SHORT PAPER

Hardware implementation of the Lehmer random number generator

A.P. Papliński
N. Bhattacharjee

Indexing terms: Lehman random number generator, Matrix

Abstract: Multiplicative linear congruential pseudorandom number generators are a popular choice for many software routines. The paper describes fast hardware implementation of the Lehmer generator which belongs to the above class. First, using the Sylvester resultant matrices it is shown that the algorithm to generate the next random number, which is based on multiplication mod$M$, can be reduced to the problem of addition/subtraction of six appropriately rotated copies of the current random number. Secondly, addition/subtraction of six numbers mod$M$ can be performed by means of three carry-save adders, one carry-propagate subtracter, and one carry-propagate adder.

1 Introduction

The Lehmer generator is a popular choice of software implementation of random number generators [1]. Our interest in this device stems from its application in a popular scientific and engineering software package, MATLAB [2].

Properties of the Lehmer generator, which belongs to the class of the multiplicative linear congruential generators [3, 4], have been exhaustively studied [5]. Generation of a pseudorandom sequence of integer numbers: $z^{(1)}, z^{(2)}, z^{(3)}, \ldots$, is in this case described by the following iterative eqn. [1]:

$$z^{(n+1)} = f(z^{(n)}) \qquad \text{for } n = 1, 2, \ldots$$

where the generating function $f(\cdot)$ is defined as

$$r = f(z) = a \cdot z \bmod M \quad \text{for all } z \in 1, 2, \ldots, M-1 \qquad (1)$$

For the Lehmer generator the modulus $M = 2^{31} - 1$ is a Mersenne prime [6], and the multiplier $a = 7^5 = 16807$, which can be represented by the 15-bit binary numeral: $\mathbf{a}_{14:0} = [1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1]$. Random numbers $z^{(n)}$ are integers represented by 31-bit numerals. As an additional step, the integer sequence of pseudorandom numbers is typically mapped into the fractional sequence $u^{(n)}$ through division by the modulus M:

$$u^{(n)} = z^{(n)}/M \simeq z^{(n)} 2^{-31} \qquad (2)$$

2 Parallelised form of the algorithm

In this Section we consider a parallelised version of the algorithm for generation of the next random number r from the previous one z using binary multiplication modulo M as in eqn. 1.

When describing hardware implementation of numerical algorithms, it is often convenient to make a distinction between numbers and their binary numerals. If a is an integer number, its $(n + 1)$-bit binary numeral will be denoted as $\mathbf{a}_{n:0}$. Using a matrix notation, a number a can now be represented as an inner product of its binary numeral $\mathbf{a}_{n:0}$ and a vector of powers of 2 (weights) $\mathbf{w}_{n:0}$ as follows:

$$a = \mathbf{a}_{n:0} \cdot \mathbf{w}_{n:0} = \sum_{i=0}^{n} a_i \cdot 2^i \qquad (3)$$

where $\mathbf{a}_{n:0} = [a_n\ a_{n-1} \ldots a_1\ a_0]$ is the numeral of a ($a_i \in \{0, 1\}$), $\mathbf{w}_{n:0} = [2^n\ 2^{n-1} \ldots 2^1\ 2^0]'$ is the vector of binary weights.

If we use the above notation, multiplication of numbers can be expressed as a matrix product of a multiplier numeral and the Sylvester resultant matrix of the multiplicand [7, 8]. If the m-bit multiplicand numeral is $\mathbf{z}_{m-1:0} = [z_{m-1}\ z_{m-2} \ldots z_1\ z_0]$, the product of two numbers a and z may be represented by the following matrix equation for binary numerals:

$$a \cdot z = \mathbf{a}_{n:0} \cdot \langle \mathbf{z} \rangle_n \cdot \mathbf{w}_{n+m-1:0} \qquad (4)$$

where $\langle \mathbf{z} \rangle_n$ is the $(n + 1) \times (n + m)$ Sylvester resultant matrix (also known as the convolution matrix), which is formed from shifted numeral $\mathbf{z}_{m-1:0}$ in the following way:

$$\langle \mathbf{z} \rangle_n = \underbrace{\begin{bmatrix} z_{m-1} & z_{m-2} & \cdots & z_0 & & & \\ & z_{m-1} & z_{m-2} & \cdots & z_0 & & \mathbf{0} \\ & \mathbf{0} & \ddots & \ddots & & \ddots & \\ & & & z_{m-1} & z_{m-2} & \cdots & z_0 \end{bmatrix}}_{n+m} \Bigg\updownarrow n+1 \qquad (5)$$

The angle brackets have been used to denote the resultant matrix. The 0s in the resultant of eqn. 5, represent appropriate triangles of zeroes.

