PeerJ Computer Science

Enhanced architecture and implementation of spectrum shaping codes

Bingrui Wang[1], Zhaopeng Xie[2] and Xingang Zhang[1]

[1] Henan Engineering Research Center of Service and Guarantee for Intelligent Emergency, Nanyang Normal University, Nanyang, Henan, China

[2] The School of Advanced Manufacturing, Fuzhou University, Jinjiang, Fujian, China

ABSTRACT

Spectral shaping codes are modulation codes widely used in communication and data storage systems. This research enhances the algorithms employed in constructing spectral shaping codes for hardware implementation. We present a parallel scrambling calculation with a time complexity of $O(1)$. Second, in the minimum accumulated signal power (MASP) module, the sine-cosine accumulation needs to be determined by remainder with time complexity $O(n^2)$. We offer reduced MASP computations for short bit-width data, ROM storage, and addition pipelines. It can remove the remainder operation, reducing accumulated complexity to $O(1)$. In addition, we present a search algorithm to generate segmented lines to replace the square operations in the MASP module. By employing the search algorithm and shift operations, we can reduce the complexity of the square from $O(n^2)$ to $O(1)$. The implementation results reveal that the original and proposed MASPs yield nearly identical spectrum nulls. The encoder-decoder of the spectral shaping codes with proposed approaches consumes just 6% of the hardware resources when carried out with a Spartan6 XC6SLX25.

Subjects Algorithms and Analysis of Algorithms, Computer Networks and Communications

Keywords Spectrum shaping, K-constraint, Guided scrambling, Accumulated signal power, Spectrum null

Submitted 28 November 2023
Accepted 29 January 2024
Published 21 February 2024

Corresponding author
Zhaopeng Xie, xzp_fzu@outlook.com

Academic editor
Sedat Akleylek

Additional Information and Declarations can be found on page 16

DOI **10.7717/peerj-cs.1883**

Copyright
2024 Wang et al.

Distributed under
Creative Commons CC-BY 4.0

OPEN ACCESS

INTRODUCTION

Spectrum shaping codes are categorized as modulation codes, and they are applied initially to digital communications utilizing transformers to connect two lines. Transformers cannot transfer signals without significant distortion if the power spectral densities of signals include low frequency components. The shaping codes are designed to adjust source data to satisfy the features of the communication channel. These codes are also employed for digital recording systems to translate an arbitrary data sequence into a sequence with particular characteristics required by the systems (*Immink & Cai, 2021*). More recently, a novel concept of integrated microwave photonics spectral shaping is introduced to open avenues to advanced functionalities (*Daulay et al., 2021*).

Spectrum shaping technologies are utilized in a variety of fields. (1) In information processing and transmission fields, *Chai et al. (2014)* discuss the practical obstacles to implementing dynamic spectrum access (DSA) devices and offer solutions. In order to accommodate DSA in commercial off-the-shelf wireless devices, they also propose a general per-frame spectrum-shaping protocol. A simple spectrum shaping technique

based on switching three loads has been presented for backscatter modulation-based Internet of Things (IoT) systems (*Nagaraj, 2017*). *Danila (2021)* describes theoretical research conducted in the terahertz G-band for a piezoelectrically-responsive ring-cone element metasurface composed of polyvinylidene fluoride (PVDF)/silicon and PVDF/silica glass. Utilizing the longitudinal piezoelectric effect of PVDF, this study examines the spectrum shaping ability of a polymer-based metasurface. Three distinct filter functions, such as Fano-like resonances, wavelength interleaving, and variable resonance mode splitting, are accomplished in *Arianfard et al. (2021)*. The outcomes theoretically validate the proposed device as a compact photonic filter with many functions for adjustable spectral shaping. *Dobre et al. (2021)* developed spectrum-skirt-filled pulse-shaping filters corresponding to spectral mask response. The suggested system design achieves more excellent data rates in a dispersive microwave propagation environment than conventional transmission using Nyquist pulse shaping. *Nasarre et al. (2021)* present a novel concept of frequency-domain spectral shaping (FDSS) with spectral extension for the enhancement of the uplink (UL) coverage in 5G New Radio (NR), based on discrete Fourier transform spread orthogonal frequency-domain multiplexing. The results demonstrate that the spectrally-extended FDSS method is a highly effective solution for improving the 5G NR UL coverage. Furthermore, we can create dependable systems by integrating modern modulation techniques and rate-diverse error-correcting codes (*Fang et al., 2023*; *Chen et al., 2023*; *Lin et al., 2023*). (2) In information storage fields, spectrum shaping codes have spectrum nulls at specific frequencies (*Pelusi et al., 2015*). In addition, it is expected to enhance the performance of dedicated servo recording systems by using the shaping codes (*Ng et al., 2015*; *Yuan et al., 2015*), which is a promising technology for ultra-mobile hard disk drives. Shaping codes with spectrum nulls at non-zero frequencies effectively reduces interference between data signals and narrow band signals. In a dedicated servo recording system, there are two frequencies for servo signals, *i.e.,* a frequency of f_1 on even tracks and a frequency of f_2 on odd tracks. In addition to avoiding interference between data and servo signals, it also permits filtering of low-frequency disc noise. Moreover, the applied recording systems require a run-length limit constraint (*Tandon, Motani & Varshney, 2019*), also known as the k-constraint. *Kahlman & Immink (1995)* concern the spectral shaping of both embedded pilot tones and spectral nulls in digital magnetic video tape recordings. The spectral notches are essential to prevent interference between the written data and the servo detecting mechanism. (3) In medical fields, *Greffier et al. (2020)* investigate the influence of tin filter-based spectral shaping computed tomography (CT) on image quality and radiation dose for use in ultralow-dose CT protocols. Tin filtering enhances the quality of the X-ray beam and the image quality characteristics of phantom images. *Baldi et al. (2020)* suggest a spectral shaping and third-generation dual-source multidetector CT scanner for evaluating osteolytic lesions caused by multiple myeloma. The outcome validates the benefits of whole-body low dose computed tomography for diagnosing patients with multiple myeloma. *Agostini et al. (2021)* investigate the function of third-generation iterative reconstruction (ADMIRE3) in a dual-source, high-pitch chest CT protocol with spectral shaping at 100 kVp coronavirus disease 2019 (COVID-19). The low-dose CT with spectral shaping and ADMIRE3 provides acceptable image quality for

evaluating COVID-19 patients while significantly reducing radiation dose and motion anomalies. Hardening the X-ray beam, tin prefiltration is established for imaging high-contrast subjects in energy-integrating detector computed tomography (EID-CT) (*Grunz et al., 2022*). This study aims to examine the potential dose-saving effect of spectral shaping *via* tin prefiltration in photon-counting detector CT (PCD-CT) of the temporal bone. Seeking for matched image noise, high-voltage scan methods with tin prefiltration enables more significant dose savings in EID-CT. However, superior inherent denoising reduces the dose reduction potential of spectral shaping in PCD-CT.

Based on the excellent performance of the research in *Cai et al. (2017)*, this study presents the implementation of spectrum shaping codes deploying a field programmable gate array (FPGA). The shaping code architecture consists of scrambling and descrambling, k-constrained encoding and decoding (*Immink, 2012*), and a minimum accumulated signal power (MASP) module. We provide simplified approaches for these modules, which are suitable for hardware implementation. Scrambling is a highly effective technique (*Park & Son, 2020*; *Xiao et al., 2020*; *Liu et al., 2021*). In the proposed scrambling and descrambling, we use only XOR (exclusive or) logical operations and no other arithmetic operations. The algorithm for k-constrained encoding and decoding is then described. Furthermore, we propose improved calculations to reduce parameter storage and processing complexity in the MASP module.

