IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 22, 2021, accepted January 10, 2022, date of publication January 14, 2022, date of current version January 25, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3143808

Real-Valued Spreading Sequences for PSSS Based High-Speed Wireless Systems

LUKASZ LOPACINSKI[1], NEBOJSA MALETIC[1], (Member, IEEE), GORAN PANIC[1], ALIREZA HASANI[1], JESÚS GUTIÉRREZ[1], AND ECKHARD GRASS[1,2]

[1]IHP Leibniz-Institut für innovative Mikroelektronik, 15236 Frankfurt (Oder), Germany
[2]Department of Computer Science, Humboldt-Universität zu Berlin, 10117 Berlin, Germany

Corresponding author: Lukasz Lopacinski (lopacinski@ihp-microelectronics.com)

This work was supported in part by the German Research Foundation (DFG), project PSSS-FEC, under Project 442607813.

ABSTRACT In past years, Parallel Sequence Spread Spectrum (PSSS) has attracted significant attention as a modulation technique for wireless communication systems targeting data rates of 100 Gb/s and beyond. PSSS allows designing high-speed baseband processors, which can be partially implemented in the analog domain. It uses multiple analog-to-digital converters (ADCs) to sample the received baseband signal in parallel, significantly relaxing the sampling rate and ADC complexity. However, due to the sidelobe effects of bipolar *m*-sequences, PSSS shows lower performance than standard digital modulation schemes. This paper proposes real-valued PSSS spreading sequences with attenuated autocorrelation sidelobes. Such sequences show excellent bit error rate (BER) performance. Moreover, our sequences do not have length restrictions of $2^m - 1$, like in the case of *m*-sequences, and reduce the chip area required to implement PSSS transceiver. The proposed sequences also reduce the peak-to-average power ratio (PAPR) of PSSS.

INDEX TERMS PSSS, high-speed wireless communications, baseband, genetic algorithm, spreading sequences, sidelobe mitigation.

I. INTRODUCTION AND MOTIVATION

The increase of wireless data rates in the last decade drives the need for new baseband processing solutions for the next generation of ultra-high-speed wireless communications targeting 100 Gbps and beyond. Due to the high bandwidth available above 200 GHz, as depicted in Fig. 1, it is possible to provide channels exceeding 50 GHz of continuous bandwidth, which allow for very high data rates, assuming uncomplicated modulation schemes [1]–[3]. One of the novel modulation techniques that could be considered for such wide radio frequency (RF) channels is the so-called Parallel Sequence Spread Spectrum (PSSS) [4]–[6], which considerably simplifies transmitter and receiver baseband design due to its inherent parallelism and analog-friendly architecture [7]. The main advantage of PSSS is a lightweight baseband implementation that allows to sample and process the baseband signal in parallel ADC structures and baseband threads (Fig. 2) [7]. This feature is especially important for future high-speed wireless channels located in the THz band, where the ADCs and DACs need to process large bandwidth. Instead of employing a pair of I-Q ADCs like in conventional systems with QAM modulation, the received signal can be sampled by N ADCs, where each ADC instance operates on N-times reduced clock frequency. Moreover, a large portion of PSSS processing can be performed directly in the analog domain due to PSSS's analog-friendly architecture. The initial PSSS processing on the receiver side consists of correlation and integration, which can be easily done in analog hardware. Thus, the PSSS baseband is a kind of mixed-signal implementation.

Wireless communication with PSSS modulation has been practically demonstrated already some years ago. In [8] and [9], the authors have implemented a hardware node for industrial wireless communication based on 5.8 GHz PHY with a 20 MHz channel. In [10], a similar module for IEEE 802.15.4-2006 is presented. The peak data rate is 250 kbps, with the system operating in 2.4 GHz. A significantly higher data rate is reported in [11], where the authors perform a hardware-in-the-loop experiment achieving 80 Gbps with analog frontends at 230 GHz. The Digital Signal Processing (DSP) pre- and post-processing are performed in Matlab and Simulink. PSSS is also deployed in orthogonal frequency-division multiplexing (OFDM) systems [12], is a part of

The associate editor coordinating the review of this manuscript and approving it for publication was Yan Huo.

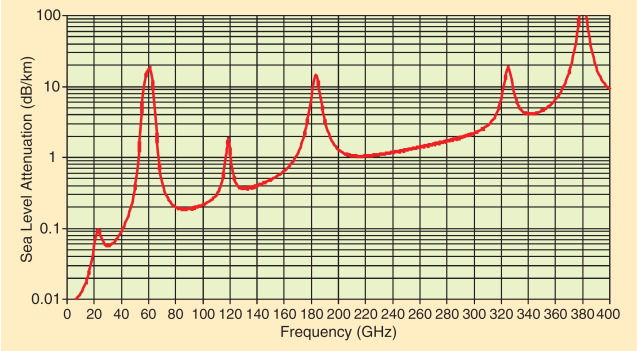


FIGURE 1. Average millimeter- and THz-wave atmospheric absorption. Figure retrieved from [1].

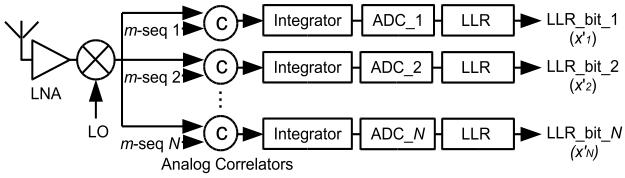


FIGURE 2. PSSS receiver with 1b/s/Hz. Figure adopted from [7].

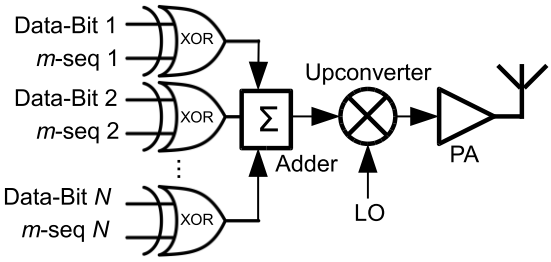


FIGURE 3. The simplest PSSS transmitter with spectral efficiency of 1b/s/Hz.

the IEEE 802.15.4-2006 standard [13], and was targeted for universal serial bus (USB) communication [14].

The main drawback of the PSSS variant used by us is the correlation DC-offset caused by the bipolar spreading sequences' sidelobes, which significantly reduce the BER performance. The mentioned correlation DC-offset caused by the standard *m*-sequences employed for PSSS leads to ∼2.5 dB EbN0 penalty [15]. In the case of PSSS, the orthogonal Walsh–Hadamard sequences cannot be used to transfer the data streams. PSSS needs sequences with good cyclic-autocorrelation properties due to its cyclic nature and orthogonalization methods proposed in [16] and [17].

This paper proposes yet another method for mitigating sidelobes in PSSS. Using a genetic algorithm, we construct new real-valued spreading sequences, defined in $\mathbb{R}$ domain, with attenuated autocorrelation sidelobes. Due to removed sidelobes, the improved PSSS scheme shows excellent BER performance and approaches QPSK and QAM modulations in AWGN. To the best of our knowledge, such a method has never been employed for generating PSSS sequences [5]–[7], [16]–[19]. Our algorithm has five main advantages when compared to the state of the art PSSS schemes:

- In contrast to the solutions shown in [4], [15], and [18], our method does not introduce any EbN0 penalty for BER performance. The resulting sequences achieve the highest BER performance and approach standard modulation schemes.
- Our idea is simpler to implement than solutions based on matrix factorization proposed in [16].
- The proposed genetic algorithm generates sequences with adjustable lengths ranging from 3 to 30 chips, which has not yet been demonstrated for PSSS in any paper. This allows designing PSSS transceivers with a more flexible hardware architecture. This is especially interesting for analog implementations considered in [7] and [20]–[22], where the *m*-sequences are problematic due to the integrator complexity. Using our sequences, the designer can specify the PSSS sequence length in 1 chip (bit) precision. The competitive *m*-sequences are available only in the lengths of 7, 15, 31, . . . , and follow the 2^m-1 length restrictions.
- PSSS systems based on the resulting real-valued sequences require up to 40% less ASIC area than PSSS combined with two external matched filters. External matched filters are one of the possible high-speed hardware realizations of PSSS orthogonalization proposed in [16] (more in sections II.D and II.E).
- The proposed real-valued sequences reduce the resulting PAPR of the PSSS transmission. In our case, we can reduce the PAPR up to 38% without any penalty in BER and EbN0.

The organization of the paper is as follows. Section II introduces the PSSS and gives a quick overview of PSSS literature. Section III presents our algorithm for PSSS sequences generation. The achieved results, and their analysis together with a comparison with state-of-the-art, are provided in section IV. The final conclusions are made in section V.

II. PSSS PHYSICAL LAYER

In this section, we introduce PSSS modulation, discuss the problem of autocorrelation sidelobes, propose our solution to mitigate these sidelobes, and compare it to the solutions from the literature.

A. PSSS BASICS

Figures 2 and 3 depict a wireless PSSS receiver and transmitter in the most straightforward, high-speed architecture with 1 bit/s/Hz spectral efficiency [7], [19]. PSSS can be defined over different spreading sequence lengths, and usually, $15 \leq N \leq 255$ [7], [9], where N denotes the used spreading code length. We target the shortest sequences ($3 \leq N \leq 30$), which ease the hardware representation, mainly the analog receiver's integrator. In [7] and [20]–[22], the N parameter is set to 15 and corresponds to *m*-sequences with 15 chips.

Due to hardware limitations explained in [7] and [20]–[22], we investigate spreading sequences with similar lengths, to be sure that analog PSSS realizations are possible.

In our scheme, the input user bits, denoted by $\mathbf{x}$:

$$\mathbf{x} = [x_1, x_2, x_3, \ldots, x_N], \quad x_k \in \{-1, +1\}, \tag{1}$$

are multiplied with cyclically shifted real-valued spreading sequences, represented as:

$$s = [s_1, s_2, s_3, \ldots, s_N], \quad s_k \in \mathbb{R}, \ -1 \leq s_k \leq +1, \tag{2}$$

and then added in the time domain resulting in transmitter baseband signal:

$$\mathbf{t} = \mathbf{xM}, \tag{3}$$

where $\mathbf{M}$ (spread matrix) is defined as:

$$\mathbf{M} = \begin{bmatrix} s_1 & s_2 & s_3 & \ldots & s_N \\ s_2 & s_3 & \cdots & s_N & s_1 \\ s_3 & \cdots & s_N & s_1 & s_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ s_{N-1} & s_N & s_1 & \cdots & s_{N-2} \\ s_N & s_1 & s_2 & \cdots & s_{N-1} \end{bmatrix}. \tag{4}$$

Accordingly, the multivalent elements of $\mathbf{t}$ in (3) are:

$$\mathbf{t} = [t_1, t_2, t_3, \ldots, t_N], \quad t_k \in \mathbb{R}, \ -N \leq t_k \leq +N,$$
$$t_k = \sum_{i=1}^{n} s_{(k+i-1) \bmod N} x_i. \tag{5}$$

At the receiver side, the received baseband sequence becomes $\mathbf{r}$, defined by:

$$\mathbf{r} = \mathbf{t} + \mathbf{n}, \tag{6}$$

where $\mathbf{n}$ vector denotes channel noise originated from AWGN or Rayleigh channel. The receiver correlates the $\mathbf{r}$ signal with spreading sequences stored in the spreading matrix $\mathbf{M}$ by performing:

$$\mathbf{c} = \mathbf{rM}. \tag{7}$$

The received bit values $\hat{\boldsymbol{x}}$ are estimated by checking the sign of the correlation elements of $\mathbf{c}$ in (7) (hard decision detection):

$$\hat{\boldsymbol{x}} = \text{sign}(\mathbf{c}),$$
$$\hat{x}_k = \begin{cases} +1, & \text{c}(k) > 0 \\ -1, & \text{c}(k) \leq 0, \end{cases} \quad 1 \leq k \leq N. \tag{8}$$

The equivalent receiving circuit can be realized, as shown in Fig. 2. The analog correlators and integrators in Fig. 2 are equivalent to the vector by matrix multiplication in (7). Digital implementations of similar PSSS systems in FPGA are demonstrated in [5] and [9]. A high-speed hard-decision detection can also be realized by replacing the ADCs and LLR estimators (Fig. 2) with binary comparators. PSSS allows processing the baseband signal in an analog circuit and sampling the soft bit values in a parallel ADC array (Fig. 2). In our case, we use up to 30 ADCs, and therefore the conversion speed can be reduced by 30 times ($N \leq 30$).

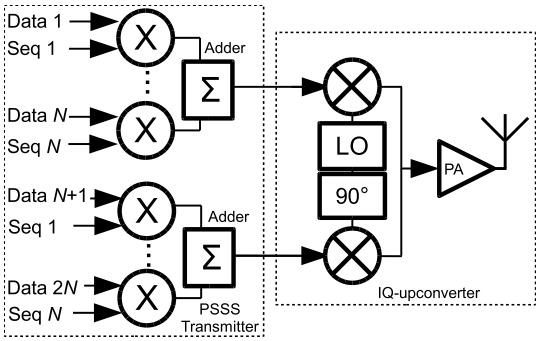


FIGURE 4. An example PSSS modulator with an IQ-upconverter and spectral eff. of 2 b/s/Hz.

This reduction is the main reason for selecting PSSS for high-speed wireless communication at 100 Gbps.

The PSSS algorithm investigated in this paper is slightly different from the original method published in 2004 [5], [6]. In our case, we use real-valued sequences on the transmitter and receiver sides in (3) and (7), while the standard PSSS method is derived for a combination of unipolar and bipolar *m*-sequences defined in the binary domain. In the standard PSSS, the transmitter uses the unipolar $\{0, +1\}$ *m*-sequence for encoding, while the receiver correlates using the same *m*-sequence in the bipolar $\{-1, +1\}$ representation. For example, sequence '1001110' is used in the transmitter, and the receiver uses '+1−1−1+1+1+1−1.' For the transmitter and receiver, the data is encoded as $\{-1, +1\}$. In our variant, we use PSSS dedicated sequences defined in $\mathbb{R}$ domain with values in the range $[-1, +1]$ for the transmitter and receiver (identical sequences are used on both sides). These sequences are the critical element of our scheme and are specially generated for the PSSS system defined in (1)-(8).

