Mieczysław JESSA
POLITECHNIKA POZNAŃSKA, WYDZIAŁ ELEKTRONIKI I TELEKOMUNIKACJI,
ul. Polanka 3, 60-965 Poznań

Przetwarzanie liczb losowych o rozkładzie nierównomiernym na liczby losowe o rozkładzie równomiernym

Dr hab. inż. Mieczysław JESSA

Adiunkt na Wydziału Elektroniki i Telekomunikacji Politechniki Poznańskiej, Katedra Systemów Telekomunikacyjnych i Optoelektroniki. Autor lub współautor ponad 120 publikacji, 15 patentów oraz kilkunastu rozwiązań konstrukcyjnych wdrożonych w krajowej sieci telekomunikacyjnej. Kierował ponad trzydziestoma projektami wykonanymi na rzecz podmiotów gospodarczych. Najważniejsze prace dotyczą synchronizacji sieci telekomunikacyjnej, zastosowań zjawiska chaosu oraz losowości i pseudolosowości.

e-mail: mjessa@et.put.poznan.pl

Streszczenie

Podstawową wadą liczb losowych otrzymywanych sprzętowo jest ich nierównomierny rozkład. W rezultacie w ciągu wyjściowym liczba otrzymanych zer może się znacząco różnić od liczby jedynek. Sposobem na eliminację tej wady jest tzw. postprocessing. W artykule zaproponowano nową metodę postprocessingu, łatwą do zaimplementowania w każdym układzie cyfrowym. Stosując przekształcenia analityczne wykazano, że na wyjściu otrzymujemy liczby o rozkładzie równomiernym, niezależnie od postaci rozkładu liczb na wejściu. Metodę zilustrowano przykładem.

Słowa kluczowe: generatory losowe, rozkład liczb losowych, kryptografia.

Converting random numbers with non-uniform distribution into uniformly distributed random numbers

Abstract

Uniformly distributed random numbers play a key role in many fields of science. The basic disadvantage of random number generators is that the properties of a physical implementation differ from the theoretical expectations. Most sources of noise have a non-uniform distribution function, which eliminates them as a direct source of uniformly distributed random numbers. If the distribution is symmetric, we can use a threshold function, but this reduces the output bit rate and the output sequences are biased when the design is implemented in a real circuit. In this paper, there is proposed a novel method for producing uniformly distributed random numbers from non-uniformly distributed random numbers. The method uses an algorithm for improving the statistical quality of multiplicative congruential generators described in the literature. There is analytically shown that the bitwise exclusive-or sum of independent random numbers with non-uniform distribution provides, in the limit, numbers with uniform distribution. The proposed method also eliminates bias for constructions that use a threshold function and for sources with theoretically uniform distribution but implemented in real physical systems. Consequently, the set of systems that can be considered for use as sources of uniformly distributed random numbers is increased significantly to include practically all known sources of randomness. The method can be easily implemented in contemporary digital circuits.

Keywords: random number generator, distribution of numbers, cryptography.

1. Wprowadzenie

Liczby losowe są stosowane w wielu dziedzinach nauki. Używane są liczby o rozkładzie równomiernym i nierównomiernym, przy czym rozkłady inne od równomiernego są najczęściej otrzymywane metodami deterministycznymi z rozkładu równomiernego. Niestety fizyczne generatory liczb losowych nie dostarczają liczb o rozkładzie równomiernym lecz innym, często mocno odbiegającym od oczekiwań teoretycznych. Rozkład ten prawie nigdy nie jest znany a priori, a na jego postać mają wpływ takie czynniki jak zasilanie, temperatura, wilgotność, potencjalne ataki i wiele innych. Stąd też zawsze istnieje potrzeba dodatkowego przetworzenia liczb wytwarzanych przez generator fizyczny. Proces ten nazywany jest postprocessingiem, a jego celem, poza otrzymaniem rozkładu równomiernego, może być także eliminowanie korelacji pomiędzy sąsiednimi bitami, które mogą się pojawić w ciągu wytwarzanym przez generator fizyczny.

