However, this implies that

$$|\{s|\alpha(s) \in \{0,1,2\} - \{a\} - \{b\}, s \in V_d^T\}| < C_m(d) \quad \text{for all } m \in \{0,1,2\}$$

in contradiction to (A4). □

Choose some x as in the last lemma, and let

$$S_1 = \{s|s \in V^S, \alpha(s) = x\};$$
$$S_2 = S - S_1.$$

Every vertex $s \in V_d^{T_0}$ must have a descendent s' such that $\alpha(s') = x$, and another descendent s'' such that $\alpha(s'') \in \{0,1,2\} - \{x\} - \{\alpha(s)\}$. In other words

for all $c \in C$ there exists $e \in S_1$ such that $e \in c$ and such that $e \in S_2$ there exists $e \in c$.

Therefore we have a set splitting.

The $\langle v \rangle^t$ G-DAG code decision problem can be proven to be NP complete for all $v \geq 3$ by a similar construction in which the binary tree is replaced by a $(v-1)$-ary tree.

References

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability.* San Francisco, CA: Freeman, 1979, pp. 221–222.

[2] R. M. Karp, "Reducibility among combinatorical problems," in *Complexity of Computer Computation*, R. N. Miller and J. W. Thatcher, Eds. New York: Plenum, 1972, pp. 85–104.

[3] L. Lovász, "Covering and colorings of hypergraphs," in *Proc. 4th Southeastern Conf. Combinatorics. Graph theory, and Computing.* Winnipeg, Manitoba: Utilitas Mathematica Publ., 1973, pp. 3–12.

[4] R. L. Rivest and A. Shamir, "How to reuse a "write-once" memory," *Inform. Control*, vol. 55, pp. 1–19, Oct./Nov./Dec. 1982.

Algorithms for the Generation of Full-Length Shift-Register Sequences

TUVI ETZION AND ABRAHAM LEMPEL, FELLOW, IEEE

***Abstract*—Two algorithms are presented for the generation of full-length shift-register cycles, also referred to as de Bruijn sequences. The first algorithm generates $2^{k \cdot g(n,k)}$ full cycles of length 2^n, using $3n + k \cdot g(n,k)$ bits of storage, where k is a free parameter in the range $1 \leq k \leq 2^{((n-4)/2)}$, and $g(n,k)$ is of the order of $n - 2\log k$. The second algorithm generates about $2^{n^2/4}$ full cycles of length 2^n, using about $n^2/2$ bits of storage. In both algorithms, the time required to produce the next bit from the last n bits is close to n. A possible application to the construction of stream ciphers is indicated.**

I. Introduction

This paper deals with the construction of full-length nonlinear shift-register cycles, also referred to as de Bruijn sequences. A comprehensive survey of past work on this subject can be found in [1]. The common approach to this construction is to consider a shift-register producing many short cycles, e.g., the pure cycling register, which are then joined together to form a full cycle.

The same practice is followed in this paper. We propose two methods of constructing full cycles. One produces full cycles by joining those generated by the pure cycling register (PCR); the other employs the pure summing register (PSR) for the same purpose. It is well known [2] that the number of full cycles of length 2^n is $2^{2^{n-1}-n}$. The various methods proposed so far differ in the number of distinct full cycles of same length that the method produces, and in the complexity per produced cycle. Fredricksen [3] shows how to generate 2^{2n-5} full cycles of length 2^n from a PCR of length n (PCR_n) using $6n$ bits of storage, and n units of time to produce the next cycle bit from the last n bits.

In Section III of this paper, we show how to construct $2^{k \cdot g(n,k)}$ full cycles of length 2^n from those of PCR_n using $3n + k \cdot g(n,k)$ bits of storage, where k is a constant in the range $1 \leq k \leq 2^{(n-4)/2}$, $g(n,k)$ is approximately $(n - 2\log k)(1 - (1/(1 + \log k)))$, and logarithms are taken to the base 2. The time required to produce the next bit from the last n bits is $o(n)$, as $n \to \infty$.

