International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V73I2P111
Volume 73 Issue 2, 135-140, February 2025

Original Article

Correlation Comparison of Kasami Sequences with Gold Codes and Novel Method of Generating Balanced Sequences from Kasami Codes

M. Dileep Reddy[1], G. Sreenivasulu[2]

[1,2]Department of ECE, Sri Venkateswara University, Andhra Pradesh, India.

[1]Corresponding Author : mdileep21@gmail.com

Received: 23 September 2024 Revised: 16 January 2025 Accepted: 27 January 2025 Published: 21 February 2025

Abstract *- This paper presents the total number of kasami codes generated for each valid tap of 'n' bit (n is even) linear feedback shift register, generalised formulae to evaluate the total number of type 1 and type 2 kasami codes and the improvement in peak value of autocorrelation function side lobe for kasami codes with respect to the gold sequences. Also, the improvement in peak cross-correlation function for Kasami codes w.r.t. gold codes is also evaluated. The objective of this paper is to present a new method for converting non-balanced kasami codes of length to balanced codes of higher length.*

Keywords *- Autocorrelation side lobe peak, Normalised cross-correlation, Gold sequences, Kasami codes, Balance property.*

1. Introduction

Linear Feed Back Shift Register (LFSR) has 'n' flip-flops that generate maximal sequences of $2^n - 1$ bits. The output sequence can be classified as either a maximal sequence or a nonmaximal sequence. The sequence is maximal if the feedback for the taps in LFSR is provided by generator polynomial or primitive polynomial of Galois Field (GF (2)). Autocorrelation denotes the degree of similarity between a code and a phase-shifted copy of the same code. ACF should have a peak value for code generated with a linear feedback shift register [1]. Good autocorrelation property is used to precisely calculate the transmission interval; hence, it is used in radar or positioning systems and in wireless communication to bifurcate dissimilar propagation signals to avoid inter-symbol interference [2]. Preferably, the auto-correlation value of a code should be impulsive, and the cross-correlation function should be zero all the time [3]. The large set of Kasami sequences achieves low correlation values [4]. Zero cross-correlation the entire time is required in order to remove the effect of multi-user interference at the receiver [5]. Binary code sequences with low autocorrelation, low cross-correlation, enormous linear extent and big family scope are preferred [6, 7]. If the cross-correlation value among spreading sequences from the same chipset is non-zero, then that chipset's sequences are considered non-orthogonal sequences [8]. Three sets of codes, i.e., Maximal, Gold and Kasami, proved themselves to be non-orthogonal binary code sequences. Small set Kasami sequence is a non-orthogonal sequence generated from the Maximal sequence [9]. Existing Kasami codes are non-balanced sequences, i.e., the total number of ones and zeros in the code sequence does not differ by one. In this paper, an innovative method is suggested for generating a balanced Kasami code of higher length. The remainder of this paper is organized as follows. Section II gives a correlation and comparison of kasami codes with gold sequences. Section III presents the kasami code sequence and its balance property. In Section IV, a new method is proposed for generating balanced codes from kasami codes, and Section V proposes a method of converting non-balanced kasami codes to balanced kasami codes. Section VI concludes the paper.

2. Methods for Generating Kasami Codes

Kasami sequences are obtained from maximal codes. The degree 'm' of the equivalent polynomial has to be even where $m = 2k$. The sequence length $2^m - 1$ can be written as $M = (2^k - 1)\,(2^k + 1)$, where $k = \frac{m}{2}$. Starting from a maximal code C_0, the corresponding decimated sequence C_d is obtained by considering all d^{th}chip from C_0 where $d = 2k + 1$ for a Kasami sequence generation and reiterating those $2k - 1$ chips $2k + 1$ times.

The consequential sequence C_d has an equal length as C_0 but a period of $2^k - 1$. The set of Kasami codes is created in an analogous way to the set of Gold codes by taking C_0 and the modulo-2 sum of C_0 and entirely $2^k - 1$ shifted versions of C_d. The total number of Kasami codes for the 'n' bit feedback register of even size is a product of the number of Kasami codes generated from each valid tap of the maximal sequence and the total number of valid taps.

Table 1. Total Kasami codes obtained for different lengths of LFSR

SR	Total Valid Taps	No. of Kasami Codes for Each Tap	Total Kasami Codes
2	1	1	1
4	2	3	6
6	6	7	42
8	16	15	240
10	60	31	1,860
12	144	63	9,072
14	756	127	96,012
16	2048	255	5,22,240

3. Auto and Cross-Correlation Comparison of Kasami Codes with Gold Sequences

The total number of kasami codes for ‘n’ bit feedback Register of even length is a product of the number of kasami codes generated from each valid tap of maximal sequence and the total number of valid taps.