B. SYSTEM MODEL WITH 2 b/s/Hz SPECTRAL EFFICIENCY

This subsection shows the investigated PSSS model with an IQ-upconverter (Fig. 4). The multiplication and add functionality shown on the left side in Fig. 4 is the equivalent of a vector by matrix multiplication defined in (3). The right side of Fig. 4 is a standard IQ-upconverter, which is identical to QPSK and QAM upconverters. In this paper, we target a carrier frequency of 240 GHz [23]. Fig. 5 depicts BER performance with 2 b/s/Hz spectral efficiency for the presented system. All curves are generated by typically used *m*-sequences [5]–[7], [16]–[19]. PSSS, as defined in (1)–(8) with bipolar *m*-sequences used as encoding and decoding spreading codes (blue curve in Fig. 5), shows disappointing BER performance compared to QPSK and the standard PSSS method [5], [6] (red-dashed curve in Fig. 5). In the next subsection, we prove that autocorrelation sidelobes cause the poor BER performance. To the best of our knowledge, only three publications address and solve this issue [15], [16], [18]. The authors in [18] propose an iterative symbol detection algorithm in the receiver. This solution causes bit detection dependencies, leading to hardware complexity, especially

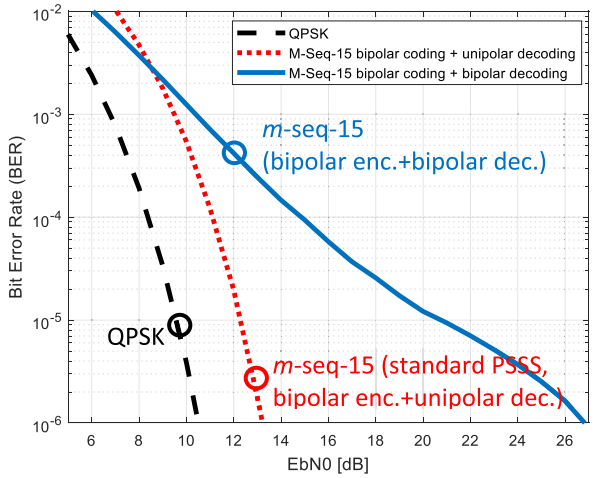


FIGURE 5. AWGN BER performance for 2 b/s/Hz PSSS.

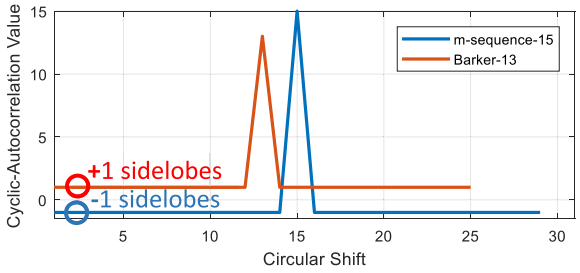


FIGURE 6. Autocorrelation functions of bipolar Barker-13 and *m*-sequence-15.

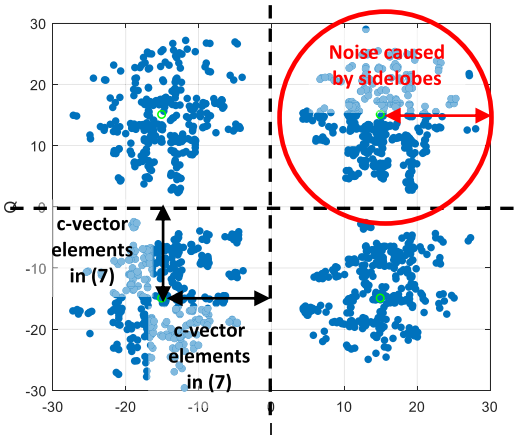


FIGURE 7. PSSS (1)-(8) constellation for random data in AWGN at SNR = 30 dB.

when analog correlators and integrators at ≥100 Gbps are targeted [20], [22]. In [15], a method based on DC-level correction for binary Barker codes is presented. Authors in [16] propose a matched filter technique based on matrix factorization to eliminate the sidelobes of bipolar *m*-sequences. Currently, this technique shows the highest BER performance and is the most promising solution for PSSS.

C. AUTOCORRELATION SIDELOBES IN BASIC PSSS SCHEME

The *m*-sequence of length 15 (denoted as *m*-sequence-15) has sidelobes of value −1 for each circular shift (Fig. 6). These sidelobes strongly influence the DC levels at the outputs of the correlators ('C'-elements in Fig. 2) by adding an individual DC offset for each PSSS symbol in (8). Assuming that the data bits in (1) follow a uniform distribution, and have a similar count of −1's and +1's, the sidelobes with positive and negative amplitude accumulate, and therefore are canceled. In such a case, the PSSS decoder with bipolar codes can operate as described in (1)-(8). If the count of positive and negative bits in user data is not balanced, and the count of positive or negative bits is increased, the signal-to-noise ratio (SNR) is reduced. When PSSS symbol data in (1) is dominated by −1's or +1's, the decoder cannot extract data bits from the symbol in (8). The sidelobes interfere constructively and cause a strong DC offset in the correlators. Such a demodulator breaks down and cannot work. To demonstrate this effect, we show an IQ-constellation in an AWGN channel at SNR = 30 dB, i.e., the signal power is 1000 times higher than the noise power (Fig. 7). The distance between the constellation points is defined by PSSS correlation values (elements of $\mathbf{c}$ in (7)). Although the constellation is similar to QPSK, the QPSK mapper and demapper are not used. Instead, PSSS with 2 b/s/Hz, as shown in Fig. 4 and (1)-(8), is employed for modulation. The noise marked in the red circles in Fig. 7 is the main reason for the poor BER performance of bipolar PSSS. A different DC offset influences each PSSS symbol in (8), and therefore, we observe four "clouds" in the constellation. To overcome this problem, we use real-valued spreading sequences with attenuated autocorrelation sidelobes. To generate these sequences, a genetic algorithm is used.

D. SIDELOBE MITIGATION ALGORITHMS

Until the work in [15]–[18] was not released, PSSS had problems achieving the BER performance of standard modulation systems due to the sidelobes and the resulting DC-offsets. The authors in [18] propose an iterative detection algorithm for PSSS, and this is the first published solution that closes the performance gap to the standard BPSK, QPSK, and QAM systems. Unfortunately, the iterative detection algorithm shows residual error floor for some sequences. Its implementation also increases PSSS receiver complexity, especially for analog receiver realizations. Authors in [16] and [17] propose a more robust algorithm to orthogonalize the PSSS. This solution shows excellent performance when applied for *m*-sequences, but requires matrix decomposition methods. The key idea is to factorize a circulant matrix initialized with cyclically shifted *m*-sequences (since the matrix is circulant, it is diagonalizable by the DFT matrix).

Authors in [15] show a trivial method to improve PSSS BER performance for the PSSS algorithm combined with Barker spreading codes. This is the simplest and most straightforward solution, but shows a constant ∼0.2 dB EbN0

penalty compared to matrix factorization proposed in [16] and [17]. The main advantage of this method is its low complexity and straightforward hardware realization. Some of the previously mentioned methods are explained and compared in [24].

E. SIMILARITIES BETWEEN MATCHED FILTERS AND REAL-VALUED SEQUENCES

The authors in [16] orthogonalize the circulant matrix **M** in (4), which is initialized with binary *m*-sequences, to remove the constant cyclic-autocorrelation sidelobes caused by the sequences. In this paper, we employ fully orthogonal real-valued sequences for **M** in (4). Our sequences do not have sidelobes and are generated primarily for PSSS. In some cases, both approaches might lead to the same values in **M**, but the development of these two ideas is entirely independent. In both solutions, the values of **M** are manipulated so that all N rows in **M** are entirely orthogonal to each other. The methods have different advantages but allow achieving identical BER performance, which is ∼2.5 dB higher than for the standard PSSS [4] with binary *m*-sequences. The matched filters orthogonalization focuses on **M** matrix factorization. At the same time, our method uses the idea of finding spreading sequences by a heuristic procedure proposed in 1974 in [25], with the difference that we employ more powerful genetic algorithms that have become widespread some years later. We also search the codes in an application-oriented approach, because we focus on cyclic-autocorrelation and PAPR properties, not standard linear autocorrelation.

In general, the authors in [16] propose three orthogonalization methods for *m*-sequences. In this paper, we selected the solution based on external matched filters due to its hardware-friendly CMOS implementation. The PSSS variant based on matched filters consists of PSSS transmitter, transmitter filter, receiver filter, and PSSS receiver. Although the authors do not mention it in their paper, this architecture has one crucial practical advantage. Their scheme's DSP comprises four modules. PSSS transmitter and receiver work in the binary domain, and their architectures are effortless (achieve very high throughput). The effort to reject sidelobes is divided among two filters. Due to the matrix by vector implementation in these filters, the DSP critical path is split into concise sections of similar delay. This allows high clock frequencies and throughput in digital CMOS realizations due to its pipelined nature. Moreover, the filter logic consists only of two coefficient values, due to the repetitive nature of the K_{OPSSS} matrix in [16]. Therefore, its implementation can be realized with minimal energy overhead. In the case of *m*-sequence-15, 87% of the filter logic does not update its values and remains idle due to the repetitive multiplication coefficients. Although the authors of [16] have overlooked all these significant advantages, the architectural benefits of their matched filter scheme are visible after performing VHDL optimizations on the RTL level.

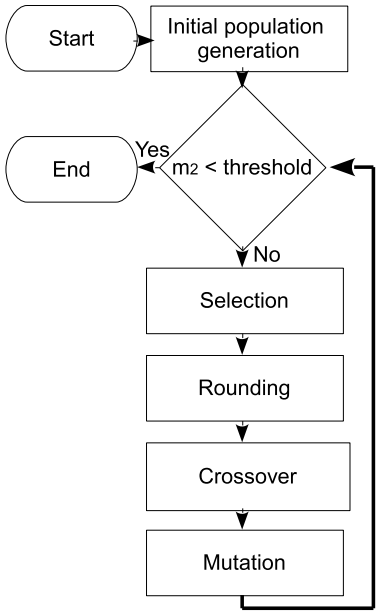


FIGURE 8. Flow chart of the genetic algorithm.

We implement and compare our real-valued sequences with the matched filters in an FPGA and ASIC technology in section IV.E.

III. WORK DETAILS

A. GENETIC ALGORITHM-BASED PSSS SEQUENCES

The best BER performance is achieved when a PSSS sequence with cyclic-autocorrelation sidelobes ≈0 is selected. We search for sequences with possible low sidelobes and possible high autocorrelation peaks using a genetic algorithm prototyped in Matlab (Fig. 8). This algorithm is a six-stage procedure that starts with randomly generated sequences. These sequences are called the ''initial population.'' In the selection stage (Fig. 8), the sequences with favorable cyclic-autocorrelation properties are selected for later processing. All sequences with weak cyclic-autocorrelation are deleted and replaced with new sequences called ''offspring generation.'' The offspring sequences are derived as a modification of the parent sequences with the highest cyclic-autocorrelation in the ''rounding,'' ''crossover,'' and ''mutation'' stages (Fig. 8). These recombination schemes are performed with the expectation that one of the newly created sequences may have better cyclic-autocorrelation properties than the parent sequences. After that, the procedure is repeated until at least one sequence achieves the targeted correlation properties. Termination criterion, in our case, is to have the sidelobe amplitudes lower than the predefined threshold $t = 0.001$ (more in sections III.C and III.D). This algorithm is iteratively repeated at least 10^4 times and has to be executed only once. When the desired sequences are found, the coefficients are saved, and there is no need to repeat the procedure. Therefore, a precise algorithm's optimization can be avoided, and only coarse algorithm

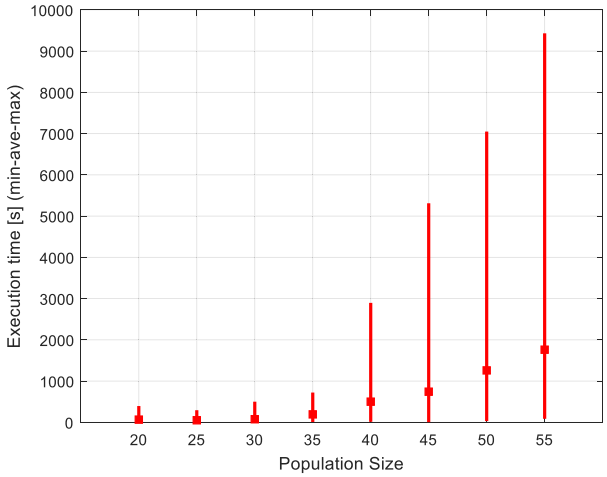


FIGURE 9. Population size impact on the genetic algorithm's execution time for $N = 10$.

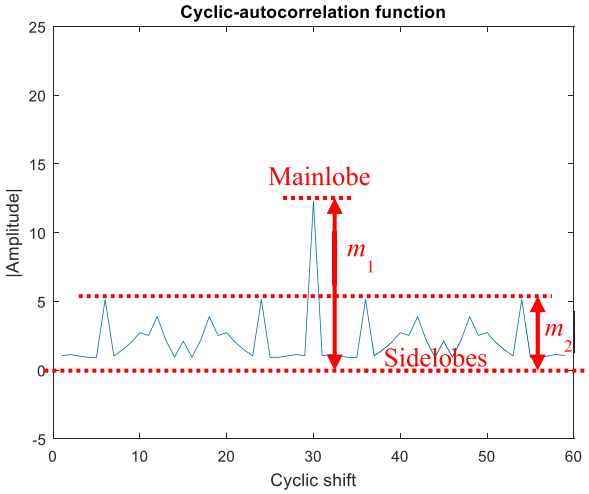


FIGURE 10. Justification of the fitness function f defined in (11) for a random sequence with $N = 30$.

adjustment is required to achieve the termination criterion in a deterministic time.

B. POPULATION GENERATION AND CHROMOSOME CODING

The initial population is generated randomly by uniformly distributed binary random numbers in $\{-1, +1\}$. Each chromosome belonging to the population consists of $3 \leq N \leq 30$ real numbers (genes). The number of genes equals to the targeted spreading sequences length (N) and directly represents the $s_1, \ldots, s_N$ elements of $\mathbf{s}$ in (2). The algorithm has been prototyped with an arbitrary selected population size of 30 individuals, typical for similar implementations [26]. After that, the fitness function, mutation scheme, and recombination strategies have been investigated and optimized for the initial population attributes (size, encoding, starting values). Thus, changing the starting population parameters requires modifications in other algorithm sections. The algorithm shows good convergence when the population size is in a range of 25-35 individuals, and it is recommended to keep this boundary. Fig. 9 depicts the relation between the population size and execution time for finding sequences with $N = 10$.

