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

Losowość generatora TRNG zaimplementowanego w FPGA

Dr hab. inż. Mieczysław JESSA

Adiunkt na Wydziale Elektroniki i Telekomunikacji Politechniki Poznańskiej. Autor lub współautor ponad 100 publikacji, 15 patentów oraz kilkunastu rozwiązań konstrukcyjnych wdrożonych w krajowej sieci telekomunikacyjnej. Kierujący ponad dwudziestoma projektami wykonanymi na rzecz podmiotów gospodarczych. Najważniejsze prace dotyczą zastosowań zjawiska chaosu w telekomunikacji i kryptografii oraz synchronizacji sieci telekomunikacyjnej.

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

Mgr inż. Łukasz MATUSZEWSKI

Doktorant na Wydziale Elektroniki i Telekomunikacji Politechniki Poznańskiej. Ukończył studia na tym samym wydziale w roku 2010. Jego zainteresowania to projektowanie urządzeń z wykorzystaniem układów reprogramowalnych w szczególności generatorów liczb losowych.

e-mail: lukasz.matuszewski@et.put.poznan.pl

Mgr inż. Michał JAWORSKI

Doktorant na Wydziale Elektroniki i Telekomunikacji Politechniki Poznańskiej. Ukończył studia na tym samym wydziale w roku 2008. Jego zainteresowania to projektowanie urządzeń z wykorzystaniem układów reprogramowalnych oraz nowe technologie internetowe.

e-mail: mj_2000@o2.pl

Streszczenie

W pracy przedstawiono wyniki badań generatora TRNG (ang. *True Random Number Generator*) zbudowanego z wielu niezależnych generatorów pierścieniowych zaimplementowanych w tym samym układzie FPGA. Wykorzystując nową metodę odróżniania losowości od pseudolosowości wykazano, że zmniejszenie częstotliwości próbkowania wyjścia generatora pierścieniowego może zwiększyć losowość ciągu wytwarzanego przez generator TRNG. Otrzymany wynik oznacza, że generator może dostarczyć ciągów losowych użytecznych w kryptografii z większą szybkością od tej obserwowanej dla większej częstotliwości próbkującej.

Słowa kluczowe: generator losowy, generator pierścieniowy, losowość i pseudolosowość, kryptografia.

Randomness of TRNG implemented in FPGA

Abstract

One of the simplest sources of purely digital true random bit sequences is the ring oscillator with output sampled by a signal coming from a low-frequency quartz oscillator. Combining XOR bit streams produced by many such generators (see Fig. 1) can significantly improve the statistical properties of the output sequence. As it is shown in the literature, this statement is true for deterministic and non-deterministic sources of random numbers. In cryptography, a user needs sequences with very good statistical properties but originating from a non-deterministic system. Therefore a method for distinguishing pseudo and true randomness for sequences produced by a combined true random number generator (TRNG) is necessary. In this paper the authors show that even a small amount of true randomness, present in a single ring oscillator, accumulates as a function of the number of ring oscillators used to produce the output stream. There is experimentally proved that in a real field programmable gate array (FPGA), the amount of randomness offered by the generator of Fig. 2 can be greater for smaller sampling frequency. Fig. 3 illustrates the behaviour of parameter m_{min} introduced in [6] as a function of the number K of source generators for four sampling frequencies f_L: 100 MHz, 150 MHz, 200 MHz, and 250 MHz. The basic result of this paper is the statement that the efficient bit rate of streams useful for cryptography can be greater for smaller sampling frequencies than that observed for greater sampling frequencies.

Keywords: random number generator, ring oscillator, randomness and pseudo-randomness, cryptography.

1. Wstęp

Istotnym elementem współczesnych systemów kryptograficznych są generatory liczb prawdziwie losowych TRNG. Ponieważ systemy kryptograficzne to prawie wyłącznie konstrukcje cyfrowe oczekuje się, że generator TRNG będzie zbudowany wyłącznie z układów cyfrowych. Do roku 2008 autorzy proponowali jako źródło losowości prawie wyłącznie jeden układ, najczęściej analogowy, a otrzymywane szybkości wytwarzania ciągów liczb prawdziwie losowych na potrzeby kryptografii, to zaledwie kilka Mbit/s [1]. W roku 2008 Wold i Tan zaproponowali, aby w tym samym układzie reprogramowalnym zaimplementować wiele generatorów pierścieniowych RO (ang. *Ring Oscillators*), których wyjścia są próbkowane sygnałem tego samego generatora, np. kwarcowego [2]. Po połączeniu XOR strumieni bitów wytwarzanych przez wiele generatorów, nazywanych dalej źródłowymi, otrzymali ciąg, który spełnia wszystkie testy z pakietu NIST 800-22 bez potrzeby stosowania dodatkowego przetwarzania poprawiającego właściwości statystyczne, tzw. post-processingu.

