PARALLEL GENERATION OF RECURRING SEQUENCES

Christoph G. Günther
Asea Brown Boveri
Corporate Research
CH-5405 Baden, Switzerland

ABSTRACT

In applications, such as radar ranging or test pattern generation, linear recurring sequences are needed at rates that require a parallel generation of the sequences. Two parallelisation methods for the generation of these sequences are discussed and previous results are made applicable to arbitrary degrees of parallelisation and arbitrary sequences. In particular, a previously known technique (sometimes called windmill technique) is shown to be explainable in a very simple way and to be equally appropriate for the parallelisation of non-linear recursions. The method is, furthermore, shown to be suitable for VLSI-realisations and software implementations.

I. INTRODUCTION

Linear recurring sequences for high rate applications can often not be generated directly by the associated linear feedback shiftregister (LFSR), at least not at reasonable costs. Examples of such applications are radar ranging [1], direct sequence spread spectrum (DSSS) communication [2], test pattern generation [3], stream cipher cryptography [4] and decoding by error trapping [5]. In some of these examples, such as radar ranging or the encryption of TV-pictures, the problem is a technological one, which can in principle be solved by faster, low price and low consumption semiconductors. For many applications, however, the difficulty is of a

J.J. Quisquater and J. Vandewalle (Eds.): Advances in Cryptology - EUROCRYPT '89, LNCS 434, pp. 503-522, 1990.
© Springer-Verlag Berlin Heidelberg 1990

more fundamental nature, since the linear recurring sequences are not needed at a higher rate on an absolute scale but relative to other rates. Examples are the hashing of the data bits by the chipping sequence in DSSS communication [2] or the clocking of the driven register in a binary rate multiplier [6].

Consequently, methods have been developed for the parallel generation of several phases of a decimated sequence. These methods can be subdivided into two classes: those that generate one phase with an appropriate LFSR and construct the other phases by the use of a linear feedforward network [7]-[11] (Fig. 1.a) and those that generate at once all phases in parallel by the use of a network of interconnected shift registers [12]-[17] (Fig. 1.b). Warlick and Hershey have based their approach to the latter class of generators on a windmill-like arrangement of LFSR's [15]. For this reason such generators are sometimes called windmill generators. As we wish to emphasize the characteristics of the two types of generators introduced above, we will, however, prefer to call the generators, from Fig. 1.a and Fig. 1.b, *parallel feedforward (PFF)* and *parallel feedback generators (PFB)*, respectively.

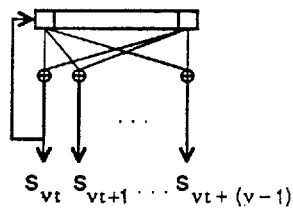

Fig. 1.a. A parallel feedforward generator

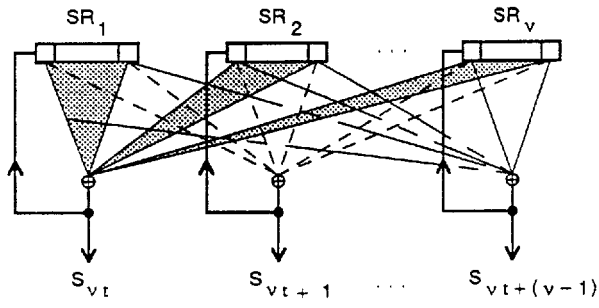

Fig. 1.b. A parallel feedback generator

The main purpose of the paper is to present a simple and intuitive explanation of the PFB-scheme and to give construction rules for such generators which are valid for arbitrary recursions and arbitrary degrees of parallelism (Sec. III).

The importance of these generators is due to their structure: They show a high degree of periodicity. This makes them very suitable for VLSI realisations and for software implementations.

The PFF-generators do not seem to have a corresponding periodicity in their structure and there is also no obvious generalisation to arbitrary recursions. In the linear case, however, we will see in Sec. II that they are as universal as the PFB-generators.

II. PARALLEL FEEDFORWARD GENERATORS

The decimation by ν of a sequence s is a sequence $\tilde{s}$ defined by $\tilde{s}_t := s_{\nu t}$. In the present context it is natural to also consider $\tilde{s}_t^{(i)} := s_{\nu t+i}$ with $i \in \{0, 1, \ldots, \nu - 1\}$. The objective is to generate the sequences $\tilde{s}^{(0)}, \tilde{s}^{(1)}, \ldots, \tilde{s}^{(\nu-1)}$ in parallel.

We note, that since the period of any sequence decimated by ν divides $T/\gcd(\nu, T)$, there are at most $\gcd(\nu, T)$ different decimated sequences.

Consider sequences s with an irreducible minimal polynomial of period T and degree l. For the decimations $\tilde{s}$ of s by ν, Zierler [18] has proved that the minimal polynominal of $\tilde{s}$ is irreducible, of period $\tilde{T} = T/\gcd(T, \nu)$ and of degree $\tilde{l} = \min\{l : \tilde{T} \mid 2^l - 1\}$ if $\tilde{s} \neq 0$.

Based on this observation, Lempel and Eastman [7] have proposed to generate $\tilde{s}^{(0)}, \tilde{s}^{(1)}, \ldots, \tilde{s}^{(\nu-1)}$ in parallel by the use of ν LFSR's. This was the first step towards PFF-generators.

The first PFF-generator was then introduced by Möhrmann [8]. He has noted that the decimation of an m-sequence s by $\nu = 2, 4, \ldots, 2^{l-1}$ does not change the minimal polynomial and has observed that the sequences $\tilde{s}^{(i)}$ can be obtained by linearly combining $\tilde{s}_{t-l+1}, \tilde{s}_{t-l+2}, \ldots, \tilde{s}_t$. (An m-sequence is a linear recurring sequence of maximal period.) Eier and Malleck [9] have further formalised these results and have indicated a general construction scheme for the feedforward network.

