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

Mikrosystem z układem Zynq do dystrybucji strumienia danych z chaotycznych generatorów PRBG w sieci LAN

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 dystrybucji strumienia danych z chaotycznych generatorów pseudolosowych (PRBG) w sieci LAN. Opisano implementację kilku wariantów architektur chaotycznych generatorów binarnych sekwencji pseudolosowych. Kompletny system zajmuje 2% przerzutników i 7% bloków LUT dostępnych w układzie XC7Z020. Szybkość transmisji danych w sieci LAN, w zależności od konfiguracji systemu, wynosi od 8,8 Mb/s do 53,4 Mb/s. Opracowano aplikację do badań i wspomagania prac projektowych z wykorzystaniem proponowanego mikrosystemu.

Słowa kluczowe: generatory pseudolosowe, chaos, SoC, FPGA.

A microsystem with Zynq device for distribution of bit-streams from chaotic PRBG generators in LAN

Abstract

This paper presents a concept, design and experimental results of a SoC-based microsystem with Zynq device from *Xilinx*, for distribution of chaotic pseudo-random bit-stream from PRBG via LAN. Several variants of PRBGs architectures have been described and tested. The complete system requires about 2% of flip-flops and 7% of LUTs available in the XC7Z020 device. The maximum speed of data transmission on LAN, depends on the system configuration, and varies from 8.8 Mbps to 53.4 Mbps. A dedicated computer application has been developed to support the research and design with use of the proposed microsystem. Pseudo-random bit-stream generators are used e.g. in cryptography and for testing digital systems. Often there is a need for high-speed transmission of data streams to multiple recipients at the same time. The described system supports the distribution of data obtained from embedded PRBGs over the LAN. In order to manage the distribution process, a dedicated client-server has been proposed. The hardware platform and objectives of the system for generation and distribution of pseudo-random sequences are discussed. There are presented the main features of the tools used for development of the project, the software and the library of utility modules that can be used in dedicated user applications.

Keywords: pseudo-random generators, chaos, SoC, FPGA.

1. Wstęp

Jednym z nowych kierunków rozwoju techniki cyfrowej jest połączenie w jednym układzie scalonym sprzętowego procesora z logiką programowalną. Dotychczas były to rdzenie, które wymagały implementowania w obrębie logiki programowalnej magistral, kontrolerów, oraz interfejsów. Układy te były do niedawna oferowane przez firmę *Xilinx* w wersji szybkiej (Virtex 4FX, Virtex 5FXT), co wiązało się ze znacznym kosztem opartych na nich konstrukcji. Pojawienie się nowej rodziny układów Zynq [1], łączącej system mikroprocesorowy z logiką programowalną, przyczyniło się do powstania nowych możliwości projektowych. Podobne układy oferuje również *Altera* w rodzinach Arria V i Cyclone V.

W opisywanym projekcie opracowano niezbędne elementy funkcjonalne umożliwiające zastosowanie generatorów chaotycznych przedstawionych wcześniej w pracy [2] w złożonych systemach cyfrowych. Zbadano możliwość ich praktycznego użycia do dystrybucji danych pseudolosowych w sieci LAN. Określono wielkość wymaganych zasobów układu SoC FPGA oraz porównano ją z analogiczną implementacją w starszych układach Virtex-5. Jest to próba innego podejścia do zagadnienia praktycznej implementacji PRBG w odniesieniu do rozwiązania przedstawionego w [3].

Artykuł jest zorganizowany w następujący sposób. W sekcji 2 przedstawiona została idea praktycznego użycia chaotycznych PRBG wbudowanych w układzie Zynq jako źródła strumieni danych pseudolosowych w sieci LAN. Sekcja 3 przedstawia sposób implementacji, oprogramowanie oraz użyte narzędzia. Rozdział 4 podaje wyniki pomiarów i ich dyskusję. Na koniec, w rozdziale 5 przedstawione zostały wnioski.

2. Dystrybutor sekwencji pseudolosowych

Dotychczasowe prace badawcze opisane w [2, 4, 5] dotyczyły analizy różnych konfiguracji generatorów chaotycznych implementowanych w układach programowalnych. Do testowania tych generatorów powstał system przedstawiony w [6]. W niniejszej pracy opisano jego rozbudowaną, uniwersalną wersję przeznaczoną do wykorzystania w złożonych systemach cyfrowych, opartą na płycie prototypowej ZedBoard z układem Zynq 7020 [7].