The generalised formulae for gold codes autocorrelation are:

(i) For odd length shift register (n = odd, i.e., $n \neq 0$ mod 4), ACF values of balanced/non-balanced gold codes are $2^{(n-1)}$, $2^{\frac{(n+1)}{2}} - 1, -1$, $-(2^{\frac{(n+1)}{2}} + 1)$.

(ii) For an even length shift register (n = even and not multiple of four, i.e., $n = 2 \bmod 4$), the generalised formulae for ACF values are $2^{(n-1)}$, $2^{\frac{(n+2)}{2}} - 1, -1$, $-(2^{\frac{(n+2)}{2}} + 1)$.

(iii) With $n = 0$ mod 4, there are no gold codes, hence no autocorrelation function [10, 11].

Therefore $(2^{\frac{n}{2}} - 1)$ kasami codes can be generated from each valid tap of linear feedback shift register of ‘n’ bit length. Total Kasami codes are the product of total valid taps and Kasami codes generated from each tap.

Table 2 presents an ACF comparison of kasami and gold codes, and Table 3 provides comparison values of improvement in the peak value of autocorrelation function side lobe for kasami codes with respect to the gold codes.

Table 2. ACF comparison of Kasami codes with gold codes

n bit SR	ACF (Kasami Codes)	ACF (Gold Codes)
2	3,-1	No codes
4	15,3,-1,-5	No codes
6	63,7,-1,-9	63,15,-1,-17
8	255,15,-1,-17	No codes
10	1023,31,-1,-33	1023,63,-1,-65
12	4095,63,-1,-65	No codes
14	16383,127,-1,-129	16383,255,-1,-257

Table 3. Kasami code peak value of ACF side lobe in comparison with gold code side peak value of ACF side lobe in dB

n bit SR	Peak Value of Side Lobe (Kasami Code ACF)	Peak Value of Side Lobe (Gold Code ACF)	Improvement in ACF Peak Value Side Lobe (w.r.t Gold Code Side Lobe Peak)
2	-9.54 dB	-	-
4	-13.97 dB	-	-
6	-19.08 dB	-12.46 dB	6.62 dB
8	-24.60 dB	-	-
10	-30.37 dB	-24.21 dB	6.16 dB
12	-36.25 dB	-	-
14	-42.21 dB	-36.15 dB	6.06 dB

Generalised formulae for autocorrelation of Kasami codes are:

(i) n = even, ACF values are $2^n - 1, 2^{\frac{n}{2}} - 1, -1, -(2^{\frac{n}{2}} + 1)$

(ii) n = odd, not possible to generate Kasami codes, hence no autocorrelation

From Table 4, it was clear that Kasami codes have lower cross-correlation values when compared with maximal codes of the same length. Autocorrelation of code should be as high as possible for signal detection and to identify the intended user with lower side lobe levels.

When the spreading sequence is correctly assigned with the received signal, it provides peak SNR for the intended receiver; hence, it is not possible for other users to decode the signal.

Codes with low cross-correlation are preferred in CDMA multi-user systems to distinguish signals between different users at the receiver and thus avoid interference. Higher values of cross-correlation result in more interference, hence reducing system capacity.

Table 4. Comparison of cross correlation of Kasami sequences with maximal codes

SR (Length)	Kasami CCF	Maximal Preferred Pair (PP) CCF
2 (3)	Only one seq.	No preferred pair
4 (15)	3,-1,-5	No preferred pair
6 (63)	7,-1,-9	15,-1,-17
8 (255)	15.-1,-17	No preferred pair
10 (1,023)	31,-1,-33	63,-1,-65
12 (4,095)	63,-1,-65	No preferred pair
14 (16,383)	127,-1,-129	255,-1,-257
16 (65,535)	255,-1,-257	No preferred pair
18 (2,62,143)	511,-1,-513	1023,-1,-1025
20 (10,48,575)	1023,-1,-1025	No preferred pair
22 (4,194,304)	2047,-1,2049	4095,-1,4097

Table 5. Comparison of normalised peak cross-correlation values of gold and Kasami codes

