Cross-Joins in de Bruijn Sequences and Maximum Length Linear Sequences

Taejoo CHANG†, Ickho SONG†, Nonmembers, Hyung Myung KIM†, Member and Sung Ho CHO††, Nonmember

SUMMARY In this paper, a construction of de Bruijn sequences using maximum length linear sequences is considered. The construction is based on the well-known cross-join (CJ) method: Maximum length linear sequences are used to produce de Bruijn sequences by the CJ process. Properties of the CJ pairs in the maximum length linear sequences are investigated. It is conjectured that the number of CJ pairs in a maximum length linear sequence is given by $(2^{n-3}+1)/3-2^{n-3}$, where $n \geq 2$ is the length of the linear feedback shift register associated with the sequence. The CJ pairs for some special cases are obtained. An algorithm for finding CJ pairs is described and a method of implementation is discussed briefly.

key words: cross-join, de Bruijn sequences, m-sequences, shift register

1. Introduction

The $n$-th order binary de Bruijn graph $G_n$ is a directed graph with $2^{n+1}$ arcs (edges) and $2^n$ vertices (nodes) of all binary $n$-tuples. Two arcs are directed from $x=(x_0, x_1, \ldots, x_{n-1})$ to $y=(x_1, x_2, \ldots, x_{n-1}, y_{n-1})$, where $y_{n-1}=0, 1$. A de Bruijn cycle is a closed path visiting each vertex of $G_n$ once and only once, and may usually be represented by a de Bruijn sequence.

A binary de Bruijn sequence of span $n$ is a binary sequence $(s_i)$ of period $2^n$ such that the $n$-tuples $s_i=(s_0, s_1, \ldots, s_{n-1})$ for $i=0, 1, \ldots, 2^n-1$ are all distinct. It is well known [1] that the number of distinct de Bruijn sequences of span $n$ is given by $2^{2^n-n}$. De Bruijn sequences may be generated by nonlinear feedback shift registers. The de Bruijn sequences, which correspond to maximum length shift register sequences, have many applications including stream ciphers [2], [3]. Much effort has been devoted to the problem of synthesizing de Bruijn sequences [2], [4].

A linear shift register sequence of length $2^n - 1$ can be converted into a de Bruijn sequence by inserting an extra 0 in the longest run of 0's [5]. Many authors have suggested several algorithms to produce de Bruijn sequences. One of the algorithms is as follows: start with $n$ zeros and append a one as the next bit of the sequence as long as the $n$-tuple so-formed has not previously appeared; otherwise append a zero. The sequences obtained from this algorithm are called the lexicographically least de Bruijn sequences [6]. This algorithm was generalized in [2] to form all de Bruijn sequences using the preference function technique. A method of constructing de Bruijn sequences by joining the cycles of a pure summing register has been proposed in [7]. In [8], the lexicographically least de Bruijn sequence was redefined and $2^{2^n-n}$ de Bruijn sequences corresponding to the sequence was generated by the cross-join process. In addition, a recursive construction of de Bruijn sequences by a mapping can be found in [e.g., 9]. In this method, a new de Bruijn sequence of span $n+1$ can be obtained from a de Bruijn sequence of span $n$.

In this paper, a construction of de Bruijn sequences by the cross-joining process with the maximum length sequences [2] is considered. In Sect. 2, basic concepts of sequences and finite fields are investigated briefly. In Sect. 3, cross-join properties in the maximum length sequences are described. In Sect. 4, an algorithm for finding cross-join pairs in the maximum length sequences is discussed.

2. Basic Concepts

2.1 Feedback Shift Registers

A feedback shift register (FSR) of length $n$ is a collection of $n$ binary storage elements, $x_0, x_1, \ldots, x_{n-1}$, together with an associated feedback function. The storage elements of an FSR are called the stages, and the vector composed of the contents of the stages is called the state of the FSR. An FSR is called linear if the feedback function has the form

$$f(x_0, x_1, \ldots, x_{n-1}) = c_0x_0 + c_1x_1 + \cdots + c_{n-1}x_{n-1},$$

(1)

where $+$ denotes the modulo-2 addition and $c_i \in \{0, 1\};$ otherwise it is called nonlinear.
Fig. 1 A general form of a binary $n$-stage FSR.