W literaturze znane są różne metody postprocessingu. Są stosowane do ciągów binarnych otrzymywanych bezpośrednio, na przykład za pomocą funkcji progowej lub do liczb losowych kodowanych za pomocą l bitów, gdzie $l>1$. Jeżeli wytwarzamy liczby losowe, to liczby te zamieniane są najpierw na l-biowe bloki, które następnie są łączone w jednolity ciąg bitów. Najprostsze metody postprocessingu wykorzystują korektor von Neumanna czy rejestry przesuwne. Bardziej skomplikowane używają funkcji skrótu MD5, SHA-1, SHA-2 lub algorytmów szyfrujących takich jak DES czy AES. Poddanie ciągu wyjściowego postprocessingowi sprawia, że prawdopodobieństwa otrzymania na wyjściu zera i jedynki są równe, nawet jeżeli na wejściu istotnie się od siebie różniły. Ciąg binarny, w którym prawdopodobieństwo wystąpienia zera jest równe prawdopodobieństwu wystąpienia jedynki nazywamy ciągiem nieobciążonym (ang. *unbiased*). Jeżeli prawdopodobieństwa wystąpienia zera i jedynki są różne, to taki ciąg nazywamy obciążonym (ang. *biased*). Podany termin przenosi się także na generator wytwarzający ciągi. A więc, jeżeli generator losowy dostarcza zer i jedynek z równym prawdopodobieństwem, to jest on nazywany generatorem nieobciążonym. W przeciwnym razie, jest to generator obciążony.

W pracach [1] i [2] metodami eksperymentalnymi pokazano, że użycie dodatkowej tablicy oraz operacji modulo 2 wykonywanej na l-bitowych słowach cyfrowych kodujących liczby pseudolosowe może znacząco poprawić ich właściwości statystyczne. Założono jednak, że statystycznie niedoskonałe ciągi liczb mają na wejściu rozkład równomierny. Celem tej pracy jest wykazanie metodami analitycznym, że ten sam algorytm może być użyty do zamiany liczb losowych o dowolnym rozkładzie na liczby o rozkładzie równomiernym oraz, że taka metoda postprocessingu jest obliczeniowo bardziej wydajna od innych metod.

2. Metoda

W proponowanej metodzie wykorzystano algorytm CMCG opisany w artykule [2]. Algorytm w znaczący sposób poprawia właściwości statystyczne ciągów pseudolosowych wytwarzanych przez multiplikatywny generator kongruencyjny dla wszystkich wartości parametru, dla których generator wytwarza ciągi o maksymalnej długości. Pseudokod algorytmu IMP proponowanego w tej pracy bazuje na CMCG i ma następującą postać:

Inicjalizacja:
Wybierz rozmiar L pomocniczej tablicy T;
Wypełnij T kolejnymi liczbami losowymi p_n, $n = 0,1,...,L-1$
Obliczenia:
for $n := 0$ to *N-1* do

$$\begin{cases} \text{Generuj liczbę } p_{n+L} \\ s_{n+L} := 1 + \text{trunc}_i(p_{n+L}) \\ j := n \bmod L, \\ T[j] := p_{n+L} \\ u_n := T[j] \oplus T[(j + s_{n+L}) \bmod L] \oplus ... \oplus T[(j + R \cdot s_{n+L}) \bmod L] \end{cases}$$ end;

gdzie $0 < i < l$, $L \geq 2^i R$.

W algorytmie IMP kolejne liczby p_n są zapisywane cyklicznie do tablicy T o L komórkach, adresowanych od 0 do $L-1$. Rozmiar tablicy spełnia warunek $L \geq 2^i R$, gdzie $0 < i < l$. W każdym kroku z T wybieramy R+1 liczb $t_0, t_1, ..., t_R$, gdzie t_0 jest liczbą

aktualnie wytworzoną przez generator losowy, a $t_1,...,t_R$ są wyczytane z *T*. Adresy liczb $t_1,...,t_R$ zależą od liczby utworzonej z *i* najstarszych bitów liczby losowej p_{n+L} powiększonej o 1. Dodanie jedności ma zapobiec sytuacji, w której liczba u_n będzie otrzymana w wyniku sumowania modulo 2 tych samych *l*-bitowych słów cyfrowych, gdy wszystkie *i* bitów liczby p_{n+L} jest równych zeru. W każdym kroku iteracji odległość pomiędzy komórkami tablicy *T*, z których wyczytujemy liczby $t_1,...,t_R$ ulega zmianie, zależnie od wartości s_{n+L}, które także są liczbami losowymi. Zwróćmy uwagę, że odległość s_{n+L} pomiędzy adresami czytanych liczb nie może być stała, gdyż dla $s_{n+L} = s$ po s krokach będą sumowane modulo 2 te same liczby co poprzednio, za wyjątkiem liczb t_0 i t_R.