SR	Kasami Normalised CCF peak	Gold Normalised CCF peak	Improvement in Peak CCF for Kasami Codes w.r.t. Gold Codes
2	-	-	-
4	0.2	-	-
6	0.1111	0.2381	53.34%
8	0.0588	-	-
10	0.0303	0.06158	50.79%
12	0.01538	-	-
14	0.00772	0.01556	50.38%
16	0.00389	-	-
18	0.00195	0.00390	50.06%
20	0.000975	-	-
22	0.000488	0.0009763	50.01%

From Table 5, it was clear that the cross-correlation of Kasami codes may take one of the three values, i.e.,

$$2^{\frac{n}{2}} - 1, -1, \ -(2^{\frac{n}{2}} + 1).$$

4. Kasami Code Sequences of Different Lengths and Number of Balanced/Unbalanced Codes

4.1. Shift Registers =4 and Tap Combinations = (4, 1)

C_1 = [1 0 1 0 1 0 0 0 0 1 1 0 1 0 0] - Non-balanced {6 ones and 9 zeros} (say sequence of type 1)

C_2 = [1 1 0 0 0 1 0 1 1 1 0 1 1 1 1] - Non-balanced {10 ones and 5 zeros} (say sequence of type 2)

C_3 = [0 1 1 1 0 0 1 1 0 0 0 0 0 1 0] - Non-balanced {6 ones and 9 zeros}

4.2. Shift Registers =4 and Tap Combinations = (4, 3)

C_1 = [0 1 1 1 1 1 1 0 1 1 1 0 1 0 0] - Non-balanced {10 ones and 5 zeros} (say the sequence of type 2)

C_2 = [1 0 1 0 0 1 0 1 1 0 0 0 0 1 0] - Non-balanced {6 ones and 9 zeros} (say sequence of type 1)

C_3 = [1 1 0 0 1 0 0 0 0 0 1 1 0 0 1] - Non-balanced {6 ones and 9 zeros}

4.2.1. Result 1

For a 4-bit shift register for any valid tap combination, there are 3 Kasami codes, out of which 2 six one's sequences and 1 ten one's sequences.

4.3. Shift Registers =6 and Tap Combinations = (6, 5)

C_1 = [0 1 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1 1 1 0 1 0 1 0 0 1 1 0 0 1 1 1 1 0 0 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 0 0] Non balanced {28 ones and 35 zeros}(say the sequence of type 1)

C_2 = [1 0 1 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 0 1 1 1 1 1 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 0 1 0 1 1 1 0 0 1 1 0 1 1 0 1 1 0 0] Non balanced {36 ones and 27 zeros}(say the sequence of type 2)

C_3 = [1 1 0 1 0 1 1 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 0 1 1 0] - Non balanced {36 ones and 27 zeros}

C_4 = [1 1 1 0 1 1 0 1 1 1 1 0 0 0 1 0 1 1 1 0 1 0 0 0 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 1 0 1 1] - Non balanced {28 ones and 35 zeros}

C_5 = [0 1 1 1 0 0 0 0 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 1 1 0 1 0 1 1 0 0 1 0 0 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1] - Non balanced {28 ones and 35 zeros}

C_6 = [0 0 1 1 1 1 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 1 1 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 1 1 1 0 1 0 1 1 0 1 0 1 0 0 0 0 1 0 0 0 1 0] - Non balanced {28 ones and 35 zeros}

C_7 = [1 0 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 1 1 0 0 0 1 1 1 1 0 1 1 1 1 1 0 0 0 1] - Non balanced {36 ones and 27 zeros}

4.4. Shift Registers =6 and Tap Combinations = (6, 5, 3, 2)

C_1 = [1 1 0 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 0 1 1 1 1 1 1 0 1 1 0 1 1 0 0 1 1 0 0 0 1 0] - Non balanced {36 ones and 27 zeros}

C_2 = [1 1 1 0 0 0 0 0 0 1 1 0 0 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 0 0 0 1 0 1 1 0 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1] - Non balanced {36 ones and 27 zeros}

C_3 = [0 1 1 1 0 1 1 1 0 0 0 1 1 1 0 0 1 0 0 1 1 0 1 0 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 1 1 1 1 1 0] - Non balanced {28 ones and 35 zeros}

C_4 = [1 0 1 1 1 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 1 1 0 1 0 1 0 1 1 0 1 1 0 1 1] - Non balanced {36 ones and 27 zeros}

C_5 = [0 1 0 1 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 1 1 1 0 1 1 0 0 1 0 0 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 0 1 1 1 0 1 0 1 0 0 1] - Non balanced {28 ones and 35 zeros}

C_6 = [0 0 1 0 1 0 1 1 1 0 1 0 0 1 0 1 1 1 1 0 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0] - Non balanced {28 ones and 35 zeros}

C_7 = [1 0 0 1 0 0 1 0 1 1 0 1 0 1 1 1 0 0 0 0 1 1 0 1 0 0 1 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0] - Non balanced {28 ones and 35 zeros}

4.4.1. Result 2

For the 6-bit shift register of any valid tap combination, there are 7 Kasami codes, out of which 4 sequences have twenty one's and 3 sequences have thirty-six ones.