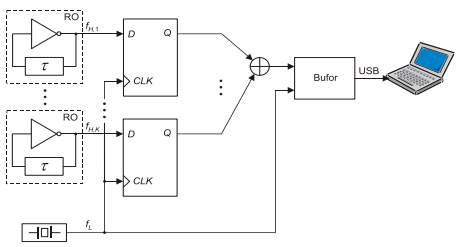
Metoda z próbkowaniem sygnału wyjściowego generatora pierścieniowego, w której źródłem losowości są szybkozmienne fluktuacje fazy przebiegu wytwarzanego przez RO pracujący swobodnie była analizowana wcześniej przez kilku autorów (zob. np. [1]-[3] wraz ze spisem literatury). Binarny sygnał losowy jest otrzymywany w wyniku próbkowania sygnału generatora pierścieniowego o częstotliwości f_H przez generator o częstotliwości f_L. Zakłada się przy tym, że $f_H >> f_L$. Nowością metody Wolda i Tana jest to, że połączyli oni za pomocą sumy modulo 2 strumienie binarne wytwarzane przez $K>1$ niezależnych generatorów TRNG tego typu. W 2009 roku Wold i Petrović przedstawili optymalizację wcześniejszej wersji generatora. Otrzymany generator wytwarza ciągi binarne z szybkością 300 Mbit/s [4]. Równocześnie N. Bochard, F. Bernard oraz V. Fischer wykazali w eksperymencie symulacyjnym, że generator Wolda i Tana może spełnić wszystkie testy NIST 800-22 nawet wtedy, gdy generatory pierścieniowe są pozbawione szybkozmiennych fluktuacji fazy, a więc nie posiadają źródła losowości [5]. Konkluzją tej pracy jest stwierdzenie, że bardzo dobre właściwości statystyczne generatora Wolda i Tana mogą być rezultatem mechanizmu deterministycznego, co wyklucza zastosowanie takiego generatora w kryptografii. W roku 2010 M. Jessa i M. Jaworski przeanalizowali ten problem dokładniej. Wykorzystując mechanizm restartów oraz test chi-kwadrat pokazali, że nawet mała ilość losowości obecna w pojedynczym RO ulega akumulacji ze wzrostem liczby K generatorów źródłowych [6, 7]. W rezultacie otrzymanie ciągu losowego dla zastosowań w kryptografii jest jak najbardziej możliwe z tym, że ciąg wyjściowy powinien być utworzony z co j-tego elementu ciągu wytwarzanego przez generator Wolda i Tana, gdzie j jest większe od pewnej minimalnej liczby m_{min} zależnej od liczby K generatorów źródłowych. Gdy K rośnie, to j maleje, co oznacza, że spadek efektywnej szybkości wytwarzania ciągu jest mniejszy dla dużych K. W pracach z 2010 roku założono, że częstotliwości $f_{H,k}$, $k=1,2,\ldots,K$ generatorów pierścieniowych są

tylko nieznacznie większe od częstotliwości próbkującej f_L=300 MHz. Przepływność ciągu wyjściowego była równa 300 Mbit/s. W świetle badań przedstawionych w pracach [6] i [7] za ciąg prawdziwie losowy możemy uważać ciąg utworzony z co 42 bitu ciągu 300 Mbit/s dla K=50. Daje to efektywną szybkość ciągu równą około 7,14 Mbit/s zamiast 300 Mbit/s.