Losowa zmiana odległości pomiędzy liczbami losowymi wybieranymi do sumowania modulo 2 oznacza, że u_n jest utworzone w wyniku losowego wyboru *R*+1 liczb losowych spośród dostępnych w każdym kroku $2^i R$ liczb. Wytworzenie kolejnej liczby powoduje „wypadniecie" z tego zbioru jednej liczby i „wejście" do niego liczby nowo wytworzonej przez generator losowy. W takim zbiorze jest realizowane ponowne losowanie z nowym s_{n+L+1}. U podstaw opisanego mechanizmu leży spostrzeżenie, że dowolny podciąg utworzony z ciągu liczb losowych jest także ciągiem losowym. Warunkiem koniecznym jest oczywiści to, aby liczby losowe tworzące ciąg, z którego wybieramy liczby, były wytworzone w sposób niezależny (zob. [3], s. 10]).

Zastanówmy się obecnie jaki rozkład będą miały liczby u_n gdy liczby p_n są rozłożone nierównomiernie w przedziale $[0, 2^l)$. Niech

$$p_n(k) = \sum_{i=0}^{l-1} b_{n,i}(k) \cdot 2^i \ , \ n = 0,1,..., \ k = 1,2,...,K \ , \ K = R+1 \quad (1)$$

są wartościami liczb $t_0, t_1,...,t_R$ wybranych w *n*-tym kroku algorytmu IMP. Symbol $b_{n,i}(k)$ oznacza *i*-ty bit kodujący *k*-tą liczbę $p_n(k)$. Dla każdego *k* ciąg liczb $\{p_n(k)\}$, *n*=0,1,… można zamodelować jako ciąg zmiennych losowych $\{Y_n(k)\}$. Zmienne $Y_n(k)$ przyjmują wartości z przedziału $[0, 2^l)$. Każde $Y_n(k)$ jest reprezentowane wzajemnie jednoznacznie przez binarne zmienne losowe $i = 0,1,...l-1$, tj.,

$$Y_n(k) = \{X_{n,l-1}(k)X_{n,l-2}(k)...X_{n,0}(k)\} \ . \quad (2)$$

Ciąg $\{X_{n,l-1}(k)X_{n,l-2}(k)...X_{n,0}(k)\}$ modeluje słowa cyfrowe $\{b_{n,l-1}(k)b_{n,l-2}(k)...b_{n,0}(k)\}$. Liczby $p_n(k)$ otrzymane dla danego *n* są dodawane bitowo modulo 2. Otrzymujemy liczbę

$$q_n = p_n(1) \oplus p_n(2) \oplus ... \oplus p_n(K) = \{c_{n,l-1}c_{n,l-2}...c_{n,0}\} \ , \quad (3)$$

gdzie

$$\begin{cases} c_{n,l-1} = b_{n,l-1}(1) \oplus b_{n,l-1}(2) \oplus ... \oplus b_{n,l-1}(K) \\ c_{n,l-2} = b_{n,l-2}(1) \oplus b_{n,l-2}(2) \oplus ... \oplus b_{n,l-2}(K) \\ \vdots \\ c_{n,0} = b_{n,0}(1) \oplus b_{n,0}(2) \oplus ... \oplus b_{n,0}(K) \end{cases} . \quad (4)$$

Liczby $q_n = \{c_{n,l-1}c_{n,l-2}...c_{n,0}\}$ także należą do przedziału $[0, 2^l)$. Ciąg $\{q_n\}$ możemy zamodelować za pomocą ciągu zmiennych losowych $\{W_n\}$, gdzie

$$W_n = Y_n(1) \oplus Y_n(2) \oplus ... \oplus Y_n(K) \ . \quad (5)$$

Każde W_n jest jednoznacznie reprezentowane przez ciąg binarnych zmiennych losowych $Z_{n,i}$, to znaczy, $W_n = \{Z_{n,l-1}X_{n,l-2}...Z_{n,0}\}$, przy czym

$$\begin{cases} Z_{n,l-1} = X_{n,l-1}(1) \oplus X_{n,l-1}(2) \oplus ... \oplus X_{n,l-1}(K) \\ Z_{n,l-2} = X_{n,l-2}(1) \oplus X_{n,l-2}(2) \oplus ... \oplus X_{n,l-2}(K) \\ \vdots \\ Z_{n,0} = X_{n,0}(1) \oplus X_{n,0}(2) \oplus ... \oplus X_{n,0}(K) \end{cases} . \quad (6)$$