In Section IV, we propose a method of constructing full cycles by joining the cycles of a PSR. To the best of the authors' knowledge, this is the first method employing the PSR. We show how to produce about $2^{n^2/4}$, or more exactly,

$$\prod_{k=1}^{\lfloor (n-2)/2 \rfloor} \binom{n-1}{2k}$$

full cycles of length 2^n using about $n^2/2$ bits of storage and $o(n)$ units of time to produce the next bit.

Manuscript received February 8, 1983; revised August 20, 1983.

The authors are with the Computer Science Department, Technion, Israel Institute of Technology, Haifa, Israel.

The number of full cycles and the amount of stored information required to generate them via the proposed algorithms, make it worthwhile to consider their use in cryptographic applications. The main problem in the design of a stream cipher [4] is the construction of the key stream from a so-called "key seed." In our case, the full cycle acts as the key stream, while the stored information required to run the algorithm plays the role of the key seed. The full cycles produced by the proposed algorithms possess some of the important properties desired of key streams. For instance, the number of distinct keys is exponential in the length of the seed which, in the algorithm of Section III, depends on a controllable parameter k.

In addition any full cycle has many of the randomness properties [5] and a large linear span [6] required of key streams.

II. The Joining of Cycles

A feedback shift-register (FSR) of length n has 2^n states corresponding to the set B^n of all binary n-tuples. The feedback function $f(x)$, $x = (x_1, x_2, \cdots, x_n) \in B^n$, of the FSR induces a mapping F: $B^n \to B^n$ under which $xF = y$, where

$$y_i = x_{i+1}, \qquad i = 1, \cdots, n-1, \text{ and } y_n = f(x).$$

The *conjugate* $\hat{x}$ and the *companion* x' of a state $x = (x_1, x_2, \cdots, x_n)$ are defined by

$$\hat{x} = (x_1 \oplus 1, x_2, \cdots, x_n),$$
$$x' = (x_1, \cdots, x_{n-1}, x_n \oplus 1),$$

where $\oplus$ denotes modulo 2 addition.

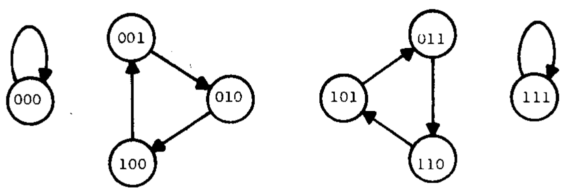
A *k-cycle* C of a FSR is a (cyclic) sequence of k distinct states $C = \langle x_1, x_2, \cdots, x_k \rangle$, $x_i \in B^n$, such that $x_1 = x_k F$ and $x_{i+1} = x_i F$, $i = 1, 2, \cdots, k-1$. The state diagram of a FSR is called a *factor* if each state belongs to a cycle. Two cycles C_1 and C_2 are said to be *adjacent* if they are (state) disjoint and there exists a state x on C_1 whose conjugate $\hat{x}$ or companion x' is on C_2.

Theorem 1 [5]: Two adjacent cycles C_1 and C_2, with x on C_1 and $\hat{x}(x')$ on C_2, are joined into a single cycle when the successors (predecessors) of x and $\hat{x}$ (x') are interchanged.

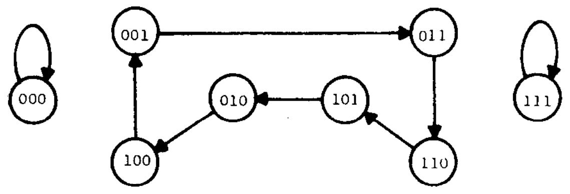
Example 1: Consider the PCR$_3$ with $f(x_1, x_2, x_3) = x_1$. Its state diagram is the factor of Fig. 1(a). Its four cycles are

$$C_1 = \langle 000 \rangle,$$
$$C_2 = \langle 001, 010, 100 \rangle,$$
$$C_3 = \langle 011, 110, 101 \rangle,$$
$$C_4 = \langle 111 \rangle.$$

C_1 and C_2 are adjacent, with 000 being the conjugate of 100. Similarly, C_2 and C_3 are adjacent, with 010 being the companion of 011. Applying Theorem 1 to, say, C_2 and C_3 we obtain the cycle $C = \langle 001, 011, 110, 101, 010, 100 \rangle$. The new factor, consisting of C_1, C, and C_4, is shown in Fig. 1(b).



