Binary Sequences With Low Aperiodic Autocorrelation for Synchronization Purposes

Şükrü Ekin Kocabaş and Abdullah Atalar

***Abstract*—An evolutionary algorithm is used to find three sets of binary sequences of length 49–100 suitable for the synchronization of digital communication systems. Optimization of the sets are done by taking into consideration the type of preamble used in data frames and the phase-lock mechanism of the communication system. The preamble is assumed to be either a pseudonoise (PN) sequence or a sequence of 1's. There may or may not be phase ambiguity in detection. With this categorization, the first set of binary sequences is optimized with respect to aperiodic autocorrelation which corresponds to the random (PN) preamble without phase ambiguity case. The second and third sets are optimized with respect to a modified aperiodic autocorrelation for different figures of merit corresponding to the predetermined preamble (sequence of 1's) with and without phase ambiguity cases.**

***Index Terms*—Aperiodic autocorrelation, binary sequences, evolutionary algorithm, modified aperiodic autocorrelation, phase ambiguity, preamble, synchronization.**

I. Introduction

BINARY sequences with low autocorrelation are used extensively in the synchronization of digital communication systems and the modulation of radar pulses. There are two frequently used binary sequence classes. The first class is named pseudonoise (PN) sequences. They have optimum *periodic* autocorrelation property and are used in synchronization when the sequence can be sent several times in succession. The second class has optimized *aperiodic* autocorrelation property and is used when the synchronization sequence is to be sent only intermittently [1]. Barker sequences, with their peak sidelobe level (PSL) of unity, are the best known examples of the second class. No Barker sequence of length greater than 13 exists up to length 12 100 [1], [2] and that is why, it is generally assumed that there are no undiscovered Barker sequences.

For many applications, sequences of length greater than 13 are needed. There is no known analytical technique to construct sequences with low aperiodic autocorrelation, and exhaustive searches have to be made in order to find the *least autocorrelated binary sequence* (LABS) for a given length. Several optimization techniques have been adapted to the LABS problem [2]–[6]. Among those, evolutionary algorithm performs the best [2]. In this paper we present three sets of sequences of length 49–100 obtained by the use of the evolutionary algorithm. The first set of sequences are less correlated compared to the ones in [2] and are suitable for synchronization in digital communication systems without phase ambiguity and with random (PN) frame preamble. The second and third sets are optimized with respect to a modified aperiodic correlation function which we claim is well suited to the synchronization of digital communication systems when the frame preamble is a sequence of 1's.

Manuscript received June 5, 2002. The associate editor coordinating the review of this letter and approving it for publication was Dr. N. Van Stralen.

The authors are with the Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey (e-mail: kocabas@alumni.bilkent.edu.tr).

Digital Object Identifier 10.1109/LCOMM.2002.807438

II. Definitions

On a synchronous data link, there are two levels of synchronization to be achieved: bit synchronization and frame synchronization. Bit synchronization refers to the adjustment of the receiver timing so that it knows whether a 1 bit or a 0 bit is currently received. Frame synchronization allows the receiver to determine which bit of the received bit stream actually starts the data in a data frame [7]. Bit synchronization is obtained by transmitting a bit sequence, which is called *preamble*, in the beginning of data frames. Frame synchronization is achieved by transmitting a predetermined sync sequence, $\{a_n\}$, of length N before the data to be transmitted starts. A data frame can be depicted as [preamble, sync seq., DATA].

The receiver calculates the correlation of the incoming bits in its input shift register of length N with the sync sequence $\{a_n\}$ held in a reference register. The peak in the correlation, when input and reference registers have similar bits, signals the start of valid data.

The aim is to find $\{a_n\}$ with minimal correlation values when there is nonzero offset between the two copies of $\{a_n\}$. During the search for suitable $\{a_n\}$, the structure of the preamble should be taken into account. In this paper we investigate two types of preambles: pseudo-noise and predefined. We assume that the pseudo-noise preambles behave like random sequences and do not contribute to the correlations calculated by the receiver. For predefined preambles, we only investigate the case when the preamble is a sequence of 1's.

Another property which should be considered when searching for $\{a_n\}$ is the phase ambiguity of the communication system. That is, whether it is possible for the receiver to interpret a 0 sent as a 1 and vice versa. The preamble and the phase ambiguity properties of the communication system together give rise to the following cases:

1) random preamble with phase ambiguity;
2) random preamble without phase ambiguity;
3) predefined preamble with phase ambiguity;
4) predefined preamble without phase ambiguity.

We map the 0's of $\{a_n\}$ to -1's and define the following correlation functions accordingly.