Surböck and Weinrichter[10] have also extended the previous results. They have shown how sequences with irreducible polynomials and non-prime periods can be generated in parallel by PFF-generators of lengths shorter than the original LFSR's if $\nu \mid T$.

For arbitrary rates $\nu \in \mathbf{N}$, we prove in the following that $\tilde{s}_t^{(0)}$, $\tilde{s}_t^{(1)}, \ldots, \tilde{s}_t^{(\nu-1)}$ is a linear function of $\tilde{s}_{t-\tilde{l}+1}, \tilde{s}_{t-\tilde{l}+2}, \ldots, \tilde{s}_t$, with $\tilde{l}$ bounded from above by the linear complexity l of s. (The linear complexity of s is the length of the shortest linear recursion that can generate s.) This statement is obtained in three steps: i) the sequences $\tilde{s}^{(0)}, \tilde{s}^{(1)}, \ldots, \tilde{s}^{(\nu-1)}$ all satisfy the same linear recursion (theorem 1), ii) this recursion has a length $\tilde{l}$ which is bounded from above by l (corollary 2) and iii) there exists an initial condition $\tilde{s}_0, \tilde{s}_1, \ldots, \tilde{s}_{\tilde{l}-1}$ for $\tilde{s}$ such that $\tilde{s}_t^{(0)}, \tilde{s}_t^{(1)}, \ldots, \tilde{s}_t^{(\nu-1)}$ can be obtained by linearly combining $\tilde{s}_{t-\tilde{l}+1}, \tilde{s}_{t-\tilde{l}+2}, \ldots, \tilde{s}_t$ (lemma 3).

Theorem 1 extends Zierler's result [18] to arbitrary sequences and describes the generating polynomial $\tilde{p}(z)$ of the decimated sequence explicitly. For the formulation and the later proof of theorem 1, we define α_k and α_i to be μ-conjugate $(k, i \in J)$ if their images under $z \longrightarrow z^\mu$ are conjugate,*i.e.*, if there exists j such that $\alpha_i^\mu = \alpha_k^{\mu 2^j}$. With

$$\tilde{J}_k := \{i \in J : \alpha_i \text{ is } \mu - \text{conjugate to } \alpha_k\}$$

and

$$\tilde{J} := \{k \in J : k = \text{smallest element in } J_k\},$$

this defines a unique partition of J :

$$J = \cup_{k \in \tilde{J}} \tilde{J}_k.$$

Theorem 1. Let $p(z) = \prod_{i \in J} p_i(z)^{m_i}$ be the decomposition of $p(z)$ into distinct irreducible factors and let $p_i(z) = \prod_{j=0}^{d_i-1}(z - \alpha_i{}^{2^j})$ be the canonical representation of the irreducible factors. For $\nu = 2^\kappa \mu \in \mathbf{N}$, with $2 \nmid \mu$, let

$$\tilde{d}_k = \min\{d : \frac{(2^{d_k}-1)}{\gcd(\mu, 2^{d_k}-1)} | 2^d - 1\}(|d_k), \tag{1}$$

and let $\tilde{m}_k = \max_{i \in \tilde{J}_k} \lceil \frac{m_i}{2^\kappa} \rceil$. Then the following three assertions hold:

a. The polynominal $\tilde{p}(z)$ *of smallest degree, that generates all decimations* $\tilde{s}^{(i)}$, $i \in \{0, \ldots, \nu - 1\}$ *of all sequences* s which have minimal polynomial $p(z)$ is given by

$$\tilde{p}(z) = \prod_{k \in \tilde{J}} \tilde{p}_k(z)^{\tilde{m}_k}, \tag{2}$$

$$\tilde{p}_k(z) = \prod_{j=0}^{\tilde{d}_k - 1} (z - {\alpha_k}^{\mu 2^j}). \tag{3}$$

b. If ν is relatively prime to the period T of s, we have $\tilde{J} = J$, $\tilde{d}_k = d_k$ and $\tilde{m}_k = m_k$.

c. In this case, the polynomial $\tilde{p}(z)$ *is minimal for all decimations* $\tilde{s}^{(i)}, i \in \{0, \ldots, \nu - 1\}$ *of all sequences* with minimal polynomial $p(z)$.

As far as we could trace it back, Duvall and Mortick [19] were the first to prove that $\prod_{k \in J} \tilde{p}_k(z)^{\tilde{m}_k}$ can generate $\tilde{s}^{(0)}$. Recently, a very elegant and compact proof of the assertion that $\prod_{k \in J} \tilde{p}_k(z)^{m_k}$ generates $\tilde{s}^{(i)}, \forall i$ was given by Niederreiter [20]. Smeets [16] has proved the strongest result, so far, *i.e.*, $\tilde{p}(z)$ generates $\tilde{s}^{(i)}, \forall i$. Based on Niederreiter's result, we give a compact proof of this assertion and a proof of the theorem in the appendix.

The following corollary is an immediate consequence of theorem 1.

Corollary 2. Let $\tilde{s}$ be a sequence s decimated by ν. Then its linear complexity $\tilde{l}$ is related to the linear complexity l of s by

$$\tilde{l} \leq l. \tag{4}$$

If ν is relatively prime to T, we have

$$\tilde{l} = l. \tag{5}$$

By theorem 1, the sequences $\tilde{s}^{(0)}$, $\tilde{s}^{(1)}, \ldots,$ $\tilde{s}^{(\nu-1)}$ can thus be generated by ν identical LFSR's, each of maximal length $\tilde{l}$. The following lemma shows that in fact only one such LFSR is needed:

Lemma 3. Let $\tilde{p}(z)$ be an arbitrary polynomial of degree $\tilde{l}$, then there is a sequence $\tilde{r}$ generated by $\tilde{p}(z)$ such that any sequence $\tilde{s}$ generated

by $\tilde{p}(z)$ can be expressed as

$$\tilde{s}_t = \sum_{i=0}^{\tilde{l}-1} \tilde{\sigma}_i \tilde{r}_{t-i}. \tag{6}$$

The proof of this lemma is easily obtained from observing that the vector of initial conditions $(\tilde{r}_{\tilde{l}-1}, \tilde{r}_{\tilde{l}-2}, \ldots, \tilde{r}_0) = (1, 0, \ldots, 0)$ is transformed into $\tilde{l}$ linearly independent vectors by applying the linear recursion.