Fig. 2 The de Bruijn diagrams of order $n=1, 2, 3, 4$.

At each clock, the value of $x_{i+1}$ is transferred to $x_i$, for $i=0, 1, \ldots, n-2$, and $x_{n-1}$ is set to the value $f(x_0, x_1, \ldots, x_{n-1})$ computed just before the transition. A general form of a binary $n$-stage FSR is shown in Fig. 1. The dependence on the clock is not shown in the figure for simplicity.

The conjugate $\bar{x}$ and the companion $x'$ of a state $x=(x_0, x_1, \ldots, x_{n-1})$ are defined by [4]

$$\bar{x}=(\bar{x}_0, \bar{x}_1, \ldots, \bar{x}_{n-1})$$

(2)

and

$$x'=(x_0, x_1, \ldots, \bar{x}_{n-1}),$$

(3)

respectively, where $\bar{x}_i$ is the binary complement of $x_i$. A state has two possible successors (the two companion states at just one clock after) and two possible predecessors (the two conjugate states at just one clock before).

The state diagram showing superposition of all possible states and transitions between states for a linear FSR is often referred to as the de Bruijn diagram [1], as shown in Fig. 2, where de Bruijn diagrams are shown for $n=1, 2, 3, 4$.

2.2 Maximum Length Linear Sequence

A binary linear sequence is a binary sequence which satisfies a recurrence relation of the form

$$s_{k+n}=\sum_{i=0}^{n-1} c_i s_{k+i}, \quad k=0, 1, 2, \ldots$$

(4)

where $c_i \in \{0, 1\}$ and the arithmetic is performed in $GF(2)$, the binary Galois field. The polynomial

$$f(x)=c_0+c_1x+\cdots+c_{n-1}x^{n-1}+x^n$$

(5)

of degree $n$ with the coefficients $c_i$ in $GF(2)$ is referred to as the characteristic polynomial of the sequence.

A maximum length linear sequence or $m$-sequence of length $2^n-1$ is a linear sequence which satisfies a linear recurrence whose characteristic polynomial of degree $n$ is primitive [2], that is, for an $m$-sequence the characteristic polynomial is an irreducible polynomial and divides $(x^n-1)$ for $k=2^n-1$ but for no smaller $k$ [10]. The number of binary primitive polynomials of degree $n$ [11] is known to be $\phi(2^n-1)/n$, where $\phi$ is Euler's $\phi$-function. (In other words, $\phi(m)$ is the number of integers in the set $\{1, 2, \ldots, m\}$ which are relatively prime to $m$). In general de Bruijn sequences and $m$-sequences are synthesized by FSRs with nonlinear and linear feedback functions, respectively. The $m$-sequences have many useful properties and wide areas of applications. A considerable literature may be found on these sequences [e.g., 12].

2.3 The Cross-Join Construction

The cross-join (CJ) construction is a method to generate another de Bruijn sequence from a de Bruijn sequence. In a de Bruijn sequence, consider two conjugate state pairs $[\theta, \theta_1]$ and $[\theta, \theta_2]$ connected by one line (chord) each, where $\theta_1$ and $\theta_2$ are the conjugate states of $\theta$ and $\theta_2$, respectively. If the two chords cross each other, then $[\theta, \theta_1]$ and $[\theta, \theta_2]$ are called a pair of cross-joins or a CJ pair. We will denote a CJ pair by $[\theta, \theta_1; \theta, \theta_2]$ or $[\theta, \theta_2]$ briefly. A CJ construction is to convert one de Bruijn sequence to another de Bruijn sequence by interchanging the successors of the conjugate state pairs in a CJ pair. For example, if

$$\theta:=(\theta_0, \theta_1, \ldots, \theta_i, \theta_{i+1}, \ldots, \theta_j, \theta_{j+1}, \ldots, \theta_k)$$

is a de Bruijn cycle, then the new cycle obtained by the
CJ pair \([\theta_i, \theta_{i+1}, \theta_{i+2}, \theta_{i+3}]\) is

\[
\theta_{i+1} = (\theta_0, \theta_1, \ldots, \theta_i, \theta_{i+1}, \ldots, \theta_{i+3} = \theta_0, \theta_1, \ldots, \theta_i, \theta_{i+3})
\]

for which the CJ construction is shown graphically in Fig. 3. The CJ construction is based on the following Lemmas [2].