The study is organized as follows. In 'Shaping Code Algorithms', we describe the overall architecture of the FPGA system implementation and present the algorithms of spectrum shaping codes. 'The enhanced algorithms' enhances the algorithms. 'FPGA implementation of a spectrum shaping code' demonstrates a specific hardware implementation of the shaping algorithms with reduced computations. The shaping code is synthesized, and the consumed resources are analyzed. 'Discussion and conclusion' gives the conclusion and discussion.

SHAPING CODE ALGORITHMS

Figure 1 illustrates a block diagram of an encoder and a decoder for spectrum shaping codes. In the encoder, the first step is to generate 0 to $2^p - 1$ numbers in decimal form and convert them to binary vector A with the size of $2^p \times p$, and p denotes the length of a scrambler. Then, we append A to the user data of length m bits, generating a vector B of size $2^p \times (m+p)$, that is, $B = [B(0), B(1), \ldots, B(2^p - 1)]$. Second, B is fed into a guided scrambler module and then is scrambled. Then, we can generate a vector C of size $2^p \times (m+p)$. Third, the scrambled vector C is encoded using a k-constrained encoder, yielding a vector D of size $2^p \times (m+p+1)$. Finally, the accumulated signal power is calculated from $D(0)$ to $D(2^p - 1)$, and the one with the least power vector T is chosen and sent. In the decoding process, the received signal R with a bit-width $(m+p+1)$ is fed into the k-constrained decoder, which produces the data Y with a bit-width $(m+p)$. By descrambling Y, we can obtain the data Z with a bit-width $(m+p)$. The original user data can be recovered by eliminating the redundant p bits.

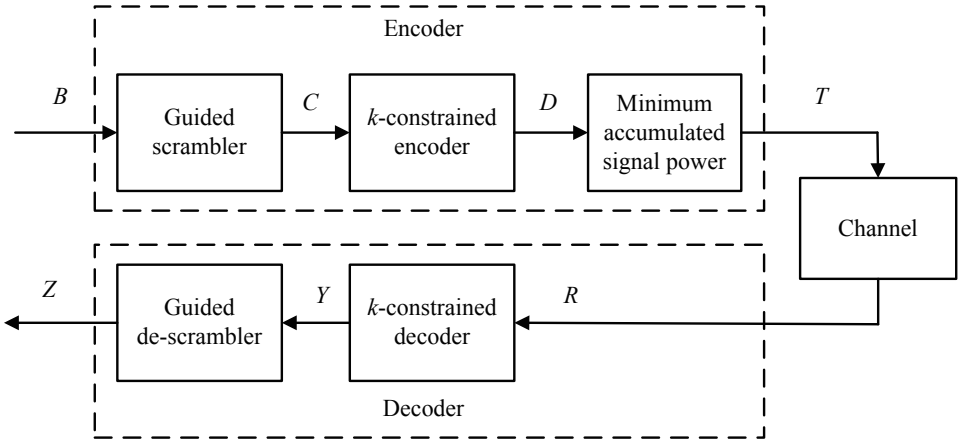

Figure 1 **Schematic diagram of the spectrum shaping code with guided scrambling.**
Full-size DOI: 10.7717/peerjcs.1883/fig-1

Simplified scrambling and unscrambling algorithms

In this study, the guided scrambler (GS) polynomial is

$$g(x) = g_0 + g_1 x + \cdots + g_{p-2} x^{p-2} + g_{p-1} x^{p-1} + x^p. \tag{1}$$

Where g_0 is a constant bit of value 1, and g_i is binary bit of 1 or 0, $0 < i < p$. The $\{b_0, b_1, b_2, \ldots, b_{n-2}\}$ is the bit set to be scrambled. The $\{c_0, c_1, c_2, \ldots, c_{n-2}\}$ represents the scrambled bit set. The b_i and c_i are the binary bits, $0 \le i \le n-2$. Each value of $\{c_0, c_1, \ldots, c_{p-1}\}$ is initialized to zero. The bits of $\{c_p, c_{p+1}, \ldots, c_{n-2}\}$ can be generated by employing the encoding as

$$c_i = b_i + \sum_{k=0}^{p-1} g_{p-k-1} c_{i-k-1}, p \le i \le n-2. \tag{2}$$

As b_i, c_i, and g_j are binary values of 1 or 0, $0 \le i \le n-2$, $0 \le j < p$, and then the multiplication result of $g_{i-k-1}c_{i-k-1}$ is also binary. Moreover, logical XOR operation can replace the addition involved in Eq. (2). The XOR compares two bits and returns a bit 1 if the two bits are different, 0 if they are equal (*Qiu et al., 2024*). An XOR operation between a variable and 0 returns the variable itself. Let the operation $\oplus$ indicate bitwise XOR. Therefore, we can modify Eq. (2) as

$$c_i = g_0 c_{i-p} \oplus g_1 c_{i-p+1} \oplus \cdots \oplus g_{p-1} c_{i-1} \oplus b_i, p \le i \le n-2. \tag{3}$$

Since Eq. (3) has a recursive structure, we perform a serial implementation, which takes n clock cycles to complete. Thus, the time complexity of GS encoding is $O(n)$.

Next, we show the GS decoding process for restoring the original bit set $\{b_0, b_1, b_2, \ldots, b_{n-2}\}$ from the encoded bit set $\{c_0, c_1, c_2, \ldots, c_{n-2}\}$. The GS encoding and decoding involve only XOR operations. In order to get b_i from c_i, we use Equation (2) or (3) and add $b_i \oplus c_i$ on the both hands side of Eqs. (2) or (3). Hence, we get the desired last

Eq. (4) generating the values of b_i.

$$\begin{aligned} b_i &= b_i \oplus \sum_{k=0}^{p-1} g_{p-k-1} c_{i-k-1} \oplus b_i \oplus c_i \\ &= c_i \oplus \sum_{k=0}^{p-1} g_{p-k-1} c_{i-k-1}, p \le i \le n-2. \end{aligned} \tag{4}$$

By comparing Eqs. (2) with (4), we observe that we only need to change positions between c_i and b_i. Since the c_i is the encoded bit and can be known, the GS decoding can be implemented in parallel.

The algorithm of *k*-constrained encoding and decoding

The k-constrained encoding algorithm is described as follows:

Step 1: Add a bit 1 to the scrambled bit set $c = \{c_0, c_1, \ldots, c_{n-2}\}$, and then generate $e = \{e_0, \ldots, e_{n-1}\}$, where $e_0 = 1$ and $(c_0, c_1, \ldots, c_{n-2}) = (e_1, e_2, \ldots, e_{n-1})$.

Step 2: The bit set e is splitted into L blocks of q bits, where $L = 2^{q-1}$ and $n = L * q$. The L blocks consist of $L_0 = (e_0, e_1, \ldots, e_{q-1})$, $L_1 = (e_q, e_{q+1}, \ldots, e_{2q-1})$, $\cdots$, $L_i = (e_{i*q}, e_{i*q+1}, \ldots, e_{(i+1)*q-1})$, $\cdots$, $L_{2^{q-1}-1} = (e_{n-q}, e_{n-q+1}, \ldots, e_{n-1})$, $0 \le i \le (2^{q-1} - 1)$.