IEE Proceedings online no. 19960100
Paper first received 22nd December 1994 and in revised form 3rd October 1995
The authors are with the Department of Digital Systems, Monash University, Wellington Road, Clayton 3168, Australia

In the case of the Lehmer generator, we have $m = 31$ and $n = 14$. Therefore, in eqn. 4 the numeral $\mathbf{a}_{14:0}$ is of the size 1×15, the size of the resultant matrix $\langle \mathbf{z} \rangle_{14}$ is 15×45, and the result, the product of a and z, is represented by the 1×45 numeral.

In general, eqn. 4 can be thought of as a parallelised description of multiplication of two numerals. Each row of the resultant represents a shifted numeral $\mathbf{z}_{m-1:0}$ which is multiplied by the respective digit of the multiplier and the columns of the resultant are summed up to give the 'pseudodigits' of the result.

The next step in the specification of the algorithm is to determine a remainder from division of $a \cdot z$ by $M = 2^{31} - 1$, that is, to perform the operation mod$M$. To expand the product into the form:

$$a \cdot z = q \cdot M + r$$

where q is the quotient and r is the remainder, the resultant matrix is first partitioned into two submatrices: the left one C of the size $(n + 1) \times n$, and the right one D of the size $(n + 1) \times m$, as follows:

$$\langle \mathbf{z} \rangle_n = \left[\begin{array}{ccc|ccccc} z_{m-1} & \cdots & z_{m-n} & z_{m-n-1} & \cdots & z_0 & \\ 0 & \ddots & \vdots & z_{m-n-2} & \cdots & z_1 & z_0 & \mathbf{0} \\ \vdots & \ddots & z_{m-1} & \vdots & & & \ddots & \ddots \\ 0 & \cdots & 0 & z_{m-1} & \cdots & \cdots & z_1 & z_0 \end{array}\right] = [C \quad | \quad D] \tag{6}$$

Combining eqns. 4 and 6, we have:

$$\begin{aligned} a \cdot z &= \mathbf{a}_{n:0} \cdot \langle \mathbf{z} \rangle_n \cdot \mathbf{w}_{n+m-1:0} \\ &= \mathbf{a}_{n:0} \cdot [C \quad D] \cdot \mathbf{w}_{n+m-1:0} \\ &= \mathbf{a}_{n:0} \cdot (C \cdot \mathbf{w}_{n-1:0} \cdot 2^m + D \cdot \mathbf{w}_{m-1:0}) \\ &= \mathbf{a}_{n:0} \cdot (C \cdot \mathbf{w}_{n-1:0} \cdot 2^m - C \cdot \mathbf{w}_{n-1:0} + D \cdot \mathbf{w}_{m-1:0} \\ &\qquad + C \cdot \mathbf{w}_{n-1:0}) \\ &= \mathbf{a}_{n:0} \cdot (C \cdot \mathbf{w}_{n-1:0} \cdot M + D \cdot \mathbf{w}_{m-1:0} + C \cdot \mathbf{w}_{n-1:0}) \end{aligned}$$

where we used the relationship $\mathbf{w}_{n+m-1:0} = \mathbf{w}_{n-1:0} \cdot 2^m + \mathbf{w}_{m-1:0}$. Taking into account that, for any integer q,

$$(q \cdot M + r) \bmod M = r \bmod M$$

eqn. 1 can now be written as:

$$\begin{aligned} r &= (a \cdot z) \bmod M \\ &= \mathbf{a}_{n:0}(D \cdot \mathbf{w}_{m-1:0} + C \cdot \mathbf{w}_{n-1:0}) \bmod M \\ &= \mathbf{a}_{n:0}(D \cdot \mathbf{w}_{m-1:0} + [0 \quad C] \cdot \mathbf{w}_{m-1:0}) \bmod M \end{aligned}$$

Finally, we have:

$$r = (\mathbf{a}_{n:0} \cdot E \cdot \mathbf{w}_{m-1:0}) \bmod M \tag{7}$$

where

$$E = D + [0 \quad C] = \left[\begin{array}{cccccc} z_{m-n-1} & \cdots & z_0 & z_{m-1} & z_{m-2} & \cdots & z_{m-n} \\ z_{m-n-2} & \cdots & z_1 & z_0 & z_{m-1} & \cdots & z_{m-n+1} \\ \ddots & & \ddots & \ddots & \ddots & & \ddots \\ \ddots & & \ddots & \ddots & \ddots & & \ddots \\ z_{m-2} & \cdots & z_{n-1} & z_{n-2} & z_{n-3} & \cdots & z_{m-1} \\ z_{m-1} & \cdots & z_n & z_{n-1} & z_{n-2} & \cdots & z_0 \end{array}\right] \begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{l} \longleftarrow n \longrightarrow \\ \\ \\ \\ \\ \longleftarrow m \longrightarrow \end{array} \tag{8}$$

is a $(n + 1) \times m$ cyclic convolution matrix formed from the numeral $\mathbf{z}_{m-1:0}$. If we now take into account that the multiplier a is represented by the binary numeral in which only bits a_{14}, a_8, a_7, a_5, a_2, a_1, a_0 are 1, then eqn. 7 may be written as:

$$\begin{aligned} r &= (\mathbf{a}_{14:0} \cdot E \cdot \mathbf{w}_{30:0}) \bmod M \\ &= (E_{14:} + E_{8:} + E_{7:} + E_{5:} \\ &\qquad + E_{2:} + E_{1:} + E_{0:})\mathbf{w}_{30:0} \bmod M \end{aligned} \tag{9}$$