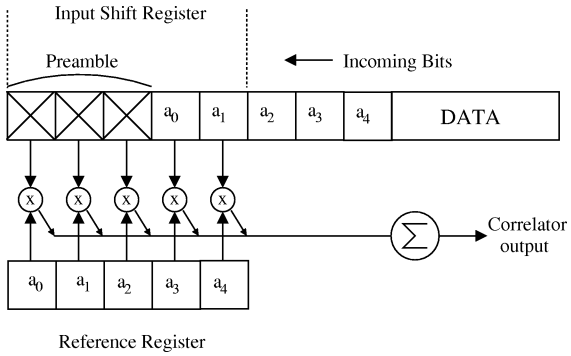

Fig. 1. The correlation calculated at the receiver for $N = 5$ and $\tau = 3$.

When the preamble is random, the correlation calculated by the receiver (Fig. 1) for a binary sequence $\{a_n\}$ of length N is the *aperiodic autocorrelation function* (ACF). For $0 \leq \tau \leq N-1$, ACF is given by;

$$C_a(\tau) = \sum_{n=0}^{N-\tau-1} a_n a_{n+\tau}.$$

When there is phase ambiguity, the received sequence may be interpreted as the inverse of the original sequence which is $-\{a_n\}$. In such a case, the receiver will calculate $-C_a(\tau)$. If the magnitude of the ACF is bounded, the receiver can resolve the ambiguity by determining the sign of the correlation peak. That is the reason the figure of merit for case 1 is the *peak sidelobe level* (PSL);

$$\text{PSL} = \max_{\tau \neq 0} \mid C_a(\tau) \mid .$$

For case 2, it is not necessary to bound ACF in absolute value. Therefore, a relaxed version of PSL (PSL^r) can be used as the figure of merit;

$$\text{PSL}^r = \max_{\tau \neq 0} C_a(\tau).$$

If the preamble is a sequence of 1's, then the correlation calculated by the receiver is the *modified aperiodic autocorrelation function* (MACF). It is defined as:

$$M_a(\tau) = \begin{cases} C_a(0) = N, & \tau = 0 \\ C_a(\tau) + \sum_{n=0}^{\tau-1} a_n, & \tau \in [1, N-1] \\ \sum_{n=0}^{N-1} a_n, & \tau = N. \end{cases}$$

When there is phase ambiguity, the absolute value of MACF should be bounded. The figure of merit for case 3 is the *modified peak sidelobe level* (MPSL);

$$\text{MPSL} = \max_{\tau \neq 0} \mid M_a(\tau) \mid .$$

For case 4, a relaxed version of MPSL (MPSL^r) can be used;

$$\text{MPSL}^r = \max_{\tau \neq 0} M_a(\tau).$$

The most suitable sequences for synchronization purposes have the smallest figures of merit.

III. Algorithm

Evolutionary algorithms can be applied to the optimization of many discrete problems where an analytical solution is not available. For the problem at hand, we chose to employ the *plus evolution strategy* with crossover [6]. The algorithm is a modified version of the one used in [2]. The implementation is as follows:

1) Generate an initial population of P parent sequences at random.
2) Apply crossover to two randomly chosen parent sequences to create two offsprings by selecting a random position and splicing the section that appears before the selected position in the first parent with the section that appears after the selected position in the second parent, and vice versa. Repeat this operation until O offsprings are produced.
3) Evaluate the figure of merit of $P + O$ sequences.
4) Apply competition to the population by setting the P sequences with the lowest figures of merit as the parents of the next generation.
5) If more than one parent share the same figure of merit, keep one of the parents untouched and for every other parent with the same figure of merit:
   - If the figure of merit is less than or equal to the aimed figure of merit, A, mutate the parent by randomly flipping one of its bits.
   - Else mutate the parent by randomly flipping two of its bits.
6) If number of generations that have been evolved is less than G go to step 2), else terminate.

In our simulation, we chose $P = 10N$, $O = 100N$ and $G = 10N$. The A values for each sequence were initially given as 7. In the subsequent runs of the program we modified A values by looking at the best result of the previous run and setting A equal to one less than the best result found so far.

IV. Results

We have obtained three sets of binary sequences of lengths 49–100 by running the evolutionary algorithm for PSL^r, MPSL and MPSL^r. The least significant N bits of the hexadecimal numbers in Table I specify the sequences found. In the tables, the length of the sequences (N), the figure of merit (M) and an example sequence are given.

V. Conclusion

By using the evolutionary algorithm, we found binary sequences suitable for the synchronization of communication systems. During the search for the sequences, we took into account the structure of the preamble and the phase ambiguity of the system and defined respective figures of merit for different cases. For case 1 defined in Section II, the sequences presented in [2] are suitable. For case 2, the sequences in Table I(a) are suitable. For case 3, the sequences in Table I(b) are suitable. Finally, for case 4 the sequences in Table I(c) are suitable. Obviously, if the predefined preamble is a sequence of 0's, the complement of the sequences in Table I(b) and c can be used.