(a)



(b)

Fig. 1. The factors of Example 1.

III. Construction of Full Cycles from PCR$_n$

The PCR$_n$ is an n-stage FSR whose feedback function $f(x_1, x_2, \cdots, x_n) = x_1$. It is well known [5] that the length of a cycle from PCR$_n$ is a divisor of n.

The *weight $W(S)$ of a state S* is the number of ONES in $S = (s_1, s_2, \cdots, s_n)$, i.e., $W(S) = \sum_{i=1}^{n} s_i$.

Clearly, states belonging to the same cycle of PCR$_n$ have the same weight.

The *weight $W(C)$ of a cycle C* from PCR$_n$ is the weight of each of its states.

Fredricksen [3] shows how to join the cycles of PCR$_n$ to form a full cycle of length 2^n. There are four possibilities for consecutive states on a full cycle (or any other cycle) of a FSR:

a) $(0, x_1, \cdots, x_{n-1}) \to (x_1, \cdots, x_{n-1}, 0)$

b) $(1, x_1, \cdots, x_{n-1}) \to (x_1, \cdots, x_{n-1}, 1)$

c) $(0, x_1, \cdots, x_{n-1}) \to (x_1, \cdots, x_{n-1}, 1)$

d) $(1, x_1, \cdots, x_{n-1}) \to (x_1, \cdots, x_{n-1}, 0)$.

In a) and b) both states come from the same PCR$_n$ cycle. In c) the weight of the second state exceeds by 1 that of the first state. In d) the weight of the first state exceeds by 1 that of the second state. Hence, in c) and d) the two states come from different PCR$_n$ cycles.

Lemma 1: Let C_1 be a cycle of weight $k > 0$ from PCR$_n$. Then there exists a state S on C_1 such that its companion S' is on a cycle C_2 whose weight is $k - 1$.

Proof: Since $W(C_1) > 0$ there exists a state of the form $S = (s_1, \cdots, s_{n-1}, 1)$ on C_1. Hence, $S' = (s_1, \cdots, s_{n-1}, 0)$ and $W(S') = W(S) - 1 = k - 1$. Therefore S' is on a PCR$_n$ cycle C_2, with $W(C_2) = k - 1$. Q.E.D.

Lemma 1 and Theorem 1 lead to a simple way of constructing a full cycle. At each step we have a *main cycle*, obtained by joining a subset of PCR$_n$ cycles, and the remaining PCR$_n$ cycles. Initially, the main cycle is chosen to be the unique PCR$_n$ cycle of weight zero. Next, the main cycle is extended by joining to it the (unique) cycle of weight one. In a general step i, we extend the main cycle by joining to it all the PCR_n cycles of weight i (in arbitrary order). This is always possible because the current main cycle contains all of the states whose weight is less than i and, since each PCR_n cycle of weight $i > 1$ has a state ending in a ONE, it can be joined (see Theorem 1 and Lemma 1) to the current main cycle.

This procedure ends when all the PCR_n cycles have been joined together.

We proceed now to a precise and detailed description of the proposed construction.