Similarly, either 28 one's sequence or 36 one's sequence is generated from all other valid taps of the 6-bit shift register, i.e., (6,1), (6,5,4,1), (6,5,2,1) (6,4,3,1).

4.5. Shift Registers = 8 and Tap Combinations = (8,6,5,4) Taps

C_1 = [0 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0 1 1 0 1 1 0 1 0 0 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 0 0 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0 1 0 1 0 0 1 1 0 1 1 0 0 1 0 0 0 0 1 1 1 0 0 1 1 1 0 1 0 0 1 1 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 0 0 0 1 1 0 1 1 0 1 0 0 0 1

1 0 0 0 1 0 1 0 1 1 1 1 0 1 1 0 1 1 1 0 0 1 1 1 0 1 1 0 1 0
0 1 0 0 1 1 0 0 0 1 1 1 0 1 1 0 0 1 1 0 1 0 0 1 0 1 0 0 1 1
0 0 1 1 1 1 0 0 0 1 0 0 0 1 0] {120 ones and 135 zeros}
(say sequence of type 1)

C_2 = [1 0 0 0 1 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 1 0 0 0 1 1 0 1 1 1
1 1 1 0 1 1 0 1 0 0 1 1 0 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0
0 0 1 1 1 1 0 0 0 0 1 1 1 0 0 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0
0 1 0 1 1 1 1 0 1 1 1 0 1 0 1 1 0 0 1 0 0 1 0 1 0 1 1 0 0 0
1 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 1 1 1 0
1 1 0 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1 1 0 1 0 1 0 0 1
0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 1 0 1 0 0 0 1 0
1 1 0 1 0 1 1 0 1 0 0 0 0 1 1 1 0 1 0 1 1 1 0 0 1 0 1 0 1 1
1 0 1 0 0 1 1 0 1 0 1 1 0 1 0] {120 ones and 135 zeros}

C_3 = [1 1 0 0 0 1 0 1 1 1 0 0 1 0 1 1 1 1 0 0 0 0 1 0 0 0 1 0 1
1 1 0 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 1 0 0 1 1 1 1 1 1
0 0 1 1 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 1 0 1 1 0 1 1 0 0 0
0 0 0 0 1 0 0 1 1 1 0 0 1 0 0 1 1 1 0 1 1 1 1 1 1 1 0 0 1 0
0 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1
0 1 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 0 1 1 0 1 0 0 0 1 0 1 0
1 0 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 1 0 0 1 0 0 0 0 1 1 1 1
0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 1 0 1 1
1 1 1 1 0 1 0 1 1 1 1 0 0 1 1 0] {136 ones and 119 zeros}
(say sequence of type 2)

C_4 = [1 1 1 0 0 0 1 1 0 1 1 1 0 1 1 1 1 0 0 0 1 1 1 1 0 1 0 1 0 1
1 0 0 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 0 1 0 1 0 0 1 0 0 0 0 0
0 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0
0 0 1 1 0 1 0 1 0 0 1 0 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 0 1 0
1 1 0 1 1 0 1 1 1 1 1 0 0 0 0 0 1 0 0 0 1 1 0 0 0 1 1 1 0 0
1 0 1 1 0 0 0 1 1 1 0 1 0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 0 1 1
0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0
1 0 1 1 1 1 0 1 0 1 0 0 0 0 1 1 1 0 0 0 1 0 1 1 0 0 1 0 0 1
1 1 0 0 1 1 0 1 0 1 1 1 0 0 0] {136 ones and 119 zeros}S=136