The sequence of three ones in the multiplier a can be replaced by +1 and –1 on the appropriate positions, because $(1\ 1\ 1)_2 = (1\ 0\ 0\ \bar{1})_2$. Therefore, finally, the implementation equation takes on the following form:

$$r = (E_{14:} + E_{8:} + E_{7:} + E_{5:} + E_{3:} - E_{0:})\mathbf{w}_{30:0} \bmod M \tag{10}$$

where

$$\begin{bmatrix} E_{14:} \\ E_{8:} \\ E_{7:} \\ E_{5:} \\ E_{3:} \\ E_{0:} \end{bmatrix} = \begin{bmatrix} z_{16} & z_{15} & z_{14} & \cdots & z_0 & z_{30} & \cdots & z_{24} & z_{23} & z_{22} & z_{21} & z_{20} & z_{19} & z_{18} & z_{17} \\ z_{22} & z_{21} & z_{20} & \cdots & \cdots & & & z_{30} & z_{29} & z_{28} & z_{27} & z_{26} & z_{25} & z_{24} & z_{23} \\ z_{23} & z_{22} & z_{21} & \cdots & \cdots & & & z_0 & z_{30} & z_{29} & z_{28} & z_{27} & z_{26} & z_{25} & z_{24} \\ z_{25} & z_{24} & z_{23} & \cdots & \cdots & & & z_2 & z_1 & z_0 & z_{30} & z_{29} & z_{28} & z_{27} & z_{26} \\ z_{27} & z_{26} & z_{25} & \cdots & \cdots & & & z_4 & z_3 & z_2 & z_1 & z_0 & z_{30} & z_{29} & z_{28} \\ z_{30} & z_{29} & z_{28} & \cdots & \cdots & & & z_7 & z_6 & z_5 & z_4 & z_3 & z_2 & z_1 & z_0 \end{bmatrix} \tag{11}$$

Thus, the Lehmer's algorithm for generation of pseudorandom numbers has been reduced to the problem of adding/subtracting six 31-bit numbers mod$M$, as in eqn. 10. Each of the numbers is the appropriately rotated current random number z.

3 Hardware implementation

In this Section we discuss details of the hardware implementation of eqn. 10, which describes addition and subtraction of six 31-bit numbers mod$M$.

First, let us consider the way in which operation of mod$M$ is performed in conjunction with the operation of summation or subtraction of two m-bit numbers. Let $\mathbf{a}_{m-1:0}$ and $\mathbf{b}_{m-1:0}$ represent two numbers to be added mod$M$. Let $\mathbf{s}_{m-1:0}$ and d_m represent the sum output from an adder and the output carry so that we can write:

$$\mathbf{a}_{m-1:0} + \mathbf{b}_{m-1:0} = d_m 2^m + \mathbf{s}_{m-1:0}$$

The output carry $d_m \in \{-1, 0, +1\}$, in order to account for both addition and subtraction. If the output carry is nonzero, we subtract M from the result (mod M operation), which gives:

$$(d_m 2^m + \mathbf{s}_{m-1:0}) \bmod M = \mathbf{s}_{m-1:0} + d_m 2^m - d_m(2^m - 1)$$

and finally, we have

$$(\mathbf{a}_{m-1:0} + \mathbf{b}_{m-1:0}) \bmod M = \mathbf{s}_{m-1:0} + d_m \tag{12}$$

Secondly, we can observe that operations described in eqn. 10 are equivalent to the problem of counting number of ones in each column of the matrix of eqn. 11. Having six elements in each column, one of which is to be subtracted, the number of ones in each column can vary from –1 to +5. Hence, the problem is to build a circuit which represents this number of ones in a form of the output carry bits propagated to the next position and the sum bit. If we take also into account the input carry bits which are required to make operation complete, the equation which describes

operation performed at the ith position of the generator is of the following form (the i subscript omitted for brevity):

$$(z_a + z_b + z_c + z_d + z_e - z_f + c_1 + c_2 - c_3) = 2(d_1 + d_2 - d_3 + d_4) + s_4 \quad (13)$$