Consider the ordered set $V = \{V(i)\}_{i=0}^{k-1}$ of k states, $1 \le k \le 2^{((n-4)/2)}$, constructed as follows:

1) The first $\lceil \log k \rceil + 1$ bits of $V(i)$ form the base-2 representation of i. (Note that the first bit is always ZERO.)
2) The last $\lfloor \log k \rfloor + 2$ bits of each $V(i)$ are ONES preceded by a single ZERO.
3) In positions $\lceil \log k \rceil + 2 + (\lfloor \log k \rfloor + 1)j$, for integers j satisfying
$$0 \le j < \left\lceil \frac{n - \lceil \log k \rceil - \lfloor \log k \rfloor - 3}{\lfloor \log k \rfloor + 1} \right\rceil$$
each $V(i)$ has a ZERO.
4) The remaining bits for each $V(i)$ are chosen arbitrarily.

Example 2: $n = 16$, $k = 8$. The set V for these values of n and k takes the form

$$\begin{array}{c}
00000x_1^{(1)}x_2^{(1)}x_3^{(1)}0x_4^{(1)}x_5^{(1)}01111\\
00010x_1^{(2)}x_2^{(2)}x_3^{(2)}0x_4^{(2)}x_5^{(2)}01111\\
00100x_1^{(3)}x_2^{(3)}x_3^{(3)}0x_4^{(3)}x_5^{(3)}01111\\
00110x_1^{(4)}x_2^{(4)}x_3^{(4)}0x_4^{(4)}x_5^{(4)}01111\\
01000x_1^{(5)}x_2^{(5)}x_3^{(5)}0x_4^{(5)}x_5^{(5)}01111\\
01010x_1^{(6)}x_2^{(6)}x_3^{(6)}0x_4^{(6)}x_5^{(6)}01111\\
01100x_1^{(7)}x_2^{(7)}x_3^{(7)}0x_4^{(7)}x_5^{(7)}01111\\
01110x_1^{(8)}x_2^{(8)}x_3^{(8)}0x_4^{(8)}x_5^{(8)}01111
\end{array}$$

where the $x_j^{(i)}$ are free parameters.

It can be easily verified that the righthand block of $\lfloor \log k \rfloor + 1$ ONES form the unique largest run of ONES in each $V(i)$, and that every pair of states differ in their first $\lceil \log k \rceil + 1$ bits. Therefore we have Lemma 2.

Lemma 2: No two states of V belong to the same cycle of PCR_n.

The construction of a full cycle from the PCR_n cycles proceeds by a sequence of joins where at each step a cycle of least weight from among the remaining PCR_n cycles is joined to the current main cycle. A join is performed by means of a pair of companion states S and S', with S on the next PCR_n cycle C in line and S' on the current main cycle. The states S and S' are called the *bridging* states of the join. The bridging state S on C is determined as follows: if C contains a state from V then it is chosen as the bridging state of C. Otherwise, the choice of S is as in Fredrickson [3], [7]. Let M be the state on C whose value $|M|$, when viewed as a number in base-2 notation is maximal. If $|M| = l \cdot 2^r$, where l is odd and $r \ge 0$, then the state S such that $|S| = l$ is also on C, and we take S to be the bridging state of C.

In any case the chosen bridging state S for the current PCR_n cycle C always ends in a ONE. By Lemma 1, its companion S' belongs to a PCR_n cycle whose weight is smaller than that of C. Therefore, S' must be on the current main cycle. By Theorem 1, interchanging the predecessors of S and S' will create the next main cycle by joining the current one with C.

A full cycle obtained by joining PCR_n cycles as described above, can be generated bit-by-bit following a procedure based on the underlying rules for the joining of cycles. In this procedure, the $(i+n)$th bit b_{i+n} of the full cycle is determined from the preceding n-bit state $\beta_i = (b_i, b_{i+1}, \cdots, b_{i+n-1})$. If β_i served as a predecessor of a bridging state (S or S') then $b_{i+n} = b_i \oplus 1$; otherwise, $b_{i+n} = b_i$. The formal steps for determining b_{i+n} are presented in the following algorithm.