Step 3: If $(e_{i*q}, e_{i*q+1}, \ldots, e_{(i+1)*q-1}) = [0]_{bin}$, then $(e_{i*q}, e_{i*q+1}, \ldots, e_{(i+1)*q-1}) = (e_{u*q}, e_{u*q+1}, \ldots, e_{(u+1)*q-1})$, where $0 \le u \le i \le (2^{q-1} - 1)$ and the $[]_{bin}$ represents to take a binary number with q bits.

Step 4: $(e_{u*q}, e_{u*q+1}, \ldots, e_{(u+1)*q-1}) = [i]_{bin}$. Repeat steps 3 and 4 $L-1$ times.

Step 5: If $(e_{i*q}, e_{i*q+1}, \ldots, e_{(i+1)*q-1}) = [2^q - 1]_{bin}$ and the first bit of $(e_{u*q}, e_{u*q+1}, \ldots, e_{(u+1)*q-1})$ is 1, then $(e_{i*q}, e_{i*q+1}, \ldots, e_{(i+1)*q-1}) = [0]_{bin}$. Repeat step 5 L times.

The k-constrained decoding algorithm is presented as follows:

Step 1: If $(e_{i*q}, e_{i*q+1}, \ldots, e_{(i+1)*q-1}) = [0]_{bin}$, we can have $(e_{i*q}, e_{i*q+1}, \ldots, e_{(i+1)*q-1}) = [2^q - 1]_{bin}$, $0 \le i \le (2^{q-1} - 1)$. Repeat step 1 L times.

Step 2: If the first bit of $(e_{i*q}, e_{i*q+1}, \ldots, e_{(i+1)*q-1})$ is 0, we have $z = [(e_{i*q}, e_{i*q+1}, \ldots, e_{(i+1)*q-1})]_{dec}$ and $(e_{z*q}, e_{z*q+1}, \ldots, e_{(z+1)*q-1}) = [0]_{bin}$, where the $[]_{dec}$ represents to take a decimal number. Repeat step 2 L times. Note that if the first bit e_0 is 1, the step 2 is executed only once, and the decoded data is obtained by simply removing the first bit e_0.

The algorithm of minimum accumulated signal power

As given in *Cai et al. (2017)*, the MASP criterion is

$$\sum_{s=1}^{t} \left| \sum_{i=1}^{n(l-1)} w_i^* e^{-j2\pi f_s i} + \sum_{i=(l-1)n+1}^{nl} w_i e^{-j2\pi f_s i} \right|^2. \tag{5}$$

Where $j = \sqrt{-1}$, t is the number of spectrum nulls at frequencies $f_1, f_2, \ldots, f_t$ and n is the length of one codeword, l indicates the number of codewords that need to be computed. The w and w^* express the current unencoded codeword and previous encoded codeword, respectively.

Let

$$\begin{aligned} Re_s^{l-1} &= \left[\sum_{i=1}^{n(l-1)} w_i^* e^{-j2\pi f_s i}\right]_{\text{realpart}} = \sum_{i=1}^{n(l-1)} w_i^* \cos(2\pi f_s i), \\ Im_s^{l-1} &= \left[\sum_{i=1}^{n(l-1)} w_i^* e^{-j2\pi f_s i}\right]_{\text{imaginarypart}} \times -1 = \sum_{i=1}^{n(l-1)} w_i^* \sin(2\pi f_s i), \end{aligned} \tag{6}$$

where Re_s^{l-1} and Im_s^{l-1} are the sine-cosine accumulations of the $l-1$ codewords and have been calculated. In Eq. (5), the left part is already obtained and can be directly added to the right part. By application of Euler's formula, the unencoded codeword is computed by

$$\begin{aligned} &\sum_{s=1}^{t} \left| Re_s^{l-1} - jIm_s^{l-1} + \sum_{i=(l-1)n+1}^{nl} w_i \left[\cos(2\pi f_s i) - j\sin(2\pi f_s i)\right] \right|^2 \\ &= \sum_{s=1}^{t} \left| Re_s^{l-1} + \sum_{i=(l-1)n+1}^{nl} w_i \cos(2\pi f_s i) - j\left[Im_s^{l-1} + \sum_{i=(l-1)n+1}^{nl} w_i \sin(2\pi f_s i) \right] \right|^2 \\ &= \sum_{s=1}^{t} \left\{ \left[Re_s^{l-1} + \sum_{i=(l-1)n+1}^{nl} w_i \cos(2\pi f_s i) \right]^2 + \left[Im_s^{l-1} + \sum_{i=(l-1)n+1}^{nl} w_i \sin(2\pi f_s i) \right]^2 \right\}. \end{aligned} \tag{7}$$

THE ENHANCED ALGORITHMS

Parallel scrambling algorithms

The GS scrambler polynomial is employed as $1+x^2$, where p, g_0, and g_1 are equivalent to the digits 2, 1, and 0, respectively. Based on the scrambling Eq. (2), the encoding is given by

$$c_i = c_{i-2} \oplus b_i, 2 \le i \le n, \tag{8}$$

where c_0 and c_1 are initially set to zero. We can see that this calculation is recursively executed in serial. In other words, one clock is taken to produce one c_i. It takes n clocks to calculate all c_i. Thus, the time complexity of the initial scrambling polynomial is $O(n)$. To reduce this time consumption, we transform Eq. (8) as

$$\begin{cases} c_{2j} = c_0 \oplus b_2 \oplus \cdots \oplus b_{2j}, \\ c_{2j+1} = c_1 \oplus b_3 \oplus \cdots \oplus b_{2j+1}, 1 \le j \le \dfrac{n-2}{2}. \end{cases} \tag{9}$$

In this way, the calculation Eq. (9) is not recursive since the right side of Eq. (9), *i.e.*, input data b and c_0 and c_1, are known in advance. Then, we can independently compute $c_3, c_4, \ldots, c_n$ at one clock in parallel, with a time complexity of $O(1)$. It means that the time complexity is reduced from $O(n)$ in Eq. (8) to $O(1)$.

The corresponding decoding is given by

$$\begin{aligned} b_i &= c_{i-2} \oplus b_i \oplus c_{i-2}, \\ &= c_i \oplus c_{i-2}, 2 \le i \le n. \end{aligned} \tag{10}$$

Recall that the encoding and decoding of scrambling only use XOR operations.

Improved MASP with remainder operation

Solving for sine and cosine is a critical step in Eq. (7), and we propose a minimum accumulated signal power with remainder (MASP-R), which is stated as follows.

Step 1: Convert the radian value $2\pi f_s i$ to the degree value h_1, $(l-1)n+1 \le i \le ln$.

Step 2: The h_1 may be greater than 360 degrees, so we need to perform the remainder of the operation on h_1. Furthermore, since $sin(h_1) = cos(h_1 + 270°)$, we also need to calculate the remainder of $h_1 + 270°$ to 360°. Thus, the sine and cosine of $2\pi f_s i$ can be given by

$$\begin{aligned} h_{r1} &= mod\left(h_1, 360°\right), \\ \cos(2\pi f_s i) &= \cos h_{r1}, 0° \le h_{r1} \le 360°, \\ h_{r2} &= mod\left(h_1 + 270°, 360^o\right), \\ \sin(2\pi f_s i) &= \sin(h_1) = \cos h_{r2}, 0° \le h_{r2} \le 360°. \end{aligned} \tag{11}$$

Step 3: We construct a ROM and store the cosine values from 0 degrees to 359 degrees in the ROM. Determine the cosine values of h_{r1} and h_{r2} from the ROM.