C_5 = [1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 1 0 1 0 1 0 0 1 1 1 1 0 1 0
1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 1
0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1
0 0 1 0 0 1 1 0 0 1 1 1 0 0 0 1 1 1 0 1 0 1 0 0 0 1 0 1 0 1
1 1 0 0 1 0 0 0 1 0 1 1 1 1 1 0 1 0 1 0 1 0 1 0 1 1 0 0 1 1
1 0 1 0 0 0 1 0 1 0 0 0 1 1 1 0 1 1 0 1 1 1 1 1 1 0 0 1 0 0
0 1 1 0 1 0 0 0 1 0 0 1 1 1 0 1 0 0 1 0 0 0 1 1 1 0 1 1 1 1
1 0 1 0 1 1 1 0 0 0 0 1 1 1 0 1 1 0 1 0 1 1 0 1 1 0 0 1 1 0
1 1 0 1 1 1 1 0 0 0 1 0 1 1 1] {136 ones and 119 zeros}

C_6 = [0 1 1 1 1 0 0 1 1 0 0 0 0 1 1 0 1 0 1 1 1 0 1 0 1 0 1 1 0 1
0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0
1 1 0 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1 1 0 1 1 1 0 0 1 0 1 1 0
1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 1 1 0 0 0 0 1 0
0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 1 1 1 0 0 1 1 0 0 1 0 0
0 0 1 0 1 0 1 1 0 0 1 0 0 0 0 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1
1 1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0
0 0 1 0 0 1 1 1 1 0 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 0 1
0 1 0 1 0 1 1 1 1 0 0 0 0 0 0] {120 ones and 135 zeros}

C_7 = [1 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 0 0 1 1 0
1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 1 1 0 1 0 0 0 1 1 0 0 1 1
0 0 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1
0 1 1 0 1 0 1 1 0 0 0 0 1 0 0 1 0 1 0 0 1 1 1 0 1 0 1 0 0 1
1 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 1 1 1 1
1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 0 1 0 0 0 1 0 1 0 1 1 0 0 0
0 0 1 0 0 1 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1 0 0 1 0 1 0 0 1 1
1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 1 0 0 1 1 0 1 1 1 0 1 1 0 1 0
1 0 0 1 0 0 1 1 0 1 0 1 0 1 1] {120 ones and 135 zeros}

C_8 = [0 1 0 1 1 1 1 1 0 0 1 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 0 0 1 1
0 0 1 1 1 0 1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 1 0
1 1 1 0 1 0 1 1 1 0 1 1 0 0 0 0 1 0 1 0 0 0 1 1 0 0 1 0 0 0
1 0 0 0 1 0 0 1 0 1 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 1 1 1 0 0
0 1 1 0 0 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 1 0 0 1 1 1 0 1 0
0 0 0 0 1 1 0 1 1 0 0 1 1 1 0 1 1 0 0 0 0 0 0 1 1 0 1 1 0 1
1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 1 1 1 1 1 0 1 1 0 0 1 1 0
0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 1 1 1
0 1 1 1 0 0 0 1 0 0 1 1 1 1 0] {136 ones and 119 zeros}

C_9 = [1 0 1 0 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 1 0 1 0 0
1 1 1 0 0 1 0 1 1 1 0 0 0 1 0 0 0 1 1 0 0 1 1 1 0 0 1 1 1 0
0 0 0 0 1 1 0 1 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1
0 0 1 1 1 1 0 0 0 0 1 0 1 0 1 1 1 0 1 1 0 1 0 0 0 0 0 0 1 1
0 1 0 0 1 0 1 1 0 1 0 0 1 1 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0
0 1 1 1 1 1 1 0 0 1 0 1 0 1 0 0 0 0 1 1 0 0 0 1 1 1 1 0 1 1
1 0 0 1 1 0 1 1 0 1 0 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0
0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 0 0 0 1 1 1 0 1 0
1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0] {120 ones and 135 zeros}

C_{10}= [1 1 0 1 0 1 1 0 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 0 1 0 0 1 0
0 1 0 1 1 0 0 1 1 0 0 0 1 0 0 1 0 0 0 1 1 1 1 1 1 1 0 1 0 0 0
1 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1
1 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 0 1 0 0 1 0 1
1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 1 0 1 1 1 0 0 1 1 1 1 0 1 1 0
1 1 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 1 0 0 1 0 0 1 0 1 1 1 0 1
0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 0 1 0 1 1 0 1 1 1 0 1 1 0 0 0
1 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0
0 1 1 1 1 1 0 0 0 1 0 0 1 0 0 1] {120 ones and 135 zeros}