Algorithm A: Choose a constant k such that $1 \le k \le 2^{((n-4)/2)}$. Choose and store an ordered set of bridging states $V = \{V(i)\}_{i=0}^{k-1}$. Initially, set $\beta_0 = (0, 0, \cdots, 0) = 0^n$. Given $\beta_i = (b_i, b_{i+1}, \cdots, b_{i+n-1})$, proceed to produce $\beta_{i+1} = (b_{i+1}, \cdots, b_{i+n-1}, b_{i+n})$ as follows.

(A1) Examine the cyclic shifts of $\beta_i^* = (b_{i+1}, \cdots, b_{i+n-1}, 1)$ for the existence of a shift α that begins with a ZERO and ends with $1 + \lfloor \log k \rfloor$ ONES. If no such α exists go to (A3).

(A2) Let α^* be the first $1 + \lceil \log k \rceil$ bits of α and let $|\alpha^*| = j$, the base-2 value of α^*. If $j > k - 1$ go to (A3); otherwise, if $\alpha = V(j) = \beta_i^*$ go to (A5); if $\alpha = V(j) \ne \beta_i^*$ go to (A4).

(A3) Let M be the cyclic shift of β_i^* with the largest base-2 value $|M| = l \cdot 2^r$, l odd, $r \ge 0$. Let S be the shift of β_i^* such that $|S| = l$. If $S = \beta_i^*$ go to (A5).

(A4) Set $b_{i+n} = b_i$ and stop.

(A5) Set $b_{i+n} = b_i \oplus 1$.

Theorem 2: a) For every choice of k, in the indicated range, and of the set V Algorithm A produces a full cycle of length 2^n.

b) For a given choice of k there are $2^{k \cdot g(n,k)}$ distinct choices for the set V, where
$$g(n,k) = n - 3 - \lceil \log k \rceil - \lfloor \log k \rfloor - \left\lceil \frac{n - 3 - \lceil \log k \rceil - \lfloor \log k \rfloor}{\lfloor \log k \rfloor + 1} \right\rceil;$$
thus, Algorithm A can be used to produce $2^{k \cdot g(n,k)}$ distinct full cycles.

c) The working space that Algorithm A requires to produce a full cycle is $3n + k \cdot g(n,k)$ bits and the work required to produce the next bit is $2n$ cyclic shifts and about the same number of n-bit comparisons.

Proof: a) follows directly from the discussion preceding Algorithm A.

b) is due to the fact that each $V(i)$ is specified up to exactly $g(n,k)$ free parameters and that no state except for $0^{n-a}1^a$, $a = 1 + \lfloor \log k \rfloor$, may serve as a bridging state via both of the two criteria: either by being a member of the set V or by representing the odd part of a maximal shift. This, together with Lemma 2, imply that distinct choices for the set V correspond to distinct sets of bridging states and, hence, to distinct full cycles.

c) follows directly from Algorithm A. Note that only information about members of the set V has to be stored and, there, only the $g(n,k)$ free bit-values of each $V(i)$ require storage. Q.E.D.

IV. Construction of Full Cycles from PSR$_n$

The PSR$_n$ is an n-stage FSR whose feedback function $f(x_1, x_2, \cdots, x_n) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$.

An *extended representation* $E(C)$ of a cycle C of PSR$_n$ is given by an $(n+1)$-tuple $[x_0 x_1 \cdots x_{n-1} x_n]$ where $(x_0, x_1, \cdots, x_{n-1})$ is a state on C and $x_n = x_0 \oplus x_1 \oplus \cdots \oplus x_{n-1}$.

The *extended weight* $W_E(C)$ of C is defined as the number of ONES in $E(C) = [x_0 x_1 \cdots x_{n-1} x_n]$, i.e., $W_E(C) = \Sigma_{i=0}^n x_i$.

The following lemma is an immediate result of the above definitions.

Lemma 3: For every cycle C from PSR$_n$ we have $W_E(C) = 2k$, for some $0 \le k \le \lfloor (n+1)/2 \rfloor$, and for each state S on C $2k - 1 \le W(S) \le 2k$.