The initially generated population should consist of $\{-1, +1\}$ binary numbers only. If the algorithm starts with random $\mathbb{R}$ values in the range of $[-1, +1]$, then the average execution time is extended by 29%. Moreover, the data type representation for the genes should be set to double-precision floating-point numbers. Single precision data type increases the average execution time by 18%. In general, the algorithm is susceptible to changes, and it is recommended to keep all the given boundaries. Otherwise, the recombination and mutation methods may not work efficiently. Most algorithm parameters were determined experimentally, and were prototyped in a Matlab implementation.

C. FITNESS FUNCTION

We use two values to assess each individual s in (2) in the population. Firstly, we compute the maximum of each individual's cyclic-autocorrelation function in (9).

$$m_1 = \max(|\mathbf{sM}|), \tag{9}$$

$$m_2 = \max_2(|\mathbf{sM}|), \tag{10}$$

$$f = m_1 - \alpha m_2 - p + g_{\mathrm{PAPR}}, \tag{11}$$

$$\alpha = 0.3N. \tag{12}$$

This value is symbolized by m_1 in Fig. 10 and represents the amplitude of the cyclic-autocorrelation mainlobe. The m_1 is obtained by finding the maximal element of $\mathbf{sM}$ multiplication in (9), which is equivalent to evaluating the cyclic-autocorrelation function for $\mathbf{s}$. Secondly, we search the second maximum, denoted as $\max_2$ function in (10), to get the highest sidelobe amplitude. This value is symbolized by m_2 in Fig. 10 and (10)-(11). The difference between m_1 and m_2 gives the fitness value f in (11) and is computed for each individual in the population at each iteration. The α and p in (11) are needed to reduce the algorithm's execution time, g_{PAPR} is used to reduce the resulting PAPR, and all these additional variables are explained later. For now, please assume that $g_{\mathrm{PAPR}} = 0$, and this parameter is not used. We introduce it in section IV.C. The fitness value f indicates sequences, which are favorable in terms of cyclic-autocorrelation. If the m_1 and m_2 difference is high, the f value is also high, and the investigated sequence has good cyclic-autocorrelation properties, and should be used at a later algorithm stage. All sequences with low f in the population are removed. Thus, the algorithm searches for sequences, which give the highest f values. This means, m_1 mainlobe is maximized, and sidelobes m_2 are minimized ($m_1 = \max f(\mathbf{s})$; $m_2 \approx 0$). As a result, sequences as shown in Fig. 11, are obtained.

For sequences with $N = 30$, at least 10^7 iterations are needed. The algorithm's operation as a function of the time for $N = 10$ is shown in Fig. 12.

Eq. (11) uses α and p to improve algorithm convergence additionally. The maximal sidelobe amplitude m_2 is

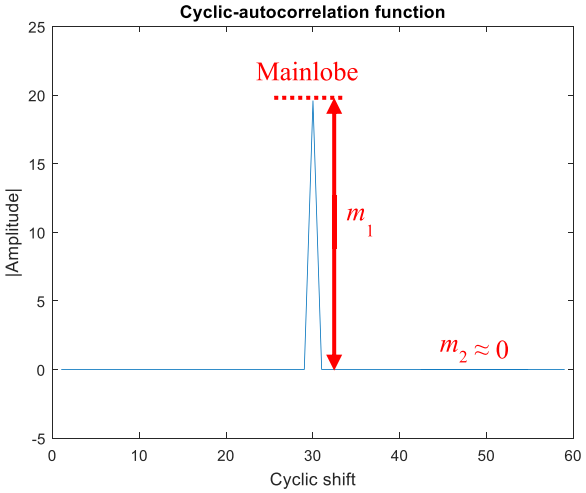


FIGURE 11. Genetic algorithm optimization results for a sequence with $N = 30$.

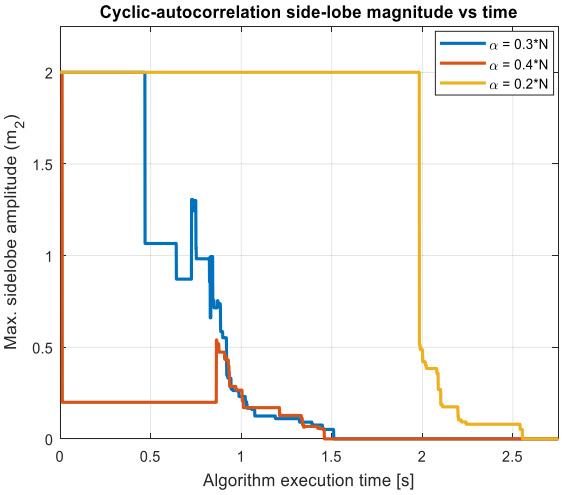


FIGURE 13. The impact of α value on max-sidelobe amplitude (m_2) in a function of the genetic algorithm's execution time for $N = 10$.

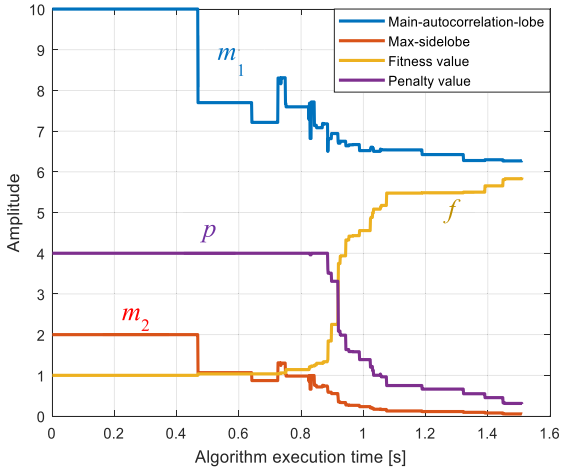


FIGURE 12. Mainlobe (m_1), max-sidelobe (m_2), f, and p in a function of the genetic algorithm's execution time for $N = 10$.

manipulated by the α amplification factor in (11). The α value is proportional to the code length, and is defined in (12). The higher the α is, the algorithm optimizes the sidelobe-amplitudes more aggressively at the mainlobe amplitude reduction cost. The α increases the convergence and reduces execution time, but its value should not be higher than $0.35N$. Otherwise, the optimal sequence cannot be found, and as a result, a sequence with reduced main-peak amplitude is returned.

α depends on N due to the reason that longer sequences have higher autocorrelation peaks. To keep a quasi-constant relation between the mainlobe and sidelobes amplitude, and keep quasi-constant convergence, m_2 is scaled by α. This is required because the algorithm can be executed for various N. Short sequences with $N = 3$ have significantly lower mainlobe amplitude than sequences with $N = 30$. Therefore we adjust the slope of f in (11) by employing α, which depends on N. The value of α has been estimated experimentally to $0.3N$ in our scripts. The influence of the α on the execution time for $N = 10$ is depicted in Fig. 13. Additionally, we use the penalty value p in (11), explained in the next subsection. In general, the genetic algorithm has to find the $\mathbf{s}$ argument giving the maximum of the $f(\mathbf{s})$ function in (11), which corresponds to the sequence with the highest cyclic-autocorrelation peak (m_1) and sidelobe amplitudes approaching zeros ($m_2 \approx 0$). The function $f(\mathbf{s})$ is handled as simple floating-point value f, calculated for each $\mathbf{s}$,and stored in a one-dimensional array of 30 elements (one f value for each individual in the population; population size is 30). Thus, in this paper, we refer to f values instead of the $f(\mathbf{s})$ function, which may look complicated to the reader and do not explain how to implement it in a program routine. If we investigate only 30 values of $f(\mathbf{s})$ stored in a one-dimensional array at each iteration, there is no point in using complicated terminology.

Genetic algorithms allow us to investigate complicated problems in a simple semi-Monte-Carlo fashion, and in our case, we evaluate the $f(\mathbf{s})$ properties avoiding analytical approaches. The $f(\mathbf{s})$ domain is $\mathbb{R}$ and can consist of 30 variables when a sequence with $N = 30$ is scheduled. We limit the applicability of our algorithm to $N \leq 30$. Otherwise, the execution time might take a while to find the proper sequence, and the algorithm is not optimized for solving such complex problems (long spreading sequences in PSSS cause high PAPR and should be avoided). Nevertheless, for practical applications targeting 100 Gbps wireless communication with analog PSSS, N is limited to $\sim$15 due to the correlator complexity [7], [20], [22]. Also, for digital realizations, the correlator and integrator sizes limit the clock frequency (more in section IV). Our solution well supports the targeted length of $N \leq 15$. The genetic algorithm is executed only once, and the generated $\mathbf{s}$ sequences required for $\mathbf{M}$ in (4) are stored in a file. Therefore, the execution time of the genetic algorithm is not critical.

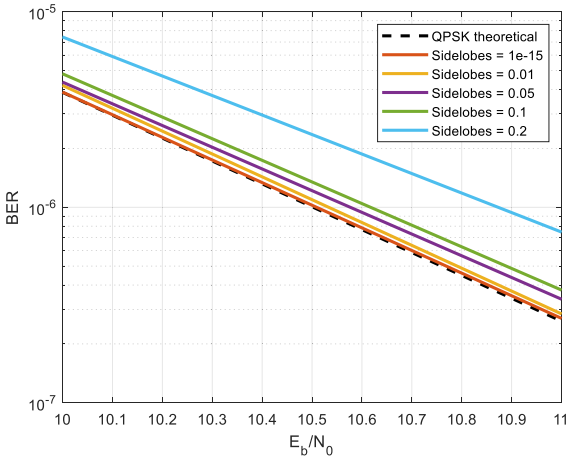


FIGURE 14. Influence of the *t parameter* value on the PSSS BER performance.

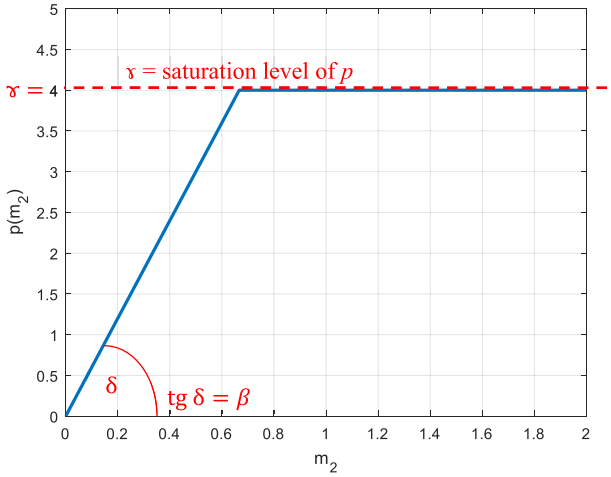


FIGURE 15. The *p* penalty value in a function of m_2.

D. PENALTY VALUE (P)

In genetic algorithms, constraints are handled using the penalty functions, which penalize infeasible solutions by reducing their fitness values (f in (11)). We use the p to exclude all codes with sidelobe amplitude $m_2 \geq t$. In such a case, we set p as follows:

$$p = \begin{cases} 0, & m_2 < t \\ \beta\, m_2, & m_2 \geq t,\ \beta\, m_2 < \gamma \\ \gamma, & m_2 \geq t,\ \beta\, m_2 \geq \gamma, \end{cases} \tag{13}$$

where $\beta = 6$, $\gamma = 4$, and t = 0.001.

The $t = 0.001$ threshold for m_2 is an arbitrarily selected value for implementation purposes in our Matlab scripts. In fact, t should not be higher than 0.01, as indicated in Fig. 14. Figure 14 depicts the BER performance of PSSS, compared to analytical QPSK, when spreading sequences with different sidelobes amplitudes are used. This means that setting t to 0.001 allows us to generate $\mathbf{s}$ sequences with ∼10 times higher precision than required for our floating-point simulation model. In section IV, we also investigate the influence of the quantization for the spreading sequences. Setting 6-bit precision for the sequence coefficients leads to sidelobes with an amplitude of ≈0.05 usually. Thus, setting t below 0.05 does not bring any visible benefits in the BER performance of our CMOS implementation, because the sidelobes are increased to ≈0.05 after the 6-bit quantization. Thus, the prototyped genetic algorithm provides sequences with ≈50 times higher precision than required for our CMOS-related investigations shown in section IV.

β and γ are also selected arbitrarily and have a relatively low impact on the execution time. This is mainly caused by the fact that algorithm quickly reduces sidelobes, and only at the beginning of the execution, p value dominates the f evaluation. After the initial sidelobe reduction, the main peak amplitude is dominant in f (Fig. 12), and the sidelobes are low enough that $p \ll f$. When the algorithm starts, the relation is opposite ($p \gg f$). The γ is used as a saturation level for p to additionally reduce the importance of p at the beginning of the program execution. β represents the slope value of p (Fig. 15). These values have been selected experimentally in our case, and any value in the range [4, 6] for both parameters is acceptable. In some cases, selecting the parameters out of this range might require adopting the α. Different parameter values lead to different penalty function results, but the relation between the individuals is preserved in most cases, and allows selecting better and worse individuals in the selection phase. This allows to assess the individuals in the relation lower-sidelobes or higher-sidelobes, but the absolute difference between the sidelobes amplitude is unimportant, and is fully ignored. In general, p is required to exclude all m-sequences, Barker, APAS [27], Arasu [28], and similar binary sequences, which the algorithm can identify. In some cases, the algorithm gets stuck in a local maximum of $f(\mathbf{s})$ caused by one of these sequences, and p is required to prevent it. Therefore, absolute values of β and γ parameters are less important, until we successfully drop these binary sequences from the population. How much these sequences are penalized, and the absolute value of p is unimportant, until the algorithm does not get stuck in local maxima. For controlling the algorithm convergence, we use the α parameter instead. Therefore, the β and γ are constant in our implementation and we do not pay attention to them. Note however that all three parameters are related in the penalty function evaluated in (11). Manipulation of all these parameters at the same time is not advised.

The p-value in a function of m_2 is shown in Fig. 15. The p function may be composed differently, and the algorithm will work correctly and converge to correct results.

