FAST HARDWARE RANDOM NUMBER GENERATOR FOR THE TAUSWORTHE SEQUENCE

Meir Barel
Technical University of Aachen, W.Germany

Abstract. Many simulation programs require m-dimensional uniformly distributed random numbers. A linear recurrence modulo two generator, based on N-bits and producing L-bit numbers ($L \leq N$), according to Tausworthe theory, may yield a sequence of m-tuples uniformly distributed in $m = (N/L)$ dimensions. When using software computing algorithms on a binary computer, for large N (e.g. N = 159), the generation speed is for many purposes too slow. To overcome this disadvantage we present a new concept of a hardware random number generator, to give the Tausworthe sequence with high generation speed independent of the number of bits per word N. For a 32-bit data word computer we have performed statistical tests on three generators, two of them gave good results.

1. INTRODUCTION

Increasing accessibility to digital computers and the need for simulation of complex systems gave rise to a variety of arithmetic techniques for generating uniformly distributed random numbers. These include the class of one-step linear congruential generators, usually attributed to Lehmer (1).

The most important issue for random number generators concerns the extent to which each generated random number behaves in an INDEPENDENT and PURELY RANDOM fashion. Many simulation programs require m-dimensional uniformly distributed random numbers. The term "m-dimensional uniformity" means that a sequence of m-tuples yields a sequence of points uniformly distributed in an m-dimensional hypercube.

It was shown by Marsaglia (2), for linear congruential generators, that all the random points generated in the unit m-cube will be found to lie in a relatively

small number of parallel hyperplanes. Coveyon and MacPherson (3, P.108) recognized too that any method of generation of random numbers of the one-step linear congruential type, cannot be made arbitrarily good, in the sense of multidimensional uniformity, by any choice of parameters.

In this paper, we present a new concept of hardware random number generator, to produce the Tausworthe sequence with generation time independent of the number of bits per word N. It is suggested here to integrate this hardware generator into computers to give random numbers in high speed and with better random characteristics.

2. LINEAR RECURRENCE MODULO 2 RELATIONS

Given the well established imperfections inherent in the class of one step linear congruential generators, one is naturally curious about a method free of these imperfections. Tausworthe (4) has suggested an approach to random number generation on a computer via the linear recurrence modulo 2 techniques.

Let $\{B_i\}$ be the binary sequence of 0's and 1's generated by the linear recursion

$$B_i \equiv C_1B_{i-1} + C_2B_{i-2} + \ldots + C_NB_{i-N} \pmod 2 \qquad (1)$$

for any given set of integers $C_1, \ldots, C_{N-1}$ each having the value 0 or 1 and $C_N = 1$. From the recursion, B_i is determined solely by the preceding N-tuple $(B_{i-1},\ldots,B_{i-N})$. Each such N-tuple thus has a unique successor N-tuple determined by (1), and the period p of $\{B_i\}$ is clearly the same as the period with which an n-tuple repeats. Since B_i is based on N bits it is clear that the number of unique representations of $(B_{i-1},\ldots,B_{i-N})$ is 2^N-1, the vector $(0,0,\ldots,0)$ being excluded. The necessary and sufficient condition that $p = 2^N-1$ is that the polynomial

$$f(x) = 1 + C_1X + C_2X^2 + \ldots + X^N \qquad (2)$$

be primitive over the Galois field of order 2.

Virtually all polynomials considered for use in (1) are primitive trinomials (5)

$$f(X) = 1 + X^Q + X^N . \tag{3}$$

This leads to a computational convenience since one can now write (1) as

$$B_i \equiv B_{i-Q} + B_{i-N} \pmod 2 \tag{4}$$

which is amenable to the "exclusive or" instruction on binary computers. That is,

$$B_i = B_{i-Q} \oplus B_{i-N} \tag{5}$$

where the "exclusive or" operation $\oplus$ is defined as $0\oplus0 = 0$, $0\oplus1 = 1$, $1\oplus0 = 1$ and $1\oplus1 = 0$.

Figure 1 illustrates a convenient method for generating a binary pseudonoise (PN) sequence as $\{B_i\}$ according to (5) long used to generate binary codes for communication. These are fully described by Golomb (5). This technique involves the use of an N-stage shift register with a feedback term formed by the modulo 2 addition of the Qth and Nth stage.

