architecture, format 0 RACH preambles with N_{CS} = 13 and corrupted with additive white Gaussian noise (AWGN) were sent to the PRACH receiver model. When N_{CS} is equal to 13, all the 64 RACH preambles that are allocated to a given cell can be created out of a single root ZC sequence.

Through the execution of only one circular cross-correlation operation in the frequency domain between the noisy RACH preambles and the corresponding local root ZC sequence, the PRACH receiver is able to detect random access attempts by several UE devices [4]. The RACH detection process follows the algorithm presented in [12]. For the next results, the bit width of the output data path of the proposed architecture was set to 8, 12, 16 and 24 bits, resulting in the fixed-point representations Q8.6, Q12.10, Q16.14 and Q24.22, respectively.

Figures 16 and **17** present the complementary results of preamble detection when the time-domain frequency shifter is employed along with the preamble detection algorithm proposed in [12]. They depict comparisons between floating-

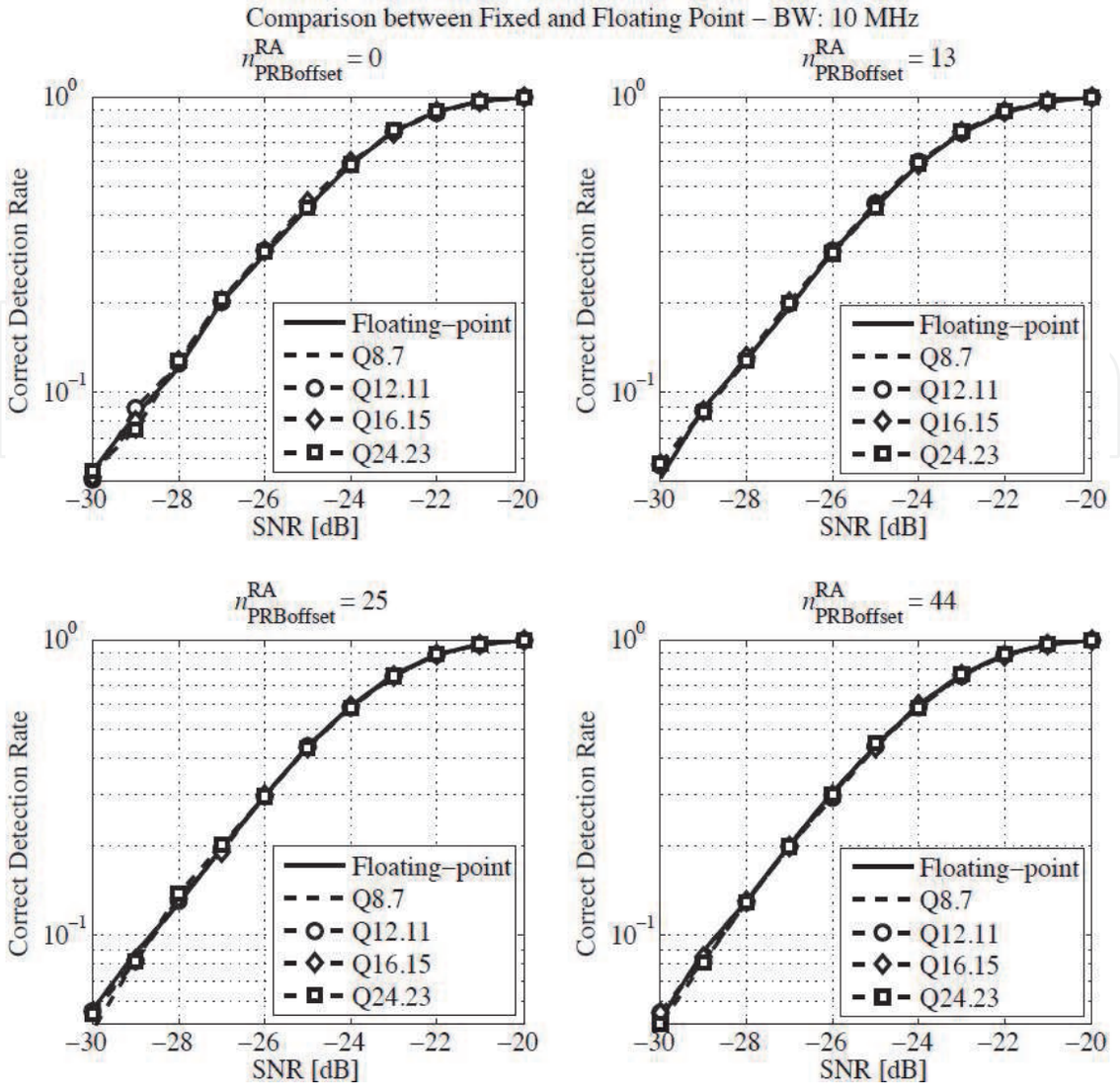

Figure 16.
Comparison of the correct detection rate between the bit-accurate and the floating-point model.

point and fixed-point detection results when bandwidth parameter, N_{RB}^{UL}, is set to 50 RBs (10 MHz).