The importance of p can be explained by analyzing the following examplary spreading sequences:

$$\mathrm{A} = [0.9, 1, -0.51, 0.9, -0.58, -0.63, 1],$$
$$\mathrm{B} = [-1, -1, 1, 1, 1, -1, 1],$$

where sequence-B is a typical m-sequence of length 7. Cyclic-autocorrelation functions for both sequences are shown in

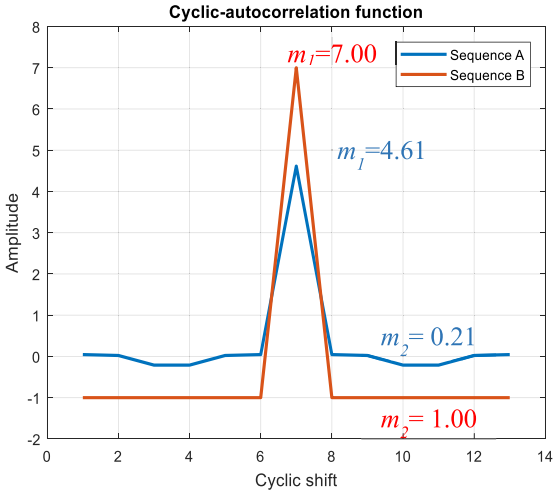


FIGURE 16. Cyclic-autocorrelation functions for two selected sequences. The sequence A is an example of a sequence defined in $\mathbb{R}$. Sequence B is a standard *m*-sequence of length 7.

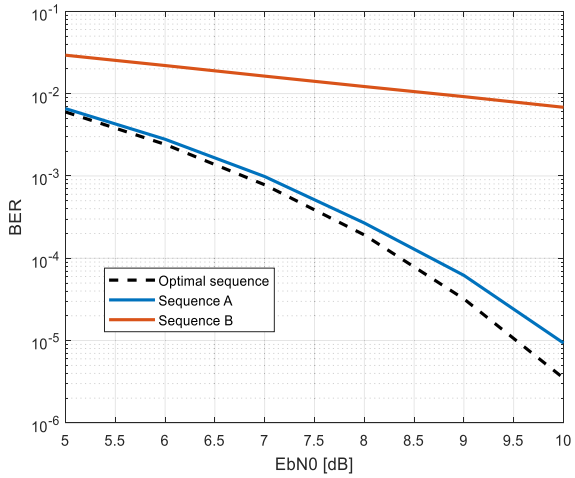


FIGURE 17. PSSS BER performance for sequences A and B defined in Fig. 16.

Fig. 16. Considering fitness values f without p, sequence-A achieves $f_{A-p} \approx 4.61 - 0.3 \times 7 \times 0.21 \approx 4.17$, while sequence-B achieves $f_{B-p} = 7 - 0.3 \times 7 \times 1 = 4.9$. Thus, sequence-B has a higher fitness value and is preferred by the algorithm. Sequence-B will be used in the successive iterations, and A will be deleted. Fig. 17 depicts the BER performance of PSSS based on both sequences, where the resulting power for modulated signals is normalized to 1W in both cases. Sequence-A achieves good performance and underperforms the optimal solution by ∼0.5 dB only. Sequence-B does not allow to perform reliable data transfer. Thus, the genetic algorithm has selected the wrong sequence and cannot reach the optimal solution. The fitness function f without p is poorly composed. If both fitness values are recalculated, including the penalty values p, then $f_A \approx 2.91$ and $f_B = 0.9$. Thus, sequence-B is removed, and sequence-A is used in a later processing stage. Note that the absolute values of p and f are unimportant. Only the relation greater/lower between f values for both sequences is considered. Thus, the parameters α, β, γ used in the genetic algorithm are relatively flexible and do not have to be estimated precisely. The main reason for sequence-B weak performance is high sidelobes that cause strong interferences between the parallel data streams (the lack of orthogonality, Fig. 16). Thus, p is needed to amplify m_2 selectively when f is evaluated.

E. SELECTION

We select 18 fittest individuals and use their genes for reproduction in the selection phase. The offspring generation replaces 12 sequences having the lowest f in the population. The combination of 18 + 12 ensures the fastest convergence of the algorithm, and is explained in detail in the following subsections.

F. GENERATION OF THE OFFSPRING POPULATION

To generate the offspring population, we use two recombination schemes described in the following two paragraphs.

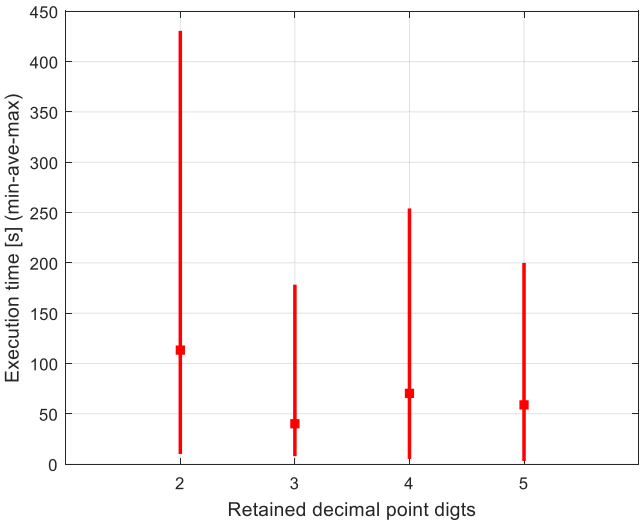


FIGURE 18. The impact of the rounding precision on the genetic algorithm's execution time.

1) ROUNDING

Two offspring individuals are formed by a randomly selected parent pair, in which single gen has been rounded (one gen in each parent). This strategy allows convergence of sequence coefficients to $\{-1, +1\}$, or to any other number with limited floating-point precision. Without this rounding, the genetic algorithm has enormous problems to approach $\{-1, +1\}$ and numbers with limited floating-point precision. E.g., the coefficients in $\mathbf{s}$ are equal to $+0.89(9)\ldots$ and can approach neither $+1.0$ nor $+0.9$, although it would improve the value of f. In our case, we round the code coefficient to retain 3 significant digits, and we provide exactly two rounded genes per iteration. In the other case, execution time is extended. Fig. 18 depicts the impact of the rounding precision on the average algorithm's execution time for $N = 10$, while Fig. 19 depicts the impact of the number of rounded genes per iteration on the execution time.

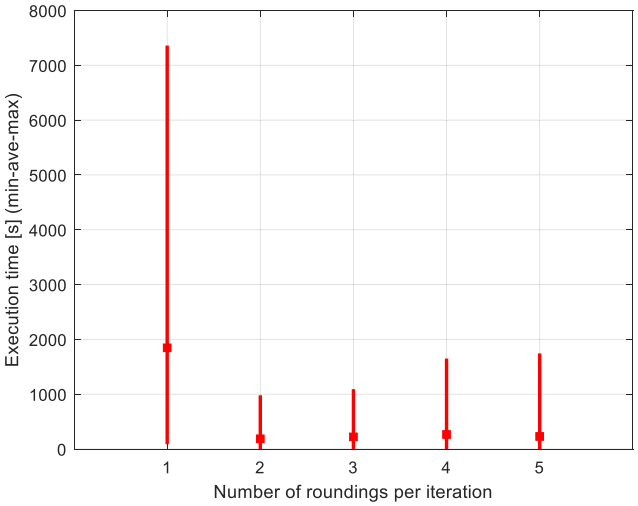


FIGURE 19. The impact of the number of roundings per iteration on the genetic algorithm's execution time.

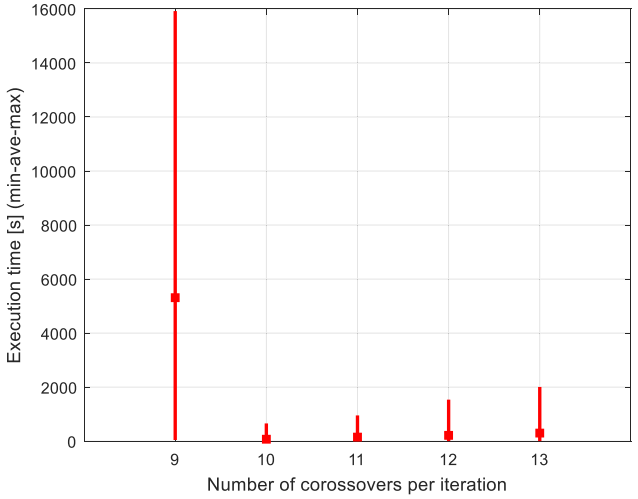


FIGURE 20. The impact of the number of crossovers on the genetic algorithm's execution time.

2) CROSSOVER

Ten new individuals are generated by the arithmetic average of two randomly selected parents and added to the offspring population. Fig. 20 depicts the impact of the number of crossovers on the average algorithm's execution time for $N = 10$. The fastest execution time is achieved, when the crossover is executed ten times and generates ten individuals. Thus, the offspring population consists of 12 individuals in total. The crossover function generates ten of them, and two by the rounding strategy.

G. MUTATION

The mutation is required to prevent being stuck in local maxima of $f(\mathbf{s})$ and significantly affects the algorithm execution time. After the recombination phase, we insert exactly 15 genes on random positions in random individuals. If the number of mutated genes is <15, the algorithm works slower

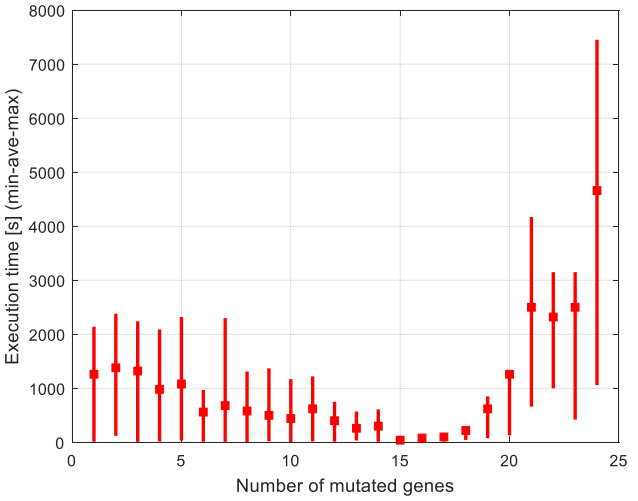


FIGURE 21. The impact of the number of mutations on the genetic algorithm's execution time.

but is able to find the approximated maximum of $f(\mathbf{s})$ (Fig. 21). If the value of mutations is >15, the method works like a Monte-Carlo method, and for long codes ($N > 20$) it does not work (unable to solve (11) with more than 20 variables in $\mathbb{R}$). Therefore, we limit the number of the mutations to 15. After the mutation process, the procedure repeats, and the newly generated offspring population is assessed by (11). The stopping criterion is the maximal sidelobe value $m_2 < t$, and in our case, $t = 0.001$.

IV. RESULTS

This section discusses the BER performance of the proposed PSSS sequences in AWGN and Rayleigh channels. Later, we compare our solution with other methods from the literature. In the end, hardware implementation in 28 nm CMOS is investigated.

A. THEORETICAL ANALYSIS OF BER PERFORMANCE IN AWGN

The theoretical BER for 1 b/s/Hz PSSS systems is derived in [18] and, in a simplified form, is expressed as

$$BER_{PSSS} = \sum_{k=0}^{N-1} \left(\Pr(N,k) \cdot \frac{1}{2} erfc\left(\sqrt{\frac{Eb}{N0}} \cdot \gamma(N,k) \right) \right). \tag{14}$$

The EbN0 penalty factor γ is equal to [18]

$$\gamma(N,q) = \frac{\varepsilon_i(N,q)}{\sum_p [\Pr(N,q) \cdot \varepsilon_i(N,q)]}, \tag{15}$$

where ε_i represents the cyclic correlation result for sequence i defined as

$$\varepsilon_i(N,q) = d \cdot (N - q + u), \tag{16}$$

with the following definitions:

d	represents a bipolar bit value $\{-1, +1\}$,
N	spreading sequence length,
q	number of bits with the same value (one can suppose that q represents the number of positive bits),
u	number of bits with the opposite value (one can suppose that u represents the number of negative bits),
Pr(N,q)	the probability of occurring a bit sequence with q-positive bits in the data sequence of length N.

Pr(N, q) probability follows a binomial distribution function and is equal to [18]

$$\Pr(N, q) = \binom{N-1}{q} \cdot 2^{-2q}. \tag{17}$$

The value of ε variable represents the amplitude of the main autocorrelation peak in the receiver correlators. Due to sidelobes of m-sequences (Fig. 6), this peak is usually different from N. Its value is influenced by ± 1 depending on the sidelobes amplitude values, which depend on the transmitted data. Thus, the above-mentioned equations depend on q and u. The q and u allow estimating the height and the sing of the accumulated sidelobe values. In our case, we use quasi-orthogonal spreading sequences generated by the genetic algorithm. The algorithm's main optimization criterion is to minimize the sidelobe amplitudes, which equals the improvement of the orthogonality between the PSSS sequences. If the sequences become orthogonal, the sidelobes will be 0. Thus, the ε variable in our case is approximately equal to:

$$\varepsilon_i \cong dN, \tag{18}$$

where $d \in \{-1, +1\}$.

Let estimate now the value of $\gamma(N, q)$ in (15) for our system. Equation (18) proves that either $\varepsilon_i(N, q) \cong N$ or $\varepsilon_i(N, q) \cong -N$. Moreover, the sum of probabilities $\sum_p [\Pr(N, q)]$ equals 1. Thus, the value of $\gamma(N, q)$ approaches 1, and the BER equation in (14) reduces to

$$BER_{PSSS} \cong \sum_{k=0}^{N-1} \left(\Pr(N, k) \cdot \frac{1}{2} erfc\left(\sqrt{\frac{Eb}{N0}} \right) \right). \tag{19}$$

The sum of probabilities in (19) equals 1, and (19) can further be reduced to

$$BER_{PSSS} \cong \frac{1}{2} erfc\left(\sqrt{\frac{Eb}{N0}} \right), \tag{20}$$

which is approximately equal to the BPSK BER curve. Thus, our variant of PSSS approximately equals the BER performance of a BPSK system. The same can be proven for higher-order modulation schemes.

