Paweł DĄBAL, Ryszard PEŁKA
WOJSKOWA AKADEMIA TECHNICZNA, WYDZIAŁ ELEKTRONIKI,
INSTYTUT TELEKOMUNIKACJI, ZAKŁAD TECHNIKI CYFROWEJ

Implementacja algorytmu szyfrującego Rijndael (AES) w układzie FPGA Virtex 4FX

Paweł DĄBAL

Jest studentem V roku Wydziału Elektroniki Wojskowej Akademii Technicznej, gdzie studiuje na specjalności Systemy Cyfrowe w trybie indywidualnym. Jego zainteresowania naukowe dotyczą projektowania układów i systemów cyfrowych z wykorzystaniem struktur programowalnych FPGA.

e-mail: paweldabal@gmail.com

Prof. dr hab. inż. Ryszard PEŁKA

Ukończył studia na Wydziale Elektroniki Politechniki Warszawskiej, obronił pracę doktorską w 1984 roku w Wojskowej Akademii Technicznej. W roku 2004 otrzymał tytuł profesora. Obecnie pełni funkcję kierownika Zakładu Techniki Cyfrowej w Instytucie Telekomunikacji w WAT. Jego zainteresowania naukowe dotyczą metod projektowania, optymalizacji i testowania systemów cyfrowych z układami FPGA i SoC.

e-mail: rpelka@wel.wat.edu.pl

Streszczenie

W artykule przedstawiono budowę, działanie i wyniki badań eksperymentalnych bloku IP-core, który może równolegle szyfrować/ deszyfrować dwa strumienie danych przy użyciu algorytmu Rijndael ze 128-bitowym kluczem, dostarczanych za pośrednictwem magistrali Processor Local Bus (PLB). Podany został kompletny opis systemu składającego się z procesora MicroBlaze oraz podłączonego do niego IP-core. Dokonano pomiarów szybkości przetwarzania w zależności od wybranego trybu pracy.

Słowa kluczowe: szyfrowanie, algorytm Rijndael, FPGA.

Implementation of the ciphering algorithm Rijndael (AES) in Virtex 4FX FPGA device

Abstract

The paper presents design, principle of operation and experimental results of a dedicated IP-core developed for parallel data encryption/decription of two data streams provided by the Processor Local Bus (PLB). The encryption process is based on the standardized Rijndael algorithm with an 128-bit encryption key. The algorithm is performed by two cooperating with each other PicoBlaze processors, with extended internal RAM and shared 2kB ROM. An architecture of IP-core block is shown in Fig. 2. The extended RAM stores the generated sub-keys for consecutive rounds. Using the substitution tables stored in ROM it is possible to achieve a uniform speed of data encryption and decryption. There is also proposed a special operating mode that changes the encryption key when a single data stream is processed. The detailed description of the complete digital system consisting of the IP-core and MicroBlaze processor is given. The experimental results of data encryption throughput are also presented. The comparison with similar solutions reported by other authors is discussed.

Keywords: data encryption, Rijndael algorithm, FPGA.

1. Wstęp

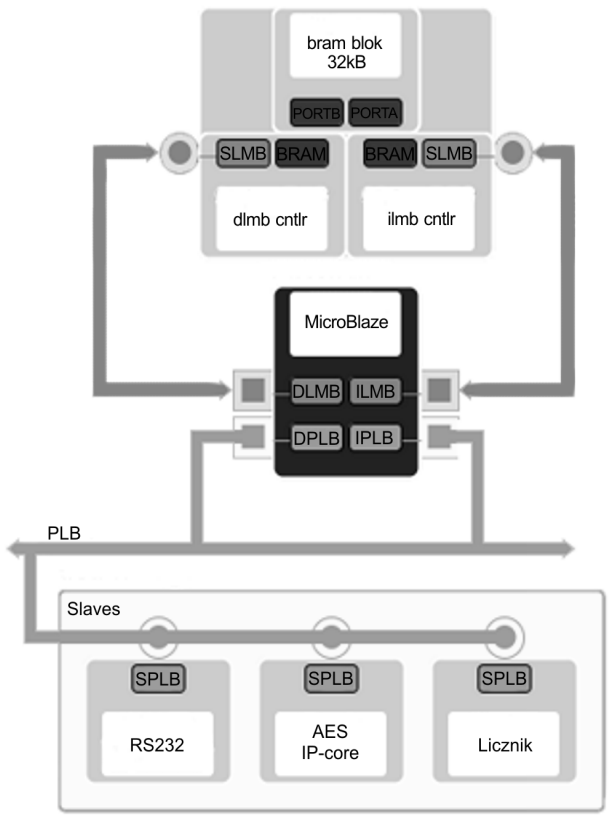
Współczesny świat wymusza na projektantach konieczność implementacji rozbudowanych algorytmów szyfrujących, które wymagają dużej mocy obliczeniowej i pamięci. Wyróżnić można trzy podstawowe sposoby implementacji tych algorytmów. Pierwszy, w pełni sprzętowy jest oparty o dedykowane układy ASIC. W drugim wariancie algorytm jest zapisany w postaci programu wykonywanego przez mikroprocesor. Trzecie, mieszane rozwiązanie, stosuje algorytm podzielony na elementy wykonywane programowo i sprzętowo, które można implementować między innymi z wykorzystaniem struktur programowalnych FPGA.

