Paweł DĄBAL, Ryszard PEŁKA
WOJSKOWA AKADEMIA TECHNICZNA, WYDZIAŁ ELEKTRONIKI, INSTYTUT TELEKOMUNIKACJI, ZAKŁAD TECHNIKI CYFROWEJ
ul. gen. Sylwestra Kaliskiego 2, 00-908 Warszawa

Implementacja pakietu testów statystycznych do badania generatorów pseudolosowych w układzie programowalnym

Mgr inż. Paweł DĄBAL

Ukończył studia na Wydziale Elektroniki Wojskowej Akademii Technicznej. Obecnie jest doktorantem IV roku Wydziału Elektroniki WAT. Jego zainteresowania naukowe dotyczą projektowania układów i systemów cyfrowych z wykorzystaniem struktur programowalnych FPGA dla zastosowania w kryptografii.

e-mail: pdabal@wat.edu.pl

Prof. dr hab. inż. Ryszard PEŁKA

Studiował na Wydziale Elektroniki Politechniki Warszawskiej. Doktor nauk technicznych (1984), doktor habilitowany (1997) i profesor (2004). Profesor zwyczajny na Wydziale Elektroniki w Wojskowej Akademii Technicznej. Kierownik Zakładu Techniki Cyfrowej w Instytucie Telekomunikacji WAT. Specjalizuje się w dziedzinie techniki cyfrowej, w obszarze precyzyjnej metrologii odcinków czasu oraz projektowania, optymalizacji i testowania systemów cyfrowych z układami FPGA i SoC.

e-mail: rpelka@wat.edu.pl

Streszczenie

W artykule przedstawiono projekt i wyniki badań eksperymentalnych mikrosystemu w układzie SoC Zynq (Xilinx) przeznaczonego do analizy statystycznej z użyciem pakietu NIST SP800-22 binarnych sekwencji pochodzących z implementowanych chaotycznych generatorów pseudolosowych. Omówiono sposób implementację pakiet testów NIST oraz wskazano potencjalne możliwości zrealizowania wybranych operacji sprzętowo. Kompletny system zajmuje 4% przerzutników i 19% bloków LUT dostępnych w układzie XC7Z020. Zastosowanie proponowanych mechanizmów pozwoliło na uzyskanie wydajności na poziomie 100 Mb/s.

Słowa kluczowe: analiza statystyczna, generatory losowe, FPGA.

Statistical tests of pseudo-random number generators in a programmable device

Abstract

This paper presents the concept, design and experimental results of a SoC-based microsystem with Zynq device from Xilinx, for statistical testing of bit-streams from pseudo-random bit generators (PRBGs). In order to detect any symptoms of non-random behavior of PRBGs, we apply the commonly used statistical tests proposed by NIST as a standard package SP800-22. Five basic tests out of 15 tests from the NIST package have been converted from PC platform and adopted to specific embedded ARM architecture. Key elements of statistical analysis are performed by a dedicated analyzer implemented in programmable logic while the other functions are executed by an integrated dual-core processor. The complete microsystem uses 4% of flip-flops and 19% of LUTs available in the XC7Z020 SoC device. The operation of the microsystem has been optimized by assumption of fixed confidence level of statistical tests and constant data sample size equal to 220. Using these values we get the maximum throughput of data analysis at the level of 100 Mbps. The proposed system may be used for real-time analysis and tracing of pseudo-random binary sequences obtained from integrated PRBGs. This feature is an important improvement in statistical testing of high bit-rate data streams since conventional NIST tests running on the PC platform can be executed in the off-line mode only. Our further work will be focused on the implementation of some other tests from the NIST package and speedup techniques based on multiple bit analysis in a single clock cycle.

Keywords: statistical tests, random number generators, FPGA.