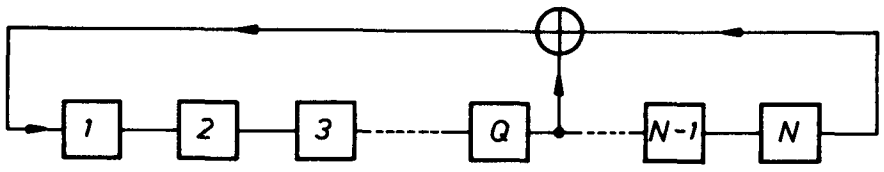

Figure 1. N-bit shift register (PN) sequence.

We now consider the binary integer

$$
\begin{aligned}
Z_0 &= B_0\ B_1\ B_2 \cdots B_{N-1} \\
Z_1 &= B_1\ B_2\ B_3 \cdots B_N \\
Z_2 &= B_2\ B_3\ B_4 \cdots B_{N+1} \\
&\ \ \vdots
\end{aligned}
\tag{6}
$$

generated in a PN generator. If we now properly decimate[1] $\{Z_i\}$ with $q \leq N$, we obtain

$$
\begin{aligned}
Z_0 &= B_0\ B_1\ B_2 \cdots B_{q-1}\ B_q \cdots B_{N-1} \\
Z_q &= B_q\ B_{q+1}\ B_{q+2} \cdots B_{2q-1}\ B_{2q} \cdots B_{q+N-1} \\
Z_{2q} &= B_{2q}\ B_{2q+1}\ B_{2q+2} \cdots B_{3q-1}\ B_{3q} \cdots B_{2q+N-1} \\
&\ \ \vdots
\end{aligned}
\tag{7}
$$

from which it is apparent that, if we read the first q or less bits of Z_i, followed by the first q or less bits of Z_{i+q} and so on, we run over the entire PN sequence.

To form decimal fractions on (0,1) Tausworthe suggests the representation

$$
V_k = 0.B_{kq+r-1}\ B_{kq+r-2} \cdots B_{kq+r-L} \quad \text{(base 2)} \tag{8}
$$

where

L = number of bits in the binary representation of V_k with $L \leq N$

q = integer such that $q \geq L$ and the greatest common divisor (g.c.d.) of q and 2^N-1 is unity $(q, 2^N-1) = 1$.

---

1) In his description of the properties of PN sequences, Golomb /5/ used the term "proper decimation" in which "decimation" means selecting every qth mod(2^N-1) term of the PN sequence, and "proper" means that $(q, 2^N-1) = 1$, so that the decimated sequence is also of full period.

r = any integer in $[0,2^N-1]$, determines the start position of the sequence

The restriction $q \geq L$ guarantees that V_k and V_j for $k \neq j$ share no bits in common. The restriction g.c.d. $(q,2^N-1) = 1$ guarantees a full period 2^N-1 for V_k.

We can also express V_k by

$$V_k = \sum_{j=1}^{L} 2^{-j} B_{kq+r-j} \qquad k = 1,\ldots,2^N-1 \quad . \tag{9}$$

3. ALGORITHM

For primitive trinomials one can write (1) in the form (6)

$$B_i = B_{i+Q-N} \oplus B_{i-N} \quad . \tag{10}$$

One sees that except for the binary point the binary representation of a linear recurrence modulo 2 sequence is now

$$Z_k = Z_{k+Q-N} \oplus Z_{k-N} \qquad k = 1,\ldots, 2^N-1 \tag{11}$$

Then one sees that

$$v_K = Z_{kq}2^{-L} \tag{12}$$

so that

$$Z_{kq} = Z_{kq+Q-N} \oplus Z_{kq-N} \tag{13}$$

provides the binary representation of V_k in (9) for r = 0.

Imposing the restriction q = N leads to

$$Z_{Nk} = Z_{Nk+Q-N} \oplus Z_{N(k-1)} \tag{14}$$

As an example, consider the linear recurrence modulo 2 generator based on N = 5 and Q = 2. The generator period is $2^5-1 = 31$.

Table 1 shows $\{Z_k\}$ for k = 0,...,31.