2.1. Platforma sprzętowa z układem Zynq

Jako platformę do realizacji części sprzętowej wybrany został jeden z najnowszych produktów firmy Xilinx, układ typu SoC – Zynq XC7Z020 umieszczony na płycie ewaluacyjnej GSC-AES-Z7EV-7Z020-G - ZedBoard firmy Avnet. Dysponuje ona 512MB pamięci DDR3, 32MB pamięci Flash Quad-SPI, gniazdem karty SD, zintegrowanym interfejsem do programowania JTAG, kontrolerem 10/100/1000 Ethernet, kontrolerem USB OTG 2.0, konwerterem USB-UART, kodekiem audio, kodekiem HDMI, 8-bitowym wyjściem VGA oraz zestawem przycisków, przełączników i diod LED. Źródłem sygnału zegarowego są dwa oscylatory 33,333 MHz (dla podsystemu procesorowego) oraz 100MHz (dla logiki programowalnej) [7]. Układy serii Zynq złożone są z dwóch bloków: procesorowego (Processing System - PS) oraz logiki programowalnej (Programmable Logic - PL) [1].

Blok procesorowy zwiera dwa rdzenie ARM Cortex-A9 taktowane zegarem 666 MHz, wbudowane kontrolery pamięci dynamicznych DDR2/3 oraz statycznych SRAM/Flash, podwójne kontrolery interfejsów Ethernet MAC, USB 2.0, CAN 2.0B, SD/SDIO, SPI, UART oraz I2C, wbudowaną pamięć: BootROM, 256kB RAM, cache L1 i L2. Możliwa jest praca w konfiguracji

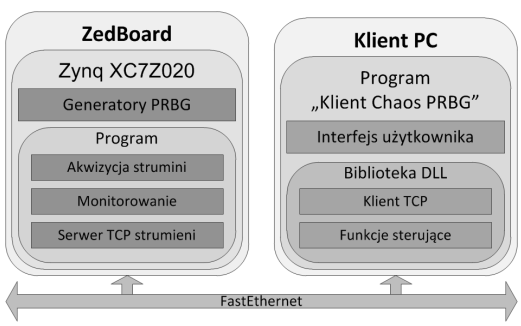
jedno- lub dwurdzeniowej (symetrycznej – z jednym programem lub asymetrycznej – gdy każdy rdzeń realizuje własny program). Rdzenie dysponują własnymi jednostkami zmiennoprzecinkowymi oraz dedykowanymi blokami przetwarzania danych multimedialnych (NEON). Tak bogate wyposażenie umożliwia uruchomienie na tej platformie systemów operacyjnych takich jak Linux czy Android. Blok logiki programowalnej jest analogiczny jak w pozostałych układach serii 7 firmy Xilinx. Komunikacja z blokiem procesorowym odbywać się może za pomocą klasycznego portu GPIO lub dedykowanej magistrali Advanced eXtensible Interface 4 (AXI4).

Wbudowana pamięć BootROM umożliwia w zależności od stanu dedykowanych wejść układu, konfigurację systemu z jednej z wybranych pamięci: QuadSPI, Flash lub karty SD. Konfigurację można także wykonać za pomocą interfejsu JTAG, który ma najwyższy priorytet.

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), 220 blokami DSP oraz 8 blokami zarządzania sygnałem zegarowym (DCM & PLL).

2.2. Koncepcja mikrosystemu

System do dystrybucji strumieni danych pseudolosowych składa się z kilku komponentów. Pierwszym z nich są chaotyczne generatory PRBG, dla których można ustawić wartość początkową (*seed*) determinującą generowaną sekwencję. Uzupełnione są one o interfejs komunikacyjny z magistralą AXI-Lite, która za pomocą zestawu rejestrów pośredniczy w komunikacji procesor-generator. Kolejnym elementem jest skonfigurowany system mikroprocesorowy odpowiedzialny za akwizycję sekwencji binarnych oraz obsługę komunikacji za pomocą interfejsu Ethernet. Wybrana platforma umożliwia uruchomienie aplikacji bezpośrednio na jednym z rdzeni procesora lub w systemie Linux. Ostatnim elementem jest aplikacja kliencka uruchamiana na komputerze PC. Umożliwia ona pobranie określonej ilości danych z wybranego generatora dostępnego w platformie dystrybucyjnej i zapisanie ich do pliku na komputerze klienta. Rysunek 1 przedstawia diagram blokowy proponowanego mikrosystemu.