**Lemma 1:** A single cycle \(C\), with \(x\) and \(\bar{x}\) on \(C\), is divided into two cycles when the successors of \(x\) and \(\bar{x}\) are interchanged.

**Lemma 2:** Two cycles \(C_1\) and \(C_2\), with \(x\) on \(C_1\) and \(\bar{x}\) on \(C_2\), merge into a single cycle when the successors of \(x\) and \(\bar{x}\) are interchanged.

In this paper, we apply the CJ construction to obtain de Bruijn sequences from \(m\)-sequences. This seems to be a reasonable approach because an \(m\)-sequence can always be converted into a de Bruijn sequence by adding the all-zero state and thus we may apply the CJ construction to \(m\)-sequences in order to obtain a new de Bruijn sequence.

3. Cross-Join Properties in \(m\)-Sequences

3.1 Representation of \(m\)-Sequences

For mathematical treatment, we will represent \(m\)-sequences using concepts in finite fields. Let \(s_t\), \(t=0, 1, \ldots\), be an \(m\)-sequence satisfying the recurrence relation (4) and the primitive polynomial corresponding to this sequence is given by (5).

It is known [13] that a finite field of \(2^n\) elements can be obtained from an \(m\)-sequence if we add the \(n\)-tuple of all-zero (the all-zero state), and that there is a one-to-one correspondence between the \(n\)-tuples of \(n\) successive elements in an \(m\)-sequence, and the powers of \(a\), where \(a\) is a fixed root of the primitive polynomial corresponding to the \(m\)-sequence, i.e., \(f(a) = 0\). Thus \(m\)-sequences can be represented by

\[
\theta_0, \theta_0 a, \theta_0 a^2, \ldots, \theta_0 a^{2^n-2},
\]

where \(\theta_0\) is the initial state of the \(m\)-sequence.

3.2 Enumeration of CJ Pairs in \(m\)-Sequences

For small values of \(n\), an experimental program is executed as an initial attempt to obtain the number of CJ pairs in \(m\)-sequences by considering all possible \(\phi(2^n-1)/n\) \(m\)-sequences. The result is given in Table 1.

An observation on the experimental values is as follows. Let \(f_n\) be the number of CJ pairs in an \(m\)-sequence of the register with length \(n\), then Table 1 shows that \(f_{n+1}/f_n\) is approximately equal to 4 as \(n\) increases. Based on this observation, we have the following conjecture.

**Conjecture 1:** The number of CJ pairs in \(m\)-sequences of length \(2^n-1\) is

\[
f_n = \frac{1}{3} (2^{2n-3} + 1) - 2^{n-2}
\]

\[
= \frac{(2^{n-1} - 1)(2^{n-1} - 2)}{(2^n - 1)(2^{n-2})}, \quad n = 1, 2, \ldots. \tag{6}
\]

It is interesting to note that Eq. (6) is the same as the formula for the number of distinct sub-groups of \(G_{n-1}\) of order of \(2^2\), where \(G_{n-1}\) is a group of binary \((n-1)\)-tuples under the operation of addition modulo-2.

3.3 Cross-Join Pairs in \(m\)-Sequences

Let us now consider the CJ pairs in \(m\)-sequences for some special cases.

**Definition 1:** The distance between two states \(\theta_i\) and \(\theta_j\) in an \(m\)-sequence, denoted by \(d(\theta_i, \theta_j)\), is defined by the integer \(d_i\) such that

\[
\theta_i a^{d_i} = \theta_j, \quad 0 \leq d_i \leq 2^n - 2,
\]

where \(n\) is the length of the FSR and \(a\) is a fixed root of the primitive polynomial corresponding to the \(m\)-sequence.

