Pseudo Random Number Generation on FPGA

Tamás Herendi[*,**]
[*] University of Debrecen/Faculty of Informatics, Debrecen, Hungary

***Abstract*—The aim of the present paper is to show the theoretical background of the construction of uniformly distributed (UD) pseudo random number (PRN) sequences with good properties and efficient implementation on FPGA.**

I. Introduction

Pseudo random numbers play an essential role in many applications, such as simulations, Monte-Carlo methods, coding and cryptography. There are numerous ways to produce sequences of pseudo random numbers see e.g. [4]. The use of linear feedback shift registers (LFSR) for generating pseudo random bit sequences is a well known application. It has several advantages and disadvantages, too. One of the good properties is the very simple structure of the generators. This gives the possibility for easy implementation, both on classical architecture computers and in FPGAs. On the negative side, however, the sequences contain only single bits and since the consecutive members have a strong correlation, it is not possible to construct sequences of several bit numbers from them, keeping the properties of the original ones. Knuth in [4] shows that one should be very careful when combining uniformly distributed (UD) sequences to achieve a new one, since the new sequence not necessarily remains UD. Lidl and Niederreiter in [5] give a generalization of LFSR to finite fields and show a method for constructing UD sequences in these structures. However, since they use a more complex algebraic structure, the implementations of such generators are not quite simple. In the next chapters, we give a generalization of LFSR to residue ring structures and show the possibility of internal representation in the general system.

II. Basic definitions and properties

Definition 1

Let $a_0, \dots, a_{d-1}$ be integers and $u = \{u_n\}_{n=0}^{\infty}$ is an infinite sequence of integers, satisfying the recurrence relation

$$u_{n+d} = a_{d-1}u_{n+d-1} + \cdots + a_0 u_n$$

for any $n = 0,1, \dots$ Then u is called a linear recurring sequence (LRS), where $a_0, \dots, a_{d-1}$ are the coefficients and $u_0, \dots, u_{d-1}$ are the initial values of u.

The order of the recurrence is d and the corresponding characteristic polynomial is

$$P(x) = x^d - a_{d-1}x^{d-1} - \cdots - a_0 .$$

Definition 2

Let u be the integer LRS defined by the coefficients $a_0, \dots, a_{d-1}$ and initial values $u_0, \dots, u_{d-1}$.

Denote by

$$\bar{u}_n(k) = (u_n, \dots, u_{n+k-1})$$

the n^{th} k-dimensional state vector and by

$$M(u) = \begin{pmatrix} 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \\ a_0 & a_1 & \dots & a_{d-2} & a_{d-1} \end{pmatrix}$$

the companion matrix of u.

Remark

With the above notations we may write

$$\bar{u}_n(d) = M(u)^n \bar{u}_0(d).$$

Definition 3

Let u be a sequence of integers and s be an integer. We say that u reduced modulo 2^s is periodic with period length ρ, if there exists a positive integer ρ_0, such that $u_n \equiv u_{n+\rho} \bmod 2^s$, i.e. u_n and $u_{n+\rho}$ has the same residue divided by 2^s, for all $n \geq 0$. The least such ρ is called the minimal period length of the sequence.

If $\rho_0 = 0$ then u is called purely periodic.

Remark

Let u be LRS of integers. By definition, u reduced by 2^s is periodic for any positive s.

Remark

If we set $s = 1$ in Definition 3, we arrive to the mathematical model of the well known LFSR. Here the coefficients of the recurrence are 0 and 1. The practical meaning is that the values in registers corresponding to coefficients equal to 1 are fed back through an exclusive or filter while the values in registers corresponding to coefficients equal to 0 are simply stored for future use.

Definition 4

Let u be a sequence of integers and s be a positive integer. We say that u is uniformly distributed (UD) reduced modulo 2^s, if

$$\lim_{N\to\infty} \frac{1}{N} \#\{n \leq N | u_n \equiv a \bmod 2^s\} = \frac{1}{2^s}$$

for all integers a, i.e. if we observe a reasonable long segment of the sequence, the relative frequencies of the residues derived from the members of the sequence by reducing modulo 2^s are close to each other.

Remark

In the case of the most common LFSR ($s = 1$) the UD property implies that the relative frequency of the 0s (and

similarly the relative frequency of 1s *are approaching* 0.5 .

Remark

General conditions for the UD of LRSs can be found in [6], [7], [1] and [2].

III. THEORETICAL BACKGROUND

The main idea of the presented construction of PRN generators is that we choose an initial UD sequence and try to extend to some more complex one. In our case the basic generator is a proper LFSR which is converted by a simple but high time complexity transformation.