Thus, given the cosine value in the first quadrant, we can determine the values in the other three quadrants. The ROM only needs to store 91 numbers from 0 to 90 degrees instead of 360 values, thus saving 3/4 of the storage space. When MASP solves for sine and cosine, it solves for $\cos(h_{r1})$ and $\cos(h_{r2})$. Next, we show an example of a modified cosine solution using h_{r1}.

$$\cos(hr_1) = \begin{cases} \cos hr_1, 0° \le hr_1 \le 90°, \\ -\cos(180° - hr_1), 90° < hr_1 \le 180°, \\ -\cos(hr_1 - 180°), 180° < hr_1 \le 270°, \\ \cos(360^o - hr_1), 270^o < hr_1 < 360^o. \end{cases} \tag{12}$$

Improved MASP with no remainder and square

Remove the remainder operation

Next, we propose an improved MASP algorithm with no remainder and square (MASP-NRS). According to the MASP formula Eq. (7), we compute a 360-degree remainder, obtain the related sine-cosine value, and perform an addition. A sine-cosine accumulation requires n clocks. The parallel execution for the accumulation is complex, and serial operation is employed instead. The time complexity of the remainder operation is $O(n)$, while a sine-cosine accumulation requires n remainders. It leads to the time complexity of accumulation $O(n^2)$. To reduce this time complexity, we propose an algorithm to remove the remainder operation that includes the following methods.

Improvement 1: Reduce the number of codewords involved in accumulation. The $\sum_{i=(l-1)n+1}^{nl} w_i \cos(2\pi f_s i)$ and $\sum_{i=(l-1)n+1}^{nl} w_i \sin(2\pi f_s i)$ of the current l-th codeword need to be added to the $\sum_{i=1}^{nl} w_i^* \cos(2\pi f_s i)$ and $\sum_{i=(l-1)n+1}^{nl} w_i^* \sin(2\pi f_s i)$ of the previous $(l-1)$

codewords. We have

$$\begin{aligned} Re_s^l &= \sum_{i=1}^{n(l-1)} w_i^* \cos(2\pi f_s i) + \sum_{i=(l-1)n+1}^{nl} w_i \cos(2\pi f_s i), \\ Im_s^l &= \sum_{i=1}^{n(l-1)} w_i^* \sin(2\pi f_s i) + \sum_{i=(l-1)n+1}^{nl} w_i \sin(2\pi f_s i). \end{aligned} \tag{13}$$

The values of Re_s^l and Im_s^l increase as the number of codewords increases. After accumulating 64 codewords, we reset Re_s^l and Im_s^l to 0 to limit these values.

Improvement 2: Eliminate the remainder of the operation and use ROM storage instead. Let the shaping code utilize two dual servo frequencies, f_1 and f_2. A codeword w has n bits that is multiplied by four groups of sine-cosine $cos(2\pi f_1 i)$, $sin(2\pi f_1 i)$, $cos(2\pi f_2 i)$, $sin(2\pi f_2 i)$, and $0 \leq i \leq n-1$. Each sine-cosine group contains n data points. A total of $64 \times 4 \times n$ sine-cosine values are stored.

Improvement 3: Adopt small bit-width. The initial sine-cosine values need to be transformed from decimals to integers to calculate on the FPGA. Multiply the initial sine-cosine values by 15 and round to the nearest integer number, which is approximately equivalent to moving the values left by four bits. As a result, including the sign bit, the bit width of four groups of sine-cosine values is 5, with a maximum value of 15.

Improvement 4: The parallel operation. Each item in codeword w has a value of either -1 or 1. The select operations can multiply w by sine and cosine. We can acquire n sine-cosine values from ROM at the same time and perform parallel selection operations to complete the sine-cosine accumulation in a single clock cycle. Thus, we eliminate the remainder operation, reducing the accumulation time complexity from $O(n^2)$ to $O(1)$.

Remove the square operation

Equation (7) involves a square operation, which has a calculating cost of $O(n^2)$ and is challenging to compute. We provide a segmented line search algorithm with dynamic error. The search algorithm seeks segmented points, which are combined to produce a segmented curve. Applying the curve, we obtain an approximate estimation. The operation of this curve only involves deterministic shifts and additions/subtractions with a complexity of $O(1)$. The complexity of the proposed search algorithm is two orders of magnitude lower than that of the square operation. The key features of the algorithm are the usage of dynamical error and the balanced coefficient of mean square error. The search algorithm is described in Algorithm 1.

We generally use expression Eq. (14) to calculate the mean square error.

$$error = \frac{\sum_{i=1}^{n} \left[\hat{f}(x_i) - f(x_i)\right]^2}{n} \tag{14}$$

where $\hat{f}(x_i)$ represents the predicted value and $f(x_i)$ the actual value.

The large, varied item significantly influences the error expression Eq. (14), but the little various item has a minor impact. So, in Algorithm 1, we propose a balanced-coefficient

Algorithm 1 Segmented line search algorithm with dynamical error

Require: $f(x)$: the square function; x: the independent variable;
Ensure: a set of segmented points;

$x \in [x_0, x_1, \cdots, x_{n-1}]$, k=1;
$x_b = x_0, x_v = x_2, sp_0 = x_0$;
for $j = 2; j <= n-1; j++$ **do**
  while $x_b < x_i < x_v$ **do**
    compute $\hat{f}(x_i) = \frac{f(x_v)-f(x_b)}{x_v-x_b}(x_i - x_b) + f(x_b)$;
  end while
  compute $error = \sum_{i=b+1}^{v-1} \frac{\left[\hat{f}(x_i)-f(x_i)\right]^2}{(v-b-1)e^{\frac{|\hat{f}(x_i)-f(x_i)|^\alpha}{\beta}}}$;
  if $error \geq \left|f(x_v) - f(x_b)\right| \mu$ **then**
    $sp_k = x_{v-1}$;
    $x_b = x_v$;
    $x_v = x_{v+2}$;
    $k = k+1$;
  else
    $x_b = x_b$;
    $x_v = x_j$;
  end if
end for
return $sp = [sp_0, sp_1, sp_2, \ldots]$;

mean square error expression Eq. (15) to accurately describe the importance of each item.

$$error^* = \frac{\sum_{i=1}^{n}\left[\hat{f}(x_i) - f(x_i)\right]^2}{n\left[e^{\frac{|\hat{f}(x_i)-f(x_i)|^\alpha}{\beta}}\right]} \tag{15}$$

where α and β are called fast and slow decay factors, respectively.