Each subplot in **Figure 16** presents the comparison of the achieved correct detection rates versus signal-to-noise ratio (SNR) in dB for a given offset, n_{PRB}^{RA}. **Figure 17** presents the comparison of the achieved error detection rates versus SNR for a given offset. For both plots the probability of false alarm (P_{fa}) is made equal to 0.1%. The plots demonstrate the high accuracy of the proposed algorithm and corresponding architecture in translating the received PRACH preambles to baseband.

An important requirement for the PRACH receiver is that it must be capable of serving a huge number of UE devices per cell maintaining a reasonable detection probability and providing them with quasi-instantaneous access to the radio resources, while keeping the false alarm rate to low levels. The probability of a correct detection of the RACH preambles at the receiver side ought to be greater than or equal to 99% at an SNR of −8.0 dB, as defined in Section 8.3.4.1 of [17].

By analysing **Figure 16**, it is possible to see that the proposed algorithm achieves a probability of correct detection greater than 99% at a SNR of −21 dB, clearly outperforming [17] in 13 dB.

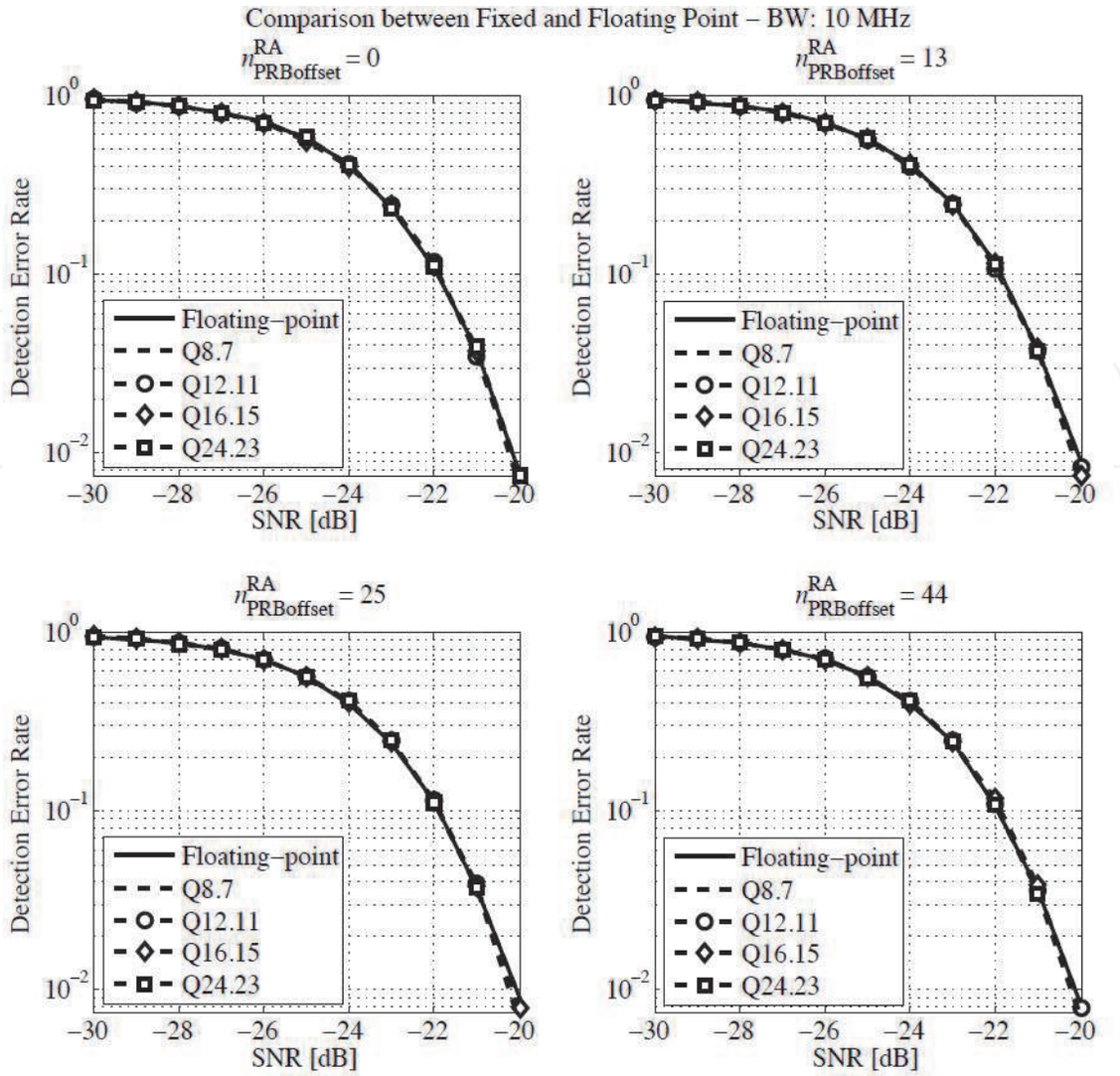

Figure 17.
Comparison of the error detection rate between the bit-accurate and the floating-point model.