1. Wstęp

Kluczowym elementem projektowania i weryfikacji poprawności pracy generatorów losowych RNG (*Random Number Generators*) jest badanie pozyskiwanych sekwencji binarnych pod kątem wystąpienia nieprawidłowości świadczących o braku losowości. Podstawową miarą losowości ciągu binarnego jest stosunek liczby wystąpień „0” do liczby „1”. Miara ta jednak nie pokazuje wielu potencjalnych słabości jakimi może cechować się RNG. Wymagane jest użycie bardziej złożonych testów ukierunkowanych na zbadanie różnych aspektów statystycznych generowanych sekwencji takich jak korelacje, cykliczność lub rozproszenie. Testy te są grupowane w zestawy spośród których najbardziej rozpowszechnione są dwa. Pierwszy został opracowany na potrzeby NIST i jest znany jako SP800-22 [1], drugi to DIEHARD [2]. Umożliwiają one dogłębną analizę, która odbywa się jednak kosztem dużej złożoności obliczeniowej, jest czasochłonna i uniemożliwia analizę danych w czasie rzeczywistym.

Współczesna technologia układów programowalnych oferuje nowe klasy układów SoC łączących logikę programowalną z wbudowanym mikroprocesorem w jednej obudowie. Układy te mogą stanowić dobrą platformę do projektowania rozwiązań hybrydowych łączących wydajną architekturę procesora z dedykowanymi blokami sprzętowymi implementowanymi w logice programowalnej. Przykładem takich układów jest wprowadzona przez firmę *Xilinx* rodzina układów Zynq oraz rodziny Cyclone V i Arria V firmy *Altera*.

Artykuł jest zorganizowany w następujący sposób. W sekcji 2 opisany został pakiet testów NIST 800-22 wraz z oceną możliwości sprzętowej realizacji wybranych składników testu oraz wprowadzenia optymalizacji programowych. Sekcja 3 przedstawia sposób implementacji pakietu, budowę oprogramowania oraz użyte narzędzia. W sekcji 4 omówiono wyniki badań eksperymentalnych mikrosystemu z układem SoC, który porównano z rozwiązaniem programowym. Na koniec, w rozdziale 5 przedstawione zostało podsumowanie i uwagi na temat dalszych prac.

2. Analiza sekwencji losowych

Dotychczasowe prace [3, 4] skupiały się na analizie wybranych konfiguracji generatorów chaotycznych. W celu przyspieszenia analizy statystycznej dużych strumieni danych pozyskiwanych z użyciem badanych generatorów zaprojektowano system rozproszony [5]. Niniejsza praca dotyczy projektu zintegrowanego mikrosystemu z układem SoC z wbudowanymi testami dla zapewnienia ciągłego monitorowania pracy generatorów.

2.1. Pakiet testowy NIST SP800-22

Pakiet do analizy statystycznej binarnych sekwencji losowych zarekomendowany przez NIST o oznaczeniu SP800-22 składa się z 15 testów, które badają sekwencje pod kątem wystąpienia zachowania nielosowego. Każdy z testów przeprowadzany jest dla tej samej sekwencji bitów o długości n w wyniku czego otrzymujemy wartość *P-value*. Jest to miara prawdopodobieństwa, że dana sekwencja jest losowa. Jeżeli poziom ufności $\alpha = 0{,}01$ (wartość typowa) to można przyjąć, że sekwencja jest nielosowa w 1%. Sekwencja zalicza test jeżeli *P-value* $> \alpha$. Jest to weryfikacja pierwszego poziomu. Następnie dla m kolejnych wyników testu standard SP800-22 proponuje dwa sposoby dalszego postępowania. Pierwszy z nich polega na sprawdzeniu

czy wartości *P-value* dla danego testu są równomiernie rozłożone w przedziale <0; 1> za pomocą testu dopasowania, wynikiem którego jest wartość $P\text{-}value_T$. Jeżeli $P\text{-}value_T > 0{,}0001$ wyniki testów można uznać za równomiernie rozłożone. Drugi sposób polega na obliczeniu proporcji testów zaliczonych do liczby sekwencji m i sprawdzeniu czy mieści się ona w przedziale ufności. Jeżeli wszystkie rezultaty testów są pozytywne, można powiedzieć, że z założonym poziomem ufności α sekwencja jest losowa.