Multiple segmented line Eq. (16) can be generated when segmented points are provided. The product of x and $k_0, k_1, \ldots,$ can be replaced by shift operation on x. The complexity of calculating $x2$ is $O(n2)$, whereas applying Eq. (16) and combining with the shift operation to compute the square of x decreases the complexity to $O(1)$. As a result, we can rewrite Eq. (7) as Eq. (17).

$$\hat{f}(x) = \begin{cases} k_0 x + b_0, sp_0 \leq x \leq sp_1 \\ k_1 x + b_1, sp_1 < x \leq sp_2 \\ k_2 x + b_2, sp_2 < x \leq sp_3 \\ \quad \vdots \end{cases} \tag{16}$$

$$\sum_{s=1}^{t}\left\{\hat{f}\left[Re_s^{l-1} + \sum_{i=(l-1)n+1}^{nl} w_i \cos(2\pi f_s i)\right] + \hat{f}\left[Im_s^{l-1} + \sum_{i=(l-1)n+1}^{nl} w_i \sin(2\pi f_s i)\right]\right\} \tag{17}$$

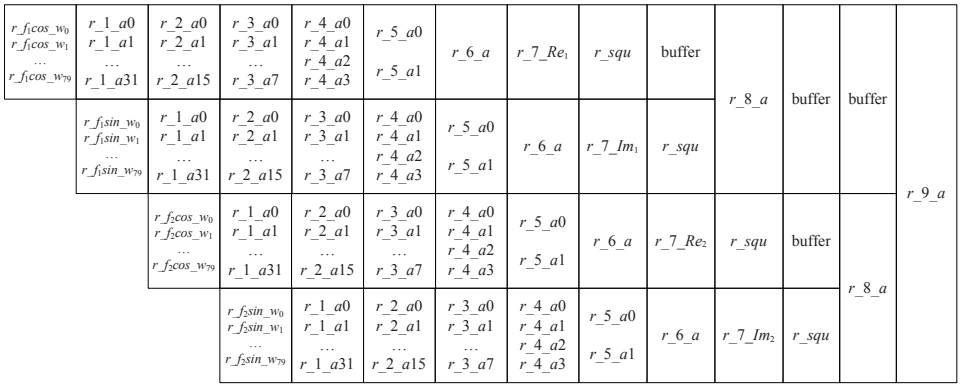

Figure 2 **The pipeline of sine-cosine accumulation.**

Full-size DOI: 10.7717/peerjcs.1883/fig-2

The values of the cosine and sine functions range from −1 to 1 in Eq. (17). It can obtain large values of $\left[Re_s^{l-1}+\sum_{i=(l-1)n+1}^{nl} w_i \cos(2\pi f_s i)\right]^2$ and $\left[Im_s^{l-1}+\sum_{i=(l-1)n+1}^{nl} w_i \sin(2\pi f_s i)\right]^2$, when the length n of a codeword is large and the binary bits w_i are all positive 1. Note that we use the MASP algorithm only for comparison. Thus, we can simultaneously reduce the sum of the two trigonometric functions without affecting the comparison. Then, we can modify Eq. (17) as

$$\sum_{s=1}^{t}\left\{\hat{f}\left[\frac{Re_s^{l-1}+\sum_{i=(l-1)n+1}^{nl} w_i \cos(2\pi f_s i)}{num}\right]+\hat{f}\left[\frac{Im_s^{l-1}+\sum_{i=(l-1)n+1}^{nl} w_i \sin(2\pi f_s i)}{num}\right]\right\}. \quad (18)$$

Where *num* expresses an integer.

FPGA IMPLEMENTATION OF A SPECTRUM SHAPING CODE

Here, we employ a specific shaping code as an example of FPGA implementation. Let the lengths of the shaping code and the original message be 80 and 77 bits. Then we get $L = 16$ and $q = 80/L$ =5. The GS scrambler polynomial is $1+x^2$, and the length of the scrambler is 2. Two bits are chosen from the binary set 00, 01, 10, 11 and appended to the original message. Next, the 79 bits need to be scrambled utilizing the parallel scrambling algorithms described in 'Parallel scrambling algorithms'. After that, we add a bit 1 to the scrambled 79 bits to create 80 bits. The k-constrained and MASP algorithms are then executed.

The implementation of MASP-NRS

The implementation of removing reminder operation

Let the shaping code utilize two kinds of dual servo frequencies, $f_1 = 1/90$ and $f_2 = 1/60$. Each sine-cosine group contains 80 data points. We store $64 \times 4 \times 80$ groups of sine-cosine values, requiring a total of $64 \times 4 \times 80 \times 5 = 12.5$ KB.

Next, using a pipelined operation, we implement the sine-cosine accumulation in Eq. (18). Figure 2 illustrates the pipeline structure.

Step 1: In Fig. 2, we use $r_f_k cos_w_i$ and $r_f_k sin_w_i$ to denote the product of w_i with $cos(2\pi f_k i)$ and $sin(2\pi f_k i)$, $k = 1, 2$, and $0 \le i \le 79$. According to the value of w_i, we use selectors to determine the 80 values of $r_f_1 cos_w_i$, $0 \le i \le 79$.

Step 2: Accumulate $r_f_1cos_w_i$. If 80 data points are added two by two, the four-stage accumulation will need 40, 20, 10, and 5 addition operations, respectively. Thus, five operands are remaining after the four-stage addition. However, adding these five operands two by two is inconvenient. We construct a segmented accumulation equation because the $r_f_1cos_w_i$ has a small five-bit width Eq. (19). Applying the equation, the first stage accumulation of 80 data requires only 32 addition operations.

$$\begin{cases} r_1_ai{=}r_f_1cos_w_{i*2} + r_f_1cos_w_{i*2+1}, 0 \le i \le 15, \\ r_1_ai{=}r_f_1cos_w_{i*2} + r_f_1cos_w_{i*2+1} + \\ r_f_1cos_w_{i+48}, 16 \le i \le 31. \end{cases} \tag{19}$$

Step 3: In Fig. 2, the variables $r_1_a0, \ldots, r_1_a31, r_2_a0, \ldots, r_2_a15, r_3_a0, \ldots, r_3_a7$, $r_4_a0, \ldots, r_4_a3$, $r_5_a0, \ldots, r_5_a1$, r_6_a, comprise a six-step sine-cosine cumulative pipeline. A two-by-two addition is then performed, *i.e.*,

$$\begin{cases} r_2_a0 = r_1_a0 + r_1_a1, \\ r_3_a0 = r_2_a0 + r_2_a1, \\ r_4_a0 = r_3_a0 + r_3_a1, \\ r_5_a0 = r_4_a0 + r_4_a1. \end{cases} \tag{20}$$

The cumulative result $r_6_a = r_5_a0 + r_5_a1$. Each r_6_a of the current codeword needs to be added to the accumulated values of the previous codewords (denoted by Re_1, Im_1, Re_2 and Im_2) to yield $r_7_Re_1$, $r_7_Im_1$, $r_7_Re_2$ and $r_7_Im_2$.

Step 4: Following accumulation, an operation instead of the square is performed, which is introduced in the next section. The accumulation of an encoded codeword is completed after 14 cycles. At the 5th clock, calculate the next encoded codeword. In Fig. 2, the buffer indicates a cache of one clock.

The implementation of removing square operation

For Eq. (18), a symbol contains 80 bits. Due to the k-constrained algorithm, there will not be five consecutive 1's and five consecutive 0's, and a symbol contains no more than 80*80% 1's. In addition, the sine-cosine values are represented by integers in the range of 0 to 15. In extreme cases, 80*80% 1's are required to multiply with these sine-cosine values. The sine-cosine values involved in the multiplication are considered as the mean value, 7.5, and then the multiplication result is 80*80% *7.5. The result of the current symbol needs to be added to that of the previous symbol, so the accumulated result can be 80*80% *7.5*2 = 960. To simplify calculating the square of large number, the *num* in Eq. (18) is set to 16. Thus, 80*80% *7.5*2/16 = 960/16 = 60. The division by 16 yields the same result as a 4-bit right shift. In order to prevent some accumulated results divided by 16 from exceeding 60, we add an overflow control. If any results are greater than 60, the results are set to 60.