6. Conclusions

A hardware-efficient algorithm and architecture for translating PRACH preambles into baseband featuring high-accuracy and low-complexity characteristics has been presented. This paper is an extension of a previous work where we only introduced some superficial aspects of the proposed hardware architecture. In this paper we present theoretical derivations showing how to arrive at the equations used to design and implement the proposed architecture. We provide the pseudocode for the proposed algorithm and discuss the advantage of the proposed architecture in terms of memory utilisation when comparing our proposed solution with an approach where the full period of the complex exponential is stored in FPGA memory. The proposed architecture is optimised to shrink the use of BRAMs, multipliers and logical resources. The low resource utilisation exhibited by the proposed architecture demonstrates its feasibility to be employed as part of large physical layer designs or to be used in FPGAs with small amount of logical elements.

The 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. Finally, this paper provides detailed information on the architectural design that was tested on an FPGA device for real-time LTE applications.

Author details

Felipe A.P. de Figueiredo[1,2*] and Fabbryccio A.C.M. Cardoso[3]

1 Instituto Nacional de Telecomunicações—INATEL, Santa Rita do Sapucaí, MG, Brazil

2 Department of Information Technology, IDLab, Ghent University—imec, Ghent, Belgium

3 CPqD—Research and Development Center on Telecommunication, Campinas, SP, Brazil

*Address all correspondence to: felipe.figueiredo@inatel.br

References

[1] Zarrinkoub H. Overview of the LTE Physical Layer. Hoboken, New Jersey, USA: John Wiley & Sons; 2014

[2] Kanchi S, Sandilya S, Bhosale D, Pitkar A, Gondhalekar M. Overview of LTE-A technology. In: IEEE Global High Tech Congress on Electronics (GHTCE). 2014

[3] Rumney M. LTE and the Evolution to 4G Wireless: Design and Measurement Challenges. Hoboken, New Jersey, USA: Wiley; 2013

[4] Sesia S, Toufik I, Baker M. LTE—The UMTS Long Term Evolution: From Theory to Practice. Hoboken, New Jersey, USA: John Wiley & Sons; 2011

[5] de Figueiredo FAP, Mathilde FS, Cardoso FACM, Vilela RM, Miranda JP. Efficient FPGA-based implementation of a CAZAC sequence generator for 3GPP LTE. In: IEEE International Conference on Re-ConFigurable Computing and FPGAs (ReConFig 14). 2014

[6] de Andrade TPC, Astudillo CA, Sekijima LR, da Fonseca NLS. The random access procedure in long term evolution networks for the internet of things. IEEE Communications Magazine. 2017;**55**(3):124-131

[7] 3GPP TS 36.211. Physical Channels and Modulation (Release 10). 2009

[8] Figueiredo FAP, Mathilde FS, Figueiredo FL, Cardoso FACM. An FPGA-based time-domain frequency shifter with application to LTE and LTE-A systems. In: IEEE Latin American Symposium on Circuits & Systems (LASCAS). 2015

[9] Frank RL, Zadoff SA, Heimiller R. Phase shift pulse codes with good periodic correlation properties. IRE Transactions on Information Theory. 1961;**7**:254-257

[10] Chu DC. Polyphase codes with good periodic correlation properties. IEEE Transactions on Information Theory. 1972;**18**(4):531-532

[11] Yang X, Fapojuwo AO. Enhanced preamble detection for PRACH in LTE. In: IEEE Wireless Communications and Networking Conference (WCNC). 2013

[12] de Figueiredo FAP et al. Multi-stage based cross-correlation peak detection for LTE random access preambles. Revista Telecomunicações. 2013;**15**:1-7

[13] Ranabhatt NA, Agarwal S, Bhattar RK, Gandhi PP. Design and implementation of numerical controlled oscillator on FPGA. In: International Conference on Wireless and Optical Communications Networks (WOCN). 2013

[14] Kadam S, Sasidaran D, Awawdeh A, Johnson L, Soderstrand M. Comparison of various numerically controlled oscillators. In: Midwest Symposium on Circuits and Systems (MWSCAS). 2002

[15] Xilinx. DS246—DDS Logic Core Product Specification—v5. 2005

[16] de Figueiredo FAP, Cardoso FACM, Lenzi KG, Bianco Filho JA, Figueiredo FL. A modified ca-cfar method for lte random access detection. In: 7th International Conference on Signal Processing and Communication Systems (ICSPCS). 2013. pp. 1-6

[17] 3GPP TS 36.104. Base Station (BS) radio transmission and reception (Release 10). 2007