B. SIMULATION OF BER PERFORMANCE IN AWGN AND RAYLEIGH

Fig. 22 depicts PSSS 2 b/s/Hz and 4 b/s/Hz BER performance with proposed sequence $N = 15$, compared to standard QPSK and QAM-16 in AWGN. Due to strongly reduced sidelobes, PSSS with our sequences approaches QPSK and QAM for each sequence length N, and does not introduce any BER penalty. This allows us to benefit from the parallel ADC architecture and parallel baseband processing without negatively impacting the BER performance. The parallel implementation architecture is the main advantage of the proposed PSSS scheme and is strongly desired for high-speed systems targeting 100 Gbps and more. PSSS based on short spreading sequences does not improve the BER performance, to the best of our knowledge.

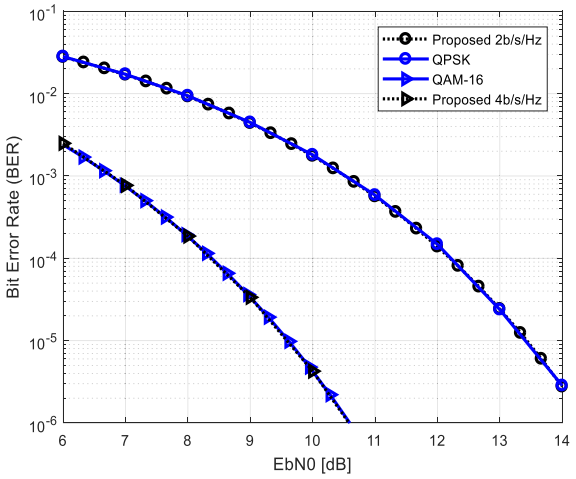


FIGURE 22. BER performance of the newly proposed sequences for PSSS compared to QPSK and QAM-16 in AWGN channel.

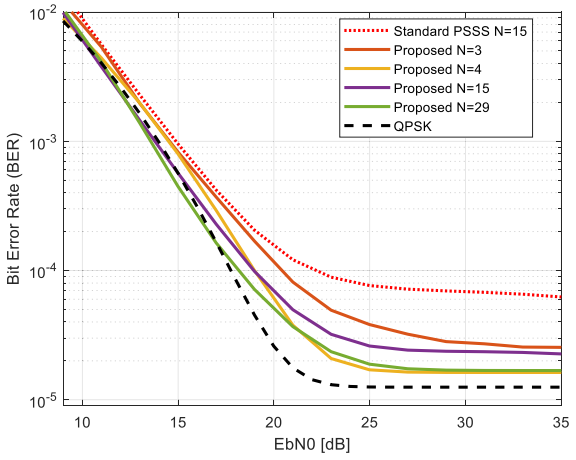


FIGURE 23. BER of the proposed sequences for 2 b/s/Hz PSSS compared to QPSK, and the standard PSSS in a Rayleigh channel defined in the ITU-R M.1225 recommendation.

Fig. 23 compares BER performance for PSSS with 2 b/s/Hz over a multipath Rayleigh channel defined in ITU-R M.1225 [29], reflecting a 3G channel for internet access (outdoor base station to indoor modem communication at 1.9 GHz and sampling frequency of 3.84 MHz). We use least squares zero-forcing equalization in the time domain

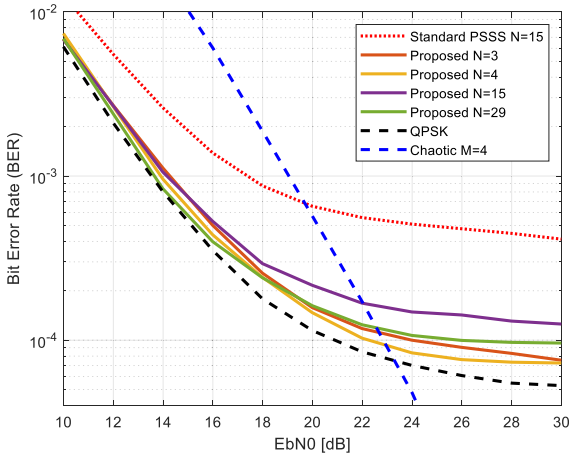


FIGURE 24. Comparisons of the simulated PSSS results for an example Rayleigh channel taken from [30] with the characteristic defined as: $\tau_1 = 0$, $\tau_2 = T_c$, $\tau_3 = 5T_c$; T_c denotes chip time period, $E(\alpha_1^2) = E(\alpha_2^2) = E(\alpha_3^2) = 1/3$.

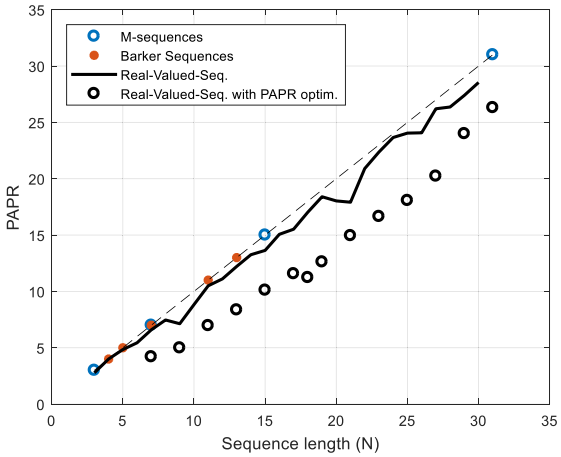


FIGURE 25. Comparison of PAPR for PSSS between *m*-sequences, Barker codes, proposed real-valued sequences, and proposed real-valued sequences with PAPR optimization.

with an FIR filter with 30 taps. For such a channel, PSSS cannot approach the performance of QPSK.

Fig. 24 compares the performance of the proposed PSSS sequences in a Rayleigh channel defined in [30], which is defined as follows: $\tau_1 = 0$, $\tau_2 = T_c$, $\tau_3 = 5T_c$, $E(\alpha_1^2) = E(\alpha_2^2) = E(\alpha_3^2) = 1/3$, where T_c denotes the chip time period. In such a case, PSSS shows better EbN0 performance in the low EbN0 region than chaos-modulation schemes proposed in [30]. However, our methods show a constant error-floor that needs to be compensated by FEC. Nevertheless, our sequences show significantly higher BER performance in both Rayleigh channels than the standard PSSS scheme based on *m*-sequences.

Although the selected 3G Rayleigh channel at 1.9 GHz allows estimating realistic multi-fading parameters, our systems targets THz-band at 240 GHz. The targeted THz frontend is shown in [23]. A PCV antenna lens with very high directivity is used, and all lateral reflections are strongly attenuated at the receiver. Moreover, the THz-band attenuation is high, and all reflections from objects have relatively low power and quickly disappear. Thus, the resulting multipath propagation is marginal, and the performance in the AWGN channel is close to the actual operating conditions at 240 GHz for point-to-point indoor communication [31], [32]. The presented Rayleigh channels can be considered for PSSS at lower frequencies.

C. PEAK TO AVERAGE POWER RATIO (PAPR) AND TRANSMITTED SIGNAL SPECTRUM

Figure 25 depicts the resulting PAPR for PSSS composed of *m*-sequences, Barker codes, real-valued sequences, and real-valued sequences optimized for PAPR. For all real-valued sequences listed in Table 1, generated by our genetic algorithm, the PAPR is better than for Barker codes and *m*-sequences. It is also possible to adjust the genetic algorithm proposed in section III to search for sequences with reduced PAPR. Thus, it is possible to optimize the searched sequences according to additional properties. In our case, we added the requirement to reduce the PAPR by adding the g_{PAPR} parameter in (11). In our case, it is defined as

$$g_{PAPR} = N - \frac{|t_{PEAK}|^2}{|t_{RMS}|^2}. \tag{21}$$

In short, (21) estimates the PAPR reduction of real-valued sequences in PSSS compared to *m*-sequences, Barker codes, and some other binary spreading sequences that could be applied in PSSS. To increase the speed of the PAPR computations, we use the following approximation

$$g_{PAPR} \approx N - \frac{\left|\sum_{k=1}^{N} |s_k|\right|^2}{\frac{1}{N}\sum_{k=1}^{N} |t_k|^2}. \tag{22}$$

It is possible to reduce further computation effort of (22), where only one specific PSSS symbol data is investigated. Then, computation of the denominator in (22) can be realized, as shown in (23).

$$g_{PAPR} \approx N - \frac{\left|\sum_{k=1}^{N} |s_k|\right|^2}{\frac{1}{N}\sum_{k=1}^{N} |s_k|^2}. \tag{23}$$

Note that the simplified PAPR estimation in (22) and (23) can be applied only for our real-valued sequences in PSSS systems, because the average energy of such composed PSSS symbols is flat (Fig. 26).

For *m*-sequences, it leads to erroneous results due to imprecisely estimated average power (Fig. 26). For PSSS with *m*-sequences, the average power needs to be estimated on a significantly more extended period than 1 PSSS symbol. In our algorithm, all binary sequences are dropped by the penalty function in (11). Therefore this simplified approximation of PAPR can be applied. It is also possible to estimate the

TABLE 1. Example real-valued spreading sequences generated by the proposed genetic algorithm.

N	Example Sequence	PAPR	Max. Corr.
3	1; 1; -0.5	2.78	2.25
4	1; -1; 1; 1	4	4
5	-0.6667; -0.6667; -0.6667; -0.6667; 1	4.84	2.78
6	0.5; 0.5; -1; 0.5; 0.5; 0.5	5.44	2.25
7	1; 1; -0.5858; 1; -0.5858; -0.5858; 1	6.59	5.03
8	-0.5774; 0.5774; -1; 0.5774; 0.5774; -0.5774; -1; -0.5774	7.46	4
9	0.9397; 0.9397; -1; -0.1736; -0.766; -0.25; -0.766; -0.1736; -1	7.13	5.06
10	-0.8332; 0.508; 0.8097; -0.9447; 0.9848; 0.2572; 1; 0.6983; 0.3686; -0.5188	8.83	5.43
11	0.634; -1; 0.634; -1; -1; 0.634; -1; -1; -1; 0.634; 0.634	10.5	8.01
12	1; -1; -1; 0.5; -0.5; -1; -1; -1; 1; 0.5; 0.5; -1	11.1	9.00
13	0.721; 1; -1; 0.918; 0.584; 0.955; 1; 0.251; -1; 0.785; -1; -1; 1	12.2	10.3
14	0.8284; 0.8284; -1; -1; 0.5147; 0.8284; 0.5147; 0.8284; 0.8284; 0.5147; 0.5147; -1; 0.8284; -1	13.3	9.18
15	1; -1; 1; 0.921; 0.9605; 0.5372; -0.6539; -1; 0.0079; -1; 0.9823; 1; -1; 1; 0.6971	13.6	11.9
16	0.9426; -0.8865; -0.9426; 0.7131; 0.4429; -0.6724; 1; -0.9439; 0.7286; -0.4584; -0.7286; -0.9439; -1; 0.6724; -0.4429; 0.7131	15.0	9.9
17	1; -0.6096; 1; 1; 0.1952; 1; 1; -0.6096; 1; -0.6096;-0.6096; -0.6096; 1; 1; -0.6096; -0.6096; -0.6096	15.5	11.0
18	-0.8122; 0.9419; -0.5989; 0.803; -0.747; 0.9969; 0.8754; -0.6891; -0.4212; -0.4435; -0.8932; 0.7266; 0.8317; -1; -0.9177; -0.4466; -0.6277; -0.8893	17.0	11.0
19	1; 1; -0.691; 1; 1; -0.691; -0.691; 1; 1; 1; 1; -0.691; 1; -0.691; 1; -0.691; -0.691; -0.691; -0.6910	18.4	14.3
20	-0.2873; 0.9684; -0.9494; -0.9642; -0.8991; 0.7124; 0.7704; -0.2928; 0.9863; 0.7859; -0.9101; 0.3241; 0.6014; -0.3025; 0.8681; 0.840; 0.8197; 0.7835; -1; 0.5986	18.0	11.9
21	-0.7279; -0.7279; -0.1508; 0.9706; 0.9706; -0.1508; 0.9706; -0.7279; -0.1508; -0.7279; 0.9706; -1; -0.7279; 0.9706; -0.1508; -0.7279; -0.7279; -1; 0.9706; -0.7279; -1	17.9	13.0
22	0.6615; 0.5953; 0.7010; 0.6637; 0.8574; -0.9705; -0.9238; 0.8895; 0.9701; 0.6482; 0.3725; -0.8826; 0.9905; -0.6633; -0.5263; -1.0000; 0.5642; -0.9225; 0.7832; 0.8977; -0.7459; 0.7447	20.9	13.7
23	-0.7101; -0.7101; 1; 1; -0.7101; -0.7101; 1; 1; -0.7101; 1; -0.7101; 1; 1; 1; 1; 1; -0.7101; -0.7101; -0.7101; -0.7101; 1; -0.7101; 1	22.3	17.5
24	0.7301; 0.8332; 0.872; 0.8581; -0.7053; 0.8332; 0.986; -0.6913; 0.872; -0.8581; 0.872; 0.8581; 0.986; -1; -0.7053; 0.8581; 0.872; -1; 0.7301; -0.6913; -0.7053; -0.8581; -0.7053; 0.8581	23.7	16.8
25	-0.9621; 0.7793; 0.8732; 1; -0.9716; 0.5803; 0.688; -0.7805; -0.5423; -0.9625; 0.6933; -0.7524; 0.887; 0.8846; -0.9732; -0.7487; -0.8772; -0.6443; 0.9847; -0.8371; -0.9496; -0.6409; -0.5551; -0.658; 0.4689	24.0	16.1
26	-0.488; -1; -1; -0.488; 0.9107; 0.9107; -0.488; -0.488; -0.488; -1; -0.488; -1; -0.488; 0.9107; -1; -1; 0.9107; -0.488; -0.488; 0.9107; 0.9107; 0.9107; -1; 0.9107; -1; 0.9107	24.1	17.6
27	0.6559; -0.6691; 0.9112; -0.8714; -0.8302; 0.9938; 0.7093; -0.939; 0.8315; 0.9786; 0.7507; -0.998; -0.8069; -0.6292; -0.8209; 0.8357; 0.9553; -0.8336; 0.4646; -0.9698; 0.8017; -0.4956; -0.826; -0.821; -0.734; -1; -0.9594	26.2	18.6
28	0.5858; 1; 0.5858; -1; -1; -0.5858; 0.5858; 0.5858; -1; 1; -1; 0.5858; -1; 1; 0.5858; -1; 0.5858; 1; -1; 0.5858; 0.5858; -0.5858; -1; -1; -1; -0.5858; -1; -1	26.4	20.1
29	1; -0.6868; 0.1566; -0.6868; 1; 1; -0.6868; -0.6868;-0.6868; -0.6868; 1; -0.6868; 1; 1; 1; -0.6868;1; 1; -0.6868; 1; 1; 1; -0.6868; 1; -0.6868; -0.6868; -0.6868; -0.6868; 1	27.4	20.6
30	0.7963; 0.4930; -0.4387; 0.5673; -0.6486; 0.6338; 0.9223; 0.4287; -0.4510; 0.7540; 0.8703; 0.9935; -0.8256; -0.9574; -1; -0.7452; -0.9659; 0.9879; 0.9398; -0.9522; 0.9488; -0.9524; 0.6987; 0.8905; -0.7177; 0.9068; 0.8246; 0.7363; -0.9533; 0.6408	28.5	19.6

average power by statistical methods (we leave it as a future work and do not consider it in this paper).