Definition 1 means that \(\theta_i\) is the \(d_i\)-shifted value of \(\theta_i\). The distance between a state and itself is zero, and if \(d(\theta_i, \theta_i) = d_i\) then it is easy to see that \(d(\theta_i, \theta_i) = 2^n - 2 = d_i\). Also, we define \(\theta_i = (10\cdots0)\) for notational convenience. Obviously, \(\theta = \theta + \theta = 1\) for all \(\theta\).

**Lemma 3:** The distance between a state \(\theta \neq \theta_i\) and its conjugate state \(\bar{\theta}\) is unique in \([1, 2^n - 2]\).

**Proof:** Let \(k = d(\theta, \bar{\theta})\). Then we have \(\theta a^k = \bar{\theta} = \theta + \theta_i\), or \(a^k = 1 + \theta \bar{\theta} = 1 - 1\). Obviously \(k\) is not zero since \(\theta \neq \theta_i\). Since the number of states except the state \(\theta_i\) in an \(m\)-sequence is \(2^n - 2\), we have the result.

**Lemma 4:** If \(d(\theta, \bar{\theta}) = 2\) and \(n > 2\), then \([\theta, \theta a]\) is a CJ pair.

<table>
<thead>
<tr>
<th>(n)</th>
<th># of CJ pairs</th>
<th>(n)</th>
<th># of CJ pairs</th>
<th>(n)</th>
<th># of CJ pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>6</td>
<td>155</td>
<td>11</td>
<td>174251</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>7</td>
<td>651</td>
<td>12</td>
<td>(\alpha 080)</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>8</td>
<td>2667</td>
<td>13</td>
<td>1794155</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>9</td>
<td>10795</td>
<td>14</td>
<td>11180715</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>10</td>
<td>43435</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\[ \theta = \theta_c a^r, \quad 1 \leq i \leq j - 1. \]  
(11)

From (9) and (11), we have \( a^{-i} = (1 + a^r)^{-1} \), or \( a^r + a^{j-i} = 0 \). This is a contradiction since \( i < j < n \).

case 2: Assume that \( \theta_a \) exists between \( \theta \) and \( \theta \). Then \( \theta_a = \theta_c (1 + a^r)^{-1} \), and there exists an integer \( i \) such that

\[ \theta = \theta_a a^{-i}, \quad 1 \leq i \leq j - k - 1. \]  
(12)

From (9) and (12), we have \( (1 + a^r)^{-1} - a^{-i} = (1 + a^r)^{-1} \), or \( a^r + a^{j-k+1} = 0 \), which is also a contradiction.

**Theorem 2:** The pair \([\theta, \theta_a]\) is a CJ pair if

\[ d(\theta, \theta_a) = 2^{n-1} - 1 \]  
(13)

and

\[ d(\theta, \theta_b) = d(\theta_a, \theta_b) - 1 \]  
(14)

Proof: Assume that \([\theta, \theta_a]\) is not a CJ pair when (13) and (14) are true. Then \( \theta, \theta_a, \theta_b, \) and \( \theta_a \) occur in this order and we have the following configurations:

\[ \theta_a = \theta b \]  
(15)

and

\[ \theta_b = \theta a. \]  
(16)

Now, letting \( p = 2^{n-1} - 1 \) for notational convenience, we have \( \theta_a = \theta_c (1 + a^p)^{-1} \) and \( \theta_b = \theta_c (1 + a^p)^{-1} \) from (13) and (14), respectively. Using these equations and \( \theta_a = \theta_b a^p \) in (15) we get \( a + a^p + a^{p+1} + a^{2p+1} = 0 \). Since \( a^{2p+1} = a^{2p} - 1 \), we have \( (1 + a) (1 + a^p) = 0 \), which is a contradiction since \( a + 1 \) and \( a^p \) are.

**Lemma 5:** Let \( S_n \) be the set of all states in an \( m \)-sequence. Let \( g: S_n \to S_n \) be a mapping such that

\[ g(\delta) = \begin{cases} \theta a & \text{if } \theta = \theta_c, \theta_a, \theta_b, \\ \theta_b a & \text{otherwise,} \end{cases} \]  
(17)

where \( \theta_i \) is defined by (10) with \( k = 1 \). Then \( g \) is bijective.