Jako algorytm szyfrujący do zaproponowanego rozwiązania wybrany został Rijndael [1], znany również pod nazwą AES. Zaimplementowana została odmiana z kluczem o długości 128 bitów. W pracy [2] przedstawione zostały rezultaty sprzętowej i programowej realizacji kilku wybranych algorytmów szyfrujących (w tym AES) realizowanych przez układ FPGA z procesorem MicroBlaze.

Podstawą do opracowania opisanego w artykule bloku IP-core była wiedza uzyskana podczas prac przedstawionych w [3], gdzie wykazana została możliwość wykorzystania procesora PicoBlaze do realizacji algorytmu AES. Celem niniejszej publikacji jest przedstawienie wydajności proponowanego bloku IP-core i porównanie jej parametrów z innymi rozwiązaniami.

2. System szyfrujący z dedykowanym IP-core

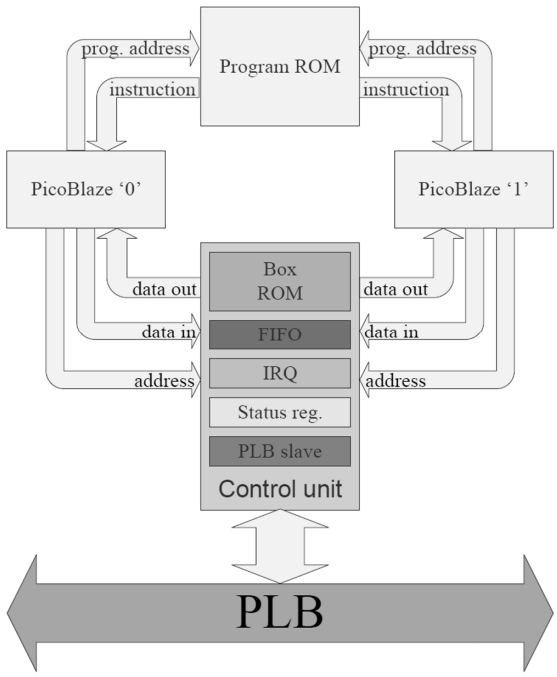
W celu zaprojektowania systemu posłużono się środowiskiem EDK, za pomocą którego zbudowano bazowy system mikroprocesorowy w oparciu o procesor MicroBlaze [4], do którego podłączone zostało 32kB pamięci BRAM przechowującej dane oraz program, magistrala PLB [5], do której podłączony został interfejs szeregowy RS232, 32-bitowy licznik oraz blok IP-core. Na rysunku 1 przedstawiony został diagram blokowy systemu.



Rys. 1. Diagram blokowy systemu do szyfrowania/deszyfrowania danych
Fig. 1. Block diagram of the data encryption/decryption system