W obecnej pracy pokazano, że efektywną szybkość wytwarzania ciągu prawdziwie losowego dla zastosowań w kryptografii można zwiększyć obniżając szybkość wytwarzania ciągu przez generator TRNG. Ten pozorny paradoks bierze się stąd, że dla danego K losowość ciągu wyjściowego zależy także od wielkości różnicy pomiędzy częstotliwością próbkującą f_L, która ustala przepływność ciągu wyjściowego, a częstotliwościami $f_{H,k}$ generatorów pierścieniowych RO. W rezultacie wybór co j-tego bitu i mniejszej f_L może dać większą efektywną szybkość ciągu bitowego. Mniejsze f_L uławia także projektowanie oraz implementację generatora TRNG w FPGA. W pracy rozważono częstotliwości próbkowania 100 MHz, 150 MHz, 200 MHz oraz 250 MHz. Dla każdej częstotliwości badano losowość ciągu w funkcji liczby K generatorów źródłowych (od 1 do 50).

2. Wytwarzanie ciągu losowego w układzie FPGA

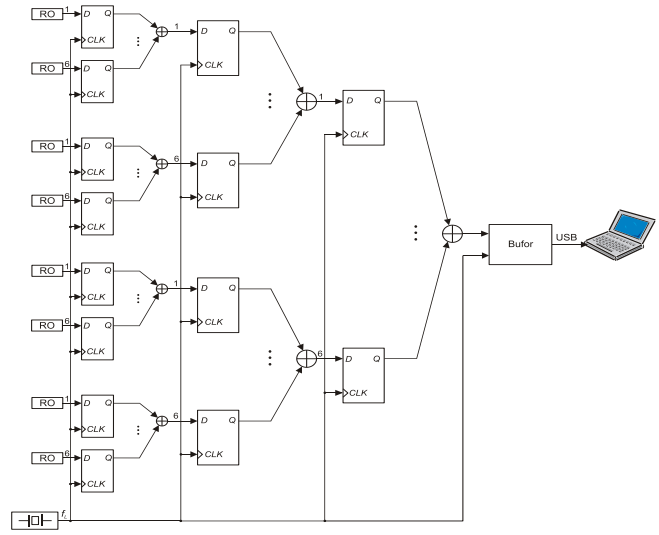
Do wytworzenia ciągu binarnego wykorzystujemy K generatorów pierścieniowych pokazanych na rysunku 1.



Rys. 1. Generator TRNG wykorzystujący K generatorów pierścieniowych
Fig. 1. TRNG using K ring oscillators

Sygnał wyjściowy generatorów pierścieniowych jest próbkowany przez sygnał generatora kwarcowego o częstotliwości f_L mniejszej od częstotliwości $f_{H,k}$. Wyjścia K przerzutników są sumowane modulo 2, a otrzymany bit jest przesyłany za pomocą dodatkowego bufora do komputera. Ponieważ dostępne bloki LUT mają za mało wejść, zaproponowano inny schemat, pokazany na rysunku 2. Każda z sum modulo 2 sumuje co najwyżej 6 bitów. Maksymalna liczba K generatorów źródłowych możliwych do zrealizowania w strukturze z rysunku 2 wynosi 6^3=216. Generator zaimplementowano w układzie Virtex-5 (XL5VLX50T) [8]. Opóźnienie τ realizuje jeden przerzutnik typu zatrzask (ang. *latch*).

Sygnał wyjściowy generatora pierścieniowego zawiera zarówno fluktuacje fazy o charakterze niedeterministycznym, co jest pożądane dla zastosowań w kryptografii oraz deterministycznym. Ponieważ testy statystyczne mogą spełnić ciągi wytwarzane przez źródła niedeterministyczne lub deterministyczne, istotne jest dysponowanie narzędziem, które rozróżni oba źródła. Jeżeli testy są spełnione dla źródeł wytwarzających wyłącznie fluktuacje fazy o charakterze deterministycznym, to mówimy o pseudolosowości generatora. Jeżeli zdają je ciągi wytwarzane dzięki obecności wyłącznie fluktuacji fazy o charakterze niedeterministycznym, to mówimy o tzw. prawdziwej losowości, lub krócej, o losowości generatora [3]. Dla odróżnienia losowości od pseudolosowości, dla generatora z rysunku 2 możemy zastosować mechanizm restartów [3]. W tym celu startujemy wielokrotnie generator z tymi samymi warunkami początkowymi. W przypadku generatorów pierścieniowych wystarczy zamienić negatory z rysunku 1 na bramki NAND wyzwalane sygnałem zewnętrznym. Jeżeli spełnienie testów wynika z pseudolosowości, to po każdym restarcie powinniśmy otrzymać ten sam ciąg. W przeciwnym przypadku ciągi będą się różnić.



