A METHOD OF GENERATING UNIFORMLY DISTRIBUTED SEQUENCES OVER *[0,K]*, WHERE *K+1* IS NOT A POWER OF TWO

Richard Kuehnel[1] and Yuke Wang[2]

1. US Dept. of Defense
9800 Savage Road, Suite 6512
Fort George G. Meade, MD 20755, USA
rjkuehn@ieee.org

2. Dept. of Computer Science
University of Texas at Dallas
Richardson, TX 75083, USA
yuke@utdallas.edu

ABSTRACT

A new methodology has been recently proposed for the efficient generation of multiple pseudo-random bit sequences that are statistically uncorrelated [1]. Random sequences that are uniformly distributed over a range $[0,K]$, where $K+1$ is a power of 2, can be constructed by forming a vector of M independent bit sequences, where $M=\log_2(K+1)$. In this paper we demonstrate that this method of construction represents a special case of a more generalized approach in which K can be any positive integer. The procedures described here can be used to efficiently generate multiple independent random sequences that are uniformly distributed over any range.

1. INTRODUCTION

Stochastic neural networks and other statistical computing systems use thousands of random number sequences. They are also massively parallel by their very nature, and thus benefit from using multiple parallel circuits, each with its own source of random sequences. This usually requires a separate pseudo-random number generator (PRNG) for each circuit. A PRNG often contains more logic gates than the circuit it supplies, so a large amount of silicon area is consumed by random number production. Moreover, to ensure that they are statistically uncorrelated, each PRNG must be designed using a different algorithm or using a different starting value. This adds complexity to the design and increases the size of hardware implementations [2].

Saarinen, et. al., analyzed several methods of generating independent sequences uniformly distributed over ranges that are a power of two, but noted that an optimum method of dealing with the complexity of the problem had not yet been developed [5]. A new methodology for the generation of multiple random bit sequences using only two pseudo-random bit generators has recently been proposed in [1]. The design reduces the routing requirements to only two signals that are passed from circuit to circuit in series. It enables new circuits to be added to the system without additional calculations – there is no need to keep track of random starting values, tap combinations, or time shifts. Using this technique, a random number sequence that is uniformly distributed over the range $[0,K]$, where $K+1$ is a power of 2, can be constructed by forming a vector of M independent bit sequences, where $M=\log_2(K+1)$. It was noted in [4], however, that in the context of random selection, sequence generation over ranges that are not a power of 2 is an area that deserves further study. In this paper we address this problem by demonstrating that the method of constructing sequences over a power of 2 represents a special case of a more generalized approach in which the range can be any positive integer.

In Section 2 we describe a method of generating a sequence distributed over a range that is not necessarily a power of 2. In Section 3 we determine how the range of the sequence impacts the number of logic gates required to construct it. Section 4 summarizes the results.

2. RANDOM SEQUENCE GENERATION

Our design method is based on the following theorem, which is proven in Appendix 1.

Generating Theorem: For every integer $K>1$ there exists a set of prime numbers $q_1,...,q_M$, unique except for order, where $q_1q_2...q_M=K+1$, such that if r_i is uniformly distributed over the set of integers $\{0,...,q_i-1\}$ for all i, $1\le i\le M$, then the sum

$$R=r_1+r_2q_1+r_3q_1q_2+...+r_Mq_1q_2...q_{M-1}$$

is uniformly distributed over the set of integers $\{0,...K\}$.