Pakiet jest dostarczany w postaci kodów źródłowych w języku C. Przygotowana aplikacja testująca został przygotowana pod kątem łatwości interpretacji i możliwości wprowadzenia rozszerzeń, np. w postaci nowych testów. Konsekwencją takiego podejścia jest ograniczenie wydajności aplikacji. Kolejnym aspektem negatywnie odbijającym się na wydajności jest przechowywanie pojedynczych bitów w pojedynczych bajtach co powoduje zwiększone zużycie pamięci. Wykonie kompletnego zestawu testów dla $m = 128$ próbek, o długości $n = 2^{20}$ bitów każda, oraz przy przyjętym poziomie ufności $\alpha = 0{,}01$ wymaga około 600 sekund (ok 3 s na próbkę + czas na wyznaczenie statystyk 2 poziomu) dla komputera PC z CPU Intel Core i5 3,2 GHz. Testowanie odbywa się w trybie *off-line* przez co utrudnione jest szybkie reagowanie na pojawienie się potencjalnych błędów.

2.2. Sprzętowe wsparcie realizacji testów NIST

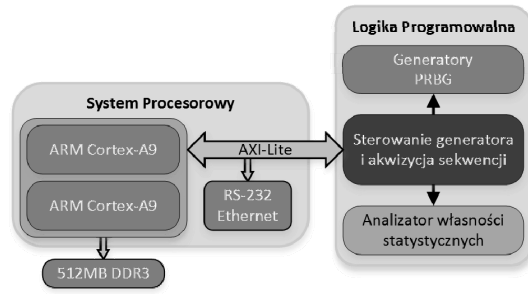
Analizując udostępniony przez NIST kod źródłowy można wskazać kilka elementów, które w przypadku aplikacji wbudowanej mogą zostać pominięte, co przyczynia się do wzrostu wydajności. W pierwszej kolejności można zrezygnować z tworzenia szczegółowych plików raportów z każdego z testów. W ten sposób odpadają wszystkie operacje związane z konwersją liczb do postaci tekstowej i obsługi systemu plików. Kluczową rolę odgrywa obliczanie złożonych funkcji statystycznych takich jak e*rf, erfc*, Γ (funkcja błędu Gausa, komplementarna funkcja błędu). Jeżeli przyjęte zostaną stałe parametry testów, takie jak poziom ufności α oraz długość badanej sekwencji n, można wyznaczyć minimalne wartości odwrotne dla tych funkcji bez potrzeby każdorazowego ich obliczania [6]. Jednakże w ten sposób rezygnujemy z testowania na poziomie drugim. Z modyfikacją tą można pójść dalej i wyznaczyć wartości graniczne przedziałów wystąpienia określonego zdarzenia w teście, np. dla testu *Frequency* można określić, że dopuszczalna różnica w liczbie zer i jedynek w sekwencji o długości $n = 2^{20}$ przy przyjętym poziomie ufności $\alpha = 0{,}01$ powinna zawierać się w przedziale <-14; 14>.

Najczęściej występującą operacją w testach jest zliczanie wystąpienia określonego zdarzenia, np. zliczanie liczby „1” (*Frequency, Block Frequency*), zliczanie liczby przejść z „0” na „1” i odwrotnie (*Runs*), zliczanie nieprzerwanej długości sekwencji „1” (*Longest Run of Ones*) oraz porównanie do określonego wzorca (*Overlapping Template Matching*). Operacje te mogą być wykonane za pomocą zestawu komparatorów oraz liczników sterowanych przez jednostkę nadzorującą wykonanie testu. Są to obliczenia stałoprzecinkowe, w związku z czym mogą być efektywnie zrealizowane sprzętowo. Korzystając z ustalonych wartości krytycznych można zbudować szybkie układy testujące z binarnym wyjściem wyniku testu co jest wystarczające do monitorowania generatorów binarnych ciągów pseudolosowych (PRBG). Alternatywnie, zliczone wystąpienia określonych zdarzeń mogą być przekazane do procesora w celu wyznaczenia pełnej statystyki.