Rys. 2. Modyfikacja generatora z rysunku 1 zachowująca funkcjonalność
Fig. 2. Modification of the generator of Fig. 1 preserving its functionality

W pracy [6] zaproponowano wykonanie M testów χ^2 zgodności rozkładu dla N-bitowych ciągów wyjściowych generatora, gdzie N jest liczbą restartów, a M jest liczbą bitów wytwarzanych dla jednego restartu. Jeżeli dla danej chwili czasowej $m=1,2,...,M$ test χ^2 jest spełniony, to uważamy, że dla danego poziomu istotności testu nie ma podstaw do odrzucenia hipotezy, iż m-ty bit otrzymano w wyniku obecności w sygnałach generatorów pierścieniowych szybkozmiennych fluktuacji fazy o charakterze niedeterministycznym. Wartość statystyki χ^2 obliczano ze wzoru:

$$\chi^2 = \sum_i^n \frac{\left(N_i - N \cdot P_i\right)^2}{N \cdot P_i}, \qquad (1)$$

gdzie n jest liczbą rozłącznych podzbiorów zawierających próbki. Liczba N_i jest liczbą elementów w i-tym podzbiorze, a $N \cdot P_i$ jest oczekiwaną liczbą elementów w tym podzbiorze. Jeżeli hipoteza o zadanym rozkładzie jest prawdziwa, to statystyka (1) dąży asymptotycznie do rozkładu chi-kwadrat z $n-r-1$ stopniami swobody, gdzie r jest liczbą estymowanych parametrów. Wartość statystyki obliczona ze wzoru (1) jest porównywana z wartością krytyczną odczytaną z tablic rozkładu χ_c^2 z $n-r-1$ stopniami swobody. W naszym przypadku jest $r=0$. Dla ciągu binarnego mamy $n=2$, co daje jeden stopień swobody. Przyjmując poziom istności testu $\alpha=0,01$ otrzymujemy wartość krytyczną statystyki równą 6,635. Jeżeli wartość statystyki (1) jest mniejsza od 6,635, to nie ma powodów do odrzucenia hipotezy (na poziomie istotności $\alpha=0,01$), że zera i jedynki w m-tym, N-elementowym ciągu występują z tą samą częstością. Jeżeli hipoteza o akumulowaniu ze wzrostem K fluktuacji fazy o charakterze niedeterministycznym jest prawdziwa, to test χ^2 powinien być spełniony dla coraz mniejszych m.

W eksperymencie założono, że liczba restartów wynosi N=2048. Dla każdego restartu wytwarzano M=20000 bitów. W czasie badań interesowała nas, podobnie jak w artykułach [6] i [7], najmniejsza wartość m ($m_{\min}$), dla której wartość statystyki jest mniejsza od wartości krytycznej z zastrzeżeniem, że dla $m \ge m_{\min}$ test chi-kwadrat może nie być spełniony co najwyżej dla dwóch kolejnych chwil a dla wszystkich pozostałych $m \ge m_{\min}$ jest spełniony. To ograniczenie wynika z faktu, że idealne źródło losowe wytwarza różne ciągi z tym samym prawdopodobieństwem. Zatem mogą także wystąpić ciągi, które nie spełniają testu chi-kwadrat, na przykład ciąg samych zer i nie może to być podstawą do twierdzenia, że źródło nie wytwarza ciągów losowych. Jeżeli p jest prawdopodobieństwem nie spełnienia testu chi-kwadrat przez ciąg restartów, to prawdopodobieństwo, że taka sytuacja wystąpi dla

dwóch kolejnych chwil wynosi p^2. Dla trzech kolejnych chwil jest to p^3 itd. Innymi słowy, jeżeli dla idealnego źródła p jest równe na przykład 0,01, to statystycznie dla co setnej chwili ciąg restartów może nie spełnić testu chi-kwadrat. To samo źródło może wygenerować ciągi, które nie spełnią testu chi-kwadrat dla dwóch kolejnych chwil co 10^4 chwil, nie spełnią testu chi-kwadrat dla trzech kolejnych chwil co 10^6 chwil itd. Przyjmując kryterium, że test chi-kwadrat nie jest spełniony co najwyżej dla dwóch kolejnych chwil, a dla wszystkich pozostałych $m \geq m_{\min}$ jest spełniony, oczekuje się, że odstęp czasu pomiędzy sytuacjami, w których test chi-kwadrat nie jest spełniony dla trzech i więcej kolejnych chwil jest dłuższy od M. Zatem jeżeli w ciągu empirycznym o długości M wystąpią jednak trzy lub więcej kolejne chwile, dla których test chi-kwadrat nie jest spełniony, to uznajemy iż jest to spowodowane brakiem dostatecznej losowości źródła.