C_{11}= [0 1 1 0 1 0 1 0 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 1
0 0 0 0 0 1 1 1 1 0 1 0 1 1 1 1 1 0 1 0 0 0 1 1 1 1 1 0 1 1
1 1 1 0 1 1 1 1 0 0 1 0 1 0 0 1 0 1 1 0 0 1 0 0 0 1 1 1 0 0
1 1 0 1 1 1 1 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 0 1 1 0
1 0 1 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 0 1
1 0 0 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 0 1 1 1 0
0 1 1 1 1 0 0 1 0 0 1 1 0 1 1 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1
1 0 0 1 1 0 1 0 0 1 1 1 0 1 1 0 0 1 0 0 1 1 0 0 0 0 1 1 1 1
0 0 1 0 0 0 1 0 0 1 1 0 1 1 1 1] {136 ones and 119 zeros}

C_{12}= [0 0 1 1 0 1 0 0 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0
1 0 1 0 1 0 0 0 1 0 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 0 0 1 0
0 1 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1
0 1 1 1 0 0 0 1 0 1 0 1 0 0 1 1 0 0 1 0 1 1 1 0 1 1 1 1 1 1
0 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 0 0
0 0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 1 0 1 1 0 0 0 1 1 1
1 1 0 1 0 1 1 0 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 1 0
0 0 1 1 0 1 0 1 0 1 1 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 1 1 0
1 0 0 0 1 1 0 1 0 1 1 1 1 1 0 0] {120 ones and 135 zeros}

C_{13}= [1 0 0 1 1 0 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0 1 1 0 0
0 1 1 1 1 1 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 0 1 0 1 1 0 1 1 0
1 0 0 1 0 1 1 1 1 0 1 1 0 0 1 1 1 0 0 1 0 1 0 1 0 1 0 0 0 1
1 0 1 0 0 1 1 0 1 1 0 1 1 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0 1 1
1 1 0 1 0 0 0 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 1 0 1 0 1 0 0 0
1 1 1 0 0 1 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0

1 1 1 0 0 0 1 0 1 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 0 0 0 0 1 0
0 1 0 1 1 0 1 0 1 1 1 1 0 1 0 1] {136 ones and 119 zeros}

C_{14}= [0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 1 0 1 1 1 0
0 0 0 1 0 1 0 0 1 1 1 1 0 0 0 1 1 0 0 0 0 1 0 1 0 1 0 1 0 0
1 1 1 1 1 1 0 0 0 1 1 1 0 1 1 1 0 1 0 0 0 0 1 0 1 1 0 0 1 1
1 1 0 0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 0 1 0 1 1 0 0 1 1 0 0 1
1 0 1 1 1 0 1 0 0 1 1 1 1 0 0 1 1 1 1 0 1 0 0 1 0 0 1 0 1 0
1 0 0 0 1 1 1 1 0 1 1 0 0 0 0 1 1 1 0 1 0 0 1 1 1 0 0 0 0 1
0 1 1 0 1 0 1 0 0 1 1 0 1 0 0 0 0 0 1 0 1 1 0 1 1 0 0 1 0 0
1 0 0 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 1 0 1 0 1 0 1 0 0 0 0 0
0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 1] {120 ones and 135 zeros}

C_{15}= [0 0 1 0 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1
0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 1 1 1 1 0 1 1 1 0 1 0 0 1 0 1
1 1 0 0 1 0 0 1 1 0 0 1 0 1 0 1 0 0 1 0 1 0 0 1 0 0 0 0 1 0
1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 0 1 1 0 1 0 0 0
1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 1 1 1 0 1 1
1 0 1 1 1 0 1 0 1 0 0 0 0 0 1 1 1 0 1 1 1 0 0 0 0 1 0 0 0 0
0 1 0 1 1 1 1 1 1 0 0 0 1 0 1 0 0 1 0 0 0 1 1 0 0 1 0 1 0 1
1 0 1 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1
0 0 0 0 0 1 0 0 1 1 0 1 0 0 1 1] {120 ones and 135 zeros}

4.5.1. Result 3

For the 8-bit shift register of any valid tap combination, there are 15 Kasami codes, out of which 8 sequences have one hundred twenty one's and 7 sequences have one hundred and thirty-six one's. Similarly, either 120 one's sequence or 136 one's sequence is generated from all remaining 15 valid taps of the 8-bit shift register, i.e., (8,7,6,1) (8,6,5,4) (8,6,5,3) (8,6,5,2) (8,6,5,1) (8,4,3,2)….etc.,

From above, the following general formulae are obtained:

1. It is clear that all Kasami codes generated with any 'n' bit shift register are non-balanced codes having the length of $2^n - 1$.

2. There are two types of sequences: one with less number of ones (type 1) and another with more number of ones (type 2).