Table 1. Linear recurrence modulo 2 sequence based on N = 5 and Q = 2

k	Z_k	k	Z_k	k	Z_k
0	10110 = 22	11	10000 = 16	22	11100 = 28
1	11011 = 27	12	01000 = 8	23	11110 = 30
2	11101 = 29	13	00100 = 4	24	11111 = 31
3	01110 = 14	14	10010 = 18	25	01111 = 15
4	10111 = 23	15	01001 = 9	26	00111 = 7
5	01011 = 11	16	10100 = 20	27	00011 = 3
6	10101 = 21	17	11010 = 26	28	10001 = 17
7	01010 = 10	18	01101 = 13	29	11000 = 24
8	00101 = 5	19	00110 = 6	30	01100 = 12
9	00010 = 2	20	10011 = 19	31	10110 = 22
10	00001 = 1	21	11001 = 25		

Imposing the following restrictions

- Q < N/2,
- the decimation q = N

then one can generate Z_{Nk} (14) with the following algorithm (7). Let A and B denote two N-bit storage locations.

Algorithm I:

1. Store a random number (V_k) in locations A and B.
2. Shift the contents of B Q positions to the right, replacing the contents of bits 1,...,Q by zeros.
3. Add the contents of A and B bitwise using the "exclusive or" instruction.
4. Store the result in A and B.

5. Shift the contents of B N-Q positions to the left, replacing the contents of bits Q+1,...,N by zeros.

6. Add the contents of A and B bitwise using the "exclusive or" instruction.

7. Store the result (V_{k+1}) in A.

8. Go to 2. for generating a new random number.

As an example consider the computation of Z_5 in Table 1 from Z_0 the initial random number, one has

Step 1	A = B = 10110	= 22
Step 2	B = 00101	
Step 3	⊕	
Step 4	A ← A = 10110, B ← 10011	
Step 5	B = 11000	
Step 6	⊕	
Step 7	A = 10011, A ← 01011	= 11

4. STATISTICAL PROPERTIES

We may generally assume, merely from the applications to which we wish to suit the numbers, that N is moderately large, so that the sequence $\{V_k\}$ is extremely numerous. For example, if N = 31, there are 2.15×10^9 numbers in $\{V_k\}$.

Tausworthe was able to present a relatively comprehensive theory for the statistical analysis of linear recurrence modulo 2 generators. He showed that for V_k in (9).

1. $E(V_k) \simeq 1/2$
2. $Var(V_k) \simeq 1/12$
3. $Corr(V_k, V_{k+s}) \simeq 0 \qquad 0 < |s| < (2^N-1-L)/q$

Property 3 implies that the period of V_k for zero correlation is $(2^N-1-L)/q$.

4. $\{V_k\}$: $k = 1,\ldots,2^N-1$ is an infinite sequence of uniform deviates.

For $\mathbf{V}_m = (V_k, V_{k-l_2}, \ldots, V_{k-l_m})$ where $0 = l_1 < l_2 < \ldots < l_m$ with $q(l_m+1) \leq N$.

5. $\mathbf{V}_m$ is an infinite sequence of m-tuples of independent deviates.

Property 5 implies that a generator producing L-bit numbers, given the condition $(q,2^N-1) = 1$, $q \geq L$, yields a sequence of nonoverlapping m-tuples uniformly distributed in $m = \lceil N/L \rceil$ dimensions.

However, as Tootill et al (8) (9) note, these results apply for a global characterization of behavior but do not reveal any insight into local behavior. In particular, an acceptable generator should exhibit good "runs up and down" properties (10). The choice of Q strongly influences these properties. They have shown that a generator based upon either X^N+X^Q+1 or $X^N+X^{N-Q}+1$ have good runs up and down properties provided that $Q \leq N/2$, $(Qm,2^N-1) = 1$, $L = Q$ and Q is neither too small nor too close to N/2.

5. STATISTICAL TESTS

For a 32-bit data word computer, we test three primitive trinomial generators to give a 31-bit[1] random numbers.

Three tests are implemented:

1. The chi-square test, testing for one-dimensional uniformity (11).
2. The serial test, testing for m-dimensional uniformity. We tested only for nonoverlapping pairs (2-dimensional uniformity) (12).
3. The runs up and down test, testing for independence (10).

We generate 200,000 consecutive numbers from each generator and compute the three tests. Each test is replicated 500 times, with consecutive sequences. We then compute the empirical cumulative distribution function (c.d.f.) of the 500

1) 31-bit and not 32-bit because the most significant bit in a 32-bit data word computer is a sign bit.

statistics and test it for departures from the theoretical c.d.f. with the Kolmogorov - Smirnov (K-S) test (11). The K-S test is relatively insensitive to deviation in the tail of the empirical c.d.f.. To compensate for this insensitivity we implemented the Anderson - Darling test (13, 14) which gives more weight to these tail effects than to deviations near the median.