C is called a *run-cycle* if all the ONES in $E(C)$ form a cyclic run.

For each cycle C of PSR$_n$, with $W_E(C) = 2k < n+1$, we define a unique *preferred state* $P(C)$. For a run-cycle, $P(C) = (1^{2k}0^{n-2k})$; for a cycle with more than one (cyclic) run of ONES the preferred state is defined as follows.

Let $E^*(C) = [0^r 1^t 0 b_1 \cdots b_{n-t-r-2} 10]$ be the unique extended representation of C which satisfies the following properties:

a) $r \ge 0$;
b) t is the length of the longest run of ONES;
c) among all extended representations of this form, with the same maximal t, $E^*(C)$ is the largest when viewed as a number in base-2 notation.

Then, the preferred state for C is $P(C) = (0^r 1^t 0 b_1 \cdots b_{n-t-r-2} 1)$.

Lemma 4: Let C_1 be a nonrun-cycle from PSR$_n$ and let $P(C_1) = (0^r 1^{t_1} 0 b_1 \cdots b_{n-t_1-r-2} 1)$. Then the states $B = (10^r 1^{t_1} 0 b_1 \cdots b_{n-t_1-r-2})$ and the companion of $P(C_1)$ are on a cycle $C_2 \ne C_1$, with $W_E(C_2) = W_E(C_1)$. Furthermore, if t_2 is the length of the longest run of ONES in $P(C_2)$ then either $t_2 = t_1 + 1$, or $t_2 = t_1$ and $|P(C_2)| > |P(C_1)|$.

Proof: Clearly $W(B) = W(P(C_1)) = W_E(C_1) = 2k$ for some k. Hence, by Lemma 3, $W_E(C_2) = 2k = W_E(C_1)$. It is also clear that $E(C_2) = [10^r 1^{t_1} 0 b_1 \cdots b_{n-t_1-r-2} 0]$. Hence, if $r = 0$, $t_2 = t_1 + 1$; if $r > 0$, then an alternate extended representation of C_2 is given by $E'(C_2) = [0^{r-1} 1^{t_1} 0 b_1 \cdots b_{n-t_1-r-2} 010]$, which implies

$$|P(C_2)| \ge \left|(0^{r-1} 1^{t_1} 0 b_1 \cdots b_{n-t_1-r-2} 01)\right| > |P(C_1)|.$$

Thus, in any case $C_2 \ne C_1$ and, since the two possible successors of B are $P(C_1)$ and the companion of $P(C_1)$, it follows that the companion of $P(C_1)$ is the successor of B on C_2. Q.E.D.

Lemma 5: Let $U = (u_1, \cdots, u_{n-1}, 1)$ be a state on a cycle C_1 of PSR$_n$ with $W(U) + 1 = W_E(C_1) = 2k$ for some $k \ge 1$. Then the companion U' of U is on a PSR$_n$ cycle C_2 with $W_E(C_2) = 2k - 2$.

Proof: This lemma follows directly from the definition of U' and Lemma 3. Q.E.D.

Lemmas 3, 4, 5 lead to a construction of a large class of full cycles from those of PSR$_n$. Lemma 4 suggests a way of joining all cycles with the same extended weight. For each extended weight $2k$, we start with the run-cycle of this weight as an initial main cycle. In each step the current main cycle is expanded by joining to it the PSR$_n$ cycle of extended weight $2k$ with the longest run of ONES; if there are two or more cycles with the same longest run of ONES, join the one with the largest preferred state. Recalling the definition of bridging states in Section III, it is easy to verify that this order of joins is always possible if the preferred state of the PSR$_n$ cycle in line is chosen as a bridging state S for the join (the described order guarantees that its companion S' belongs to the current main cycle).