Figures 26 – 28 compare PSSS based on *m*-sequences with PSSS based on the proposed real-valued sequences concerning average power, peak power, and PAPR computed within single PSSS symbols. The main advantage of the proposed sequences is constant power across transmitted symbols, reduced peak power variation, and reduced PAPR. Moreover, the real-valued spreading sequences allow obtaining a flat spectrum of PSSS signal (Fig. 29). Some resulting spreading sequences with reduced PAPR are presented in Table 2. Figure 30 compares the BER performance for PSSS with and without PAPR optimizations. Although the PAPR optimized sequences show reduced main peak amplitude, no BER performance reduction is observed (the orthogonality condition for $\mathbf{M}$ in (3) is preserved).

D. COMPLEXITY COMPARISON TO OTHER SIDELOBES MITIGATION TECHNIQUES

The problem of PSSS-sidelobes originating from the non-orthogonal codes has been investigated several times in the literature. This section compares our solution to other methods according to the BER performance and hardware complexity. Firstly, BER performance of external matched filters inspired by [16], iterative bit detection [18], DC-correction for Barker codes [18], median detection for

TABLE 2. Real-valued spreading sequences with optimized PAPR characteristics.

N	Example Sequence	PAPR	Max. Corr.
7	-1; -0.33; 0.0355; 0.8; -0.7972; 0; -0.2996	4.2	2.48
9	0; -0.9981; 0.5219; -0.09; 1; 0.6; 0.0018; 0.1069; 0.6192	4.99	3.03
11	0.0019; 0.9673; -0.18; 0.719; 0.3; 1; 0.1029; -0.6691; -0.9584; 0.9209; 0.0007	6.96	4.79
13	0.5933; -0.0850; 0.1972; 0.9956; -0.3107; -1; 0.1253; -0.5191; 0.9508; 0.1972; 0.3967; -0.0022; 0.4883;	8.36	4.11
15	-0.8692; -0.9837; -0.7929; -0.5633; -0.098; 0; -0.199; 0.932; -0.9999; 1; -0.1; -0.92; -0.1; 0.802; 0.2	10.11	7.13
17	0.4048; 0.2888; 0.9852; -0.0649; -1; 0.9527; 0.3253; 0.4722; 0.0542; 0.8459; -0.0213; 0.9042; 0.4118; -0.6143; -0.3379; -0.0868; -0.9555	11.5	6.57
18	0.5984; 0.2986; 0.0809; -0.9992; 0.7162; -0.1879; 0.1007; -0.8070; 0.2847; 0.0983; 0.0143; 0.9983; 1; 0.1115; 0.6974; 0.0874; -0.9712; 0.3999	11.23	6.35
19	0.1944; 0.4977; 0.2861; 0.9897; 0.2095; -0.8084; 0.3059; -0.105; -0.8993; 0.2865; -0.1049; 1; 0.0874; -0.3064; -0.9986; 0.7984; 0.095; 0.2971; 0.6993	12.6	6.37
21	0.0168; -1; -0.0535; -0.0299; 0.7138; -0.9562; 0.7226; 0.9292; 0.9244; -0.475; 0.8032; -0.0491; -0.9809; 0.9128; 0.9518; 0.9463; 0.8699; -0.9636; 0.225; 0.0275; -0.2295	14.95	10.9
23	0.9904; 0.2993; -0.2961; -0.1082; 0.8681; -0.9995; 0.6813; -0.8817; 0.7582; 0.9709; -0.6311; -0.4965; -0.0177; -0.0453;-0.0177; 0.7714; -0.3099; -0.9992; -0.1215; 0.7773; 0.3026; 0.7026; 1	16.6	10.2
25	0.7426; 0.923; -0.017; 0.4142; -1; 0.929; -0.0021; 0.1959; 0.13; 0.1087; 0.4752; -0.9776; 0.3437; -0.7865; 0.8596; 0.8479; 0.0177; 0.085; 0.9638; -0.9728; -0.9809; -0.8858; 0.7983; 0.4788; 0.7683;	18.0	11.9
27	-0.8204; -0.2517; -0.9155; -0.9244; 0.3902; -0.7127; 0.1195; 1; 0.7084; -0.7602; -0.9544; 0.1996; 0.9298; -0.1166; 0.8284; -0.944; -0.009; 0.2125; -0.8133; -0.0524; 0.229; 0.5384; -0.6325; 0.8249; -0.6068; -0.1192; -0.7671	20.2	11.6
29	-0.965; -0.0627; 0.4192; 0.4919; -0.8804; 0.7733; 0.3587; 0.9543; -0.9997; -0.4948; 0.7961; 1; 0.3134; -0.1947; -0.8637; -0.687; -0.8835; -0.9698; 0.516; -0.386; 0.3086; -0.7603; 0.5745; -0.4437; -0.7575; 0.8685; -0.9637; -0.9935; 0.1124	24.2	14.5
31	0.6382; -0.0045; -0.2502; -0.702; -0.1036; 0.9149; -0.0141; -0.6379; -0.7194; -0.8715; 0.9094; -0.8341; -0.8098; 0.9412; -0.864; 0.8493; 1; 0.8549; -0.9144; 0.8802; -0.52; -0.2833; -0.4749; 0.5299; 0.7092; -0.4253; -0.4793; -0.8016; -0.9027; -0.8156; -0.7834;	26.3	15.9

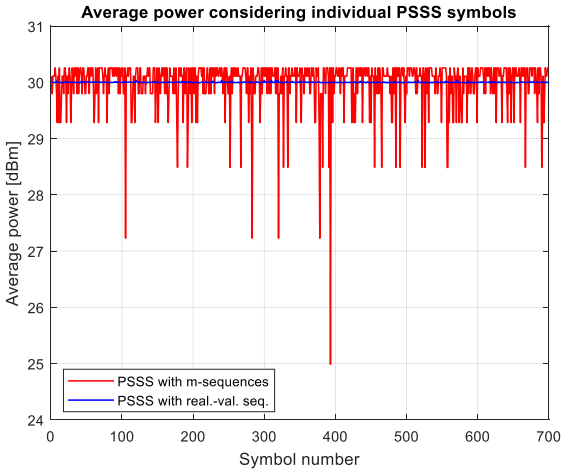


FIGURE 26. Average PSSS symbol power for PSSS composed with m-sequences and real-valued sequences.

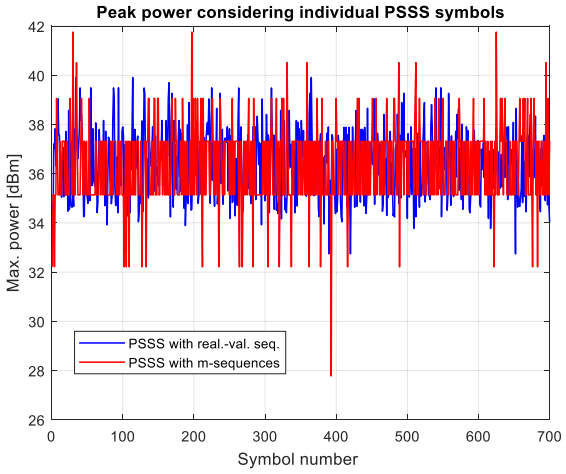


FIGURE 27. Peak power measured within individual PSSS symbols for PSSS composed with *m*-sequences and real-valued sequences.

Barker codes [18], and standard PSSS [4]–[6] are compared with our newly proposed real-valued sequences in Fig. 31.

The proposed method, together with matched filter solution, achieves the best results and approaches the performance of QPSK. The other methods underperform by at least 0.25 dB. Moreover, the DC-correction method works only with the Barker code of length 13. The same holds for the median detection technique for Barker-13. The method based on the iterative bit decoding shows an error floor for sequences with $N = 15$, and should not be used for such short sequences. This method should be applied only for sequences with the length of at least 31 chips [16], [17].

The standard PSSS decoder loses ∼2.5 dB and has the lowest performance from the tested algorithms, but also its architecture is the simplest. At this point, the matched filter technique and the proposed real-valued sequences seem to be the most promising approaches at the cost of additional processing.

It is essential to mention that the matched filters designed in [16] are needed to remove the *m*-sequences' sidelobes,

TABLE 3. PSSS transceiver complexity estimation for different PSSS schemes.

PSSS Algorithm	Supported sequence(s)	Transmitter			Receiver			AWGN penalty
		Num. of operations			Num. of operations			
		xor	add	mult	compare	add	mult	
Standard PSSS [4]	m-sequNEces, N=[3,7,15,31,63…)	450	30	0	30	450	0	~2.5 dB
Median Detection [15]	Barker, N=13	338	26	0	39	338	0	~0.6 dB
DC-correction [15]	Barker, N=13	338	52	2	26	338	0	~0.25 dB
Iterative Detection [18]	m-sequNEces, N=[3,7,15,31,63…)	450	30	0	60	510	0	Error-floor
Two matched filters	m-sequNEces, N=[3,7,15,31,63…)	450	480	450	30	900	450	0 dB
This work	$\mathbb{R}$ domain sequences, N=[3,4,…,30]	0	480	0	30	450	450	0 dB

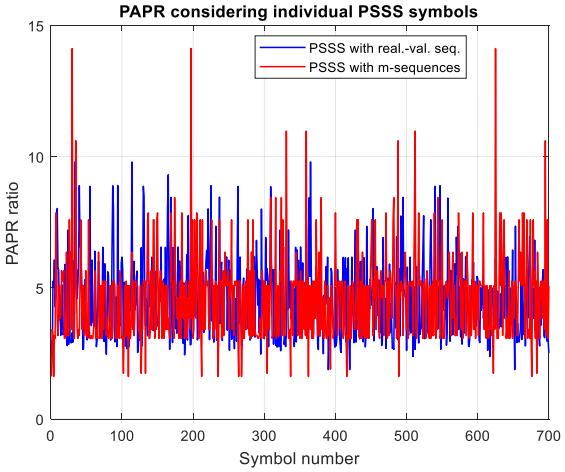


FIGURE 28. PAPR estimated within individual PSSS symbols for PSSS composed with *m*-sequences and real-valued sequences.

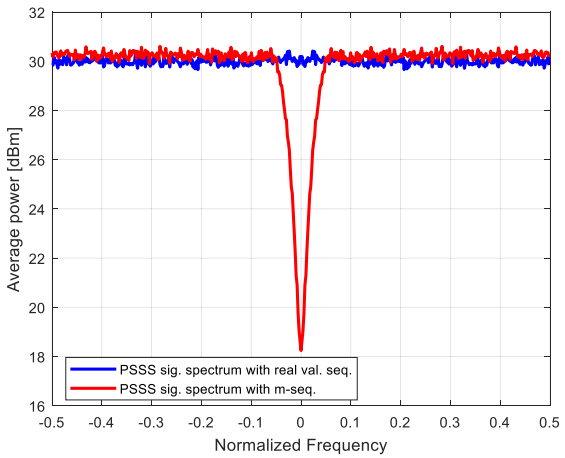


FIGURE 29. FFT-512 spectrum of the resulting transmitter signal for PSSS composed with *m*-sequences and real-valued sequences.

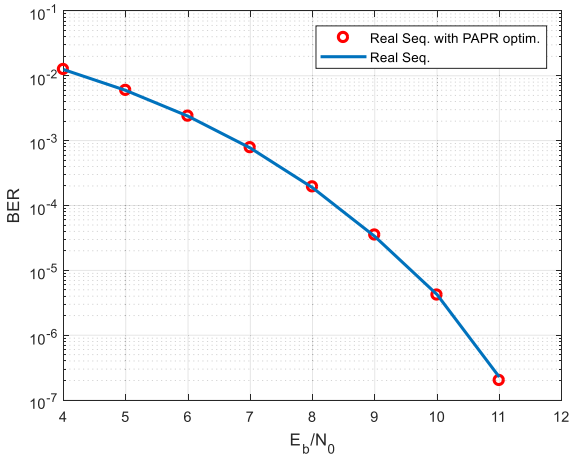


FIGURE 30. Comparison of PSSS BER performance for real-valued sequences with and without PAPR optimizations.

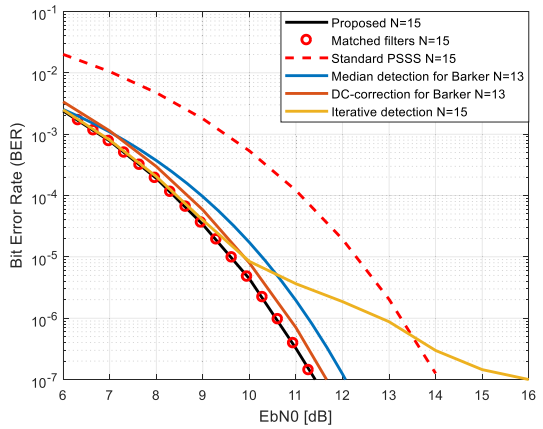


FIGURE 31. BER performance of selected sidelobes mitigation techniques for PSSS in AWGN channel.

and are not used for pulse shaping purposes. The filtering logic performs a dedicated vector by matrix multiplication and works cyclically in a PSSS symbol boundary. The filters cannot be implemented as standard FIR filters.

In general, the limit for $N \leq 15$ targeted in this paper is caused by the integrator complexity when BiCMOS implementation for PSSS is considered [7], [20], [22]. Also, longer sequences show higher PAPR, and their applicability is limited in practical systems.