Using this theorem we can create a random sequence $\{R(n)\}$ that is uniformly distributed over $[0,K]$, where K is any integer greater than one, by factoring $K+1$ into its unique primes $K+1=q_1...q_M$. We then generate M independent, random sequences $\{r_i(n)\}$ uniformly distributed over $0\le r_i(n)<q_i$, $1\le i\le M$, and concatenate them by the relation

$$R(n)=r_1(n)+r_2(n)q_1+r_3(n)q_1q_2+...+r_M(n)q_1...q_{M-1}$$

A linear feedback shift register (LFSR) can be used to generate the pseudo random sequences $\{r_i(n)\}$ as shown in figure 1. Each of the arrows denotes a sufficient number of bits to represent the

sequence's maximum value $q_i - 1$. Both the output and the input to the shift register are

$$r_i(n) = ar_i(n-L) \oplus br_i(n-\lambda)$$

where $\oplus$ denotes modulo-q_i addition, L is the length of the shift register, and λ is a fixed tap in the shift register such that $1 \le \lambda < L$. The coefficients a and b are determined by a primitive, irreducible, characteristic polynomial. More than two taps can also be used. A list of appropriate polynomials is widely available and found, for example, in [3] and on the internet. For $q_i = 2$ the tap coefficients are $a = b = 1$. Otherwise $1 \le a, b < q_i$ according to the selected polynomial. Moreover, a and b are not necessarily the same as the polynomial coefficients. Appendix 2 describes a procedure for computing the tap coefficients from the characteristic polynomial.

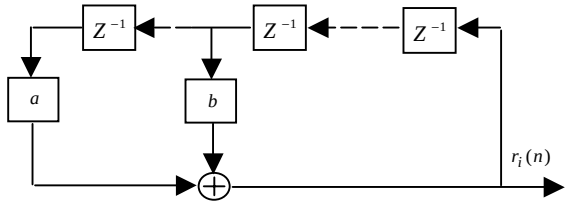

Figure 1.

It was recently shown in [1] that a separate LFSR for each random sequence is unnecessary. Instead only two LFSR circuits for each unique value of q_i are needed for the entire system, each having a different random sequence length. They generate the sequences $\{s_i(n)\}$ and $\{t_i(n)\}$ shown in figure 2.

As can be seen, $\{s_i(n)\}$ and $\{t_i(n)\}$ are not used directly. Instead a new random sequence is generated by their modulo-q_i addition, where $\{t_i(n)\}$ is delayed by 1 clock cycle from where it was last used. The k-th sequence $\{r_{i,k}(n)\}$, uniformly distributed over $[0, q_i - 1]$, is generated by

$$r_{i,k}(n) = s_i(n) \oplus t_i(n-k)$$

where $\oplus$, as before, signifies modulo-q_i addition. For simplicity of notation all further references to these random sequences will drop the second subscript. The sequence $\{r_i(n)\}$ will simply refer to an independent sequence uniformly distributed over $[0, q_i - 1]$ that is used only once.

The desired sequence $\{R(n)\}$ is generated by

$$R(n) = r_1(n) + r_2(n)q_1 + r_3(n)q_1q_2 + ... + r_M(n)q_1...q_{M-1} \quad (2)$$

One can observe that multiplication by q_1 occurs in $M-1$ terms, whereas multiplication by q_M never occurs. Since a multiplication by 2 can be efficiently implemented in VLSI, we choose to order the prime factors such that $q_1 \le q_2 \le ... \le q_M$. If $K+1$ is a power of two, then $q_1 = ... = q_M = 2$ and equation (2) reduces to

$$R(n) = \sum_{i=1}^{M} r_i(n)(2)^{i-1} \quad (3)$$

where each $\{r_i(n)\}$ is an independent random bit sequence. The summation is achieved without logic by simply ordering the random bits from the least significant to the most significant. The summation can also be performed without additional logic if $K+1 = (2^{M-1})q_M$, where $q_M > 2$. In this case equation (3) remains unchanged. If, on the other hand, $K+1 = (2^{M-2})q_{M-1}q_M$, where $2 < q_{M-1} \le q_M$, then

$$R(n) = \sum_{i=1}^{M-1} r_i(n)(2)^{i-1} + r_M(n)(2)^{M-2} q_{M-1}$$

The multiplication by q_{M-1}, which is fortunately a constant value, requires additional logic. Obviously if many of the prime number factors are not equal to 2 then significant additional circuitry may be required.