Ze wzoru 1 można dowieść, że dla danej chwili ciąg restartów nie spełnia testu chi-kwadrat gdy liczba jedynek różni się od liczby zer o co najmniej 117. Ciąg restartów spełnia test chi-kwadrat, gdy liczba jedynek różni się od liczby zer o co najwyżej 116. Możemy zapisać, że

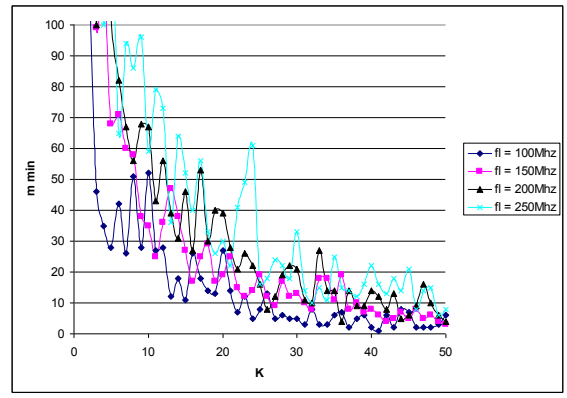
$$p = 1 - p', \tag{2}$$

gdzie p jest prawdopodobieństwem, że liczba jedynek różni się od liczby zer o co najmniej 117, a p' jest prawdopodobieństwem, że liczba jedynek różni się od liczby zer o co najwyżej 116. Ponieważ liczba restartów M=2048 jest parzysta, to różnica pomiędzy liczbą zer i jedynek też musi być parzysta. Dla idealnego źródła ciągów losowych elementy ciągu są niezależne. Stąd prawdopodobieństwo p', że liczba jedynek różni się od liczby zer o co najwyżej 116 jest sumą prawdopodobieństw, że liczba jedynek różni się od liczby zer dokładnie o 0,2,4,…,116. W pierwszym przypadku ciąg restartów o długości 2048 ma 1024 jedynki na dowolnych pozycjach i 1024 zera też na dowolnych pozycjach. W drugim ma 1023 jedynki na dowolnych pozycjach oraz 1025 zer na dowolnych pozycjach lub odwrotnie itd. Dla 116 mamy 966 jedynek na dowolnych pozycjach oraz 1082 zera na dowolnych pozycjach lub odwrotnie. Jeżeli $1/2^{2048}$ jest prawdopodobieństwem otrzymania dowolnego ciągu o długości 2048 bitów, to

$$p' = \frac{1}{2^{2048}}\left[C_{2048}^{1024} + \sum_{i=1}^{58}\left(C_{2048}^{1024-i} + C_{2048}^{1024+i}\right)\right] = 0{,}990289... \tag{3}$$

oraz $p = 0{,}009710\ldots$. Zatem odstęp pomiędzy elementami ciągu o długości M, dla którego test chi-kwadrat może nie być spełniony dla trzech i więcej kolejnych m wynosi $1/p^3 \approx 10^6$ i jest znacząco dłuższy od długości ciągu generowanego dla każdego restartu tj. od 20000 bitów. Natomiast odstęp pomiędzy elementami ciągu o długości M, dla którego test chi-kwadrat może nie być spełniony dla dwóch i więcej kolejnych m wynosi $1/p^2 \approx 10604$ i jest krótszy od długości ciągu generowanego dla każdego restartu.