Table 3 compares the tested solutions' complexities, assuming 2 b/s/Hz, $N = 15$ for *m*-sequences, and $N = 13$ for Barker sequences. Hardware overhead to support our sequences is lower than adding two external matched filters. On the one hand, we require multipliers in the receiver, and this increases the receiver complexity. On the other hand, all methods which do not use multiplications in the receiver have lower BER performance and underperform standard

modulation techniques (e.g., QPSK, QAM-16). Thus, the applicability of these methods in practical applications is questionable, especially that in modern ASIC technologies, the cost of fixed-point multiplications, or Gilbert cells in the analog domain, is not challenging anymore. Also, low-cost Kintex FPGAs are equipped with thousands of DSP blocks that support multiplication and addition operations in prefabricated hardware blocks. Thus, we do not see any prospect for reducing the receiver sensitivity due to the reduction in the number of multiplications. In a data link layer, forward error correction (FEC) is performed, and FEC usually implements thousands of transistors [33] to gain from the received soft-bits as much as possible. FEC tries to correct all bit errors at the cost of very high hardware utilization and power consumption. Thus, reducing the sensitivity on the physical layer due to the reduction in a few multiplications is an example of a horrible designing practice. In the remaining part of this paper, we discard all the methods that underperform the QPSK BER performance, and we concentrate only on the real-valued sequences and matched filters inspired by [16].

Note that our technique supports short spreading codes with any arbitrary selected length in $N \in [3, 30]$, while all other solutions work with m-sequences with the length restrictions of $2^m - 1$, or support a Barker code of length 13 only. Thus, our method is significantly more hardware-friendly. Instead of being fixed to m-sequence-7, m-sequence-15, or Barker-13, we can use any sequence in the range of 3 to 30 chips, and we pick up the sequence that gives the best performance and lowest overhead in the targeted technology. This is especially important for analog realizations [7], [20], [22].

E. HARDWARE IMPLEMENTATION RESULTS

Although table 3 gives an overview of hardware complexity for all investigated PSSS methods, our solution and matched filters inspired by [16] need to be further compared. Firstly, both solutions achieve the highest BER performance and can be considered for practical realizations. Secondly, our sequences require different hardware architecture than the matched filters. Thus, a question arises which of these two methods is more hardware friendly. For this reason, we implement both solutions in FPGA and 28 nm CMOS technology, and perform a detailed comparison.

Fig. 32 depicts the impact of binary quantization of the real-valued sequences. A 6-bit resolution for storing spreading sequences is required to approach the QPSK floating-point performance. Fig. 33 depicts that DACs and ADCs should support 7-bit to 9-bit sampling precision. The same parameters are investigated for matched filters. The filtering arithmetic requires 5-bit multiplication precision, which means that the multiplication operands can be reduced by 1 bit compared to real-valued sequences. ADC and DAC sampling precision have to support at least 7 bits, identical to the case of real-valued sequences. Fig. 34 depicts BER performance for both solutions when the above-mentioned quantization is applied. As a result, both algorithms have identical BER performance, and in this form, are implemented in VHDL.

TABLE 4. PSSS transceiver FPGA hardware utilization.

	Standard PSSS	Matched filters	This work
Tx LUTs	100	88	314
Tx-filter LUTs	---	1010	---
Tx FFs	86	86	240
Tx-filter FFs	---	620	---
Rx LUTs	672	494	2380
Rx-filter LUTs	---	1022	---
Rx FFs	374	370	974
Rx-filter FFs	---	740	---
Total LUT	**772**	**2614**	**2694**
Total FFs	**460**	**1816**	**1214**
Max. Clk.	775 MHz	571 MHz	512 MHz

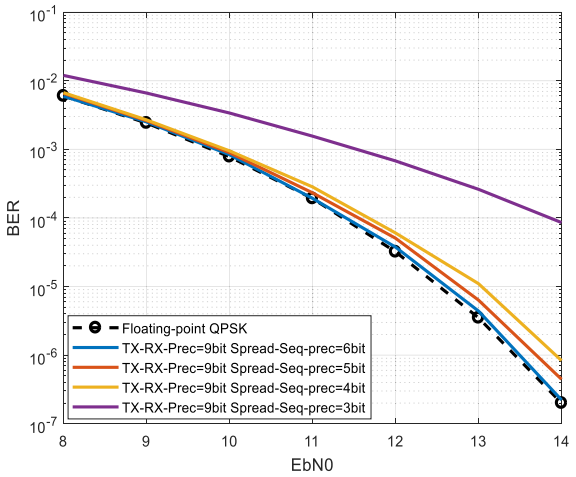


FIGURE 32. The impact of spreading-sequences quantization on BER performance for PSSS based on real-valued sequences in AWGN channel.

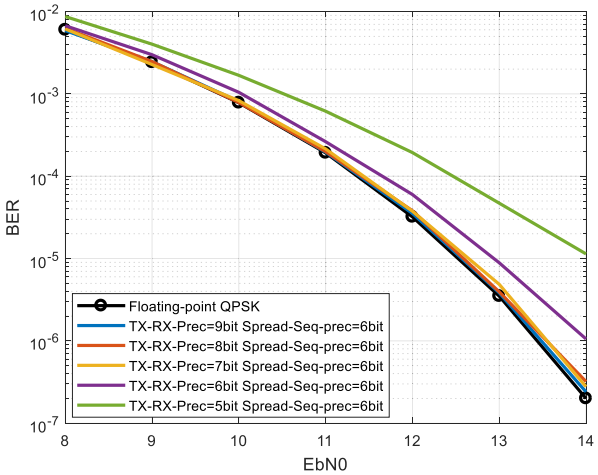


FIGURE 33. The impact of DAC and ADC quantization on BER performance for PSSS based on real-valued sequences in AWGN channel.

Table 4 compares hardware utilization in Virtex Ultrascale+ FPGA for $N = 15$ and 2 b/s/Hz. A hardware module that contains a transmitter and receiver is considered,

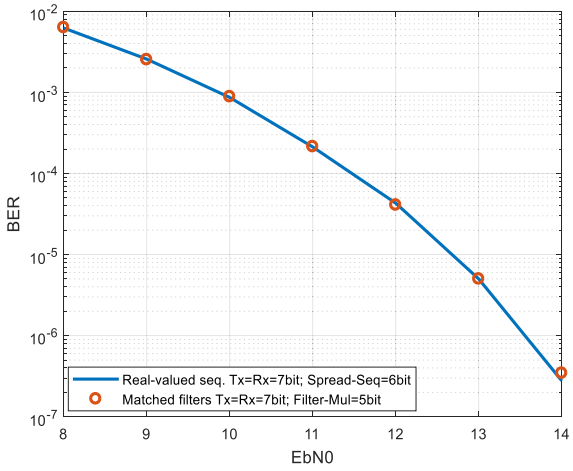


FIGURE 34. BER performance in AWGN for PSSS with real-valued sequences compared to matched filters. Both models use fixed-point arithmetic.

TABLE 5. PSSS transceivers in 28 nm CMOS.

	Matched filters	This work
Tx area	846 um^2	1942 um^2
Tx-filter area	6952 um^2	---
Rx area	3108 um^2	9394 um^2
Rx-filter area	6444 um^2	---
Total area	17350 um^2	11336 um^2 (-34.66%)
Tx power	7.80 mW	12.52 mW
Tx-filter power	24.10 mW	---
Rx power	14.36 mW	59.58 mW
Rx-filter power	23.22 mW	---
Total power	69.48 mW	72.10 mW
Tx energy	4.15 pJ/bit	1.62 pJ/bit
Rx energy	4.89 pJ/bit	7.75 pJ/bit
Total energy	9.04 pJ/bit	9.37 pJ/bit (+3.65%)
Max. clk.	3937 MHz	3610 MHz (-8.3%)
Normalized throughput	453.83 Gbps/mm^2	636.91 Gbps/mm^2 (+40.34%)

so that a transceiver functionality is implemented. Our solution requires ∼33% fewer flip-flops (FFs) than the matched filters, but the maximal clock frequency is reduced by ∼10%. On the one hand, we do not implement the logic for filters, and we save resources. On the other hand, the receiver's correlator for real-valued sequences is significantly more complicated and reduces the overall clock frequency. It is also the most resource-consuming element in the implementation and consumes ∼5 times more look-up tables (LUTs) than in the case of matched filters. It is worth mentioning that matched filters and the proposed real-valued sequences require approximately 3-4 times more resources than the standard PSSS based on *m*-sequences (table 4), but also, the standard PSSS shows the lowest BER performance and underperforms the mentioned solutions by ∼2.5 dB.

The presented FPGA results are difficult to interpret. On the one hand, our solution requires significantly fewer FFs than the matched filters. On the other hand, we achieve a lower clock rate and require ∼3% more LUT logic. Thus, CMOS implementation in 28 nm is additionally investigated, where the LUTs and FFs are reflected in a common silicon area. Thus, it is easier to compare both solutions and show normalized results. The 28 nm floorplans are made with Synopsys and Cadence software with default settings. All power-related parameters are performed using vector-based profiling. All area values include only the core area, and the space required for power rings and pads is excluded. Table 5 depicts the results of this investigation. The consumed power and energy per bit remain at a very similar level for both solutions. Again, our implementation shows a complex receiver integrator's problem and achieves ∼8% lower clock frequency at the benefit of ∼35% smaller chip area. As an effect, the PSSS transceiver based on real-valued sequences achieves ∼40% higher data rate per 1 mm^2 of silicon. This comparison shows our architecture's main advantage, which achieves the same BER performance as the matched filters, but requires significantly less CMOS area. The matched filter solution requires a significantly larger chip area but shows the advantages of well-distributed DSP. The power is dissipated equally among the receiver and transmitter. The same applies to the consumed area. In this aspect, our real-valued sequences have a very non-balanced architecture. The receiver is a couple of times more complicated than the transmitter. Thus, our solution will perform better in applications where a stationary receiver is used, and the transmitters are small battery-powered devices. Optimally, the number of transmitters should be larger than receivers. In contrast to our solution, the matched filters perfectly fit networks, where transmitting and receiving effort is equally distributed among communication nodes.

Due to the well-pipelined architecture of matched filters, this solution perfectly fits digital CMOS realizations. The filtering logic is hardly implementable in the analog domain due to its cyclical nature (which is different from FIR logic).

Our solution is significantly friendlier for analog implementations, where we can realize the multiplication as Gilbert cells and represent the sequence coefficients as current sources. Moreover, we can adapt analog correlation effort in one chip (bit) precision, because our sequences can be generated with one chip (bit) step. This is impossible for the matched filters because they are based on *m*-sequences and have to follow the 2^m-1 length restrictions.

Fig. 35 and Fig. 36 compare floorplans for both systems. Our PSSS transmitter and receiver have larger areas due to the employed real-valued sequences and fixed-point arithmetic, but do not require the filter logic. The overhead required for implementing fixed-point arithmetic is lower than the area required for filters. Thus, the implemented chip area for the proposed method is smaller. On the other side, we see that the logic in matched filters is better balanced between the transmitter and receiver. However, the effort spent on filtering is significantly higher than performing PSSS transmission and bit detection.

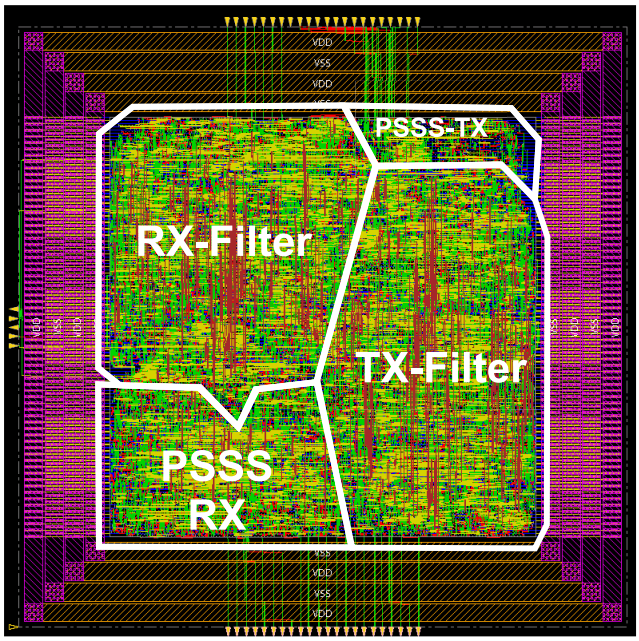


FIGURE 35. Chip floorplan for PSSS with matched filters inspired by [16].

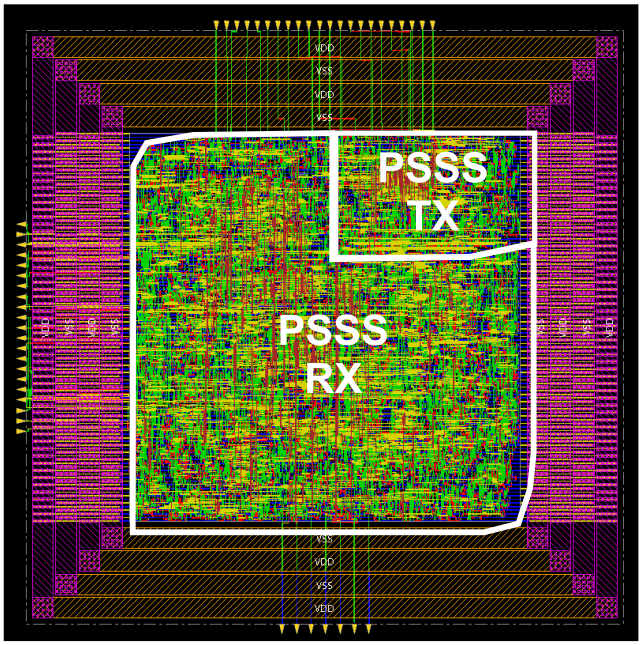


FIGURE 36. Chip floorplan for PSSS based on real-valued spreading sequences.

V. CONCLUSION

This paper addresses Parallel Sequence Spread Spectrum (PSSS) with bipolar spreading codes. Sidelobes of cyclic-autocorrelation function cause intense DC noise at the output of the receiver's correlators (Walsh–Hadamard sequences cannot be used for PSSS due to its cyclic nature). Thus, the BER performance of PSSS receivers is reduced [5]–[7], [16]–[19]. We show real-valued spreading codes (defined in $\mathbb{R}$), which have strongly suppressed cyclic-autocorrelation sidelobes. Therefore, the noise originating from the sidelobes is avoided, and excellent results are achieved. To the best of our knowledge, never before such a method to construct PSSS spreading sequences has been proposed [5]–[7], [16]–[19].