In Algorithm 1, the slow decay and fast decay factors are set to 1/2 and 4, and the $\mu$is 0.06, and then we get the segmented line equation

$$\hat{f}(x)=\begin{cases}2x, sp_0 \le x \le sp_1\\ 6x-8, sp_1 < x \le sp_2\\ 11x-28, sp_2 < x \le sp_3\\ 18x-77, sp_3 < x \le sp_4\\ 26x-165, sp_4 < x \le sp_5\\ 35x-300, sp_5 < x \le sp_6\\ 46x-520, sp_6 < x \le sp_7\\ 58x-832, sp_7 < x \le sp_8\\ 70x-1216, sp_8 < x \le sp_9\\ 83x-1710, sp_9 < x \le sp_{10}\\ 98x-2385, sp_{10} < x \le sp_{11}\\ 113x-3180, sp_{11} < x \le sp_{12}\end{cases} \tag{21}$$

where the values of $sp_0, sp_1, sp_2, \ldots, sp_{12}$ are 0, 2, 4, 7, 11, 15, 20, 26, 32, 38, 45, 53, 60. We can explore other appropriate values in conjunction with optimization algorithms such as particle swarms (*Chen et al., 2022*). By replacing the multiplication in Eq. (21) with a shift operation, Eq. (21) become

$$\hat{f}(x)=\begin{cases}x<<1, sp_0 \le x \le sp_1\\ x<<2+x<<1-8, sp_1 < x \le sp_2\\ x<<3+x<<1+x-28, sp_2 < x \le sp_3\\ x<<4+x<<1-77, sp_3 < x \le sp_4\\ x<<4+x<<3+x<<1-165, sp_4 < x \le sp_5\\ x<<5+x<<1+x-300, sp_5 < x \le sp_6\\ x<<5+x<<3+x<<2+x-520, sp_6 < x \le sp_7\\ x<<5+x<<4+x<<3+x-832, sp_7 < x \le sp_8\\ x<<6+x<<2+x<<1-1216, sp_8 < x \le sp_9\\ x<<6+x<<4+x<<1+x-1710, sp_9 < x \le sp_{10}\\ x<<6+x<<5+x<<1-2385, sp_{10} < x \le sp_{11}\\ x<<6+x<<5+x<<4+x-3180, sp_{11} < x \le sp_{12}\end{cases} \tag{22}$$

where the $<<$ indicates that the variable x is left-shifted.

As shown in Fig. 3, we compare the segmented line function $\hat{f}(x)$ with the square function $f(x)$. It is seen that the two curves exhibit a high degree of concordance, suggesting a strong resemblance between them. Using Eq. (23), the correlation coefficient *rela* between the estimated and actual square values equals 1.

$$rela=\frac{\sum_{i=1}^{n}\left\{f(x_i)-E\left[f(x_i)\right]\right\}\left\{\hat{f}(x_i)-E\left[\hat{f}(x_i)\right]\right\}}{\sqrt{\sum_{i=1}^{n}\left\{f(x_i)-E\left[f(x_i)\right]\right\}^2\sum_{i=1}^{n}\left\{\hat{f}(x_i)-E\left[\hat{f}(x_i)\right]\right\}^2}} \tag{23}$$

where $E[f(x_i)]$ denotes the expected actual value and $E\left[\hat{f}(x_i)\right]$ denotes the expected estimated value.

$$td=\frac{rela}{\sqrt{\frac{1-rela^2}{n-2}}}. \tag{24}$$

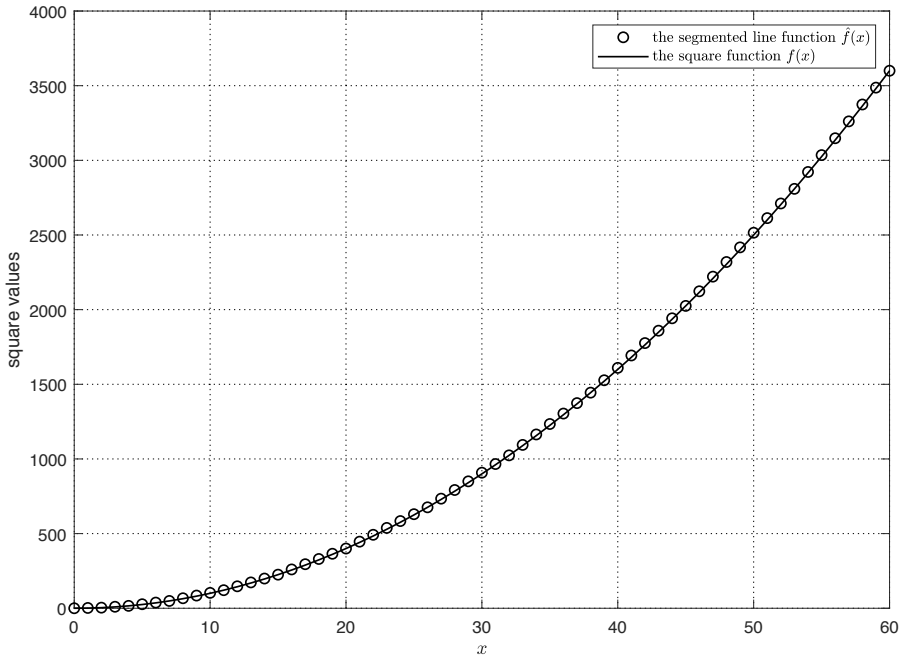

Figure 3 **The comparison of segmented line prediction and actual square.**
Full-size DOI: 10.7717/peerjcs.1883/fig-3

Then, we define a variable *td* according to Eq. (24), consult the t-distribution table, and obtain a p-value of 0 that is less than the significance level ($p = 0.05$). As a result, the correlation coefficient *rela* is regarded as significant. The $\hat{f}(x)$ and $f(x)$ are completely correlated.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left[\hat{f}(x_i) - f(x_i)\right]^2}{\sum_{i=1}^{n}\left[\bar{f}(x_i) - f(x_i)\right]^2}. \tag{25}$$

Next, we examine the R^2 relationship between $\hat{f}(x)$ and $f(x)$ as stated in Eq. (25). The calculated value of R^2 is zero, demonstrating that the variance of the difference between $\hat{f}(x)$ and $f(x)$ is 0% of the variance of $f(x)$. The variance of the difference between $\hat{f}(x)$ and $f(x)$ is extremely small, indicating that $\hat{f}(x)$ and $f(x)$ are quite close in value.

Implementation result

We use a Spartan6 XC6SLX25 to implement the FPGA. Table 1 illustrates the resources consumed by spectrum shaping encoder based on MASP-R and MASP-NRS. These two MASPs employ the same decoding technique, and the hardware resources of decoder are detailed in Table 2. We can see that the encoder consumes more resources than the decoder, since the former one implements the MASP-R/MASP-NRS algorithms. The encoder consumes more 1,500 slice registers than the decoder. Also, it consumes twice as many LUT slices as decoder, due to MASP-R/MASP-NRS needs combinatorial logics such as addition. In particular, the encoder with MASP-R employs two DSPs to calculate the remainder and square operation as in Eq. (7), whereas the encoder with MASP-NRS

Table 1 The hardware sources of spectrum shaping encoder based on MASP-R and MASP-NRS.