Rys. 1. Struktura mikrosystemu do dystrybucji strumieni danych pseudolosowych
Fig. 1. Block diagram of the microsystem for distribution of pseudo-random data

3. Implementacja mikrosystemu

Projekt mikrosystemu składa się z trzech części, które są szczegółowo opisane poniżej.

3.1. Projekt systemu wbudowanego

W bloku procesorowym skonfigurowano jeden kontroler Ethernet, jeden interfejs UART, kontrolery pamięci zewnętrznych, blok generacji zegara oraz licznik. Port szeregowy służy do monitorowania pracy programu, natomiast licznik jest przeznaczony do pomiaru wydajności akwizycji i transmisji. Za pośrednictwem magistrali AXI-Lite z kolejką FIFO o pojemności 16 słów podłączono trzy typy generatorów opisanych w [4]: z odwzorowaniem logistycznym, Hénona oraz z oscylatorem FDNR. Każdy jest dostępny w dwóch wariantach precyzji: 32- i 64-bitowej. Oprócz kolejki FIFO zaimplementowany został również zestaw rejestrów: ustawienia wartości początkowej (od 1 do 6 rejestrów), ustawienia parametru odwzorowania (od 1 do 6 rejestrów) oraz sterujący. Ich liczba zależy od typu i precyzji obliczeniowej generatora. Dla każdego z typów generatorów w bloku zegarowym PS wytwarzane są 3 sygnały zegarowe: 25 MHz (Hénon), 100 MHz (logistyczny) i 150 MHz (FDNR). Częstotliwości te określono w wyniku badań opisanych w [6].

W tablicy 1 podano zestawienie zasobów logicznych używanych przez poszczególne komponenty mikrosystemu oraz implementację analogicznego rozwiązania w układzie Virtex 5FXT z procesorem PowerPC. Liczba zasobów potrzebnych do implementacji generatorów jest podobna, a nawet identyczna w przypadku przerzutników i bloków DSP. Cały system w starszym rozwiązaniu wymaga około dwukrotnie więcej zasobów, ponieważ kontrolery pamięci oraz interfejsy są implementowane w logice programowalnej.

Tab. 1. Zasoby wymagane przez system i generatory
Tab. 1. Resources used by the system and generators

Układ	**Zynq XC7Z020**			**Virtex 5FXT**		
Rodzaj zasobu	*Flip-flop*	*LUT*	*DSP*	*Flip-flop*	*LUT*	*DSP*
Dostępne	106 400	53 200	220	20 480	20 480	64
Cały system	2 124	3 751	38	6 834	5 386	38
Logistic @ 32b	32	62	4	32	64	4
Logistic @ 64b	64	372	14	64	368	14
Hénon @ 32b	32	555	4	32	556	4
Hénon @ 64b	64	1 632	16	64	1 636	16
FDNR @ 32b	96	223	0	96	256	0
FDNR @ 64b	192	427	0	192	512	0

Za konfigurację mikrosystemu czyli uruchomienie oprogramowania i zaprogramowanie PL odpowiada wbudowany program ładujący (*bootloader*), który poszukuje w pamięci QuadSPI programu dla procesora oraz pliku konfiguracyjnego.

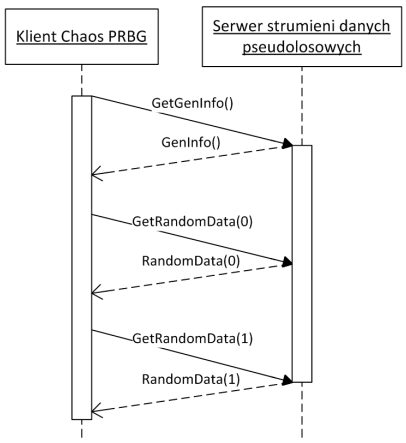
3.2. Projekt oprogramowania mikrosystemu

Program jest wykonywany niezależnie na każdym rdzeniu lub w konfiguracji dwurdzeniowej. W pierwszym przypadku aplikacja pracuje bez systemu operacyjnego, w drugim zaś z OS (Linux, RTOS). W tym projekcie zdecydowano się na pracę bez OS ze względu na możliwość wykorzystania wcześniej utworzonych fragmentów programu dla procesora MicroBlaze.