The feedforward network is now derived from the initial conditions $s_0, s_\nu, \ldots, s_{\nu(\tilde{l}-1)}; \ldots; s_{\nu-1}, s_{2\nu-1}, \ldots, s_{\nu\tilde{l}-1}$ for $\tilde{s}^{(0)}; \tilde{s}^{(1)}; \ldots; \tilde{s}^{(\nu-1)}$. These are obtained by the use of the original linear recursion and s_0, $s_1, \ldots, s_{l-1}$. The maximal depth of the XOR-tree needed to implement the network is $\log_2 \tilde{l} (\leq \log_2 l)$ and the maximum number of XOR's is $\nu \log_2 \tilde{l} (\leq \nu \log_2 l)$.

In this context, we also observe that the number of memory cells is equal to $\tilde{l} (\leq l)$. Thus, if we compare the parallel implementation by a PFF-generator with the serial implementation by an LFSR, we note that only the number of logical gates is increased by a factor ν.

III. PARALLEL FEEDBACK GENERATORS

Two different forms of the generator shown in Fig. 1.b have been introduced, one by Hsiao [12] and the other by Hurd [13], by Maritsas, Arvillias and Bounas [14], and later by Warlick and Hershey [15]. Hsiao has obtained his generator by considering iterates T^ν of the transfer matrix T, associated with the linear recursion $s_t = \sum_{i=1}^{l} \pi_i s_{t-i}$:

$$T = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & 1 & 0 \\ 0 & \ldots & 0 & 0 & 1 \\ \pi_l & \ldots & \pi_3 & \pi_2 & \pi_1 \end{pmatrix}. \tag{7}$$

This simple construction method was, however, not adopted by many authors, probably for two reasons: first the electronic components in the late 60's were not suitable for practical implementations of this construction and secondly the method did not necessarily provide structured

generators. In their papers, Hurd, and Maritsas, Arvillias and Bounas, correspondingly used a completely different approach. They started with a particular, well structured generator and searched for conditions under which the given generator yields different phases of a decimated m-sequence. More precisely, they considered a network of cyclically and linearly interconnected shiftregisters as represented in Fig. 2.

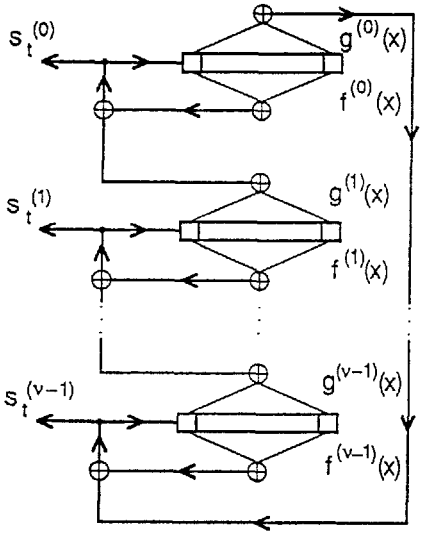

Fig. 2. Generator with a cyclic arrangement of shiftregisters as considered in [13]-[15]

The sequences obtained from such a network are solutions of the following system of linear recurrence equations

$$f^{(i)}(D) \cdot s_t^{(i)} = g^{(i+1) \bmod \nu}(D) \cdot s_t^{(i+1) \bmod \nu}, \; i \in \{0, 1, \ldots, \nu - 1\}. \quad (8)$$

Due to the cyclic nature of this system, they are also solutions of

$$p(D) \cdot s_t^{(i)} := \left(\prod_{j=0}^{\nu-1} f^{(j)}(D) + \prod_{j=0}^{\nu-1} g^{(j)}(D) \right) \cdot s_t^{(i)} = 0. \quad (9)$$

The polynomial $p(z)$ thus generates the sequences $s^{(0)}, s^{(1)}, \ldots, s^{(\nu-1)}$. If it is primitive, all sequences are correspondingly phases of one single m-sequence. Warlick and Hershey [15], who started from an even more geometric conception, have obtained similar results for the special case $f_i(z) = 1 + z^t$, $i \in \{0, 1, \ldots, \nu - 1\}$; $g_i(z) = z^l$, $i \in \{0, 1, \ldots, \nu - 2\}$ and $g_{\nu-1}(z) = z^{l+1}$.

In the work of Hurd, of Maritsas, Arvillias and Bounas and of Warlick and Hershey no assertions are made about the relative phases between the sequences $s^{(i)}$. They will usually be non consecutive. This severly limits the applicability of the scheme. Smeets [16] and Smeets and Chambers [17] have indicated sufficient conditions to insure that the sequences $s^{(i)}$ have appropiate phase relations. The assumptions they need in order to prove their assertions are, however, still quite restrictive and their proofs rather involved.

In the following a construction method is exposed, which associates to every sequence s and to every degree of parallelisation ν a PFB-generator for $\tilde{s}^{(0)}, \tilde{s}^{(1)}, \ldots, \tilde{s}^{(\nu-1)}$. The only restriction for the sequence s is that it can be implemented by a not necessarily linear recursion. The simplest case of such a construction is discussed in lemma 4.

Lemma 4. If s is a linear recurring sequence with a characteristic polynomial of the form $p(z) = 1 + z^{\lambda} f(z)$ with $\lambda > 1$ and $\deg p(z) = l$. Then, for any degree of parallelisation $\nu \leq \lambda$, the generation of s can be parallelised with a PFB-generator, graphically constructed according to algorithm A.