Logic utilization	Method	Used	Available	Utilization
Slice registers	MASP-R	2,966	30,064	9%
	MASP-NRS	1,965	30,064	6%
Slice LUTs	MASP-R	5,250	15,032	34%
	MASP-NRS	4,273	15,032	28%
Block RAM/FIFO	MASP-R	0	52	0%
	MASP-NRS	24	52	46%
BUFG/BUFGCTRLs	MASP-R	1	16	6%
	MASP-NRS	1	16	6%
DSP48E1s	MASP-R	2	38	5%
	MASP-NRS	0	38	0%

Table 2 The hardware sources of spectrum shaping decoder.

Logic utilization	Used	Available	Utilization
Slice registers	864	30,064	2%
Slice LUTs	1,686	15,032	11%
Block RAM/FIFO	0	52	0%
BUFG/BUFGCTRLs	1	16	6%
DSP48E1s	0	38	0%

needs no DSP. Since MASP-NRS eliminates the remainder operation, the corresponding encoder occupies some Block RAMs. Based on the Spartan6 XC6SLX25 implementation, the encoder and decoder with MASP-NRS can operate at frequencies of 121.560 MHz and 164.401 MHz, respectively.

Figure 4 demonstrates the power spectrum densities for the same spectrum code. The dashed curve corresponds to the result of the initial MASP which is depicted in Eq. (7), while the solid curve represents the result of the MASP-NRS. Both curves use a code length of 80 bits, and the encoding and decoding methods are similar, except for the difference in the accumulated signal power method and scrambling. In Fig. 4, we can see that the MASP can generate spectrum nulls of −22.8 dB at frequency 1/90 and −20.0 dB at frequency 1/60. The improved algorithm MASP-NRS obtains spectrum nulls of −22.5 dB and −19.4 dB at frequencies 1/90 and 1/60, respectively. The spectrum nulls of MASP-NRS are 98.7% and 97.0% of those of the MASP, with losses of 1.3% and 3% due to truncation operations in MASP-NRS.

DISCUSSION AND CONCLUSION

In this research, we enhance the encoder–decoder algorithms for spectrum shaping codes in order to facilitate hardware implementation. We improve the scrambling algorithm and provide a mathematical description of the k-constrained algorithm. Concerning both descrambling and scrambling, we employ parallel operations that can be executed within a single schedule. We propose an enhanced MASP-R algorithm to compute remainder

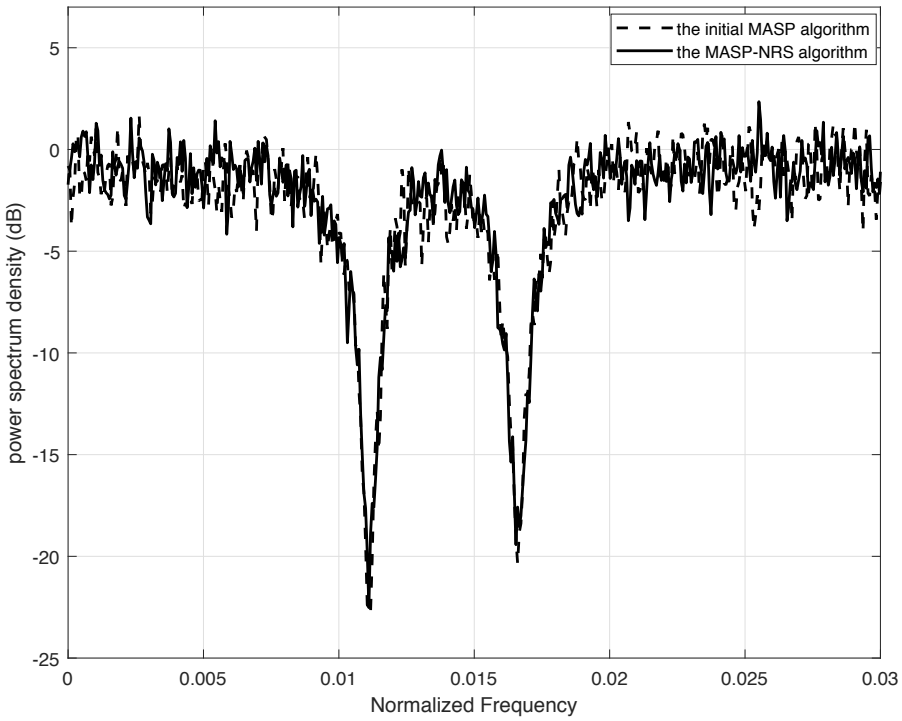

Figure 4 **Comparison of the power spectrum density of MASP and MASP-NRS.**
Full-size DOI: 10.7717/peerjcs.1883/fig-4

operations for sine-cosine accumulation; however, its execution in parallel is challenging due to its significant time complexity. Thus, we further present a MASP-NRS algorithm that quantizes sine-cosine values with short bit-width and stores them in ROM, eliminating the remainder operation. In particular, the MASP-NRS allows parallel operations for the sine-cosine accumulation within a single clock. It is capable of resolving the parallelization issue that plagued the initial MASP. Furthermore, we put forward a search algorithm that utilizes two approaches: dynamical error and balanced-coefficient mean square error. The search algorithm generates a curve $\hat{f}(x)$ similar to the square function $f(x)$. By employing correlation and R^2 analysis, it is possible to ascertain that $f(x)$ and $\hat{f}(x)$ are almost equivalent. The complexity is reduced by two orders of magnitude through substituting the square operation in MASP with $\hat{f}(x)$. Finally, the encoder–decoder of shaping codes is executed utilizing the Spartan6 XC6SLX25. The synthesis results show that the decoder is simpler than the encoder since it does not have to calculate the accumulated signal power. Furthermore, we demonstrate that the performance of initial MASP and MASP-NRS is nearly identical, yielding spectrum nulls of approximately -22.8 dB, which confirms the accuracy of the proposed algorithm.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This research was funded by Nsf of Fujian province (No. 2020J01301), the Key Scientific Research Projects of Colleges and Universities in Henan Province (No.24A520031, 24A520032) and Key Scientifc and Technological Research Projects in Henan Province (No. 222102320369). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
Nsf of Fujian province: 2020J01301.
The Key Scientific Research Projects of Colleges and Universities in Henan Province: No. 24A520031, 24A520032.
Key Scientifc and Technological Research Projects in Henan Province: 222102320369.

Competing Interests

The authors declare there are no competing interests.

Author Contributions

- Bingrui Wang performed the experiments, analyzed the data, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Zhaopeng Xie conceived and designed the experiments, prepared figures and/or tables, and approved the final draft.
- Xingang Zhang performed the computation work, authored or reviewed drafts of the article, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The code is available in the Supplemental File.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.1883#supplemental-information.

REFERENCES

Agostini A, Borgheresi A, Carotti M, Ottaviani L, Badaloni M, Floridi C, Giovagnoni A. 2021. Third-generation iterative reconstruction on a dual-source, high-pitch, low-dose chest CT protocol with tin filter for spectral shaping at 100 kV: a study on a small series of COVID-19 patients. *La Radiologia Medica* **126**:388–398 DOI 10.1007/s11547-020-01298-5.