3. CIRCUIT SIZE

The number of logic gates needed to create a random sequence varies depending on the range of the sequence. The least number of logic gates are required when as many of the prime factors as possible are equal to 2. To generate a sequence over $[0,255]$, for example, requires only eight XOR gates and eight flipflops. More generally, to create an independent pseudo-random sequence that is uniformly distributed over $[0, K]$, where $K+1 = q_1...q_M$, requires a modulo-q_i adder for each i, $1 \le i \le M$. The total number of flipflops needed is

$$\sum_{i=1}^{M} \lceil \log_2(q_i) \rceil$$

Additional logic elements for multiplication are required if $q_i > 2$, for any $i < M$. For example, if $q_{M-1} = 3$ and $q_M = 5$ then we need to compute the product $3r_M(n)$ where $r_M(n) = \{0,1,...,4\}$. Since the coefficient 3 is a fixed value, this multiplication would require only a lookup table with a 3-bit input to represent $r_M(n)$ and a 4-bit output to represent the product.

In addition to the circuit elements required for each independent random sequence, a pair of linear feedback shift registers is needed for each prime factor q_i that is unique to the overall system. This requires two modulo-q_i adders and $(L_{1,i} + L_{2,i})\lceil \log_2(q_i) \rceil$ flipflops, where $L_{1,i}$ and $L_{2,i}$ are the lengths of the shift registers. If $q_i > 2$ then, depending on the characteristic polynomial that is selected, a modulo-q_i multiplier

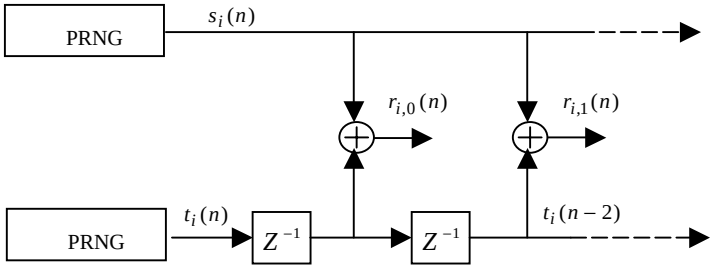

Figure 2.

may be needed for each of the tap coefficients a and b shown in figure 1. For large systems the number of logic gates used for these shift registers becomes insignificant compared to the total used to generate each independent random sequence $\{r_i(n)\}$.

An n-bit modulo m addition for $n = \lceil \log_2 m \rceil$ can be viewed as a modulo m operation performed after the addition is done. Thus the modulo m addition follows:

$$y = |x_1 + x_2|_m = \begin{cases} x_1 + x_2, & \text{if } x_1 + x_2 < m \\ x_1 + x_2 - m, & \text{if } x_1 + x_2 \geq m \end{cases}$$

Generally, two methods can be used to complete the above computation: 1) Compute the results of both $x_1 + x_2$ and $x_1 + x_2 - m$, then select the correct result of modulo m addition from them. 2) Use a correction table to correct the addition $x_1 + x_2$ to the result $|x_1 + x_2|_m$.

In the first method, two n-bit adders are used; the first adder computes $x_1 + x_2 - m$, while the second adder computes $x_1 + x_2$. The carry bit generated from the second adder indicates whether or not $x_1 + x_2$ is greater than m (figure 3). A multiplexer, controlled by the carry, selects the correct output.

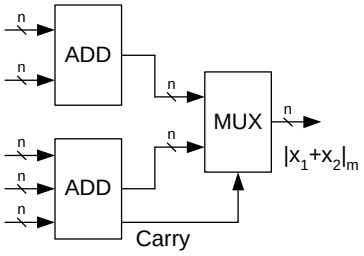

Figure 3.

In the second method, a lookup table is used to replace the first n-bit adder and the multiplexer in the first method (figure 4). When the lookup table in the ROM is small, i.e. when the modulus m is small, the second method can have better performance than the first method for fast table lookup.