Proof: Let \( T = (\theta_c, \theta_a, \theta_b) \). First, we show that if \( g(\gamma) = g(\delta) \) then \( \gamma, \delta \in S_n - T \) or \( \gamma, \delta \in T \). Suppose \( \gamma \in S_n - T \) and \( \delta \in T \), then we have \( (\gamma + \theta_a) a = \delta a \). It thus follows that \( \gamma = \delta \). Now we have \( \delta + \theta_a \) since \( \gamma \in S_n - T \), and we have \( \gamma \in T \) since \( \delta \in T \). This is a contradiction. Next, we show that \( g \) is injective. Suppose \( g(\gamma) = g(\delta) \), that is, suppose

\[ (\gamma + \theta_a) a = (\delta + \theta_a) a \quad \text{if } \gamma, \delta \in S_n - T \]  
(18)

\[ \gamma a = \delta a \quad \text{if } \gamma, \delta \in T, \]  
then we have \( \gamma = \delta \), from which it follows that \( g \) is injective. Let us now take any element \( \gamma \in S_n \). Then

\[ \gamma = (\gamma a^{-1} + \theta_c + \theta_a) a \quad \text{if } \gamma \in S_n - T \]  
(19)

\[ \gamma = a \gamma a^{-1} \quad \text{if } \gamma \in T, \]

and consequently every \( \gamma \in S_n \) is equal to \( g(\theta) \) for
\[ \theta = r^{-1} + \theta_c \quad \text{if} \quad r \in S_n - T \]
\[ \theta = r^{-1} \quad \text{if} \quad r \in T. \]

Therefore \( g \) is surjective. Since \( g \) is injective and surjective, it is bijective.

Because we know the states \( \theta_c \) and \( \theta_i \) explicitly, we get the following.

**Corollary 1**: Let \( f_i(x_0, x_1, \ldots, x_{n-1}) \) be the feedback function of an \( m \)-sequence. Then the sequence generated by the feedback function
\[ f_d(x_0, x_1, \ldots, x_{n-1}) = f_i(x_0, x_1, \ldots, x_{n-1}) + 1 + x_1x_2\cdots x_{n-1}, \]
is a de Bruijn sequence.

**Proof**: Since the states \( \theta_i \) and \( \theta_c \) are \((01 \cdots 11)\) and \((10 \cdots 00)\), respectively, we have by a CJ process (i.e., by interchanging the successors of the conjugate state pairs in a CJ pair).

\[ f_d(x_0, x_1, \ldots, x_{n-1}) = f_i(x_0, x_1, \ldots, x_{n-1}) + x_1x_2\cdots x_{n-1} \quad \text{(add the all-zero state to m-sequence)} \]
\[ + 1 \quad \text{(assume all states are CJ pairs)} \]
\[ + x_1x_2\cdots x_{n-1} \quad \text{(skip the states \( \theta_c \) and \( \theta_i \))} \]
\[ + x_1x_2\cdots x_{n-1} \quad \text{(skip the states \( \theta_i \) and \( \theta_i \))} \]
\[ = f_i(x_0, x_1, \ldots, x_{n-1}) + 1 + x_1x_2\cdots x_{n-1}. \]

As an example, let us consider the two \( m \)-sequences of shift registers with lengths of 4 and 5 as shown in Figs. 4 and 5, respectively. The primitive polynomials corresponding to the sequences are \( f(x) = x^3 + x + 1 \) and \( f(x) = x^5 + x^3 + 1 \), respectively, and the feedback functions are \( f(x_0, x_1, x_2, x_3) = x_0 + x_1 + x_3 \) and \( f(x_0, x_1, x_2, x_3, x_4) = x_0 + x_2 + x_3 \). In Figs. 4 and 5, the sequences are shown, and all conjugate pairs are explicitly connected by straight lines (chords). We can easily check the states and the CJ pairs described above in the figures.

4. An Algorithm for Finding Cross-Join Pairs

In this section, an algorithm for finding CJ pairs by a trial method is considered, and a method for implementation of the CJ sequences is discussed briefly.