Arianfard H, Wu J, Juodkazis S, Moss DJ. 2021. Spectral shaping based on coupled Sagnac loop reflectors formed by a self-coupled wire waveguide. *IEEE Photonics Technology Letters* **33(13)**:680–683 DOI 10.1109/LPT.2021.3088089.

Baldi D, Tramontano L, Alfano V, Punzo B, Cavaliere C, Salvatore M. 2020. Whole body low dose computed tomography using third-generation dual-source multi-detector with spectral shaping: protocol optimization and literature review. *Dose-Response* **18(4)**:1559325820973131 DOI 10.1177/1559325820973131.

Cai K, Immink KAS, Zhang M, Zhao R. 2017. On the design of spectrum shaping codes for high-density data storage. *IEEE Transactions on Consumer Electronics* **63(4)**:477–482 DOI 10.1109/TCE.2017.015067.

Chai E, Shin KG, Lee J, Lee S-J, Etkin RH. 2014. Fast spectrum shaping for next-generation wireless networks. *IEEE Transactions on Mobile Computing* **13(1)**:20–34 DOI 10.1109/TMC.2013.125.

Chen P, Shi L, Fang Y, Lau FCM, Cheng J. 2023. Rate-diverse multiple access over gaussian channels. *IEEE Transactions on Wireless Communications* **22(8)**:5399–5413 DOI 10.1109/TWC.2022.3233798.

Chen Y, Zhao T, Cheng P, Ding M, Chen CW. 2022. Joint Front–Edge–Cloud IoVT analytics: resource-effective design and scheduling. *IEEE Internet of Things Journal* **9(23)**:23941–23953 DOI 10.1109/JIOT.2022.3189035.

Danila O. 2021. Polyvinylidene fluoride-based metasurface for high-quality active switching and spectrum shaping in the terahertz G-band. *Polymers* **13(11)**:1860 DOI 10.3390/polym13111860.

Daulay O, Liu G, Guo X, Eijkel M, Marpaung D. 2021. A tutorial on integrated microwave photonic spectral shaping. *Journal of Lightwave Technology* **39(3)**:700–711 DOI 10.1109/JLT.2020.3033706.

Dobre E-I, Mostafa A, Lampe L, ShahMohammadian H, Jian M. 2021. Efficient spectrum utilization via pulse shape design for fixed transmission networks. *IEEE Transactions on Communications* **69(3)**:1510–1523 DOI 10.1109/TCOMM.2020.3042265.

Fang Y, Zhuo J, Ma H, Mumtaz S, Li Y. 2023. Design and analysis of a new index-modulation-aided DCSK system with frequency-and-time resources. *IEEE Transactions on Vehicular Technology* **72(6)**:7411–7425 DOI 10.1109/TVT.2023.3238379.

Greffier J, Pereira F, Hamard A, Addala T, Beregi J, Frandon J. 2020. Effect of tin filter-based spectral shaping CT on image quality and radiation dose for routine use on ultralow-dose CT protocols: a phantom study. *Diagnostic and Interventional Imaging* **101(6)**:373–381 DOI 10.1016/j.diii.2020.01.002.

Grunz J-P, Heidenreich JF, Lennartz S, Weighardt JP, Bley TA, Ergün S, Petritsch B, Huflage H. 2022. Spectral shaping via tin prefiltration in ultra-high-resolution photon-counting and energy-integrating detector CT of the temporal bone. *Investigative Radiology* **57(12)**:819–825 DOI 10.1097/RLI.0000000000000901.

Immink KAS. 2012. High-rate maximum runlength constrained coding schemes using nibble replacement. *IEEE Transactions on Information Theory* **58(10)**:6572–6580 DOI 10.1109/TIT.2012.2204034.

Immink KAS, Cai K. 2021. Spectral shaping codes. *IEEE Transactions on Consumer Electronics* **67(2)**:158–165 DOI 10.1109/TCE.2021.3073199.

Kahlman JA, Immink KS. 1995. Channel code with embedded pilot tracking tones for DVCR. *IEEE Transactions on Consumer Electronics* **41**(**1**):180–185 DOI 10.1109/30.370325.

Lin Z, Fang Y, Chen P, Chen F, Zhang G. 2023. Modeling and analysis of edge caching for 6G mmWave vehicular networks. *IEEE Transactions on Intelligent Transportation Systems* **24**(**7**):7422–7434 DOI 10.1109/TITS.2022.3147696.

Liu X, Huang Y, Chang C-Y, Chen H-H. 2021. Generalized complementary coded scrambling multiple access for MIMO communications. *IEEE Transactions on Vehicular Technology* **70**(**12**):13047–13061 DOI 10.1109/TVT.2021.3125029.

Nagaraj S. 2017. Spectrum shaping for backscatter modulation. *ICT Express* **3**(**1**):52–55 DOI 10.1016/j.icte.2017.01.001.

Nasarre IP, Levanen T, Pajukoski K, Lehti A, Tiirola E, Valkama M. 2021. Enhanced uplink coverage for 5G NR: Frequency-domain spectral shaping with spectral extension. *IEEE Open Journal of the Communications Society* **2**:1188–1204 DOI 10.1109/OJCOMS.2021.3082688.

Ng Y, Cai K, Chan KS, Elidrissi MR, Yu-Lin M, Yuan Z-M, Ong CL, Ang S. 2015. Signal processing for dedicated servo recording system. *IEEE Transactions on Magnetics* **51**(**10**):1–5 DOI 10.1109/TMAG.2015.2456851.

Park S, Son H. 2020. Near-Perfect code scrambling with limited key information for wiretap channels. *IEEE Transactions on Vehicular Technology* **69**(**11**):13410–13423 DOI 10.1109/TVT.2020.3026087.

Pelusi D, Elmougy S, Tallini LG, Bose B. 2015. *m*-ary balanced codes with parallel decoding. *IEEE Transactions on Information Theory* **61**(**6**):3251–3264 DOI 10.1109/TIT.2015.2429139.

Qiu Y, Xie Z, Kang P, Chen P, Fang Y. 2024. Polar-coded gaussian multiple-access channels with physical-layer network coding. *IEEE Transactions on Vehicular Technology* Epub ahead of print Jan 11 2024 DOI 10.1109/TVT.2024.3352980.

Tandon A, Motani M, Varshney LR. 2019. Are run-length limited codes suitable for simultaneous energy and information transfer? *IEEE Transactions on Green Communications and Networking* **3**(**4**):988–996 DOI 10.1109/TGCN.2019.2919780.

Xiao Y, Chen Y, Long C, Shi J, Ma J, He J. 2020. A novel hybrid secure method based on DNA encoding encryption and spiral scrambling in chaotic OFDM-PON. *IEEE Photonics Journal* **12**(**3**):1–15 DOI 10.1109/JPHOT.2020.2987317.

Yuan Z-M, Shi J, Ong CL, Alexopoulos PS, Du C, Kong A, Ang S, Santoso B, Leong SH, Chan KS, Ng Y, Cai K, Tsai J, Ng H, Tan HK. 2015. Dedicated servo recording system and performance evaluation. *IEEE Transactions on Magnetics* **51**(**4**):1–7 DOI 10.1109/TMAG.2014.2354379.