TABLE I
SEQUENCES OPTIMIZED WITH RESPECT TO THE (A) PSL^r, (B) MPSL, and (C) MPSL^r CRITERIA

a)

N	M	Example Sequence	N	M	Example Sequence	N	M	Example Sequence
49	3	0669FEAA465A1	67	4	710FD61528A189B1E	85	5	13CA5561FCD4F1A0090CD9
50	3	03FB972E62925	68	5	CC63F74536F49E04A	86	5	0057FA1E439CBB6B361D12
51	3	5DD4129C61F23	69	4	0F4A570C876434FBBB	87	5	7B54B15352465C22AD87F3
52	3	3A277AFE11258	70	4	248EFDC49A0E86A539	88	5	F60633407235D4CD3AF249
53	3	05965DC8DD786A	71	5	01D1D137DA756F3908	89	6	0879D591763D4DA6CDE2261
54	4	212A2C708CCBE6	72	5	0EF280B1D5BBB1E4C2	90	6	0CEBDD104741AF653F17894
55	4	6346734D8F8250	73	5	1C27E8AD3082CDEE5C5	91	6	3E125A38BD4FDD483239833
56	4	CABEE40CC74B2D	74	5	346D4939A8967CBE63E	92	6	D62C6690F4992A2357DF12D
57	4	1C52F614C36CF43	75	5	4DACE55D0B07BF0832C	93	6	1F1729710275A5CB29AC65D3
58	4	094D1A98EE9F220	76	5	9C427F2863EB76E8B05	94	6	24A162E33B28E7ADF01BA604
59	4	653158FAB072CF0	77	5	0A64F45D12BF5B78C216	95	6	1148E535E5E1FE213B63C65D
60	4	3B089B2175FAD38	78	5	1D37496F43A9819E5C0B	96	6	CAAF60998AE0C065FD6DE38D
61	4	0B54FEB8259CC4C1	79	5	623F8F28A40C6C4996AE	97	6	19DD941243C4FB1655FA6C6A7
62	4	139A01CD8B2AB96B	80	5	20EB1EFAB0FA751999A1	98	6	19A463C91095FE2EA5647E945
63	4	414FF2B9B2CF4284	81	5	13EE5C24985C41CA627AF	99	6	7690A688F45C1F3975DBB112C
64	4	7CD593ADF7818AC8	82	5	332702FD8349287559E77	100	5	9D9FC48B797A0D6058CB2EBC6
65	4	033A6A629B7D58BC0	83	5	7B3ED031226730B49E157			
66	4	3289FEB5CD7879312	84	5	4E3638F5AABFC0704CAC8			

b)

N	M	Example Sequence	N	M	Example Sequence	N	M	Example Sequence
49	5	1DA0ACCB15838	67	7	4FE6108B5AC990BAA	85	7	1F807654AE18C6D65F1D30
50	4	39911AE92BC3E	68	6	C9F0A1B32D5D433DF	86	8	3B8CCD055AD3E482F1F60E
51	5	4FB0A295BF191	69	7	1F810D7944EC9B5D8E	87	7	7B07AA06277568691DB668
52	6	9F59260ADEE07	70	6	3B6481C3B8B4277423	88	8	F43C176994C69F9A15C508
53	5	1F03A4D333A30A	71	7	378CF845A145BBBA06	89	7	1ED83056CCD61DD109E8768
54	6	17C74923997E01	72	6	CF09F0232ED736151D	90	8	165B5D1E08C3337E95A9980
55	5	3F324863F82957	73	7	0FD20A5C5D0736F32DC	91	7	6989BEA119A1AACFD659070
56	6	5E67026CABDB42	74	6	3AB40DF22396736505 8	92	8	F23A12D9655C70EC513BC40
57	5	1DA123760BA6A39	75	7	794AA09EB61C36C6C47	93	7	1369CAD1C9371390FBE41D00
58	6	3D0D78070BAD931	76	6	CF3A1B006ED675406B7	94	8	353CB8C40F4F68DC49BB28EA
59	5	7925E44563CE360	77	7	162F972434594E35AB00	95	7	7C9511950BE6DA13C58EF804
60	6	E92751738562FCB	78	6	3C355A83E63230F6D9A0	96	8	FA4D0147AEA16BC693989D12
61	5	1CC6C5542F0DB700	79	7	3B1BE2960A86D74C7B20	97	7	17E08D456E70353E659ACD397
62	6	0F5B7919041FCABD	80	6	E1DE8291576D70992D8F	98	8	2B3C35A3D1C89327EA2BE8404
63	5	7D019CE1EB46C955	81	7	1B99602C5CBC69C05F510	99	7	7AF0C216B33B82F22AEDD125F
64	6	D5AECC23826D6F4C	82	6	2E62B4A2F1C13E4FA8910	100	8	EEE1A81AD4F21F3412B8D9B01
65	7	18B5D48499EB70B84	83	7	2FF0605652FC7194CBD22			
66	6	3C07C6C7369953502	84	6	FC5035E158BA31CC96C29			

c)