Statisfactory performance in an m-dimensional simulation requires not only good statistical properties "ACROSS-DIMENSIONS" concerned with properties of the sequence as generated, but also requires satisfactory statistical properties "ALONG-DIMENSION", i.e. it needs, good performance for the subsequence obtained by taking every mth number generated (8). The results for these tests are summarized in Table 2 (15).

Table 2. Statistical test results obtained from 500 replications each consisting of 200,000 numbers, for L = 31 bits.

Generator	Subsequence of every mth number	Chi-Square-Test		Serial Test		Runs Up and Down Test	
		K-S	A-D	K-S	A-D	K-S	A-D
I. $X^{31}+X^{6}+1$	1	0,28	0,44	9,98	3714	4,93	1257
II. $X^{63}+X^{11}+1$	1	0,32	0,46	0,35	0,75	0,29	2.02
	2	0,34	0,45	-	-	0,38	3,00
III. $X^{159}+X^{31}+1$	1	0,29	0,24	0,32	0,50	0,31	1,19
	2	0,60	0,66	-	-	0,34	1,30
	5	0,44	0,75	-	-	0,70	5,00

The K-S and A-D statistics show the poor properties of the first (I) generator for the runs up and down test and as predicated for the serial test. The K-S and A-D statistics for the second (II) and third (III) generators, are all within the acceptability limits, for all three tests, according to Table 2 of (11)

and references (13, 14), and can be recommended. The third generator (III) should have been tested for 5-dimensional uniformity and should give good results according to Tausworthe theory. The implementation of the serial test for 5-dimensional uniformity was impossible because of the enormous amount of computer time and memory space required. A theoretical test for the m-dimensional uniformity for the Tausworthe generator, similar to the spectral test (3) for the linear congruential generators is still to be found.

On the assumption that investigators will prefer generators that fit the word size of their machines, compilations of primitive trinomials modulo 2 for $2 \leq N \leq 1000$ appear in Zierler and Brillhart (16), (17).

6. CONFIGURATION

A linear recurrence modulo 2 generator requires N to be fairly large, for otherwise the number of dimensions $m = \lfloor N/L \rfloor$ over which the sequence is uniformly distributed will be too small. This requirement makes software computing algorithms for the Tausworthe generator ineffective.

Hardware for direct random number generation will reduce generation time dramatically. A particularly convenient method for generating the Tausworthe sequence involves the use of N-stages shift register with a feedback term formed by the modulo 2 additions of several output stages (5). However, for N-bit random words, the word rate of the generator is reduced by a factor of N.

To overcome this disadvantage we design a hardware random number generator, based on algorithm I, to give the Tausworthe sequence with generation time independent of the number of bits per word.

Figure 2 describes the architecture of the hardware generator based on $N = 5$ and $Q = 2$. However N and Q may take any value as desired. It is extremely simple, but also surprisingly powerful. The circuit consists of two registers and two exclusive or gates operating in two levels. Let us assume that a certain

random number V_k is being stored in Register I, with the clock pulse (cp). The first level exclusive or (Exclusive Or Gate I) adds the contents of Register I to the "Q-bit right shifted word"(Step 2 and 3, Algorithm I)[1]. The second level exclusive or (Exclusive Or Gate II) adds the "(N-Q)-bits left shifted word" from Exclusive Or Gate I with the "unshifted word" from Exclusive Or Gate I (Step 5 and 6, Algorithm I). Register II[2] (and the Output) contains now the next random number V_{k+1}. With the next clock pulse the same cycle is being repeated.