3. Implementacja testów statystycznych

W celu uproszczenia projektu przyjęte zostały następujące założenia: długości badanej sekwencji $n = 2^{20}$; parametry poszczególnych testów zostały ustalone na sztywno zgodnie z zalecanymi wartościami dla danej długości sekwencji. Przy tych założeniach, dla każdego z testów wyznaczone zostały wartości odwrotne dla statystyk oraz krytyczne wartości całkowitoliczbowe.

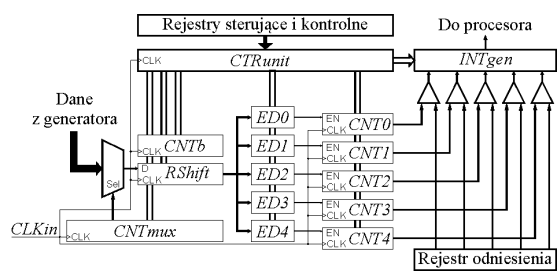
Na rys. 1 przedstawiono schemat blokowy systemu. Jak źródło sekwencji posłużyć może jeden z generatorów przedstawionych w [4]. Na potrzeby testów użyty został generator pracujący w oparciu o odwzorowanie logistyczne. Blok sterowania generatorem i akwizycji jest odpowiedzialny za ustalenie punktu początkowego, sterowanie sygnałem zegarowym oraz odczytywanie generowanych słów, które są przekazywane za pośrednictwem kolejki FIFO o rozmiarze 1kB do procesora w celu zapisania do zewnętrznej pamięci RAM na potrzeby analizy programowej lub do bloku analizy własności statystycznych.



Rys.1. Schemat blokowy zintegrowanego mikrosystemu testującego
Fig. 1. Block diagram of the integrated testing microsystem

3.1. Budowa analizatora własności statystycznych

Rysunek 2 przedstawia schemat bloku testującego. Dane do testów z postaci równoległej zamieniane są na szeregową (multiplekser i licznik *CNTmux*) i wprowadzane do rejestru przesuwającego *RShift*, który ma 24-bitowe wyjście równoległe. Licznik główny *CNTb* odpowiada za zliczanie bitów sekwencji oraz stanowi główny punkt odniesienia dla jednostki sterującej *CTRunit*, która generuje na jego podstawie sygnały sterujące do poszczególnych bloków. Tor analizy dla każdego z zaimplementowanych testów składa się z licznika *CNTi* (*i* – numer testu) odpowiedzialnego za zliczanie zdarzeń wykrytych przez blok kombinacyjny detekcji zdarzenia *EDi*. Zaimplementowane zostało wsparcie sprzętowe dla 5 testów: *Frequency* (*i* = 0)*, Block Frequency* (1)*, Runs* (2)*, Longest Run of Ones* (3) oraz *Overlapping Template Matching* (4). Element *INTgen* na podstawie wyniku porównania zliczonej liczby zdarzeń z wartością krytyczną generuje przerwanie dla procesora informujące o wystąpieniu nieprawidłowości. Komunikacja z procesorem odbywa się z użyciem banku rejestrów: sterującego, kontrolnego, wartości krytycznych oraz z wynikami pośrednimi poszczególnych testów, poprzez magistralę AXI4-Lite.



Rys.2. Schemat blokowy analizatora własności statystycznych
Fig. 2. Block diagram of the statistical parameters analyzer

Tablica 1 zawiera zestawienie wymaganych zasobów logicznych przez poszczególne elementy systemu testującego. Blok testujący zajmuje około 19% bloków LUT i 4% przerzutników FF dostępnych w układzie. Moduł może pracować z prędkością do 100 MHz co przekłada się na efektywną zdolność weryfikowania sekwencji o przepływności dochodzącej do 100 Mb/s.