Example

Fig. 1 shows an LFSR composed from 6 *storage registers and 4 XOR gates. Sending* $[0,0,0,0,0,1]$ *to the input, located on the left, one get*

$$[0,0,0,0,0,1,0,1,1,0,$$
$$1,1,1,0,0,1,1,1,1,1,$$
$$0,1,0,0,1,0,0,0,1,1,$$
$$0,0,0,0,0,1,0,1,1,\dots]$$

on the output, located on the right. The sequence repeats from the fourth line. The period length of the repetition is 30. The formal description of the shift register is

$$u_{n+6} = u_{n+4} \otimes u_{n+3} \otimes u_{n+2} \otimes u_{n+1} \otimes u_n \ ,$$

where $\otimes$ *denotes the XOR operation.*

The mathematical model of the sequence is

$$u_{n+6} = u_{n+4} + u_{n+3} + u_{n+2} + u_{n+1} + u_n \ \ mod\ 2 \ .$$

Here the order d *of the recurrence is* 6, *while the characteristic polynomial is*

$$P(x) = x^6 - x^4 - x^3 - x^2 - x - 1 \ .$$

By general theory, one can prove that u_n *is UD modulo* 2, *which can be checked by counting the members of the sequence (any consecutive subsequence of length* 30 *contains exactly* $15\ 0's$ *and* $15\ 1's$*).*

There are two weakness of the above defined sequence: first, its period is rather short and second, the generation of numbers greater than 1 is possible only by tying together some k bits using a proper selection method, e.g. the k consecutive members of the sequence. However, the computed number sequence looses the UD property. Both problem can be solved using the following results.

Definition 5

An LFSR (or in general an LRS) with initial values $[0,0,\dots,0,1]$ *is called an impulse response sequence (IR).*

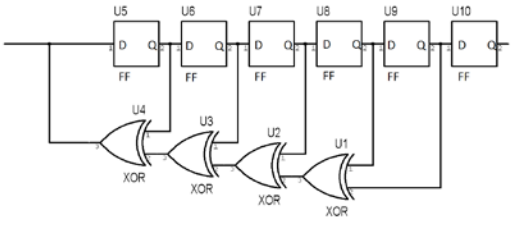

Figure 1. *LFSR with 6 register*

Theorem 1

Let u *be an IR LRS of integers with characteristic polynomial*

$$P(x) \equiv (x+1)^2 Q(x) \ mod\ 2 \ ,$$

where $Q(x)$ *is a degree* k, *modulo* 2 *irreducible polynomial. Then* u *is UD modulo* 2. *Choosing* $Q(x)$ *with a little care,* u *have period length* $2^{k+1} - 2$.

The proof of Theorem 1 can be found in [2].

Remark

Thanks to Theorem 1, theoretically, one can simply create an UD LRS generated by an LFSR with long period. Choosing the degree of $Q(x)$ *to be large enough, the period length of the sequence became reasonable. For instance, set* $k = 1000$ *then the period length is* $2^{1001} - 2$, *which is approximately* 10^{302}.

For the construction of pseudo random number (PRN) sequences the following result is important.

Theorem 2

Let $Q(x)$ *be an irreducible polynomial modulo* 2 *of degree* k *and let*

$$P(x) \equiv (x+1)^2 Q(x) \ mod\ 2 \ .$$

Further, let

$$P_1(x) = P(x),$$
$$P_2(x) = P(x) - 2,$$
$$P_3(x) = P(x) - 2x,$$
$$P_4(x) = P(x) - 2x - 2 \ ,$$

and the corresponding IR LRSs are $u^{(1)}$, $u^{(2)}, u^{(3)}$ *és* $u^{(4)}$, *respectively.*

Then exactly one of the sequences $u^{(i)}$ *is UD reduced modulo* 2^s *for all* $s \geq 1$. *With a proper* $Q(x)$ *we can assume, that the period length of the sequences are* $2^s(2^{k+1} - 2)$.

The proof of Theorem 2 can be found in [2].

Remark

By Theorem 2, if we find the LRS $u^{(i)}$ *with the strong UD property, then the sequence reduced modulo* 2^s *provides pseudo random integers of* s *bits.*

IV. ALGORITHM

The algorithm, presented below, based on the previous results and on their proofs, lets us to construct LRSs which are UD modulo 2^s for all $s \geq 1$.

Step 1.

Choose a suitable integer k and find a monic polynomial Q(x) of degree k, which reduction modulo 2 is irreducible. If k is a so called Mersenne-prime, the search for a proper Q(x) is more simple. Let $= 2^k - 1$.

Step 2.