Wykres wartości $m_{\min}$ w funkcji liczby K generatorów źródłowych dla wszystkich częstotliwości próbkujących rozważanych w pracy tj. dla 100 MHz, 150MHz, 200 MHz oraz 250 MHz pokazano na rysunku 3. We wszystkich przypadkach gdy K rośnie, to $m_{\min}$ maleje, chociaż nieregularnie. Podobnie jak w pracach [6] i [7] otrzymano, że dla danej chwili czasowej ze wzrostem liczby użytych generatorów źródłowych rośnie losowość N elementowego ciągu. Tej właściwości nie obserwuje się dla fluktuacji fazy o charakterze deterministycznym, gdyż niezależnie od K dla danego m i tych samych warunków początkowych zawsze otrzymamy ten sam bit. Na wykresie można zauważyć jednak także inną ważną cechę. Dla ustalonego K mniejsze f_L oznacza wyraźnie mniejsze m_{min}. Podobnie jak w [6] i [7], aby otrzymać ciąg prawdziwie losowy, np. dla zastosowań w kryptografii, ciąg wyjściowy powinien być utworzony z co j-tego elementu ciągu wytwarzanego przez generator z rysunku 2, gdzie $j \geq m_{\min}$. Zmniejsza to co najmniej m_{min} krotnie efektywną szybkość wytwarzania ciągu. Okazuje się jednak, że dla mniejszych f_L iloraz $f_L / m_{\min}$ może być większy od tego, który otrzymano dla większych częstotliwości próbkujących. Przykładowo, dla $K = 30$ (jest to najmniejsza wartość K, dla której są już spełnione wszystkie testy statystyczne NIST 800-22 dla wszystkich częstotliwości próbkujących f_L rozważonych w pracy) i dla $f_L = 100$ MHz mamy 20 Mbit/s ($m_{\min} = 5$), a dla $f_L = 250$ MHz jest to ~7,57 Mbit/s ($m_{\min} = 33$).



Rys. 3. Wartości $m_{\min}$ w funkcji K

Fig. 3. The values of $m_{\min}$ as a function of K

3. Wnioski

Podstawowym wnioskiem jest to, że otrzymanie większej efektywnej szybkości wytwarzania ciągu losowego na potrzeby kryptografii w oparciu o generator z rysunku 2 niekoniecznie wymaga zwiększenia częstotliwości próbkującej jak dotychczas sądzono. Jest tak przynamniej dla implementacji generatora TRNG z rysunku 2 w układzie Virtex-5. Kierunki dalszych badań to przede wszystkim sprawdzenie, czy podobny wniosek można otrzymać dla innych układów FPGA tego samego oraz innych producentów. Inny kierunek badań to poszukiwanie metod dalszego zwiększania efektywnej szybkości wytwarzania ciągu, który spełnia nie tylko wszystkie testy statystyczne NIST 800-22 lecz może być także uznany za ciąg prawdziwie losowy.

4. Literatura

[1] Bucci M. and Luzzi R.: Fully digital random bit generators for cryptographic applications, IEEE Trans. Circuits and Syst. I, Regular Papers., vol. 55, pp. 861-875, April 2008.

[2] Wold K. and Tan C.: Analysis and enhancement of random number generator in FPGA based on oscillator rings, Proc, of ReConFig 2008, 3-5 Dec, 2008, pp, 385-390.

[3] Dichtl M. and Golić J.D.: High speed true random number generation with logic gates only, CHES 2007, Lecture Notes in Computer Science, LNCS 4727, pp, 45-62, 2007.

[4] Wold K. and Petrović S.: Optimizing speed of a true random number generator in FPGA by spectral analysis," Proc. of Fourth International Conference on Computer Sciences and Convergence Information Technology, ICCIT'09, 24-26 Nov., 2009, pp. 1105-1110.

[5] Bochard N., Bernard F. and Fischer V.: Observing the randomness in RO-based TRNG," Proc, of ReConFig 2009, 9-11 Dec, 2009, pp, 237-242.

[6] Jessa M., Jaworski M.: Randomness of a combined TRNG based on the ring oscillator sampling method, Proceedings of International Conference on Signals and Electronic Systems, ICSES'10, Sept. 7-10, 2010, pp. 323-326.

[7] Jessa M., Jaworski M.: Losowość generatora łączonego TRNG wykorzystującego metodę z próbkowaniem wyjścia generatora pierścieniowego, Elektronika, vol. LI nr 12/2010, s. 47-50.

[8] http://www,xilinx,com/support/documentation/virtex-5,htm.

otrzymano / received: 13.05.2011
przyjęto do druku / accepted: 04.07.2011 *artykuł recenzowany*