N	M	Example Sequence	N	M	Example Sequence	N	M	Example Sequence
49	-1	004C550B6C729	67	-1	0069E0958DD56698D	85	-1	000B033AC83D1B8E9526A6
50	-2	0068ACC9563C9	68	-2	048499AD0EA8F661D	86	-2	02085B22B709D70726A473
51	-1	0288DA41ED59C	69	-1	00889C37569A2F9CAD	87	-1	0093190EC2D1875975283D
52	-2	018C9A5C54ED1	70	-2	00471923565C2E25AE	88	-2	00B161AC1CCCB1794CA3A6
53	-1	01054E91C989E5	71	-1	009C29668986BB9D0D	89	-1	00256A0C71647B15A236D1E
54	-2	0026A3438D65C9	72	-2	0221D8C35651AB0FB2	90	-2	0024C9A3AD2752E273AA1EC
55	-1	00B4C2A764D45C	73	-1	0028971B0ACE20E5491	91	-1	00A188D836AA72E135A621F
56	-2	0432CE46C5A175	74	-2	010B49709F188CBA36A	92	-2	020CB11AB60F115B4B44C75
57	-1	002A471E46BB6B2	75	-1	008A9310F3C2D8A6E94	93	-1	00223AC3350AB644E5B78386
58	-2	0052C93A9CF0BB9	76	-2	01230AE349AEC5A33E5	94	-2	0028A16CC6938BB4A9B1C4BC
59	-1	00C352317B255C3	77	-1	003215172E25AE1992DD	95	-1	00531824DCD61592AF488F0F
60	-2	01315349D6071AC	78	-2	000D193E16D3558E62D7	96	-2	00705856AA3232C7269B521D
61	-1	0105437991CB8972	79	-1	00851D25D6C6E176E58E	97	-1	009195017C8D0B8F79AB84B66
62	-2	0043917934AEB163	80	-2	0118A68EB41B3635C16B	98	-2	004A41C6724D954ABE3487CE4
63	-1	005AA232F92E469C	81	-1	000A9874AC16D468B3991	99	-1	0012655361B1711EA4E7094F3
64	-2	0511C916A4C7D08E	82	-2	0021C2998DA26CA9C56C3	100	-2	010664D2C672A4AAD770B3C17
65	-1	00818CA6D47C8F2A5	83	-1	0068A5AC7137648775CAC			
66	-2	0044CB0F42ACE1359	84	-2	0144BB18594FA1CBA625E			

It is interesting to note that the optimization in the MPSL^r sense [Table I(c)] resulted in MPSL$^r = -1$ for N odd and MPSL$^r = -2$ for N even for all N values we studied.

REFERENCES

[1] D. Wiggert, *Codes for Error Control and Synchronization*. Norwood, MA: Artech House, 1988, pp. 177–181.

[2] X. Deng and P. Fan, "New binary sequences with good aperiodic autocorrelations obtained by evolutionary algorithm," *IEEE Commun. Lett.*, vol. 3, no. 10, pp. 288–290, Oct. 1999.

[3] H. Deng, "Synthesis of binary sequences with good autocorrelation and crosscorrelation properties by simulated annealing," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 32, no. 1, pp. 98–107, Jan. 1996.

[4] F. Hu, P. Z. Fan, M. Darnell, and F. Jin, "Binary sequences with good aperiodic autocorrelation functions obtained by neural network search," *Electron. Lett.*, vol. 33, no. 8, pp. 688–690, Apr. 1997.

[5] S. Mertens, "Exhaustive search for low-autocorrelation binary sequences," *J. Phys. A: Math. Gen*, vol. 29, no. 18, pp. 473–481, Sept. 1996.

[6] B. Militzer, M. Zamparelli, and D. Beule, "Evolutionary search for low autocorrelated binary sequences," *IEEE Trans. Evol. Comput.*, vol. 2, no. 1, Apr. 1998.

[7] R. A. Williams, *Communication Systems Analysis and Design: A Systems Approach*. Englewood Cliffs, New Jersey: Prentice-Hall, 1987, pp. 371–372.