Opracowano program, który składa się z trzech modułów: (1) akwizycji próbek z wybranego generatora, (2) asynchronicznego serwera próbek pracującego w oparciu o protokół warstwy sesji modelu ISO/OSI, (3) funkcji monitorujących poprawność pracy generatorów.

Generatory od strony procesora widoczne są jako bank od 3 do 13 rejestrów 32-bitowych. Liczba rejestrów zależy od użytego generatora i jego precyzji. Moduł akwizycji to zestaw funkcji, które konfigurują generator, umożliwiają odczytanie określonej porcji danych i zapisanie ustawień. Podprogram ten pracuje w pętli głównej programu.

Procedura startowa uruchamia podprogram serwera TCP/IP realizujący zapytania od aplikacji klienckich, który obsługuje dedykowany protokół opisany w [6]. W szczególności serwer obsługuje trzy rodzaje zapytań: (1) dostępne generatory (GetGenInfo), (2) ustaw generator (SetGenCfg) oraz (3) udostępnij strumień pseudolosowy (GetRandomData). W pierwszym przypadku klient jest informowany o dostępnych generatorach oraz ich konfiguracji. Drugie zapytanie jest żądaniem ustawienia wybranego generatora i przygotowaniem go do pracy. Trzecie natomiast żąda przesłania określonego strumienia danych. Moduł ten pracuje asynchronicznie z przerwaniem wywoływanym w obsłudze stosu TCP/IP. Rys. 2 ilustruje przebieg komunikacji między klientem a serwerem.



Rys. 2. Diagram wymiany komunikatów między aplikacją klienta i serwerem
Fig. 2. Diagram of communication client-server

Ostatni moduł jest odpowiedzialny za monitorowanie statystycznych własności generowanych strumieni. Jest to para testów. Pierwszy z nich polega na sprawdzeniu, czy liczba występujących 1 i 0 jest zbliżona. Drugi zlicza liczbę przejść 0→1 oraz 1→0. Jest to tzw. test częstości dający podstawę do stwierdzenia wystąpienia ewentualnych nieprawidłowości w pracy generatora.

3.3. Aplikacja klienta

Komputery w sieci LAN mogą korzystać z dostępu do strumienia danych z PRBG za pomocą aplikacja napisanej w języku C#. Składa się ona z pliku wykonywalnego oraz pliku biblioteki. Podejście takie zostało wybrane aby umożliwić użycie systemu dystrybucyjnego w innych zastosowaniach niż prezentowane w tym artykule.

Biblioteka zawiera zestaw funkcji odpowiedzialnych za komunikację z aplikacją do akwizycji danych w układzie Zynq, z których korzysta program. Obsługuje ona dedykowany protokół działający w oparciu o stos TCP/IP.

Dostęp do generatora zapewnia interfejs aplikacji, który umożliwia: połączenie z układem o znanym adresie IP i numerze portu TCP, wybór dostępnego strumienia, konfigurację generatora, ustalenie rozmiaru danych jakie można pobrać i zapisanie otrzymanych danych do pliku. Alternatywnie aplikacja może zostać uruchomiona przez podanie listy argumentów bez uruchamiania interfejsu.

3.4. Metoda projektowa i użyte narzędzia

Podczas projektowania systemu zastosowano szereg narzędzi. Pierwszym z nich jest pakiet oprogramowania firmy Xilinx ISE/XPS, w wersji 14.4, który posłużył do opisu systemu wbudowanego implementowanego w układzie programowalnym. Generatory PRBG zostały opracowane w środowisku Matlab 2011a z dodatkiem System Generator firmy Xilinx, a następnie wyeksportowane do środowiska ISE, w którym za pośrednictwem opracowanego modułu kontrolera magistrali AXI zostały podłączone do magistrali systemowej. Implementację sprzętową zrealizowano z wykorzystaniem płyty ZedBoard [7].

Oprogramowanie dla mikrosystemu zostało napisane w języku C przy użyciu środowiska Xilinx SDK. Aplikacja klienta oraz biblioteka zostały napisane w języku C#, w środowisku Visual Studio 2012. Do uruchomienia aplikacji klienta niezbędne jest aby na komputerze zainstalowany był pakiet .NET Framework 4.0.