where z_α are relevant bits from the ith column of the matrix 11, c_j and d_j are input and output carry bits, respectively, and s_4 is the sum bit. The circuit which implements eqn. 13 performs compression of information with the ratio 9:5. Both 9 input bits, and 5 output bits represent numbers from –2 to +7. Eqn. 13 can now be converted into a set of elementary 3-input, 2-output summation. If we also include the final carry-propagate adder, the implementation equations take the following form:

$$\begin{aligned} \Sigma 1 &: z_a + z_b + z_c = 2d_1 + s_1 \\ \Sigma 2 &: s_1 + z_d + z_e = 2d_2 + s_2 \\ \Sigma 3 &: s_2 - z_f - c_3 = -2d_3 + s_3 \\ \Sigma 4 &: s_3 + c_1 + c_2 = 2d_4 + s_4 \\ \Sigma 5 &: s_4 + c_4 + c_5 = 2d_5 + r \end{aligned} \quad (14)$$

The ith bit-slice of the generator, which is the implementation of eqn. 14 is presented in Fig. 1. In Fig. 1, input signals z_a,...,z_f have been replaced with equivalent entries from the ith column of the matrix of eqn. 11. Input and output carry signals c_i, d_j have been replaced with the equivalent $c_{i,j}$, $c_{j,i+1}$ signals.

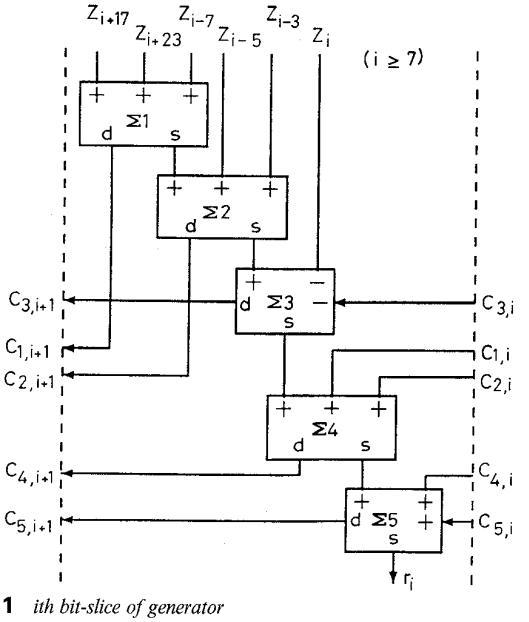

Fig. 1 *ith bit-slice of generator*

The complete generator consists of 31 such 1-bit slices. To perform operation mod$M$ as explained in eqn. 12, the output carry bits from the most significant position, are connected to the least significant outputs, that is:

$$(c_{1,0}, c_{2,0}, c_{3,0}, c_{4,0}, c_{5,0}) \Leftarrow (c_{1,31}, c_{2,31}, c_{3,31}, c_{4,31}, c_{5,31}) \quad (15)$$

Finally, we can note that the complete circuit consists effectively of three 31-bit carry-save adders (Σ1, Σ2, Σ4), one carry-propagate subtracter (Σ3), and one carry-propagate adder (Σ5). The above algorithm has been described in VHDL and implemented using an FPGA.

4 Conclusion

Using the concept of the Sylvester resultants, we have formally demonstrated that the Lehmar algorithm, which requires multiplication modulo a Mersenne prime, can be implemented using only three 31-bit CS adders, one CP subtracter and one CP adder.

A similar method can be applied to perform number-theoretical transforms [9] which involve the Fermat or the Mersenne primes of the form $2^m - \xi$.

5 Acknowledgment

This work has been supported by the Australian Research Council under the 1993 Small Grants Scheme.

6 References

1 PARK, S., and MILLER, K.: 'Random number generators: good ones are hard to find', *Commun. ACM*, 1988, **31**, pp. 1192–1201
2 MathWorks: 'MATLAB reference guide' (The MathWorks Inc., 1994)
3 KNUTH, D.: 'The art of computer programming' (Addison–Wesley, 2nd edn., 1981)
4 DEÁK, I.: 'Random number generators and simulation' (Akadémiai Kiadó, Budapest, 1990)
5 FISHMAN, G.S., and MOORE, L.R.: 'An exhaustive analysis of multiplicative congruential random number generators with modulus 2^{31}–1', *SIAM J. Sci. Stat. Comput.*, 1986, **7**, pp. 24–45
6 SCHROEDER, M.: 'Number theory in science and communication' (Springer–Verlag, 1984)
7 PAPLINSKI, A.P.: 'Matrix multiplication and division of polynomials', *IEE Proc. D*, 1985, **132**, pp. 95–99
8 STEFANIDIS, P., PAPLINSKI, A.P., and GIBBARD, M.J.: 'Numerical operations with polynomial matrices: application to multi-variable dynamic compensator design' (Springer–Verlag, 1992)
9 PROAKIS, J., RADER, C., LING, F., and NIKIAS, C.: 'Advanced digital signal processing' (Macmillan, 1992)