Calculate the polynomial

$$P(x) = x^{k+2} - p_{k+1}x^{k+1} - \cdots - p_0$$

by the relation

$$P(x) \equiv (x+1)^2 Q(x) \bmod 2\,,$$

where the coefficients $p_0, \ldots, p_{k+1} \in \{0,1\}$ and let

$$P'(x) \equiv (x+1)Q(x) \bmod 2\,.$$

Further, let

$$P_1(x) = P(x),$$
$$P_2(x) = P(x) - 2,$$
$$P_3(x) = P(x) - 2x,$$
$$P_4(x) = P(x) - 2x - 2.$$

Step 3.

Calculate the companion matrices $M_{(i)}$ corresponding to the characteristic polynomial $P_i(x)$. Check for which i the relation $M_{(i)}\bar{1} \equiv \bar{1} \bmod 4$ holds. Keep the two matrices which satisfy the congruence and denote them by M_1 and M_2 .

Step 4.

Compute the matrix $(M_1)^{2\rho} \bmod 4$. If $(M_1)^{2\rho} \equiv E \bmod 4$, where E is the unit matrix, then the sequence we are looking for is the LRS given by the recurrence relation corresponding to M_2, otherwise it is the LRS given by the recurrence relation corresponding to M_1.

Remark

The proofs of the theorems and correctness of the algorithms and the detailed construction of LRSs can be found in [2].

Remark

The most time consuming part of the algorithm is the computation of $(M_1)^{2\rho}$ *mod* 4. *If the value of* k *is around* 1000, *then we have to multiply 1000 times matrices of the size* $1000x1000$.

V. IMPLEMENTATION

The practical use of LRSs as PRN generators rise two issues. First, one has to find the parameters of a suitable LRS. Second, if one has the parameters of a proper LRS, it is still question how to compute the members of the sequence.

To find the coefficients of a good generator, as it was mentioned before, is a time consuming task. At the present state of technology, using a single processor (single core) computer one can check the UD property of a given LRS within a reasonable time (i.e. in a few weeks) for about $k = 10000$. However, [3] describes a construction for an FPGA which increases the speed of the computations by a factor around 1000. The design developed in the paper uses tiny processors for computing dot product of vectors of small integers. 400 of the mentioned processors are connected to a network to compute row-column combinations of matrices. Since the storage capacity of an FPGA is rather limited, the most difficult problem during the execution is to move a large amount of data to the processing units in time. Organizing the order of computation of partial products and data flow, however, make it possible that all important units can work continuously. It makes the design very effective, which yields, that either the running time can be reduced to some hours or the size of the recurrence can be increased to 30000, which provides already an extremely huge PRN generator with period length more than 2^{30000}.

Theoretically and finally in practice, we are able to calculate parameters of large LRSs for PRN generator purposes. Although the computation of the members of a LRS is a very simple as an algorithmic problem, the length of the core of the generator causes the reduction of the speed of the calculation. For example, if we want to use a recurrence of order 10000 for integers with 1024 bits, we have to store 10000 numbers of 128 bytes and determining the next member of the sequence, depending on the properties of the coefficients, we have to apply approximately 5000 addition on numbers of the given size. The amount and structure of operations one have to execute during computation of the sequence immediately bring the possibility of parallelization into sight. Furthermore, since the applied operations are quite simple, one feels that the abilities of a complex processor are unexploited. If one could produce small reduced instruction set processors, the efficiency will increase and the space occupied by one unit decreases. Using an FPGA is ideal for these requirements.

In the case of LFSRs, it is well known, that reorganizing the connections of the structure can considerably decrease the number of necessary clock cycles. If one implements them directly by the definition the result is the so called external LFSR representation. The speed of this representation strongly depends on the speed of the applied operation (in this case it is the XOR). The efficiency of the external representation can be increased by the use of operation networks. Furthermore, the internal representation of an LFSR reduces the operational time to one clock cycle.

Implementation of general LRS generators

The general LRS generators can have similar physical structure as the LFSRs, substituting the flip-flops by storage registers and the XOR gates by operational components. However, here some extra unit should be added for the realization of multiplications. Fig. 2 shows the flow chart of the generator, corresponding to the recurrence relation

$$u_{n+d} = a_{d-1}u_{n+d-1} + \cdots + a_0 u_n\,.$$

If the additive operation has associative and commutative properties (as the usual addition and modular addition have) we can redraw again the structure to an operation network. (See Fig. 3) Such a network provides the result of operations in logarithmic time in the best case, compared to the sequential execution. In general, the construction of an operation network would require more space than the serial one, but thanks to the special algebraic properties, in our case there are no increase in the area used by the components. One has to be careful,