4.1 Basic Operations in Finite Fields

To obtain the CJ pairs described in Sect. 3, a finite field computation is required. There exist several methods for computation over a finite field [14]. Since an \( m \)-sequence is represented by the dual basis, we can use dual basis or standard basis. In either case, the basis conversion is required, and to implement the basis conversion, the trace computation and matrix inversion over \( GF(2^n) \) are required. We also need an algorithm for multiplication, which can be written as follows:

**Algorithm GFMUL**

\[ [w = u \times v] \]

(A) \( w = 0 \)
(B) for \( i = 0 \) to \( n - 1 \)
(C) if \( v_i = 1 \) then \( w = w + u \)
(D) \( u = u \times a \)

For exponentiation in \( GF(2^n) \), the well-known binary method [15] can be used. In this method, repeated squaring of the partial results is used to reduce the required number of multiplications. For example, since \( 11 = 2^3 + 2^2 + 2^1 \), we have \( u^{11} = ((u^2)^2)^2 \cdot u^2 u \). Let \( e = (e_0, e_1, \ldots, e_n) \) be the binary representation of \( e \). Then an algorithm to compute \( w = u^e \) can be written as follows:

**Algorithm GFEXP**

\[ [w = u^e] \]

(A) \( w = e \)
(B) \( s = u \)
(C) \( w = 1 \)
(D) if \( e_0 = 1 \) then \( w = w \times s \)
(E) for \( i = 1 \) to \( n - 1 \)
\( \{ \)
(F) \( s = s \times s \)
(G) if \( e_i = 1 \) then \( w = w \times s \)
\( \} \)

4.2 An Algorithm for Finding CJ Pairs

Two states \( \theta_i \) and \( \theta_j \) form a CJ pair only if the four states, \( \theta_i, \theta_j, \theta_i, \theta_j \), occur in the order \( [\theta_i, \theta_j, \theta_i, \theta_j] \). To find the states which cross-join \( \theta_i \), we generate states successively until \( \theta_j \) is found. If the distance between \( \theta_i \) and \( \theta_j \) is very large, therefore, it is quite time-consuming to find CJ pairs. Thus we search states \( \theta_i \) for which \( [\theta_i, \theta_i] \) is a CJ pair with the distance between \( \theta_i \) and \( \theta_i \) limited. An algorithm to search for the CJ pairs, which can also be used for counting all the CJ pairs (of which the number is shown in Table 1) with some modifications, is as follows.

**Algorithm SCJ**

[Searches the CJ pairs \( [\theta, \theta] \) such that \( d(\theta, \theta) = d_0 \) given. Also searches the CJ pairs \( [\theta_i, \theta_i] \) such that \( d(\theta_i, \theta_i) < d_0 \) and \( \theta_i \) is in between \( \theta \) and \( \theta \).] Given \( d_0 \).

(A) Find the state \( \theta_i \) such that \( d(\theta_i, \theta_i) = d_0 \).
(B) Initialize: \text{SHIFT REGISTER} \rightarrow \theta_i; \text{SLIST} \rightarrow \theta_i.
(C) shift \text{SHIFT REGISTER}.
(D) \( \theta \rightarrow \text{SHIFT REGISTER} \).
(E) if \( \theta \) is not on \text{SLIST} then insert \( \theta \) to the end of \text{SLIST}.

Table 2 A result of algorithm SCI.

<table>
<thead>
<tr>
<th>loop</th>
<th>SHIFT_REGISTER(θ)</th>
<th>ˆθ</th>
<th>SLIST</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0011</td>
<td></td>
<td></td>
<td>(0011) none</td>
</tr>
<tr>
<td>1</td>
<td>0110</td>
<td>1110</td>
<td>(0011 0110) none</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1101</td>
<td>0101</td>
<td>(0011 0100 1101) none</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1010</td>
<td>0010</td>
<td>(0011 0100 0100 1101) none</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0011</td>
<td>1101</td>
<td>(0011 0100 0100) none</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1011</td>
<td>0011</td>
<td>(0011 0100 0100) none</td>
<td></td>
</tr>
</tbody>
</table>

else print ˆθ and states right of ˆθ in SLIST.
(F) remove ˆθ from SLIST.
(G) if ˆθ not equal to 0 then goto (C).