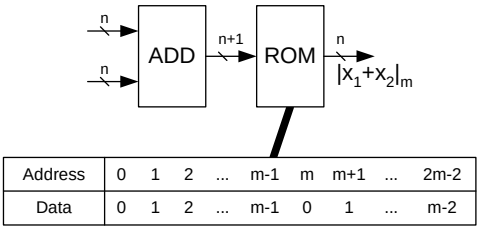

Figure 4.

However, for an n-bit modulo addition the second method requires a 2^n-entry ROM with n bits for each entry. The hardware consumption for the lookup table is much greater than the first method when the size of the modulus is large, which makes it unaffordable in practical hardware designs.

4. CONCLUSIONS

A random number sequence that is uniformly distributed over the range $[0, K]$, where $K+1$ is a power of 2, can be constructed by forming a vector of M independent bit sequences, where $M = \log_2(K+1)$. This method of construction represents a special case of a more generalized approach in which K can be any positive integer. Moreover, the specific value of K has a direct impact on the size of the circuit needed to generate the sequence. The smallest circuits can be implemented when at most only one of the prime number factors of $K+1$ is greater than 2.

APPENDIX 1: PROOF OF THE GENERATING THEOREM

Proof: According to the *fundamental theorem of arithmetic*, any integer value $K+1$ is equal to a product of primes $q_1 q_2 ... q_M$ that are unique except for order. Let x be an integer, where $0 \leq x \leq K$. (In this section a mod b denotes the remainder resulting from integer division, i.e. 7 mod 2 = 1.) Then

$$\begin{aligned} x &= x \bmod (K+1) \\ &= x \bmod (q_1 ... q_M) \\ &= x \bmod (q_1 ... q_M) \bmod (q_1 ... q_{M-1}) + a_M q_1 ... q_{M-1} \\ &\text{where } 0 \leq a_M < q_M \end{aligned}$$

$$= x \bmod (q_1...q_M) \bmod (q_1...q_{M-1}) \bmod (q_1...q_{M-2})$$
$$+ a_{M-1} q_1...q_{M-2} + a_M q_1...q_{M-1}$$
$$where \quad 0 \le a_i < q_i, \quad M-1 \le i \le M$$

$$= x \bmod (q_1...q_M) \bmod (q_1...q_{M-1})...\bmod (q_1)$$
$$+ a_2 q_1 + a_2 q_1 q_2 + ... + a_M q_1...q_{M-1}$$
$$where \quad 0 \le a_i < q_i, \quad 2 \le i \le M$$

If we define

$$a_1 \equiv x \bmod (q_1...q_M) \bmod (q_1...q_{M-1})...\bmod (q_1)$$

then by inspection $0 \le a_1 < q_1$ and thus

$$x = a_1 + a_2 q_1 + a_2 q_1 q_2 + ... + a_M q_1...q_{M-1} \quad (1)$$
$$where \quad 0 \le a_i < q_i, \quad 1 \le i \le M$$

Let X be the set of all integers in the range $[0, K]$ and let A be the set of all M-tuples $(a_1,..., a_M)$ such that $0 \le a_i < q_i$ for all i, $1 \le i \le M$. Equation (1) proves that any integer $x \in X$ is the image of an element $(a_1,..., a_M) \in A$ under a function f defined as

$$f(a_1,..., a_M) = a_1 + a_2 q_1 + a_3 q_1 q_2 + ... + a_M q_1...q_{M-1}$$

where $K+1 = q_1 q_2...q_M$ and where $q_1, q_2,..., q_M$ are prime numbers, some or all of which can be the same. Both X and A contain $K+1$ elements. For each $x \in X$ there exists at least one $(a_1,..., a_M) \in A$ such that $f(a_1,..., a_M) = x$. Thus f is a function that maps A onto X. Let $(b_1,..., b_M) \in A$ and let $f(b_1,..., b_M) = f(a_1,..., a_M)$. Since

$$f(b_1,..., b_M) \bmod q_1 = f(a_1,..., a_M) \bmod q_1$$

then $b_1 = a_1$. By repeating this procedure with $\bmod(q_1 q_2)$, $\bmod(q_1 q_2 q_3)$, etc., it can be shown that $(b_1,..., b_M) = (a_1,..., a_M)$. Thus f is one-to-one. Since f is both one-to-one and onto then there exists an inverse function $f^{-1}: X \to A$.