Algorithm A: (see also Fig. 3)

a) draw the indices $0, 1, \ldots, l+\nu-1$ on a line and their values modulo ν underneath,

b) draw the recursions for $s_l, \ldots, s_{l+\nu-1}$,

c) distribute the memory cells of the shiftregister on ν lines according to the partition induced by the indices modulo ν (the cells on line i will contain successive values of $\tilde{s}^{(i)}$),

d) interconnect the successive cells in each line and determine the new content of the first cell of line i by use of the linear recursion for s_{l+i}, drawn in step b).

The idea behind this lemma is the following: due to the particular form of the feedback polynomial the elements s_{t+i}, $i \in \{0, 1, \ldots, \nu-1\}$ do not depend on the elements s_{t+j}, $j \in \{0, 1, \ldots, i-1\}$ and, therefore, the linear recursion for these new elements can be evaluated in parallel. The construction of these recursions is straightforward and leads, *e.g.*, to the description given by algorithm A. The completion of these remarks to a proof is trivial.

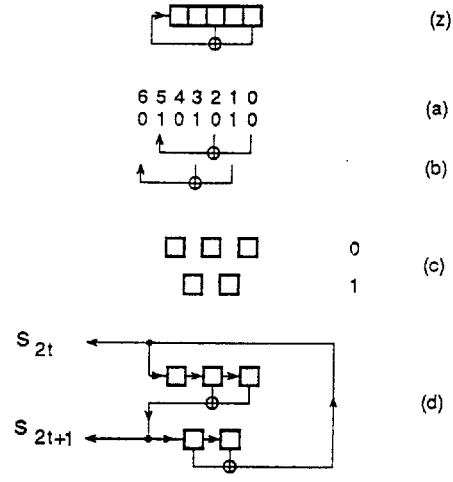

Fig. 3. Algorithm A applied to the linear recursion $p(z) = 1 + z^3 + z^5$ with rate $\nu = 2$. Step (z) shows the LFSR for a serial generation and step (d) the PFB-generator for a generation of two elements in parallel.

We note that in lemma 4 the linearity of the recursion was not used. Furthermore, the independence of the s_{t+i} from the s_{t+j}, with $0 \leq j \leq i-1$, was not essential either, as the elements s_{t+j}, $0 \leq j \leq i-1$ can be expressed in terms of the elements s_{t+k}, with $0 \leq k \leq j-1 \leq i-2$ and so on:

$$\begin{aligned} s_{t+i} &= F(s_{t+i-1}, \ldots, s_{t+i-l}) \\ &= F(F(s_{t+i-2}, \ldots, s_{t+i-l-1}), s_{t+i-2}, \ldots, s_{t+i-l}) \\ &= \ldots \\ &= G_i(s_{t-1}, \ldots, s_{t-l}) \ , \quad i \in \{0, 1, \ldots, \nu - 1\}, \end{aligned} \tag{10}$$

where F is the recursion that generates s, and where G_i is the recursion obtained by iteratively applying F to eliminate all s_{t+j}, $0 \leq j < i$. These remarks lead us immediately to:

Theorem 5. Let F be an arbitrary recursion of length l. Then the generation of any sequence s, associated with F, can be parallelised with a PFB-generator up to an arbitrary degree of parallelisation. The corresponding generator can be constructed using either algorithm B, which requires l memory cells, or algorithm B'. In the latter case the

generator has a total number of memory cells d bounded from above by $d \leq l + \nu$.

Algorithm B: (see also Fig. 4.a)

a) unchanged,

b) draw the recursion G_i for s_{l+i}, $i \in \{0, \ldots, \nu - 1\}$, *i.e.*, draw the recursion with the values of s_{l+j}, $0 \leq j < i$ eliminated by the use of the recursion F,

c) draw ν outputs on ν lines and distribute the l memory cells of the shiftregister on the $\min\{l, \nu\}$ upper lines, according to the partition induced by the indices modulo ν,

d) unchanged.

Algorithm B': (see also Fig. 4.b)

a) draw the indices $0, 1, \ldots, l+2\nu-2$ on a line and their values modulo ν underneath,

b) draw the recursion $G_{\nu-1}$ for $s_{l+\nu-1+i}$, $i \in \{0, 1, \ldots, \nu - 1\}$, *i.e.*, draw the recursion with the values of $s_{l+j-1+i}$, $j \in \{0, \ldots, \nu - 1\}$ eliminated by the use of the recursion F,

c) draw ν outputs on ν lines and distribute the memory cells ($\leq l + \nu - 1$) of the shiftregister associated with $G_{\nu-1}$, according to the partition induced by the indices modulo ν,

d) unchanged.

After completion of the contruction, the generators B and B' are loaded with the initial conditions, $s_0, \ldots, s_{l-1}$ and $s_{l+\nu-d}, \ldots s_{l+\nu-1}$, respectively, where $s_l, \ldots, s_{l+\nu-1}$ is obtained by repeated application of the recursion F.

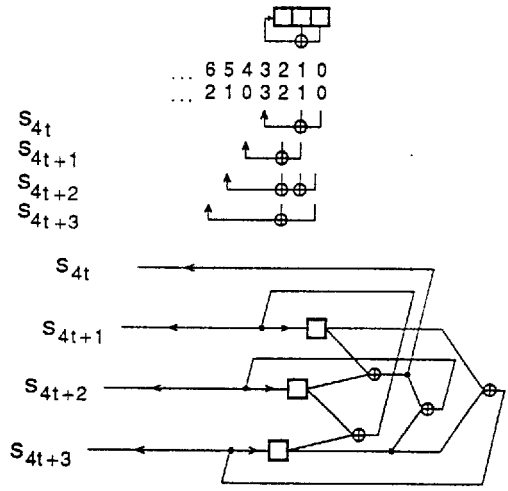

Fig. 4.a. Generator obtained by applying algorithm B to $p(z) = 1 + z^2 + z^3$ and $\nu = 4$.