3. In type 1 sequence, i.e., Kasami code with less number of ones than zeros
   (i) The total number of ones are $(2^{n-1} - 2^{\frac{(n-2)}{2}})$.
   (ii) The total zeros: $(2^n - 1) - (2^{n-1} - 2^{\frac{(n-2)}{2}})$

4. In type 2 sequence, i.e., Kasami code with much higher ones than zeros
   (i) The total number of ones are $(2^{n-1} + 2^{\frac{(n-2)}{2}})$.
   (ii) The total zeros: $(2^n - 1) - (2^{n-1} + 2^{\frac{(n-2)}{2}})$.

From Table 6, it is clear that the Whole quantity of Type 1 sequences are $2^{\frac{(n-2)}{2}}$ and Entire number of Type 2 sequences are $2^{\frac{(n-2)}{2}} - 1$.

Table 6. Total No. of type 1 and type 2 Kasami codes obtained with various length LFSR

SR	No. of Kasami Codes for Each Valid Tap	Total type 1 codes (seq. with less No. of one's)	Total Type 2 Codes (seq. with more No. of one's)
4	3	2	1
6	7	4	3
8	15	8	7
10	31	16	15
12	63	32	31
14	127	64	63
16	255	128	127

5. Proposed Method of Converting Non-Balanced Kasami Code to Balanced Kasami Code

All existing research work till now presents only on non-balanced kasami codes and never proposed a method to convert the non-balanced kasami codes to balanced codes. Consider Kasami code C_1 of length $2^m - 1$ and Kasami code C_2 of length $2^m - 1$, append code C_1 and code C_2 with zero i.e., [C_1 C_2 0] generates higher length balanced Kasami code of $2^{m+1} - 1$.

The balance property of spreading codes means the number of ones and zeros should differ by one, i.e., Even distribution of autocorrelation function values (For example, Maximal and gold codes), and this property helps to test the random nature of spreading code. This balanced nature of codes reduces interference and cross-talk and improves efficiency in high-capacity multi-user wireless communication CDMA systems.

5.1. Proof

Choose Kasami code C_1 = [0 1 1 1 1 1 1 0 1 1 1 0 1 0 0] having 10 ones and 5 zeros (sequence of type 2) and Kasami code C_2 = [1 0 1 0 0 1 0 1 1 0 0 0 0 1 0] having 6 ones and 9 zeros (sequence of type 1). Append C_1 with C_2 and append 0, i.e., [C_1 C_2 0] = [0 1 1 1 1 1 1 0 1 1 1 0 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 1 0 0]. The proposed sequence has 16 one's and 15 zero's. The difference between the total number of ones and the total zeros is one. It satisfies the balance property. Using 15 lengths of non-balanced Kasami codes of type 1 and type 2, 31 length balanced code is generated. Similarly Consider 'n=6' bit shift register with valid tap combinations = (6, 5)

Consider Kasami code C_1 = [0 1 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1 1 1 0 1 0 1 0 0 1 1 0 0 1 1 1 1 0 0 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 0 0] having 28 ones and 35 zeros (say sequence of type 1) and C_2 = [1 0 1 0 0 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 1 0 0 1 1 1 1 1 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 0 1 0 1 1 1 0 0 1 1 0 1 1 0 1 1 0 0] having 36 ones and 27 zeros (say sequence of type 2) append C_1 with C_2 and append 0 i.e.,

[C_1 C_2 0] = [0 1 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1 1 1 0 1 0 1 0 0
1 1 0 0 1 1 1 1 0 0 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 1
0 0 1 0 0 0 1 1 0 0 0 1 0 1 0 0 0 1 1 0 1 1 1 1 1 1
1 1 1 0 1 0 0 1 0 0 1 1 1 1 1 0 1 0 1 0 1 0 0 0 1 0
1 0 1 0 0 1 0 1 1 1 0 0 1 1 0 1 1 0 1 1 0 0].

The proposed sequence has 64 one's and 63 zero's. The difference between the total number of one's and the total zeros is one. It satisfies the balance property. Using 63 length-balanced Kasami codes of type 1 and type2, 127 length balanced code is obtained. The same method is followed for n=8, 10, 12… shift register.