The algorithm requires nd bits of storage for SLIST.
To illustrate the algorithm, consider the m-sequence generated from \( f(x) = x^5 + x + 1 \), shown in Fig. 4.
Assume \( d_0 = 5 \). Then \( \theta = 0011 \) and the contents of various variables are shown in Table 2.

4.3 Computational Results

The routine for computing over a finite field described in Sect. 4.1 is implemented with a program written in C
and assembler languages for multiprecision arithmetic on an IBM-PC/AT. The algorithm SCI is also implemented.
As a test sequence, the m-sequence corresponding to the primitive polynomial \( f(x) = x^127 + x + 1 \) is used.
Details of the results of the test are shown in Appendix.

4.4 Implementations

It is simple to implement a cross-joined sequence (a sequence obtained by a CJ pair of another sequence).
Namely, we add the all-zero state to an m-sequence and toggle next bit if the current state is in the CJ pair.
Let an indicator function \( \phi_\theta(x_1, x_2, \cdots, x_{n-1}) \) be
\[
\phi_\theta(x_1, x_2, \cdots, x_{n-1}) = 1
\]
if and only if
\[
\theta = (x_1, \cdots, x_{n-1}).
\]
Let \( f_c(\cdot) \) be the feedback function of an m-sequence
and let \( [\theta_i, \theta_j] \) be a CJ pair. Then the feedback function \( f_c(\cdot) \) of a cross-joined sequence may be written as
\[
f_c(x_0, x_1, \cdots, x_{n-1}) = f_c(x_0, x_1, \cdots, x_{n-1}) \quad (m\text{-sequence})
\]
\[+ \phi_\theta(x_1, \cdots, x_{n-1}) \quad \text{(add the all-zero state)}
\]
\[+ \phi_{\hat{\theta}}(x_1, \cdots, x_{n-1}) \quad \text{(cross join \( \theta \) and \( \hat{\theta} \))}
\]
\[+ \phi_{\hat{\theta}}(x_1, \cdots, x_{n-1}) \quad \text{(cross join \( \hat{\theta} \) and \( \hat{\theta} \))},
\]
where 0 is the all-zero state. The implementation uses about 3n bits of storages and requires 3n comparisons
to generate next bit.

5. Conclusion

In this paper, a construction method of de Bruijn sequences by applying the cross-join method to m-sequences is considered. The m-sequences are easy to tract mathematically since they form finite fields. The number of cross-join pairs in m-sequences is conjectured based on an observation from computer simulation: the conjecture is recently proved in [16]. Some cross-join pairs were found by a trial method, in which the distance is restricted. This restriction on distance may be removed if we employ the discrete-logarithm when we find cross-join pairs. The cross-joined sequences have a large segments of linear sequences. This possible deficiency may be improved by finding a large number of independent cross-join pairs at the expense of more storage and computation time for implementation.

It would be an interesting problem to find cross-join pairs for given constraints, for example under the constraint that we are required to use minimum terms in feedback function, etc. Another interesting problem is to compare the performance of various sequence construction methods of de Bruijn sequences.

Acknowledgements

This research was supported by Korea Advanced Institute of Science and Technology (KAIST) under an Internal Grant and by Korea Science and Engineering Foundation (KOSEF) under Grant 921-0800-002-2, for which the authors would like to express their thanks. The authors also wish to thank the anonymous reviewers for their constructive comments and suggestions which improved the quality of this paper.

References

### Appendix

The states in Theorems 1 and 2 are shown for the m-sequence corresponding to the primitive polynomial $f(x) = x^{127} + x + 1$. The period of the sequence is $2^{127} - 1$, and the state values are given in hexadecimal.