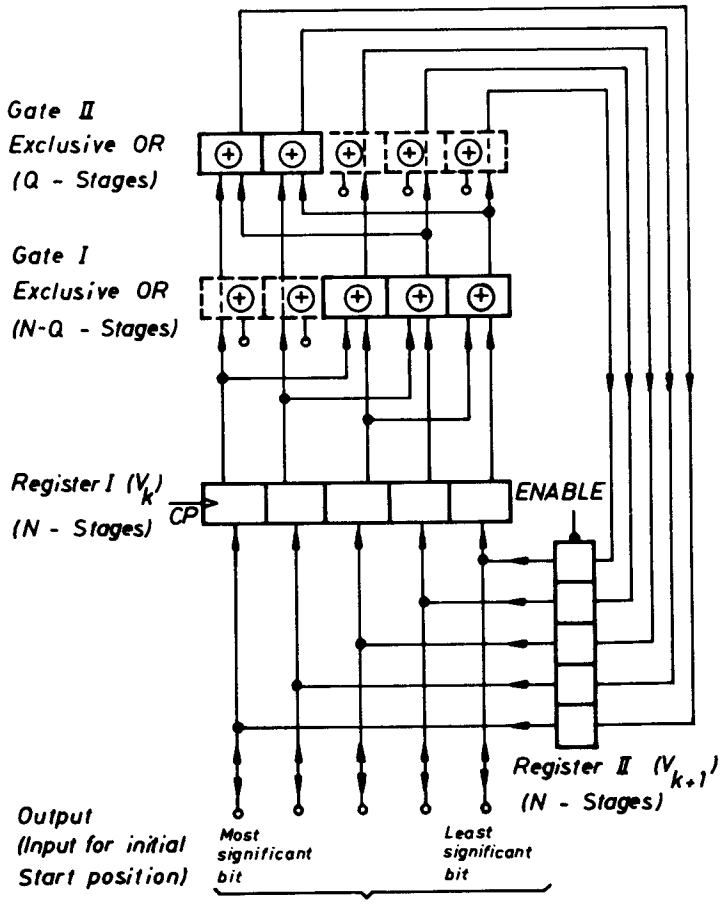

Figure 2. Linear recurrence modulo 2 hardware generator (Tausworthe type), based on N=5 Q=2. However N and Q may take any value as desired.

1) We add in the first level exclusive or just N-Q bits, because the rest Q bit (from the right shifted word) are 0 and the exclusive or operation will not change the bit pattern. The same holds for the second level exclusive or.

2) Register II is normally (output mode) transparent. During an input operation it is being switched to Tri-State position.

The sequence is available with maximum clock frequency, according to the applied technology. For TTL-technology, a speed of more than f = 20 MHz (t=50 ns!) can be reachied, see also Table 3.

Table 3. Time comparison for different random number generators.

Generation Method	Implemented in	Relative Generation Time		
		31 bit	63 bit	159 bit
1. Linear Congruential[a)]	Software	1[c)]	3	-
2. Linear Recurrance Mod.2[b)]	Software	1	3	8
3. N-Stages shift register[b)]	Hardware	$31 \cdot 10^{-3}$	$63 \cdot 10^{-3}$	$159 \cdot 10^{-3}$
4. Fast Tausworthe Sequence Generator[b)]	Hardware	10^{-3}	10^{-3}	10^{-3}

a) On 32-bits data word computer SIEMENS 4004/151
b) Tausworthe Type
c) Typical time 10...50 µs

The hardware random number generator should be connected to the Data-Bus of a binary computer or microprocessor via a Multiplexed I/O-port as Figure 3 shows. Through this I/O-port, Register I can be loaded with the initial random number, thereby controlling the start position of the sequence if desired. Using the first L-bit from an N-bit word, one can address the Multiplexed I/O-port with the Port Address Decoder to read the first L-bit random number. On the other hand, if one uses m-random numbers from the same N-bit word, the Port Address Decoder and the Multiplexed I/O-port should put out the desired random number on the Data-Bus.