4. Weryfikacja eksperymentalna

Weryfikacja poprawności pracy polegała na zbadaniu możliwości zdalnego pozyskania określonego strumienia danych i zapisania go do pliku. Zrealizowano dwa scenariusze testowe. Pierwszy z nich polegał na zestawieniu jednego połączenia przez aplikację kliencką z mikrosystemem oraz utworzeniu 6 plików o wielkości 1024 Mb (jeden plik dla każdego z generatorów). Zmierzono średnią szybkość transmisji, która wyniosła 53,4 Mb/s. Drugi scenariusz polegał na jednoczesnej obsłudze 6 klientów, z których każdy żądał strumienia z innego generatora. W tym przypadku średnia szybkość transmisji dla jednego klienta wyniosła 8,8 Mb/s. Spowolnienie transmisji wynika z faktu, że dane wysyłane są porcjami 1024 B kolejno do każdego klienta. Alternatywnie można rozważyć opcję transmisji bez jej przerywania, jednak może to powodować wydłużenie czasu oczekiwania przez kolejnych klientów.

5. Podsumowanie

W projekcie przedstawiono zintegrowany mikrosystem z układem Zynq umożliwiający wykorzystanie generatorów chaotycznych PRBG jako źródła strumienia danych pseudolosowych dystrybuowanych w sieci LAN. Zweryfikowano poprawność pracy systemu oraz określono jego potencjalną wydajność dla dwóch scenariuszy działania. Przedstawiono opis budowy systemu oraz podano wymagane wielkość zasobów logicznych.

W trakcie dalszych prac planowane jest rozszerzenie mikrosystemu o sprzętowy układ monitorowania ciągów pseudolosowych przez badanie entropii. Podobną metodę wykorzystuje się do badania losowości generatorów TRNG (True Random Bit Generator). Rozważane jest także użycie innych generatorów, które mogą być implementowane w układach FPGA, np. opisanych w pracach [8, 9], oraz wprowadzenie metod post-processingu [10], w celu poprawy jakości statystycznej i zwiększenia szybkości generacji strumienia danych.

6. Literatura

[1] Xilinx: Zynq-7000 All Programmable SoC Overview, DS 190, v. 1.2, 21 August, 2012, on-line: http://www.xilinx.com.
[2] 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.
[3] Dabal P., Pelka R.: FPGA-based cryptosystem with combined stream-block cipher and digital chaos generator, in Proc. of Int. Conf. on Signals and Electronic Systems (ICSES), 7-10 Sept. 2010, Gliwice, pp. 315-318, (on-line: IEEE Xplore).
[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), May 2012, Warszawa, pp. 260-264, (on-line: IEEE Xplore).
[5] Dabal P., Pelka R., A Chaos-Based Pseudo-Random Bit Generator Implemented in FPGA Device, in Proc. 14th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, 13-15 April 2011, Cottbus, Germany, pp. 151-154, (on-line: IEEE Xplore).
[6] Dabal P., Pelka R.: An Integrated System for Statistical Testing of Pseudo-Random Generators in FPGA Devices, in Proc. Int. Conf. on Signals and Electronic Systems (ICSES'2012), 18-21 September 2012, Wrocław, pp.1-5, (on-line: IEEE Xplore).
[7] ZedBoard: Zynq™ Evaluation and Development. Hardware User's Guide, ver. 1.9, Jan. 2013, (on-line: www.zedboard.org).
[8] Fischer V., Drutarovsky M.: True Random Number Generator Embedded in Reconfigurable Hardware, in Proc. 4th Int. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002), Springer-Verlag, LNCS 2523, 2002.
[9] Wold K., Chik How Tan., Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings, Int. Journal of Reconfigurable Computing, vol. 2009, Article ID 501672, 2009.
[10] Addabbo T., Alioto M., Fort A., Rocchi S., Vignoli V.: Efficient Post-Processing Module for a Chaos-based Random Bit Generator, in Proc. 13th IEEE Int. Conf. on Electronics, Circuits and Systems, ICECS '06, 10-13 Dec. 2006, pp. 1224-1227.

otrzymano / received: 21.05.2013
przyjęto do druku / accepted: 03.07.2013 *artykuł recenzowany / revised paper*