In the linear case, algorithm B' is equivalent to the contruction of a polynomial $q(z) = 1 + z^{\nu} f(z)$ which is a multiple of $p(z)$ and has no tabs in the ν leading positions. It is obvious that such a polynomial generates the sequence s if applied to the correct initial conditions. By Lemma 4 this polynomial can, furthermore, be used to generate $\tilde{s}^{(0)}, ..., \tilde{s}^{(\nu-1)}$ in parallel. A complete proof of the theorem is an easy consequence of remarks made prior to its formulation.

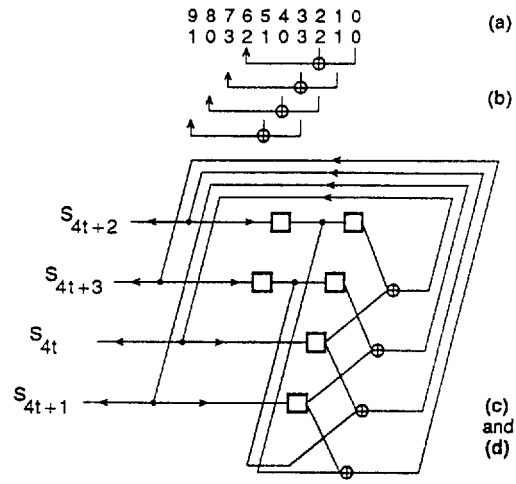

Fig. 4.b. Generators obtained with algorithm B' are always well structured. Algorithm B' was applied to the same example as in Fig. 4.a.

Hsiao's result, described at the beginning of the paragraph, is a particular case of theorem 5/algorithm B, which correspondingly does not necessarily lead to highly structured generators. The structure of the generators is much improved with algorithm B', which, however, requires up to ν additional memory cells and a corresponding extension of the initial conditions.

It is obvious that generators constructed according to algorithm B' can easily be implemented in VLSI-technique and are also well suited for software implementations.

In software implementations, when $\nu \not| l$, the structure obtained from algorithm B' is further improved by matching the length of the recursion with a multiple of the wordlength w. In the typical case $l \geq w$, it is done by changing the recursion $s_{t+\nu-1} = G_{\nu-1}(s_{t-1}, \ldots, s_{t-i}, \ldots, s_{t-l})$ into $s_{t+\nu-1} = G_{\nu-1}(s_{t-1}, \ldots, F(s_{t-i-1}, \ldots s_{t-i-l}), \ldots, s_{t-l})$ with $i + l + \nu - 1 = nw$, $n \in \mathbf{N}$, $i \in \{1, \ldots, l\}$. In the case $l < w$ this transformation is applied repeatedly. If F is linear, the tranformation corresponds to a change of the polynomial $q(z) = \gamma(z)p(z) = 1 + q_\nu z^\nu + \cdots + q_{l+\nu-1} z^{l+\nu-1}$ into $q'(z) = (\gamma(z) + z^{nw-l})p(z) = q(z) + z^{nw-l}p(z)$ with $nw - l \geq \nu$. Clearly, these recursions have the required form, and the smallest n which fulfills the above inequalities is

$$n = \lceil \frac{l + \nu}{w} \rceil. \tag{11}$$

This is exactly the number of words required anyway. Therefore the adaptation, which we shall call algorithm C, can always be performed without increasing the space complexity with respect to algorithm B'. The complexity of the software implementation becomes mainly dependent on the number of tabs in the polynomial $q(z)$. Unfortunatly, no results are known for constructing multiples of polynominals that have only few non-vanishing coefficients and, in particular, that have ν vanishing leading coefficients.

In this situation we have to use a trick in order to generate in parallel 32 bits of an m-sequence of period $2^{32} - 1$ on a 32-bit machine. We choose a polynomial $p(z) = 1 + z^{25} + z^{29} + z^{30} + z^{32}$, which has the fewest possible number of tabs and no tabs in the 16 leading positions. (We note that when $l \equiv 0 \pmod 8$, no irreducible trinomials exits [21].) Then by squaring $p(z)$ we obtain a polynomial $q(z) = (p(z))^2 = 1 + z^{50} + z^{58} +$

$z^{60} + z^{64}$, which has the same small number of tabs and no tabs in the 31 leading positions. This polynomial $q(z)$ can be used to write a very simple assembler program which generates the sequence wordwise.

IV. SUMMARY AND CONCLUSION

Recurring sequences are often needed at rates that cannot easily be achieved by sequential generation of the elements. Two methods for the parallel generation were correspondingly described: the parallel feedforward (PFF) and the parallel feedback method (PFB).

The parallel feedforward method, which was previously known as the more universal method, could be shown to be applicable to all linear recurring sequences. The parallel feedback method is even more performant: it is not only applicable to linear recurring sequences but to any sequence with a minimal recursion of reasonable length, and is highly suitable for VLSI- and for software-implementations.

In hardware implementations, both methods have a space complexity that is increased by a factor ν with respect to the number of logical gates and that remains essentially unchanged with respect to the number of memory cells. The effective gain in time complexity, *i.e.*, the effective rate, is $\nu\tau_{sr}/\max\{\tau_{sr}, \tau_{gate}\log_2 l\}$, where τ_{sr} and τ_{gate} are the time delays introduced by the shift register cells and the gates, respectively. In software implementations the parallel generation of whole words by the PFB-method is possible at practically no costs in space complexity and no costs in time complexity (rate= ν). This is due to the wordwise processing capability of arithmetic logic units (ALU's).

Finally, we note that parallel feedback generators are easily constructed by either of the algorithms, depending on the implementation intended.