Wykonywanie operacji modulo 2, chociaż bardzo szybkie, nastręcza wiele trudności w modelowaniu matematycznym. Zdecydowanie wygodniejsze jest posługiwanie się operacją dodawania czy mnożenia. Dlatego kluczowym dla naszych rozważań jest trik obliczeniowy wprowadzony w prywatnym artykule Roberta Davies'a, zamieszczonym w Internecie [4]. O istotnym znaczeniu tej pracy świadczy także to, że łącze do strony www z tą pracą można znaleźć na stronie Narodowego Instytutu Standardów i Technologii (NIST) – organizacji zajmującej się w USA standaryzacją, miedzy innymi, w obszarze kryptografii. Według pracy [4], jeżeli *X* przyjmuje wartości 0 lub 1, to $f(X) = 1-2X$ jest równe 1 lub -1. W rezultacie [4],

$$\begin{aligned} f(Z_{n,i}) &= f\left(X_{n,i}(1) \oplus X_{n,i}(2) \oplus ... \oplus X_{n,i}(K)\right) = \\ &= f\left(X_{n,i}(1)\right) \cdot f\left(X_{n,i}(2)\right) \cdot ... \cdot f\left(X_{n,i}(K)\right) \end{aligned} , \quad (7)$$

gdzie $n = 0,1,...$, $i = 0,1,...,l-1$. Jeżeli $E(Z_{n,i})$ jest wartością oczekiwaną zmiennej losowej $Z_{n,i}$, to [4]

$$\begin{aligned} E(Z_{n,i}) &= E\left[\frac{1-f(Z_{n,i})}{2}\right] = \frac{1}{2} - \frac{1}{2}E\left[f(Z_{n,i})\right] = \\ &= \frac{1}{2} - \frac{1}{2}E\left[f\left(X_{n,i}(1)\right) \cdot f\left(X_{n,i}(2)\right) \cdot ... \cdot f\left(X_{n,i}(K)\right)\right] \end{aligned} . \quad (8)$$

If $X_{n,i}(1), X_{n,i}(2),..., X_n(K)$ są niezależne, to

$$\begin{aligned} E(Z_{n,i}) = \frac{1}{2} - \frac{1}{2}\{&E\left[f\left(X_{n,i}(1)\right)\right] \cdot E\left[f\left(X_{n,i}(2)\right)\right] \cdot ... \\ &... \cdot E\left[f\left(X_{n,i}(K)\right)\right]\} \end{aligned} . \quad (9)$$

Ponieważ

$$E\left[f\left(X_{n,i}(k)\right)\right] = 1 - 2E\left(X_{n,i}(k)\right) , \quad (10)$$

mamy

$$\begin{aligned} E(Z_{n,i}) = \frac{1}{2} - \frac{1}{2}\{&\left[1-2E\left(X_{n,i}(1)\right)\right] \cdot \left[1-2E\left(X_{n,i}(2)\right)\right] \cdot ... \\ &... \cdot \left[1-2E\left(X_{n,i}(K)\right)\right]\} \end{aligned} \quad (11)$$

lub

$$\begin{aligned} E(Z_{n,i}) &= \frac{1}{2} - \frac{1}{2}\left\{2\Delta_{n,i}(1) \cdot 2\Delta_{n,i}(2) \cdot ... \cdot 2\Delta_{n,i}(K)\right\} = \\ &= \frac{1}{2} - 2^{K-1}\prod_{k=1}^{K}\Delta_{n,i}(k) \end{aligned} , \quad (12)$$

gdzie

$$\Delta_{n,i}(k) = \frac{1}{2} - E\left(X_{n,i}(k)\right) . \quad (13)$$

Uwzględniając znak $\Delta_{n,i}(k)$ otrzymujemy

$$E(Z_{n,i}) = \begin{cases} \frac{1}{2} - 2^{K-1}\prod_{k=1}^{K}\left|\Delta_{n,i}(k)\right| & dla \quad \prod_{k=1}^{K}\Delta_{n,i}(k) \geq 0 \\ \frac{1}{2} + 2^{K-1}\prod_{k=1}^{K}\left|\Delta_{n,i}(k)\right| & dla \quad \prod_{k=1}^{K}\Delta_{n,i}(k) < 0 \end{cases} . \quad (14)$$

Jeżeli zmienne $X_{n,i}(k)$, $k=1,2,...,K$, $n=0,1,...$, $i=0,1,...,l-1$, przyjmują wartości zarówno 0 jak i 1, to

$$\forall_{n=0,1,...,\ i=0,1,...,l-1,\ k=1,2,...,K} \quad 0<2\left|\Delta_{n,i}(k)\right|<1. \tag{15}$$