Tab. 1. Zasoby logiczne zajmowane przez system testujący
Tab. 1. Logical resources required by the microsystem

Komponent		LUT	FF	DSP
Generator		107	64	16
Akwizycja		1644	480	0
Analiza		388	416	1
w tym:	- CTRunit	148	256	0
	- ED0 i CNT0	32	28	0
	- ED1 i CNT1	48	36	1
	- ED2 i CNT2	64	48	0
	- ED3 i CNT3	32	16	0
	- ED4 i CNT4	64	32	0
Razem		2139	960	17

W odniesieniu do rozwiązania [7] uzyskano znaczną redukcję wymaganych zasobów logicznych (o ponad połowę) oraz nieznaczne przyspieszenie szybkości pracy. Głównym czynnikiem, który umożliwił tę poprawę jest znaczny rozwój technologii układów programowalnych między rodzinami Virtex II a Zynq, które dzielą 4 generacje postępu technologicznego.

3.2. Aplikacja testująca

Oryginalny kod źródłowy przeznaczony dla aplikacji PC został przeniesiony na architekturę ARM Cortex-A9. Dodane zostały funkcje odpowiedzialne za sterowanie modułem i obsługę generowanych przerwań. Aplikacja może pracować w kilku trybach. W trybie pełnym (*NISTv0*) wykonywane są wszystkie testy i wyznaczana jest wartość *P-value* oraz $P\text{-value}_T$. W trybie częściowym (*NISTv1*) dla wszystkich testów wyznaczana jest tylko wartość *P-value.* Natomiast w trybie minimalnym (*NISTv2*) sprawdza się czy sekwencja przekracza próg krytyczny bez wyznaczania statystyki. W tym ostatnim przypadku można ograniczyć się jedynie do realizacji monitorowania sprzętowego.

3.3. Metoda projektowa i użyte narzędzia

Implementację testów statystycznych wykonano z użyciem pakietu oprogramowania firmy Xilinx ISE/EDK, w wersji 14.7. Program ISE Project Navigator posłużył do przygotowania modułów testujących opisanych behawioralnie z użyciem języka VHDL oraz symulacyjnej weryfikacji poprawności pracy. Środowisko EDK posłużyło do opracowania zintegrowanego systemu oraz napisania stosownego oprogramowania w języku C.

Jako bazę do zbudowania systemu do analizy sekwencji posłużył układ typu SoC – Zynq XC7Z020 umieszczony na płycie ewaluacyjnej GSC-AES-Z7EV-7Z020-G - ZedBoard firmy Avnet. Układy serii Zynq składają się z dwóch zasadniczych bloków: procesorowego (*Processing System* - PS) oraz logiki programowalnej (*Programmable Logic* - PL). Blok procesorowy zwiera dwa rdzenie ARM Cortex-A9 taktowane zegarem 666 MHz. Blok logiki programowalnej (PL) jest wykonany w technologii 28 nm, odpowiada układom serii Artix-7 i dysponuje 53200 blokami LUT, 106400 przerzutnikami, 140 blokami pamięci BlockRAM (każdy o wielkości 36Kb) oraz 220 blokami DSP.

4. Weryfikacja eksperymentalna

Procedura weryfikacji polegała na zbadaniu poprawności pracy testów poprzez porównanie ich wyników z wynikami dla aplikacji oryginalnej pracującej na PC. Następnie wyznaczono czasy potrzebne do wykonania poszczególnych testów przez procesor dla każdego z trybów pracy aplikacji. Wyniki zamieszczone zostały w tablicy 2, z której wynika, że zastosowanie proponowanych założeń przynosi korzystne rezultaty. W szczególności, jeżeli użytkownika satysfakcjonuje jedynie informacja o nie przekroczeniu wartości krytycznej w pięciu podstawowych testach możliwe jest uzyskanie szybkości analizy na poziomie około 100 Mb/s.