ACKNOWLEDGMENT

I would like to thank Professor James L. Massey for his interest and encouragement.

APPENDIX: Proof of Theorem 1

In order to prove the minimality of $\tilde{p}(z)$, we need two preliminary results: theorem A.1 and lemma A.2. With

$$\binom{\xi}{\eta}_2 = \begin{cases} \binom{\xi}{\eta} & \text{if } \xi \geq \eta \\ 0 & \text{else,} \end{cases} \tag{A.1}$$

and

$$\mathrm{Tr}(\xi) := \sum_{j=0}^{d-1} \xi^{2^j}, \quad \text{if} \quad \xi \in \mathrm{GF}(2^d) \tag{A.2}$$

theorem A.1 reads:

Theorem A.1. (Milne-Thomson [22] and Ward [23], see also Key [24], Herlestam [25])
Let $p(z)$ be as in theorem 1 then all sequences s generated by the polynomial $p(z)$ have the form

$$s_t = \sum_{i \in J} \sum_{n=1}^{m_i} \binom{2^l - n}{t}_2 \mathrm{Tr}(\gamma_{i,n}\, \alpha_i{}^t) \tag{A.3}$$

and the period

$$T = 2^{\max_{i \in J} \lceil \log_2 m_i \rceil} \mathrm{lcm}\{T_i\}_{i \in J}, \tag{A.4}$$

with $\gamma_{i,n} \in \mathrm{GF}(2^{d_i})$, α_i a root of $p_i(z)$ and T_i the period of α_i. Furthermore, $p(z)$ is the minimal polynomial of s if and only if $\gamma_{i,m_i} \neq 0, \quad \forall i \in J$.

For the formulation of lemma A.2, we need the Hasse-derivative. It is defined as follows [26]:

$$D_\zeta^{(j)} \sum_{k=0}^{\infty} a_k \zeta^k := \sum_{k=j}^{\infty} a_k \binom{k}{j} \zeta^{k-j}, \tag{A.5}$$

and is formaly equivalent to $\frac{1}{j!}(\frac{d}{d\zeta})^j$.

Lemma A.2. Let $\nu = 2^\kappa \mu$ with $2 \nmid \mu$, let $\tilde{l} := l - \kappa$, $\tilde{n} := \lceil \frac{n}{2^\kappa} \rceil$ and let ω be such that $\mu\omega \equiv 1 \bmod 2^{\tilde{l}}$. Then

$$\binom{2^l - n}{\nu t}_2 = \sum_{i=0}^{\tilde{n}} c_i \binom{2^{\tilde{l}} - i}{t}_2 \tag{A.6}$$

where

$$c_i := D_\zeta^{(2^{\tilde{l}}-i)}(1+(\zeta-1)^\omega)^{2^{\tilde{l}}-\tilde{n}}|_{\zeta=0}. \tag{A.7}$$

and $c_1 = 1$.

Proof: Let $x = \sum_i x_i 2^i$ and $y = \sum_i y_i 2^i$ then Lukas' theorem states $\binom{x}{y} = \prod_i \binom{x_i}{y_i}$. This implies

$$\binom{2^l - n}{2^\kappa \mu t}_2 = \binom{2^{\tilde{l}} - \tilde{n}}{\mu t}_2. \tag{A.8}$$

We thus have only to prove

$$\binom{2^{\tilde{l}} - \tilde{n}}{\mu t}_2 = \sum_{i=0}^{2^{\tilde{l}}-1} c_i \binom{2^{\tilde{l}} - i}{t}_2 \tag{A.9}$$

and $c_i = 0, \quad \forall i > \tilde{n}$. We consider the generating functions of these expressions:

$$\sum_{t=0}^{2^{\tilde{l}}-1} \binom{2^{\tilde{l}} - \tilde{n}}{\mu t}_2 z^t = (1+z^\omega)^{2^{\tilde{l}}-\tilde{n}} \tag{A.10}$$

and

$$\sum_{t=0}^{2^{\tilde{l}}-1} \sum_{i=0}^{2^{\tilde{l}}-1} c_i \binom{2^{\tilde{l}} - i}{t}_2 z^t = \sum_{i=0}^{2^{\tilde{l}}-1} c_i (1+z)^{2^{\tilde{l}}-i}. \tag{A.11}$$

The substitution $\zeta = 1 + z$ leads to

$$\sum_{i=0}^{2^{\tilde{l}}-1} c_i \zeta^{2^{\tilde{l}}-i} = (1+(\zeta-1)^\omega)^{2^{\tilde{l}}-\tilde{n}} \tag{A.12}$$

and by taking the $(2^{\tilde{l}} - i)$-th Hasse derivative in $\zeta = 0$:

$$c_i = D_\zeta^{(2^{\tilde{l}}-i)}(1+(\zeta-1)^\omega)^{2^{\tilde{l}}-\tilde{n}}|_{\zeta=0} \tag{A.13}$$

In modulo 2 arithmetic this derivative vanishes $\forall i > \tilde{n}$. For $i = \tilde{n}$ we have

$$c_{\tilde{n}} \equiv \omega^{2^{\tilde{l}}-\tilde{n}} \equiv 1 \bmod 2 \tag{A.14}$$

since $2 \nmid \omega$. This concludes the proof of lemma A.2.

Theorem A.1 and lemma A.2 lead to an explicit representation of the decimated sequence $\tilde{s}^{(0)}$ and with some simple additions to an explicit representation of all decimated sequences. This could have been used to prove theorem 1. We shall, however, extract most results directly from the polynominal, which turns out to be much simpler.