2.1. Budowa bloku IP-core

Proponowane rozwiązanie jako podstawowy element wykonawczy wykorzystuje dwa 8-bitowe procesory PicoBlaze [6]. Dostęp do plików źródłowych umożliwił wprowadzenie modyfikacji polegającej na rozszerzeniu wbudowanej pamięci RAM o rozmiarze 64 bajtów do wielkości 256 bajtów. Wykorzystują one współdzieloną pamięć programu. W tym celu do jej implementacji wykorzystana została pamięć dwuportowa o organizacji 1024 x 18b. Procesory współdzielą również między sobą dodatkową pamięć ROM o rozmiarze 2048 bajtów, zawierającą tablice przekształceń wykorzystywanych w operacjach szyfrowania i deszyfrowania. Do komunikacji między procesorowej służą dwie kolejki FIFO, każda o wielkości 16 bajtów. Do sygnalizacji stanu kolejek służą specjalne rejestry statusu zawierające flagi określające zawartości pamięci FIFO, oraz wyzwalające przerwania. Szczegółowy opis tego rozwiązania podany został w [3]. Komunikacja z otoczeniem odbywa się poprzez uniwersalną magistrale PLB [5]. IP-core jest urządzeniem typu *slave*, reprezentowanym jako dziesięć 32-bitowych rejestrów, do których dostęp ma układ *master*. Do stworzenia odpowiedniej logiki wymaganej przez magistrale, wykorzystany został dostępny kreator w środowisku EDK. Elementem wspomagającym transmisje miedzy układami jest kontroler przerwań umożliwiający zrezygnowanie z programowego sprawdzania rejestrów sterujących. Na rysunku 2 przedstawiona została zaproponowana architektura bloku IP-core.



Rys. 2. Architektura bloku IP-core
Fig. 2. Architecture of the IP-core

Z racji 8-bitowej architektury procesorów PicoBlaze konieczne jest multipleksowanie szyny adresowej podłączonej do pamięci ROM. Obszar 2kB wymaga 11 linii adresowych w celu zaadresowania komórki pamięci. Dla pierwszych 8 bitów adresu zastosowano rejestr, do którego poprzez magistralę danych (out_data) zapisywana była młodsza część adresu. Bardziej znaczące, pozostałe 3 linie, były podłączone bezpośrednio do pamięci (port_id).

W przypadku rejestrów magistrali PLB do ich odczytu konieczne było stworzenie dwóch multiplekserów, po jednym dla każdego PicoBlaze, sterowanego 5 liniami adresowymi. Zastosowanie prostego opisu w języku VHDL z użyciem polecenia *case* dało w wyniku syntezy złożony układ kombinacyjny o dużym opóźnieniu. Zastosowanie opisu niskopoziomowego bazującego na dostępnych elementach logicznych struktury wpłynęło na zmniejszenie wymaganej ilości zasobów.

Tabela 1 zawiera zestawienie ilości wymaganych zasobów logicznych dla poszczególnych elementów systemu oraz ich procentowego udziału w projekcie.

Tab. 1. Zestawienie ilości zajmowanych zasobów przez system mikroprocesorowy
Tab. 1. Resources required by the microprocessor system

Elementy	Typ zasobów			
	FPGA Slice	Flip-Flop	LUT	Block RAM
MicroBlaze	1452	1311	1977	0
Pamięć	8	6	14	16
Licznik	252	294	303	0
RS232	108	147	136	0
PLB master	218	155	384	0
Inne	45	73	53	0
IP-core	656	739	1241	2
2 x PicoBlaze	312	152	586	0
Pamięć prog.	0	0	0	1
Pamięć ROM	13	18	4	1
2 x FIFO	21	14	34	0
PLB slave	310	555	617	0
System	2739	2725	4108	18

Widoczne jest, że zastosowanie zaawansowanej magistrali jaką jest PLB powoduje zaangażowanie zasobów porównywalne do tego, jakie potrzebne jest do budowy elementów właściwego układu realizującego szyfrowanie.

Testy przeprowadzono z użyciem płytki uruchomieniowej firmy AVNET ADS-XLX-V4FX-EVL12-G, zbudowanej w oparciu o układ FPGA Virtex 4 FX12 firmy Xilinx. Do układu doprowadzono sygnał zegarowy z oscylatora o częstotliwości 100 MHz.

2.2. Oprogramowanie bloku IP-core

Program przygotowany dla procesorów PicoBlaze został tak zaprojektowany, aby mógł być współdzielony między układy i aby możliwe było jego niezależne wykonywanie. W tym celu, po restarcie systemu procesory wczytują identyfikator, który podczas operacji na flagach statusu umożliwia komunikację międzyukładową. Kolejny krok to zapisanie do rejestru magistrali PLB informacji o statusie. Od tego momentu IP-core jest gotowy do pracy w jednym z trzech trybów.

Pierwszy i drugi tryb polegają odpowiednio na szyfrowaniu/deszyfrowaniu danych dostarczanych poprzez magistralę PLB z wcześniej ustalonym kluczem. Układy pracują w tych trybach w pełni niezależnie. Trzeci tryb polega na zaszyfrowaniu/odszyfrowaniu ciągu danych za pomocą pierwszego klucza, a następnie przekazaniu ich do drugiego procesora gdzie mogą zostać zaszyfrowane/odszyfrowane z użyciem drugiego klucza.