Once all the PSR$_n$ cycles of extended weight $2k$ are joined together into a corresponding main cycle MC$_k$, $0 \le k \le \lfloor (n+1)/2 \rfloor$, we apply Lemma 5 to joining the MC$_k$ cycles, in order of increasing k, to form a full cycle.

We proceed now to describe an algorithm for producing the $(i+n)$th bit b_{i+n} of the resulting full cycle from the following inputs:

a) the preceding n-bit state $\beta_i = (b_i, b_{i+1}, \cdots, b_{i+n-1})$,
b) the parity p_i of β_i, $p_i = b_i \oplus b_{i=1} \oplus \cdots \oplus b_{i+n-1}$, and
c) the weight $W(\beta_i)$ of β_i.

The production of b_{i+n} from the above inputs is based on the fact, that when $(x_0, x_1, \cdots, x_{n-1}) \to (x_1, \cdots, x_{n-1}, x_n)$ then $\Sigma_{i=0}^n x_i$ is even if and only if both states are on the same PSR$_n$ cycle.

Before presenting the formal steps of the algorithm, we remind the reader that the preferred state S of each PSR$_n$ nonrun-cycle and its companion S' serve as bridging states in the process of forming one of the MC$_k$ cycles. In the process of joining the MC$_k$ cycles into a full cycle, the bridging state S on the MC$_k$ cycle in line, $1 \le k \le \lfloor (n+1)/2 \rfloor$, can be chosen as any state of odd weight and a trailing ONE; i.e., the bridging state $S^{(k)}$ for MC$_k$ can be any state of the form $S^{(k)} = (s_1^k, s_2^k, \cdots, s_{n-1}^k, 1)$ with $W(S^{(k)}) = 2k - 1$ (see Lemma 5).

In Algorithm B, given below, we first check whether the given state β_i serves as a predecessor of a bridging state (S or S'). If it does, we set $b_{i+n} = p_i \oplus 1$, $p_{i+1} = b_i \oplus 1$, and $W(\beta_{i+1}) = W(\beta_i) - b_i + (p_i \oplus 1)$; otherwise, $b_{i+n} = p_i$, $p_{i+1} = b_i$, and $W(\beta_{i+1}) = W(\beta_i) - b_i + p_i$.

Algorithm B: For every k such that $1 \leq k \leq \lfloor (n+1)/2 \rfloor$ choose and store a bridging state $U^{(2k)}$ of the form $U^{(2k)} = (u_1^k, u_2^k, \cdots, u_{n-1}^k, 1)$ with $W(U^{(2k)}) = 2k - 1$. Initially, set $\beta_0 = (0,0,\cdots,0) = 0^n$, $p_0 = 0$ $W(\beta_0) = 0$. Given $\beta_i = (b_i, b_{i+1}, \cdots, b_{i+n-1})$, p_i, $w_i = W(\beta_i)$ proceed to produce $\beta_{i+1} = (b_{i+1}, \cdots, b_{i+n-1}, b_{i+n})$, p_{i+1}, w_{i+1} as follows:

```
(B1) If $p_i \oplus b_i = 1$ go to (B3).
(B2) If $(b_{i+1}, \cdots, b_{i+n-1}, 1) = U^{(w_i - b_i + 2)}$ go to (B6);
     otherwise go to (B5).
(B3) If $\beta_i^* = [b_{i+1} \cdots b_{i+n-1}10]$ is a run-cycle go to
     (B5); otherwise, find the cyclic shift $E_i^* = [0^r 1^t 0 b_s \cdots b_{n-t-r+s+3}10]$
     of $\beta_i^*$ whose first $n$ bits form a preferred state.
(B4) If $E_i^* = \beta_i^*$ go to (B6).
(B5) Set $b_{i+n} = p_i$, $p_{i+1} = b_i$, $w_{i+1} = w_i - b_i + p_i$, and
     stop.
(B6) Set $b_{i+n} = p_i \oplus 1$, $p_{i+1} = b_i \oplus 1$, $w_{i+1} = w_i - b_i + (p_i \oplus 1)$.
```