Now let $R \equiv f(r_1,..., r_M)$ where each r_i is an independent random variable uniformly distributed over the set of integers $\{0,..., q_i - 1\}$ for all i, $1 \le i \le M$. Then $(r_1,..., r_M) \in A$ and thus $R \in X$. For each $x \in X$, let $(a_1,..., a_M) = f^{-1}(x)$. Then

$$P\{R = x\} = P\{f^{-1}(R) = f^{-1}(x)\}$$
$$= P\{(r_1,..., r_M) = (a_1,..., a_M)\}$$
$$= P\{r_1 = a_1,..., r_M = a_M\}$$
$$= P\{r_1 = a_1\} P\{r_2 = a_2\}...P\{r_1 = a_M\}$$
$$= \left(\frac{1}{q_1}\right)\left(\frac{1}{q_2}\right)...\left(\frac{1}{q_M}\right)$$
$$= \frac{1}{K+1}$$

Therefore R is uniformly distributed over $[0, K]$.

APPENDIX 2: COEFFICIENT TRANSLATION

For a primitive polynomial with characteristic $q_i = 2$ the shift register coefficients can be determined directly. The polynomial $f(x) = x^7 + x + 1$, for example, translates into a shift register configuration of $r_i(n) = r_i(n-7) \oplus r_i(n-1)$. The ease with which this is accomplished is attributable to the fact that in modulo-2 arithmetic $-1 = 1$. In general, however, let $f(x) = ax^c + bx^d + e$ be a primitive polynomial with characteristic q_i. Then the shift register configuration is $-e r_i(n) = a r_i(n-c) \oplus b r_i(n-d)$. (In this section juxtaposition is interpreted as modulo-q_i multiplication.) When we eliminate the coefficient on the left hand side using modulo-q_i arithmetic then the coefficients on the right hand side will probably change unless $q_i = 2$. For example, let the polynomial be $f(x) = x^4 + x^3 + 5$ with characteristic $q_i = 17$. Then

$$r_i(n-4) \oplus r_i(n-3) \oplus 5 r_i(n) = 0$$
$$10 r_i(n-4) \oplus 10 r_i(n-3) \oplus 16 r_i(n) = 0$$
$$10 r_i(n-4) \oplus 10 r_i(n-3) \oplus (-1) r_i(n) = 0$$
$$r_i(n) = 10 r_i(n-4) \oplus 10 r_i(n-3)$$

REFERENCES

[1] Richard Kuehnel, "An Improved Design Methodology for Generating Multiple Random Bit Sequences," (to be published at ISPC/GSPx, March 2003).

[2] Dennis R. Morgan, "Autocorrelation Function of Sequential M-Bit Words Taken from an N-Bit Shift Register," IEEE Transactions on Computers, Vol. C-29, No. 5, May 1980.

[3] V. N. Yarmolik and S. N. Demidenko, Generation and Application of Pseudorandom Sequences for Random Testing, (New York: John Wiley and Sons, 1988).

[4] Bradley D. Brown and Howard C. Card, "Stochastic Neural Computation I: Computational Elements," IEEE Transactions on Computers, Vol. 50, No. 9, Sep. 2001.

[5] J. Saarinen, J. Tomberg, L. Vehmanen, K. Kaski, "VLSI Implementation of Tausworthe Random Number Generator for Parallel Processing Environment," IEE Proceedings-E, Vol. 138, No. 3, May 1991.