Iloczyn liczb większych od zera i mniejszych od jedności dąży do zera ze wzrostem liczby mnożonych czynników, tak więc

$$\lim_{K\to\infty} E(Z_{n,i}) = \lim_{K\to\infty}\left\{\begin{array}{ll} \frac{1}{2}-2^{K-1}\prod_{k=1}^{K}\left|\Delta_{n,i}(k)\right| & dla \quad \prod_{k=1}^{K}\Delta_{n,i}(k)\ge 0 \\ \frac{1}{2}+2^{K-1}\prod_{k=1}^{K}\left|\Delta_{n,i}(k)\right| & dla \quad \prod_{k=1}^{K}\Delta_{n,i}(k)<0 \end{array}\right\}=\frac{1}{2}. \tag{16}$$

Ponieważ $Z_{n,i}$ przyjmuje tylko dwie wartości zero lub jedynkę, otrzymujemy

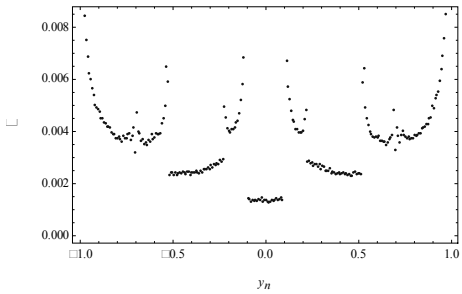
$$P(Z_{n,i}=0)=P(Z_{n,i}=1)=1/2. \tag{17}$$

Chociaż wynik (17) dotyczy każdego bitu słowa *l*-bitowego nie musi oznaczać, że *l*-bitowe liczby kodowane przez te bity są rozłożone równomiernie w przedziale $[0, 2^l)$. Zilustrujmy to przykładem. Przyjmijmy *l*=2 oraz identyczne prawdopodobieństwa otrzymania liczb 0 i 3, równe 0.5. Liczby 1 i 2 są wytwarzane z zerowym prawdopodobieństwem. Po *K* losowaniach ciąg będą tworzyły tylko liczby 0 i 3. Suma bitowa modulo 2 da nam liczbę 0 dla parzystej liczby trójek w ciągu *K* elementowym lub liczbę 3 dla nieparzystej liczby trójek. Liczby 0, 1, 2, 3 nie są więc równomiernie rozłożone chociaż na pozycji *i*=0 oraz *i*=1 bity pojawiają się z prawdopodobieństwem równym 0.5 i są spełnione założenia, dla których prawdziwy jest wzór (15). Aby wykluczyć opisany przypadek musimy założyć, że liczby nie mogą pojawić się w ciągu z prawdopodobieństwem równym zeru (nie istnieją liczby „zabronione") albo równym jedności (nie istnieją liczby „pewne"). Warunek ten oczywiście spełnia generator losowy, gdyż istnienie liczb „pewnych" lub „zabronionych" dyskwalifikuje taki generator jako źródło ciągów losowych. Zatem dopiero po spełnieniu tego warunku równość prawdopodobieństw wszystkich bitów (wzór (17)) otrzymana dla dowolnego *n* skutkuje równymi prawdopodobieństwami liczb kodowanych przez te bity.

Standardowym sposobem zamiany sygnału losowego na liczby losowe jest użycie przetwornika analogowo-cyfrowego. Źródłem losowości są najczęściej szumy układów elektronicznych, na przykład diody lawinowej. W przykładzie użyjemy jednowymiarowego odwzorowania chaotycznego, które w stosunku do źródła szumu charakteryzuje się, przynajmniej teoretycznie, większą odpornością na ataki. Inną zaletą jest znacząco wyższa przepływność ciągów binarnych otrzymywanych na wyjściu (rzędu Gbit/s). Chociaż układ chaotyczny jest układem deterministycznym, to po spełnieniu pewnych warunków może zastąpić praktycznie każde źródło fizyczne [5]. Wadą układu chaotycznego są silnie skorelowane sąsiednie próbki, co powoduje, że jako niezależne możemy uważać dopiero próbki bardzo odległe. O tym jak bardzo odległe powinny to być próbki decyduje szybkość utraty informacji o stanie układu chaotycznego, kontrolowana dla odwzorowania jednowymiarowego wartością wykładnika Lapunowa [5]. W przykładzie wykorzystano odwzorowanie sinusoidalne