6. Conclusion

The Kasami codes have a lower cross-correlation value when compared with maximal codes of the same length and approximately a 6dB Improvement in the ACF peak value of the side lobe with respect to the gold code side lobe peak value. For 'n' bit shift register {length of code =$(2^n - 1)$} of even length, there are $(2^{\frac{n}{2}} - 1)$ kasami codes generated from each valid tap of linear feedback shift register and all the Kasami codes generated with any 'n' bit shift register are non-balanced codes having the length of $2^n - 1$. Out of which $2^{\frac{(n-2)}{2}}$ are Type 1 sequences and $2^{\frac{(n-2)}{2}} - 1$ are Type 2 sequences.

Appending any type 1 sequence with any type 2 sequence and containing zero as the last bit, the new balanced sequence can be obtained having the length of $(2^{n+1} - 1)$.

The future scope of work includes studying the correlation analysis of proposed balanced codes and evaluating the performance of proposed balanced kasami codes in multi-user wireless communication systems with various modulation schemes in different channel conditions.

References

[1] D. Sri Kavya, and P. Siddaiah, "Implementation of Seven Finger RAKE Receiver using MRC Technique," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no.3, pp. 1689-1693, 2019. [CrossRef] [Publisher Link]

[2] I.A. Alimi, J.J. Popoola, and K.F. Akingbade, "A Power Efficient Rake Receiver for Interference Reduction in the Mobile Communication Systems," *International Journal of Electronics and Electrical Engineering*, vol. 3, no. 6, pp. 501-505, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[3] Vaibhav Khairanr, Jitendra Mathur, and Hema Singh, "Bit Error Rate Performance Analysis of CDMA Rake Receiver," *International Journal of Engineering Science Invention*, vol. 3, no. 6, pp. 52-58, 2014. [Google Scholar] [Publisher Link]

[4] Xiangyong Zeng, John Qingchong Liu, and Lei Hu, "Generalised Kasami Sequences: The Large Set," *IEEE Transactions on Information Theory*, vol. 53, no. 7, pp. 2587-2598, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[5] Busim Ananta Lakshmi et al., "Design and Analysis of Rake Receiver," *International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, Erode, India, pp. 1020-1027, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[6] Kyungwhoon Cheun, "Performance of Direct-Sequence Spread-Spectrum Rake Receivers with Random Spreading Sequences," *IEEE Transactions on Communications*, vol. 45, no. 9, pp. 1130-1143, 1997. [CrossRef] [Google Scholar] [Publisher Link]

[7] U. Grob et al., "Microcellular Direct-Sequence Spread-Spectrum Radio System Using N-Path RAKE Receiver," *IEEE Journal of Selected Areas in Communications*, vol. 8, no. 5, pp. 772-780, 1990. [CrossRef] [Google Scholar] [Publisher Link]

[8] B.H. Khalaj, A. Paulraj, and T. Kailath, "2D RAKE Receivers for CDMA Cellular Systems," *IEEE Conference on Global Communications (GLOBECOM)*, San Francisco, CA, USA, vol. 1, pp. 400-401, 1994. [CrossRef] [Google Scholar] [Publisher Link]

[9] Xinyue Li, Yajie Yan, and Deyue Zou "A Master-Slave Rake Receiver for Integrated Navigation/Communication Signal," *International Wireless Communications and Mobile Computing (IWCMC)*, Harbin City, China, pp.1070-1074, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] V. Umadevi, and P. Easwaran, "A Study on Rake Receivers," *IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE)*, Karur, India, pp. 1-5, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[11] J. Lehnert, and M. Pursley, "Multipath Diversity Reception of Spread-Spectrum Multiple-Access Communications," *IEEE Transactions on Communications*, vol. 35, no. 11, pp. 1189-1198, 1987. [CrossRef] [Google Scholar] [Publisher Link]

[12] Tengku Azita Tengku Aziz, and Abdul Halim Ali, "A New Rake Receiver Design for Long Term Evolution - Advance Wireless System," *IEEE Symposium on Wireless Technology and Applications (ISWTA)*, Langkawi, Malaysia, pp. 52-55, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[13] G.E. Bottomley et al., "A Generalized RAKE Receiver for Interference Suppression," *IEEE Journal on Selected Areas of Communication*, vol. 18, no. 8, pp. 1536-1545, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[14] Ahmed Faraz, "Performance Metrics of RAKE Receivers," *International Multi-Disciplinary Conference in Emerging Research Trends (IMCERT)*, Karachi, Pakistan, pp. 1-4, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] P. Nilsson, and T. Maseng, "RAKE Receiver CDMA Performance," *5th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Wireless Networks - Catching the Mobile Future*, The Hague, Netherlands, vol. 2, pp. 696-699, 1994. [CrossRef] [Publisher Link]