Proof of theorem 1. a. We first prove that $\tilde{p}(z)$ can generate all decimations $\tilde{s}^{(i)}$. Through a very elegant approach, Niederreiter [20] has shown that

$$\tilde{p}(z) \mid \prod_{i \in J} \prod_{j=0}^{d_i - 1} (z - {\alpha_i}^{\nu 2^j})^{m_i}. \tag{A.15}$$

By regrouping terms, $\tilde{p}(z)$ can thus be written in the following form

$$\tilde{p}(z) = \prod_{k \in \tilde{J}} \tilde{p}_k(z)^{m'_k}. \tag{A.16}$$

with an m'_k to be determined.

Let D and $\tilde{D}$ denote the time shift on the undecimated and decimated sequence, respectively. Then

$$\tilde{p}(\tilde{D})\tilde{s}^{(i)}|_t = \tilde{p}(D^\nu)s|_{\nu t + i}. \tag{A.17}$$

The left hand side (lhs) of this equation is zero by definition. Since s has minimal polynomial $p(z)$, a necessary and sufficient condition for this to hold on the right hand side (rhs) is

$$p(z) \mid \tilde{p}(z^\nu) = \prod_{k \in \tilde{J}} \prod_{j=0}^{\tilde{d}_i - 1} (z^\mu - {\alpha_k}^{\mu 2^j})^{2^\kappa m'_k} \tag{A.18}$$

Now $\prod_{i \in \tilde{J}_k} (z - \alpha_i^{2^j}) | z^\mu - \alpha_k^{\mu 2^j}$ holds since every factor of the lhs divides the rhs and since these factors are relatively prime. The divisibility condition (A.18) is thus satisfied whenever $m_i \leq 2^\kappa m'_i$ and $\tilde{m}_i = \lceil \frac{m_i}{2^\kappa} \rceil$ is the smallest such m'_i.

Next we prove the existence of a sequence s such that $\tilde{s}^{(0)}$ has minimal polynomial $\tilde{p}(z)$. Since $\gcd(\tilde{p}_k(z), \tilde{p}_{k'}(z)) = 1$ if $k \neq k'$, we can restrict ourselves to $p(z) = \prod_{i \in \tilde{J}_k} p_i(z)^{m_i}$ and $\tilde{p}(z) = \tilde{p}_k(z)^{\tilde{m}_k}$. We consider

$$s_t = \sum_{i \in \tilde{J}_k} \binom{2^l - m_i}{t}_2 \mathrm{Tr}(\gamma_i {\alpha_i}^t) \tag{A.19}$$

with $\gamma_i \neq 0$, which by theorem A.1, has minimal polynomial $p(z)$. By using lemma A.2 the decimation $\tilde{s}_t^{(0)}$ becomes

$$\tilde{s}_t^{(0)} = s_{\nu t} = \sum_{i \in \tilde{J}_k} \sum_{j=0}^{\tilde{m}_i} c_j \binom{2^{\tilde{l}} - j}{t}_2 \mathrm{Tr}(\gamma_i \alpha_k{}^{\mu t}). \qquad (A.20)$$

Define the set of factors with maximal multiplicity $\tilde{I}_k := \{i \in \tilde{J}_k : \tilde{m}_i \geq \tilde{m}_j, \ \forall\, j \in \tilde{J}_k\}$ and the sequence $\tilde{u}$ associated with these factors

$$\tilde{u}_t := \binom{2^{\tilde{l}} - \tilde{m}_k}{t}_2 \sum_{i \in \tilde{I}_k} \mathrm{Tr}(\gamma_i \alpha_k{}^{\mu t}), \qquad (A.21)$$

then by theorem A.1, $\tilde{s}^{(0)} - \tilde{u}$ is generated by $\tilde{p}_k(z)^{\tilde{m}_k - 1}$. If

$$\sum_{i \in \tilde{I}_k} \mathrm{Tr}_{\mathrm{GF}(2^{d_i})|\mathrm{GF}(2^{\tilde{d}_k})}(\gamma_i) := \sum_{i \in \tilde{I}_k} \sum_{j=0}^{\frac{d_i}{\tilde{d}_k} - 1} \gamma_i{}^{2^{\tilde{d}_k j}} \neq 0, \qquad (A.22)$$

the same theorem implies that $\tilde{p}_k(z)^{\tilde{m}_k}$ is minimal for $\tilde{u}$ and thus also for $\tilde{s}^{(0)} = \tilde{u} + (\tilde{s}^{(0)} - \tilde{u})$. The condition can always be fullfilled, *e.g.*, by choosing a set $\{\gamma_i\}_{i \in \tilde{J}_k}$ with $\mathrm{Tr}(\gamma_i) = 1$ for one index i and $\mathrm{Tr}(\gamma_{i'}) = 0$, $\forall i' \in \tilde{J}_k \setminus \{i\}$.

b. We prove the assertions for ν relatively prime to T. $\tilde{J} = J$ and $\tilde{d}_k = d_k$ are trivial. For the proof of $\tilde{m}_k = m_k$ we note that

$$T = 2^{\max_{i \in J} \lceil \log_2 m_i \rceil} \cdot \mathrm{lcm}\{T_i\}_{i \in J}, \qquad (A.23)$$

with T_i the period of α_i (theorem A.1). Thus $\gcd(\nu, T) = 1$, *i.e.*,

$$2^{\min\{\kappa, \max_{i \in J} \lceil \log_2 m_i \rceil\}} \gcd(\mu, \mathrm{lcm}\{T_i\}_{i \in J}) = 1 \qquad (A.24)$$

implies either $\kappa = 0$ *i.e.*, $\tilde{m}_k = m_k$ or $\max_i \lceil \log_2 m_i \rceil = 0$, *i.e.*, $m_i = 1$ and $\tilde{m}_i = \lceil \frac{1}{2^\kappa} \rceil = 1 = m_i$.

c. Finally, we have to prove that $\tilde{p}(D)$ is minimal for all decimations $\tilde{s}^{(i)}$ of all sequences s generated by $p(z)$, whenever $\gcd(\nu, T) = 1$. By the primality condition, there is an e such that $\nu^e \equiv 1 \bmod T$, *i.e.*, an e-fold decimation by ν leads back to the original sequence. Now, degree $\tilde{p}(z) \leq$ degree $p(z)$, for every one of the decimations

$$deg\ p(z) \geq deg\ \tilde{p}(z) \geq deg\ \tilde{\tilde{p}}(z) \geq \ldots deg\ p(z) \qquad (A.25)$$

which is only possible if all degrees are equal.