<table>
<thead>
<tr>
<th>state (B)</th>
<th>state (B) such that $d(k, B) = d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>distance (d)</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5555 5555 5555 5555 5555 5555 5555 5555</td>
</tr>
<tr>
<td>3</td>
<td>4e92 9e92 9e92 9e92 9e92 9e92 9e92 9e92</td>
</tr>
<tr>
<td>4</td>
<td>5Mr6 5Mr6 5Mr6 5Mr6 5Mr6 5Mr6 5Mr6 5Mr6</td>
</tr>
<tr>
<td>5</td>
<td>5e92 5e92 5e92 5e92 5e92 5e92 5e92 5e92</td>
</tr>
<tr>
<td>6</td>
<td>5555 5555 5555 5555 5555 5555 5555 5555</td>
</tr>
<tr>
<td>7</td>
<td>4e92 4e92 4e92 4e92 4e92 4e92 4e92 4e92</td>
</tr>
<tr>
<td>8</td>
<td>5Mr6 5Mr6 5Mr6 5Mr6 5Mr6 5Mr6 5Mr6 5Mr6</td>
</tr>
<tr>
<td>9</td>
<td>5e92 5e92 5e92 5e92 5e92 5e92 5e92 5e92</td>
</tr>
<tr>
<td>10</td>
<td>5555 5555 5555 5555 5555 5555 5555 5555</td>
</tr>
<tr>
<td>11</td>
<td>4e92 4e92 4e92 4e92 4e92 4e92 4e92 4e92</td>
</tr>
<tr>
<td>12</td>
<td>5Mr6 5Mr6 5Mr6 5Mr6 5Mr6 5Mr6 5Mr6 5Mr6</td>
</tr>
<tr>
<td>13</td>
<td>5e92 5e92 5e92 5e92 5e92 5e92 5e92 5e92</td>
</tr>
</tbody>
</table>

The states in Theorems 1 and 2 are shown for the m-sequence corresponding to the primitive polynomial $f(x) = x^{127} + x + 1$. The period of the sequence is $2^{127} - 1$, and the state values are given in hexadecimal.
Ikheo Song was born in Seoul, Korea, on February 20, 1960. He received the B.S. (magna cum laude) and M.S.E. degrees in electronics engineering from Seoul National University, Seoul, Korea, in 1982 and 1984, respectively. He also received the M.S.E. and Ph.D. degrees in electrical engineering from the University of Pennsylvania, Philadelphia, PA, U.S.A., in 1985 and 1987, respectively. He was a Member of Technical Staff at Bell Communications Research, Morristown, NJ, U.S.A., in 1987. Since 1988, he has been with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, where he is currently an Associate Professor. His research interests include detection and estimation theory, statistical signal and image processing, and statistical communication theory. He was awarded a University Fellowship from Seoul National University during 1978-1983. He was a recipient of the Korean Honor Scholarship in 1985 and 1986, and of the Korean American Scholarship in 1986. He received the Young Scientists Awards from the Union Radio Scientifique Internationale (URSI) in 1989 and 1990 and the Academic Award from the Korean Institute of Communication Sciences in 1991. He served as the Treasurer of the IEEE Korea Section in 1989 and as the Associate Editor for Communications Technology of the Acoustical Society of Korea in 1990.

Hyung Myung Kim was born in Soochang, Korea, on October 24, 1952. He received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1974, and the M.S. and Ph.D. degrees in electrical engineering from the University of Pittsburgh, Pittsburgh, PA, U.S.A., in 1982 and 1985, respectively. He is now an Associate Professor at the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. His research interests include digital signal/image processing, digital image transmission, and multidimensional system theory. He was the Treasurer of the IEEE Daejeon Section in 1992.

Taejoong Chang was born in Ulsan, Korea, on April 20, 1960. He received the B.S.E. degree from Ulsan University, Ulsan, Korea, in 1982 and the M.S.E. degree from Korea Advanced Institute of Science and Technology (KAIST), Seoul, Korea, in 1990, all in electrical engineering. His research interests include coding and information theory.

Sung He Cho was born in Seoul, Korea, on February 21, 1960. He received the B.E. degree in electronics engineering from Hanyang University, Seoul, Korea, in 1982, the M.S. degree in electrical and computer engineering from the University of Iowa, Iowa City, IA, U.S.A., in 1984, and the Ph.D. degree in electrical engineering from the University of Utah, Salt Lake City, UT, U.S.A., in 1989. He was a Senior Member of the Technical Staff of the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea in 1989-1992. He is now an Assistant Professor at the Department of Electronics Engineering, Hanyang University, Ansan, Korea. His research interests include adaptive filtering algorithms and structures, spectrum estimation, mobile communications, and stochastic processes. He is a member of Eta Kappa Nu and Tau Beta Pi.