Theorem 3: a) For every choice of the set of states $\{U^{(2k)}\}_{k=1}^{\lfloor (n+1)/2 \rfloor}$ Algorithm B produces a full cycle of length 2^n.

b) There are

$$\prod_{k=1}^{\lfloor (n-2)/2 \rfloor} \binom{n-1}{2k}$$

distinct choices for the set of states $\{U^{(2k)}\}$; thus Algorithm B can be used to produce

$$\prod_{k=1}^{\lfloor (n-2)/2 \rfloor} \binom{n-1}{2k}$$

distinct full cycles.

c) The working space that Algorithm B requires to produce a full cycle is about $n^2/2$ bits and the work required to produce the next bit is n cyclic shifts and about the same number of n-bit comparisons.

Proof: a) follows directly from the discussion preceding Algorithm B.

b) is due to the fact that different sets of bridging states produce different full cycles. The number of ways to choose the set $\{U^{(2k)}\}$ is

$$\prod_{k=1}^{\lfloor (n+1)/2 \rfloor} \binom{n-1}{2k-2} = \prod_{k=0}^{\lfloor (n-1)/2 \rfloor} \binom{n-1}{2k} = \prod_{k=1}^{\lfloor (n-2)/2 \rfloor} \binom{n-1}{2k}.$$

c) follows directly from Algorithm B. It is clear that most of the work consists of finding the preferred state of

$$\beta_i^* = [b_{i+1} \cdots b_{i+n-1}10] = [0^{r_1}1^{t_1}X_1 10]$$

in (B3). Let E_i^* be the shift of β_i^* whose first n bits form a preferred state. Initially, $E_i^* = \beta_i^*$. Given $E_i^* = [0^{r_2}1^{t_2}X_2 10]$ and a shift $E_i^+ = [0^{r_3}1^{t_3}X_3 10]$ of β_i^*, set $E_i^* = E_i^+$ if either $t_3 > t_2$, or $t_2 = t_3$ and $|E_i^+| > |E_i^*|$. After n shifts E_i^* will have the required form. Q.E.D.

Example 3: For $n = 6$, and the bridging states

$$U^{(2)} = (0,0,0,0,0,1),$$
$$U^{(4)} = (1,0,1,0,0,1),$$
$$U^{(6)} = (1,1,0,1,1,1),$$

the full successive bits of one period of the full cycle generated by Algorithm B are

00000011000010100111000111100110
11111101100101110101011010001001.

It should be noted that a similar algorithm can be derived for the complement of the PSR$_n$, i.e., the FSR with the feedback function $f(x_1, x_2, \cdots, x_n) = x_1 \oplus x_2 \oplus \cdots \oplus x_n \oplus 1$.

References

[1] H. Fredricksen, "A survey of full length nonlinear shift register cycle algorithms," *SIAM Rev.*, vol. 24, pp. 195–221, Apr. 1982.

[2] N. G. de Gruijn, "A combinatorial problem," in *Nederl. Akad. Wetensch. Proc.*, vol. 49, 1946, pp. 758–764.

[3] H. Fredricksen, "A class of non-linear de Bruijn cycles," *J. Comb. Theory, Ser. A*, vol. 19, pp. 192–199, Sept. 1975.

[4] A. Lempel, "Cryptology in transition," *Computing Surveys*, vol. 11, pp. 285–303, Dec. 1979.

[5] S. W. Golomb, *Shift Register Sequences.* San Francisco: Holden-Day, 1967.

[6] A. H. Chan, R. A. Games, and E. L. Key, "On the complexities of de Bruijn sequences," *J. Comb. Theory, Ser. A*, vol. 33, pp. 233–246, Nov. 1982.

[7] H. Fredricksen, "Generation of the Ford sequence of length 2^n, n large," *J. Comb. Theory, Ser. A*, vol. 12, pp. 153–154, 1972.