$$x_n = x_n - \mu\cdot\sin x_n \tag{18}$$

z $\mu=4{,}5$. Równanie (18) opisuje zmiany różnicy faz pomiędzy sinusoidalnym sygnałem wejściowego i sygnałem, najczęściej prostokątnym, lokalnego generatora dyskretnej w czasie pętli fazowej z próbkowaniem (DT-PLL) dla tzw. zerowej liczby obrotu [6, 7]. Przetwornik A/C zamienia na liczby sygnał wyjściowy y_n detektora fazy. Jego kształt jest taki sam jak kształt sygnału na wejściu, tj. $y_n=\sin x_n$ [6], [7]. Układ DT-PLL można łatwo zaimplementować w układzie cyfrowym, który może dostarczać nawet kilku GSa/s. System chaotyczny (18) z $\mu=4.5$ traci ok. 1,36 bita informacji na iterację [5]. Zatem dla na przykład 20 bitowego A/C możemy użyć próbek po utracie co najmniej 20 bitów informacji, co oznacza wybieranie jako p_n co piętnastą (lub rzadziej) liczbę z wyjścia przetwornika A/C. Zakres przetwarzania wynosi ±1 V. Rozkład empiryczny wartości y_n otrzymany w eksperymencie symulacyjnym pokazuje rysunek 1.



Rys. 1. Rozkład empiryczny 10^6 liczb $y_n=\sin x_n$

Fig. 1. Empirical distribution of 10^6 numbers $y_n=\sin x_n$

Do oceny równomierności rozkładu liczb u_n wykorzystano test chi-kwadrat z 25 stopniami swobody. Wartość krytyczna statystyki dla przedziału ufności 0.01 wynosi 42,98. W Tabeli 1 zamieszczono wyniki testu dla ciągu $\{u_n/2^{20}\}$ o długości 10^5 i kilku wartości *K*. Parametr *i* w algorytmie IMP był równy 3. Rozmiar tablicy był stały dla wszystkich *K* i wynosił 64. Daną wejściową dla IMP była co piętnasta liczba wytworzona przez DT-PLL.

Tab. 1. Wartości statystyki chi-kwadrat dla 10^5 liczb $u_n/2^{20}$

Tab. 1. The values of chi-square statistic for 10^5 numbers $u_n/2^{20}$

	K					
	1	**2**	**3**	**4**	**5**	**6**
χ^2	23 150,11	768,93	46,40	29,85	24,11	19,63

Jak można zauważyć rozkład równomierny otrzymano już dla *K*=4, co odpowiada sumowaniu modulo 2 słów cyfrowych kodujących liczbę bieżącą i trzy liczby wyczytane z tablicy *T*.

Pracę sfinansowano z projektu nr 08/83/DSPB/4707.

3. Wnioski

W pracy pokazano, że obliczeniowo mało złożony, nie wymagający wielu zasobów, algorytm IMP może służyć do wytworzenia ciągu liczb o rozkładzie równomiernym z ciągu liczb o rozkładzie nierównomiernym. Proponowany algorytm jest szybszy i mniej sprzętochłonny od algorytmów używających funkcji skrótu lub algorytmów szyfrujących. Kolejnym krokiem jest implementacja algorytmu w układzie FPGA i określenie wpływu wyboru liczby *i* najstarszych bitów liczb losowych p_{n+L} na działanie algorytmu.

4. Literatura

[1] Jessa M.: Generatory ciągów liter alfabetu skończonego, Wyd. Politechniki Poznańskiej, 2008.
[2] Jessa M.: Jaworski M.: High-Speed FPGA-Based Pseudorandom Generators with Extremely Long Periods, ReConFig'10, Dec. 13-15, 2010, Cancun, s. 286-291.
[3] Gentle J. E.: Random Number Generation and Monte Carlo Methods, Springer, 2003.
[4] Davies R.: Exclusive OR (XOR) and hardware random number generators."http://www.robertnz.net.
[5] Schuster H. G.: Just W.: Deterministic Chaos, Wiley-VCH, 2005.
[6] Bernstein G., Lieberman M. A.: Secure random number generation using chaotic circuits, IEEE Trans. Circuits and Syst. vol. 37, pp. Sept. 1990, s. 1157-1164.
[7] Kudrewicz J.: Wąsowicz S.: Equations of Phase-Locked Loops, Dynamics on Circle, Torus and Cylinder, World Scientific, 2007.

otrzymano / received: 12.04.2014
przyjęto do druku / accepted: 02.06.2014 *artykuł recenzowany / revised paper*