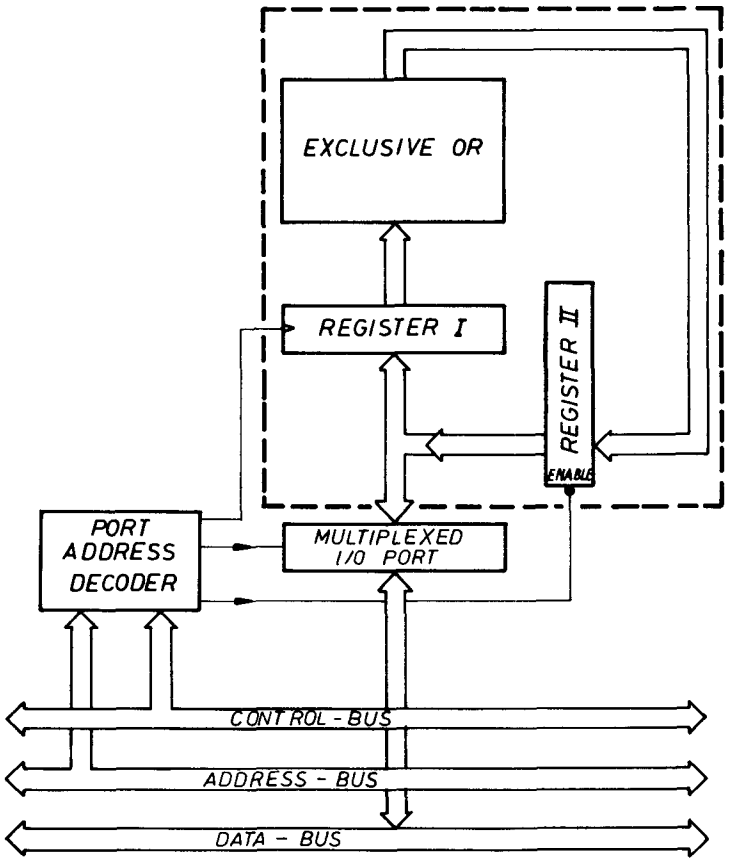

Figure 3. Input/Output configuration for the linear recurrence modulo 2 hardware generator.

REFERENCES

1. Lehmer, D.H., "Mathematical Methods in Large-Scale Computing Units", Ann. Comp. Lab., Harvard University, Vol. 26, 1951, pp. 141-146.

2. Marsaglia, G., "Random numbers fall mainly in the planes", Proc. Nat. Acad. Sci., 60,5 (Sept. 1968), S. 25-28.

3. Coveyon, R.R. and R.D. MacPherson, "Fourier Analysis of Uniform Random Number Generators", J. ACM, Vol. 14, 1967, pp. 100-119.

4. Tausworthe, R.C., "Random Numbers Generated by linear Recurrence Modulo Two", Math. Comp., Vol. 19, 1965, pp. 201-209.

5. Golomb, S.W., "Shift Register Sequences", Holden-Day, San Francisco 1967, pp. 24-89.

6. Fishman, G.S., "Notes On Linear Recurrence Generators Modulo 2" Lecture Notes on the University of North Carolina at Chapel Hill,1979.

7. Whittlesley, J.R.B., "A Comparison of the Correlation Behavior of Random Number Generators for the IBM 360", Comm. ACM, Vol. 11, 1968, pp. 641-644.

8. Tootill, J.P.R., W.D. Robinson and A.G. Adams, "The Runs Up and Down Performance of Tausworthe Pseudo-Random Number Generators", J. ACM, Vol. 18, 1971, pp. 381-399.

9. Tootill, J.P.R., W.D. Robinson and D.J. Eagle, "An Asymptotically Random Tausworthe Sequence", J. ACM, Vol. 20, 1975, pp. 469-481.

10. Levene, H. and Wolfowitz, "The Covariance Matrix of Runs Up and Down", Ann. Math. Stat., Vol. 15, 1944, pp. 58-56.

11. Knuth, D.E., "The Art of Computer Programming: Seminumerical Algorithms", Vol. 2, Addison-Wesley 1969.

12. Fishman, G.S., "Principles of Discrete Event Simulation", John Wiley & Sons 1978.

13. Anderson, T.W. and D.A. Darling, "Asymptotic Theory of Certain 'Goodness of Fit' Criteria Based on Stochastic Processes", Ann. Math. Stat., Vol. 23, 1952, pp. 193-212.

14. Anderson, T.W. and D.A. Darling, "A Test of Goodness of Fit", J. Am. Stat. Assoc., Vol. 49, Dec. 1954, pp. 765-769.

15. Jobes, B., "Untersuchung An Gleichverteilten Pseudozufallsgeneratoren", Master Thesis, Technical University Aachen.

16. Zierler, N. and Brillhart, J., "On Primitive Trinomials (Mod 2), I". Inform. and Contr. 13, 6 (1968), pp. 541-554.

17. Zierler, N. and Brillhart, J., "On Primitive Trinomials (Mod 2), II", Inform. and Contr. 14, 6 (1969), pp. 566-569.

ACKNOWLEDGEMENTS

I wish to thank Professor George Fishman/North Carolina, for directing my attention on the Tausworthe method for generating random deviates; Professor Friedrich Schreiber/Aachen, for helpful discussions; Dipl.-Ing. B. Jobes for the statistical tests he implemented on the random number generators mentioned in this paper. Acknowledgement is also made to the Deutsche Forschungsgemeinschaft, Bonn who has supported this work.