REFERENCES

[1] W.M. Siebert, "A Radar Detection Philosophy," *IRE Trans. Inform. Theory,* vol. IT-2, pp. 204-221, (1956).

[2] R.L. Pickholtz, D.L. Schilling, L.B. Milstein, "Theory of Spread-Spectrum Communications - A Tutorial," *IEEE Trans. Commun.,* vol. COM-30, pp. 855-884, May 1982.

[3] P.H. Bardell, W.H. McAnney, "Pseudorandom Arrays for Built-In Tests," *IEEE Trans. Comput.,* vol. C-35, pp. 653-658, July 1986.

[4] R.A. Rueppel, *Analysis and Design of Stream Ciphers,* Springer-Verlag, Berlin, Heidelberg 1986.

[5] T. Kasami, "A Decoding Procedure for Multiple-Error-Correcting Cyclic Codes," *IEEE Trans. Inform. Theory,* vol. IT-10, pp. 134-138, April 1964.

[6] W.G. Chambers, S.M. Jennings, "Linear Equivalence of Certain BRM Shift-Register Sequences," *Electron. Lett.,* vol. 20, pp. 1018-1019, Nov. 1984.

[7] A. Lempel, W.L. Eastman, "High Speed Generation of Maximal Length Sequences," *IEEE Trans. Comput.,* vol. C-20, pp. 227-229, Feb. 1971.

[8] K.H. Möhrmann, "Erzeugung von binären Quasi-Zufallsfolgen hoher Taktfrequenz durch Multiplexen," *Siemens Forsch.- u. Entwickl.-Ber.,* vol. 3, pp. 218-224 (1974).

[9] R. Eier, H. Malleck, "Anwendung von Multiplextechniken bei der Erzeugung von schnellen Pseudozufallsfolgen," *NTZ,* vol. 28, pp. 227-231 (1975).

[10] F. Surböck, H. Weinrichter, "Interlacing Properties of Shift-Register Sequences with Generator Polynomials Irreducible over GF(p)," *IEEE Trans. Inform. Theory,* vol. IT-24, pp. 386-389, May 1978.

[11] C.F. Woodcock, P.A. Davies, "The Generation of High-Speed Binary Sequences from Interleaved Low-Speed Sequences," *IEEE Trans. Commun.,* vol. COM-35, pp. 115-117, Jan. 1987.

[12] M.Y. Hsiao, "Generating PN Sequences in Parallel," *Proc., 3rd. Ann. Princeton Conf. Inform. Sci. Syst.,* Princeton NJ., pp. 397-401, March 1969.

[13] W.J. Hurd, "Efficient Generation of Statistically Good Pseudonoise by Linear Interconnected Shift Registers," *IEEE Trans. Comput.,* vol. C-23, pp. 146-152, Feb. 1974.

[14] D.G. Maritsas, A.C. Arvillias, A.C. Bounas, "Phase-Shift Analysis of Linear Feedback Shift Register Structures Generating Pseudorandom Sequences," *IEEE Trans. Comput.,* vol. C-27, pp. 660-668, July 1978.

[15] W.W. Warlick Jr., J.E. Hershey, "High-Speed M-Sequence Generators," *IEEE Trans. Comput.,* vol. C-29, pp. 398-400, May 1980.

[16] B.J.M. Smeets, *Some Results on Linear Recurring Sequences,* Ph.D. Thesis, Technical University of Lund, Sweden (1987).

[17] B.J.M. Smeets, W.G. Chambers, "Windmill Generators: A Generalization and an Observation of How Many There Are," *Advances in Cryptology - EUROCRYPT'88,* Lect. Notes in Comp. Science, vol. 330, pp. 325-330, Springer-Verlag (1988).

[18] N. Zierler, "Linear Recurring Sequences," *J. Soc. Indust. Appl. Math.,* vol. 7, pp. 31-48, March 1959.

[19] P.F. Duvall, J.C. Mortick, "Decimation of Periodic Sequences," *SIAM J. Appl. Math.,* vol. 21, pp. 367-372, Nov. 1971.

[20] H. Niederreiter, "A Simple and General Approach to the Decimation of Feedback Shift-Register Sequences," *Probl. of Control and Inform. Theory,* vol. 17, pp. 327-331 (1988).

[21] R.G. Swan, "Factorization of Polynomials over Finite Fields," *Pac. J. Math.,* vol. 12, pp. 1099-1106 (1962).

[22] L.M. Milne-Thomson, *The Calculus of Finite Differences,* Macmillan and Co., London 1951.

[23] M. Ward, "The Arithmetical Theory of Linear Recurring Series," *Trans. Am. Math. Soc.,* vol. 35, pp. 600-628, Jan. 1933.

[24] E.L. Key, "An Analysis of the Structure and Complexity of Nonlinear Binary Sequence Generators," *IEEE Trans. Inform. Theory,* vol. IT-22, pp. 732-736, Nov. 1976.

[25] T. Herlestam, "On Functions of Linear Shift Register Sequences," *Advances in Cryptology - EUROCRYPT'85,* Lect. Notes in Comp. Science, vol. 219, pp. 119-129, Springer-Verlag (1986).

[26] H. Hasse, "Theorie der höheren Differentiale in einem algebraischen Funktionenkörper mit vollkommenem Konstantenkörper bei beliebiger Characteristik," *J. reine angew. Math.*, vol. 175, pp. 50-54 (1936).