Tab. 2. Czas wykonania poszczególnych testów dla jednej próbki
Tab. 2. Execution time of tests (for a single data sample)

Nazwa testu	Czas realizacji dla jednej próbki [ms]		
	NISTv0	NISTv1	NISTv2
Frequency	3,06	2,45	0,00
Frequency within a Block	1,56	1,25	0,00
Cumulative Sums	5,13	4,10	3,28
Runs	4,06	3,25	0,00
Tests for Longest-Run-of-Ones in a Block	22,31	17,85	0,00
Binary Matrix Rank	142,00	113,60	90,88
Discrete Fourier Transform	877,44	701,95	561,56
Non-overlapping Template Matching	28,94	23,15	18,52
Overlapping Template Matching	90,69	72,55	0,00
Maurer's "Universal Statistical"	837,99	670,39	536,31
Approximate Entropy	352,38	281,90	225,52
Random Excursions	158,00	126,40	101,12
Random Excursions Variant	6,81	5,45	4,36
Serial	5,38	4,30	3,44
Linear Complexity	10,56	8,45	6,76
Czas całkowity	2546,30	2037,04	1551,75

5. Podsumowanie

Przedstawiona została hybrydowa implementacja pakietu testów statystycznych NIST 800-22 w układzie Zynq. Dla pięciu testów opracowane zostały sprzętowe bloki wspomagające analizę sekwencji binarnych. Zaproponowane rozwiązanie umożliwia bieżącą obserwację sekwencji generowanej przez PRBG, dzięki czemu w przypadku wystąpienia nieprawidłowości system może stosownie zareagować. Wprowadzono optymalizację polegającą na pominięciu obliczania złożonych funkcji *erfc* oraz *igmac* przez zastąpienie ich odpowiednimi stałymi wartościami krytycznymi wyznaczonymi dla ich odwrotności. Zliczanie zdarzeń realizowane jest przez dedykowane bloki liczników i komparatorów implementowanych sprzętowo. Dodatkowo ustawienie stałej liczby bitów dla każdej sekwencji pozwoliło na uzyskanie szybkości analizy na poziomie około 100 Mb/s. Przedstawiono opis budowy systemu oraz podano wymagane wielkość zasobów logicznych.

W trakcie dalszych prac planowane jest rozszerzenie zestawu testów realizowanych sprzętowego o kolejne testy z pakietu NIST oraz wprowadzenie możliwości ustalania długości badanej sekwencji n i poziomu ufności α. Rozważana jest też możliwość zwiększenia wydajności mikrosystemu przez analizę więcej niż jednego bitu w każdym cyklu zegarowym.

6. Literatura

[1] Rukhin A., et al.: A statistical test suite for random and pseudorandom number generators for cryptographic applications, NIST Special Publication 800-22, Revision 1a, Aug. 2010.

[2] Marsaglia G.: DIEHARD: a battery of tests of randomness, http://www.stat.fsu.edu/pub/diehard/.

[3] Dąbal P., Pełka R.: Implementacja generatorów cyfrowego chaosu do zastosowań w kryptografii w układzie FPGA, Pomiary Automatyka Kontrola, vol. 56, nr 07/2010, str. 711-713.

[4] Dabal P., Pelka R.: FPGA Implementation of Chaotic Pseudo-Random Bit Generators, in Proc. 19th Int. Conf. Mixed Design of Integrated Circuits and Systems (MIXDES 2012), Maj 2012, Warszawa, str. 260-264.

[5] Dabal P., Pelka R.: An Automated Method for Statistical Testing of FPGA−based Pseudo−Random Generators, Elektronika - Konstrukcje, Technolgie, Zastosowania, vol. 54, no. 2, pp. 58-63, 2013.

[6] Suresh V.B., Antonioli D., Burleson W.P.: On-chip lightweight implementation of reduced NIST randomness test suite, IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), str. 93-98, 2-3 June 2013.

[7] Hoţoleanu D., et al.: Real-Time Testing of True Random Number Generators Through Dynamic Reconfiguration, 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD), str. 247-250, 1-3 Sept. 2010.

otrzymano / received: 09.04.2014
przyjęto do druku / accepted: 02.06.2014 *artykuł recenzowany / revised paper*