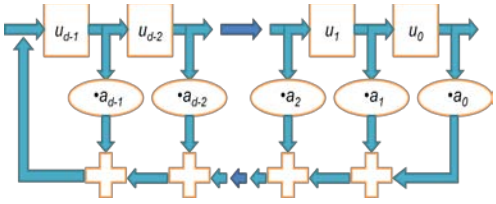

Figure 2. The LRS generator $u_{n+d}=a_{d-1}u_{n+d-1}+\ldots+a_0u_n$

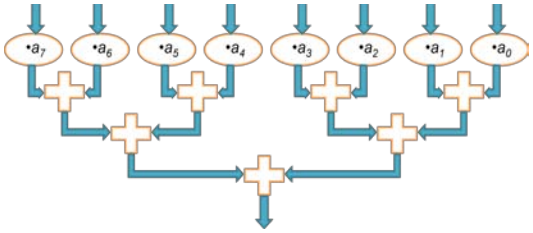

Figure 3. Network of associative and commutative operations

however, with the increasing amount of interconnections.

Further improvement can be achieved by generalization of the internal representation of LFSRs to the universal case.

Theorem 3

Let u be a LRS generated by the recurrence relation

$$u_{n+d} = a_{d-1}u_{n+d-1} + \cdots + a_0 u_n \qquad (1)$$

and let the vector sequence v be defined by the following

$$v^0{}_n = a_0 u_{n-1}$$
$$v^1{}_n = a_0 u_{n-2} + a_1 u_{n-1}$$
$$\cdots$$
$$v^{d-2}{}_n = a_0 u_{n-d+1} + \cdots + a_{d-2} u_{n-1}$$
$$v^{d-1}{}_n = a_0 u_{n-d} + \cdots + a_{d-2} u_{n-2} + a_{d-1} u_{n-1}\ .$$

Then

$$u_n = v^{d-1}{}_n \qquad (2)$$

and

$$v^i{}_n = v^{i-1}{}_n + a_i u_{n-1}\ . \qquad (3)$$

Proof.

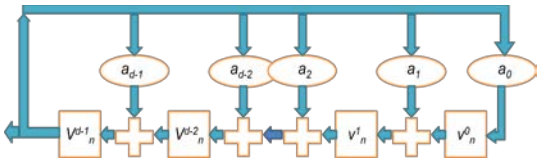

Figure 4. Internal representation of a LRS generator

$u_{n+d}=a_{d-1}u_{n+d-1}+\ldots+a_0u_n$

Substituting $n-d$ for n in (1) we obtain

$$u_n = a_{d-1}u_{n-1} + \cdots + a_0 u_{n-d}. \qquad (4)$$

The right hand side of this equation is just the definition of $v^{d-1}{}_n$, which proves (2).

Let fix i. Substituting the definition of $v^{i-1}{}_n$ in (3) we get the equation of the definition of $v^i{}_n$, which proves (3).

The corollary of Theorem 3 is that the internal representation of LFSRs can be generalized to arbitrary LRS. This gives the possibility of implementation of a LRS as it is shown on Fig. 4.

The physical realization of a LRS generator on FPGA still has some limitations. If we want to produce a sequence of 1024-bit integers, we have to place on the device 1024-bit adders. The space on even the largest FPGAs enables only a few of such an adder to implement. The solution to this problem is that we execute only a segment of the addition at once and then shift the sequence to the adder chain again. The result is an approximately 100 times faster generator than one can create on a conventional computer.

Finding special recurrences with sparse coefficients decrease the need for additions and hence increase the speed of computations of the next members of the sequence.

Acknowledgments

Research supported by the TÁMOP 4.2.1./B-09/1/KONV-2010-0007 project and TARIPAR3 project grant Nr. TECH 08-A2/2-2008-0086

References

[1] T. Herendi "Uniform distribution of linear recurring sequences modulo prime powers", *Journal of Finite Fields and Applications*, vol.10, 2004, pp. 1–23

[2] T. Herendi "Construction of uniformly distributed linear recurring sequences modulo powers of 2", in press

[3] T. Herendi and S.R. Major "Modular exponentiation of matrices on FPGA-s", in press

[4] D.E. Knuth, *The art of computer programming*, Addison-Wesley,1973

[5] R. Lidl, H. Niederreiter, *Introduction to finite fields and their applications*, Cambridge University Press, 1986

[6] H. Niederreiter, J.S.Shiue "Equidistribution of linear recurring sequences in finite fields", *Indag. Math.*, vol. 39, 1977 pp. 397--405

[7] H. Niederreiter, J.S.Shiue "Equidistribution of linear recurring sequences in finite fields II", *Acta Arith.*, vol. 38, 1980, pp. 197—207