The presented genetic algorithm and PSSS architecture with real-valued sequences have six significant advantages.

- Firstly, the genetic implementation is uncomplicated and can be used without advanced mathematics. Only a structural code of ∼80 lines in Matlab is required to generate the PSSS spreading sequences, and all methods based on matrix factorization are avoided. Therefore, it is simpler to implement than the solution based on matched filters which uses matrix factorization [16]. The general structure of PSSS is unaffected, and only the 6-bit fixed-point arithmetic needs to be added for spreading and correlation.
- PSSS transceivers based on the proposed sequences require ∼40% less CMOS area, providing the same performance as matched filters. Moreover, our algorithm has higher BER performance than the standard PSSS [4], median detection [15], DC-correction [15], and iterative bit detection [18] methods.
- The genetic algorithm generates PSSS sequences of any chosen length in the range of [3, 30], which is not shown in any other paper. We start with a random circulant matrix and improve its characteristics in each iteration. Therefore, we can generate sequences that are in any arbitrary selected length. The matched-filters solution in [16] uses a circulant-matrix initialized with m-sequences, which are factorized in a later stage. Therefore, it generates sequences of the length of 2^m-1 only. In our scheme, we do not need to stick to 2^m-1 in BiCMOS realizations like in [7], [20], and [22], but we can choose the length that is most suitable for the targeted technology.
- The proposed algorithm reduces the resulting PAPR in PSSS systems up to 38% without BER performance degradation.
- The same method can be easily adapted by modifying (9)–(12) to search sequences with other predefined properties, e.g., Kasami- and Gold-like sequences in $\mathbb{R}$ and $\mathbb{C}$.
- This is the first paper, which addresses a practical ASIC implementation of the PSSS algorithm, which does not introduce any BER penalty in AWGN channels.

The genetic algorithm described in this paper is a heuristic method to solve an optimization task of reducing spreading sequences' sidelobes and PAPR on the sequence-coefficients level, which are difficult to mitigate by analytical means. Also, the algorithm does not need precisely estimated configuration parameters. The algorithm accepts a wide range of values for almost every parameter. This is the advantage of the genetic, heuristic, and Monte-Carlo methods. We solve the optimization task without estimating the specific parameter sets and starting points. Only a rough guess is enough to

converge to the targeted results. It is quick, easy, and allows to skip all other complicated solutions.

REFERENCES

[1] J. Wells, "Faster than fiber: The future of multi-G/s wireless," *IEEE Microw. Mag.*, vol. 10, no. 3, pp. 104–112, May 2009.

[2] S. Priebe and T. Kurner, "Stochastic modeling of THz indoor radio channels," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4445–4455, Sep. 2013.

[3] Z. Hossain, Q. C. Li, D. Ying, G. Wu, and C. Xiong, "THz channel model for 6G communications," in *Proc. IEEE 32nd Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2021, pp. 1–7.

[4] A. Wolf, "Method and device for transmitting data words," Patent EP 04701288.5-1515/1584151, DE 102004033 581, US 20060256850, Jan. 10, 2004.

[5] H. Schwetlick, "PSSS-parallel sequence spread spectrum a potential physical layer for OBAN," in *Proc. 14th IST Mobile Wireless Commun. Summit*, 2005, pp. 34–38.

[6] H. Schwedick and A. Wolf, "PSSS-parallel sequence spread spectrum a physical layer for RF communication," in *Proc. IEEE Int. Symp. Consum. Electron.*, Sep. 2004, pp. 262–265.

[7] A. R. Javed, J. C. Scheytt, K. KrishneGowda, and R. Kraemer, "System design considerations for a PSSS transceiver for 100 Gbps wireless communication with emphasis on mixed signal implementation," in *Proc. IEEE 16th Annu. Wireless Microw. Technol. Conf. (WAMICON)*, Apr. 2015, pp. 1–4.

[8] R. Kraemer, M. Methfessel, R. Kays, L. Underberg, and A. C. Wolf, "ParSec: A PSSS approach to industrial radio with very low and very flexible cycle timing," in *Proc. 24th Eur. Signal Process. Conf. (EUSIPCO)*, Aug. 2016, pp. 1222–1226.

[9] L. Underberg, R. Croonenbroeck, R. Kays, and R. Kraemer, "ParSec: Wireless industrial communication first PSSS measurements in industrial environment," in *Proc. IEEE 13th Int. Workshop Factory Commun. Syst. (WFCS)*, May 2017, pp. 1–8.

[10] A. C. Wolf and M. Mahlig, "Benchmarking of WSN solutions and IEEE 802.15.4-2006 PSSS based solutions," in *Proc. 9th GI/ITG KuVS Fachgespräch Sensornetze*, 2010, p. 13.

[11] K. KrishneGowda, P. Rodriguez-Vazquez, A. C. Wolf, J. Grzyb, U. R. Pfeiffer, and R. Kraemer, "100 Gbps and beyond: Hardware in the loop experiments with PSSS modulation using 230 GHz RF frontend," in *Proc. 15th Workshop Positioning, Navigat. Commun. (WPNC)*, Oct. 2018, pp. 1–5.

[12] P. I. M. Urriza, "Parallel sequence spread spectrum-orthogonal frequency division multiplexing (PSSS-OFDM) scheme—A novel physical layer for robust wireless communication systems," M.S. thesis, College Eng., Univ. Philippines Diliman, Quezon City, Philippines, Aug. 2010.

[13] *IEEE 802.15.4-2006—Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)*, IEEE Standard 802.15.4-2006, 2006.

[14] A. C. Wolf and C. Scheytt, "15 Gbps communication over an USB3.0 cable and even more," in *Proc. Int. Multi-Conf. Syst., Sygnals Devices*, Mar. 2012, pp. 1–3.

[15] L. Lopacinski, N. Maletic, A. Hasani, K. Krishnegowda, J. Gutierrez, R. Kraemer, and E. Grass, "A study of barker spreading codes for high-speed PSSS wireless systems," in *Proc. Joint Eur. Conf. Netw. Commun. 6G Summit (EuCNC/6G Summit)*, Jun. 2021, pp. 112–117.

[16] W. Endemann, R. Kays, and E. L. Peter, "Orthogonalization of parallel sequence spread spectrum codes for high order modulation," in *Proc. 16th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2019, pp. 117–121.

[17] E. L. Peter, W. Endemann, and R. Kays, "Improved parallel sequence spread spectrum transmission for high bandwidth efficiency," in *Proc. IEEE 91st Veh. Technol. Conf. (VTC-Spring)*, May 2020, pp. 1–5.

[18] L. Underberg, R. Croonenbroeck, A. Wulf, W. Endemann, and R. Kays, "A PSSS approach for wireless industrial communication applying iterative symbol detection," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2108–2119, May 2018.

[19] K. Krishnegowda, "Investigation of PSSS technologies to achieve 100 Gbps and beyond," Ph.D. dissertation, Brandenburg Univ. Technol., Cottbus, Germany, 2020.

[20] J. C. Scheytt, A. R. Javed, E. R. Bammidi, K. KrishneGowda, I. Kallfass, and R. Kraemer, "100 Gbps wireless system and circuit design using parallel spread-spectrum sequencing," *Frequenz*, vol. 71, nos. 9–10, pp. 399–414, Sep. 2017.

[21] A. R. Javed and J. C. Scheytt, "System design and simulation of a PSSS based mixed signal transceiver for a 20 Gbps bandwidth limited communication link," in *Proc. 1st URSI Atlantic Radio Sci. Conf. (URSI AT-RASC)*, May 2015, p. 1.

[22] A. R. Javed, J. C. Scheytt, and U. V. D. Ahe, "Linear ultra-broadband NPN-only analog correlator at 33 Gbps in 130 nm SiGe BiCMOS technology," in *Proc. IEEE Bipolar/BiCMOS Circuits Technol. Meeting (BCTM)*, Sep. 2016, pp. 78–81.

[23] M. H. Eissa, A. Malignaggi, R. Wang, M. Elkhouly, K. Schmalz, A. C. Ulusoy, and D. Kissinger, "Wideband 240-GHz transmitter and receiver in BiCMOS technology with 25-Gbit/s data rate," *IEEE J. Solid-State Circuits*, vol. 53, no. 9, pp. 2532–2542, Sep. 2018.

[24] L. Lopacinski, N. Maletic, A. Hasani, J. Gutierrez, and E. Grass, "Fast algorithms for generating real-valued spreading sequences for high-speed wireless communication systems based on PSSS," in *Proc. IEEE Microw. Theory Techn. Wireless Commun. (MTTW)*, Oct. 2021, pp. 280–284.

[25] U. Somaini and M. Ackroyd, "Uniform complex codes with low autocorrelation sidelobes," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 5, pp. 689–691, Sep. 1974.

[26] J. T. Alander, "On optimal population size of genetic algorithms," in *Proc. Comput. Syst. Softw. Eng.*, 1992, pp. 65–70.

[27] J. Wolfmann, "Almost perfect autocorrelation sequences," *IEEE Trans. Inf. Theory*, vol. 38, no. 4, pp. 1412–1418, Jul. 1992.

[28] K. T. Arasu, S. L. Ma, and N. J. Voss, "On a class of almost perfect sequences," *J. Algebra*, vol. 192, no. 2, pp. 641–650, Jun. 1997.

[29] *M.1225: Guidelines for Evaluation of Radio Transmission Technologies for IMT-2000*, ITU, Geneva, Switzerland, Feb. 1997.

[30] X. Cai, W. Xu, R. Zhang, and L. Wang, "A multilevel code shifted differential chaos shift keying system with M-ary modulation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 66, no. 8, pp. 1451–1455, Aug. 2019.

[31] *IEEE 802.15.THZ WPAN Interest Group*. Accessed: Nov. 28, 2021. [Online]. Available: https://www.ieee802.org/15/pub/IGthzOLD.html

[32] *IEEE 802.15d, TG3d Channel Modelling Document (CMD)*. Accessed: Nov. 28, 2021. [Online]. Available: https://mentor.ieee.org/802.15/dcn/14/15-14-0310-19-003d-channel-modeling-document.docx

[33] L. Lopacinski, M. Marinkovic, G. Panic, M. H. Eissa, A. Hasani, K. Krishnegowda, and R. Kraemer, "Data link layer processor for 100 Gbps terahertz wireless communications in 28 nm CMOS technology," *IEEE Access*, vol. 7, pp. 44489–44502, 2019.

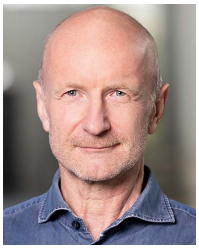
LUKASZ LOPACINSKI received the M.Sc. degree in computer science from the West Pomeranian University of Technology, Szczecin, Poland, in 2009, and the Ph.D. degree from the Brandenburg University of Technology Cottbus–Senftenberg, Germany, in 2017. From 2007 to 2013, he worked in industrial companies in embedded systems and wireless communication. From 2013 to 2016, he was a Research Assistant with the Brandenburg University of Technology Cottbus–Senftenberg. Since 2016, he has been working with IHP Microelectronics, Frankfurt (Oder), Germany.

NEBOJSA MALETIC (Member, IEEE) received the Dipl.-Ing. degree in electrical engineering (major in microwave engineering) and the M.Sc. degree in electrical engineering and computer science from the University of Belgrade, Serbia, in 2008 and 2010, respectively. In 2010, he joined the Faculty of Electrical Engineering, University of Banja Luka, Bosnia and Herzegovina, as a Teaching and Research Associate. In late 2015, he joined the IHP—Innovations for High Performance Microelectronics (Leibniz-Institut für innovative Mikroelektronik), Frankfurt (Oder), Germany, where he is a member of the Wireless Broadband Communications Group. His current research interest includes the design of high-speed wireless communication systems operating in the millimeter-wave spectrum. He participated in several European H2020 projects (5G-XHaul, 5G-PICTURE, WORTECS, 5GENESIS, 5G-VICTORI) related to millimeter waves and their applications. He is a member of the IEEE Communications Society (IEEE ComSoc) and of the IEEE Microwave Theory and Techniques Society (IEEE MTT-S).

GORAN PANIC was born in Sarajevo, Bosnia and Herzegovina, in 1976. He received the Diploma degree from the Faculty of Electronic Engineering in Nis, University Of Nis, Serbia, in 2001, and the Ph.D. degree in engineering from the Technical University of Cottbus–Senftenberg, Germany, in 2014. Following his graduation, he worked as a Research Assistant in the area of microprocessor architectures at the University of Nis. He is currently employed as a Scientist with the System Department of IHP, Frankfurt (Oder), Germany. His research areas include SoC design, low-power, and communication systems.

ALIREZA HASANI received the B.Sc. and M.Sc. degrees in electrical engineering from the Iran University of Science and Technology (IUST), Tehran, Iran, in 2012 and 2015, respectively, and the Ph.D. degree from the Brandenburg University of Technology Cottbus–Senftenberg in 2021. He is currently a Researcher at IHP Microelectronics, Frankfurt (Oder), Germany, working mainly on the design of forward-error correction codes for high-throughput communications systems.

JESÚS GUTIÉRREZ received the B.S. and Ph.D. degrees in telecommunication engineering from the University of Cantabria, Spain, in 2008 and 2013, respectively. Since 2013, he has been with IHP working on high-performance wireless communications systems and lately involved in 5G-related projects and activities on these topics.



ECKHARD GRASS received the Dr.-Ing. degree in electronics from the Humboldt University of Berlin, Germany, in 1993. He was a Visiting Research Fellow at Loughborough University, U.K., from 1993 to 1995, and a Senior Lecturer in microelectronics with the University of Westminster, London, U.K., from 1995 to 1999. Since 1999, he has been with IHP, leading several projects on the implementation of wireless broadband communication systems. He is the Team Leader of the Wireless Broadband Communications Group at IHP and a Professor with the Department of Computer Science, Humboldt University of Berlin. He published over 100 papers at international conferences and journals. He was actively involved in the definition of the 60 GHz standards IEEE802.15.3c and IEEE802.11ad. Since 2014, he has been actively involved in the definition and development of 5G, coordinating several European projects. His research topics include wireless broadband communication systems as well as digital signal processing algorithms and architectures.