W celu ułatwienia wykorzystania możliwości oferowanych przez IP-core opracowane zostały w języku C odpowiednie sterowniki, które w sposób przejrzysty udostępniają programowi głównemu wybrany tryb pracy. Konieczne było uzyskanie kompromisu pomiędzy funkcjonalnością, a opóźnieniem wprowadzanym przez nadmiarowe fragmenty kodu (sterowanie przepływem, kontrola błędów, dopełnienia).

Jednym z kluczowych elementów wpływających na szybkość przetwarzania jest dostęp właściwego algorytmu do podkluczy dla poszczególnych rund wygenerowanych w procedurze *KeyExpansion*. Przechowywanie ich wymaga dostępu do 176 bajtów pamięci RAM, co w połączeniu z blokiem 16 bajtów danych trzykrotnie przewyższa dostępny obszar pamięci procesora PicoBlaze. Wymusza to generowanie kolejnych podkluczy w miejsce starych

[3]. Zwiększenie pojemności wbudowanej pamięci do 256 bajtów umożliwia zapisanie wszystkich podkluczy, dzięki czemu nie jest konieczne powtarzanie dla każdego bloku danych tej procedury.

Jak wykazane zostało w [3] zastosowanie dodatkowej pamięci ROM z tablicą S-Box o wielkości 256 bajtów, powoduje ponad 200-krotny wzrost szybkości szyfrowania. W proponowanym rozwiązaniu pamięć rozszerzona zostaje o dodatkowe osiem 256 bajtowych tablic, z których trzy: *SBox, Xtime2SBox* i *Xtime3SBox* wspierają procedurę szyfrowania, natomiast *ISBox, Xtime9ISBox, XtimeBISBox, XtimeDISBox* oraz *XtimeEISBox* wspomagają deszyfrowanie. Wzorując się na [7] opracowano w języku asemblera procedury szyfrowania, deszyfrowania i rozszerzania klucza.

Ostatnim elementem programu jest procedura obsługi przerwania, zidentyfikowanie źródła oraz podjęcie odpowiednich operacji w zależności od zawartości zewnętrznych rejestrów sterujących.

2.3. Metoda projektowa i wykorzystane narzędzia

W projektowaniu systemu wykorzystany został szereg narzędzi programistycznych. Pierwszym z nich jest ISE 10.1.3, przy użyciu którego zbudowano opis z wykorzystaniem języka VHDL oraz dokonano weryfikacji funkcjonalnej IP-core. Środowisko EDK 10.1.3 posłużyło do stworzenia platformy testowej. Dodatkowo posiada ono moduł wspomagający tworzenie oprogramowania (SDK), który posłużył do napisania aplikacji testującej oraz sterowników IP-core. Program dla procesorów PicoBlaze powstał z użyciem środowiska pBlazeIDE 3.7.4, które poza kompilatorem asemblera, posiada rozbudowany debuger.

3. Metodologia pomiarowa

Do pomiaru szybkości przetwarzania w każdym z trybów pracy opracowany został wektor testowy o długości 1024 bajtów zawierający dane do szyfrowania, deszyfrowania i zamiany klucza oraz dwa 128-bitowe klucze. Dane te są przechowywane w pamięci programu mikroprocesora MicroBlaze. Mierzono liczbę cykli zegarowych potrzebnych do wykonania pełnego przetworzenia wektora, dla określonego trybu pracy, przy użyciu 32-bitowego licznika podłączonego do magistrali PLB. Odczytana liczba cykli była następnie przeliczona na efektywną przepustowość wyrażoną w bitach na sekundę [b/s], która za pośrednictwem interfejsu szeregowego RS232 była przesyłana do komputera.

Porównano wydajność następujących konfiguracji:

MB-A – algorytm realizowany w całości przez procesor Micro Blaze z włączonymi opcjami do przetwarzania liczb całkowitych (układ wspomagający mnożenie, dzielenie, przesuwanie bitowe);

MB-B – konfiguracja sprzętowa procesora jak w MB-A z wykorzystaniem IP-core (PicoBlaze z 64B RAM);

MB-C – konfiguracja sprzętowa procesora jak w MB-A z wykorzystaniem IP-core (PicoBlaze z 256B RAM).

4. Wyniki badań

Wyniki badań zaproponowanych konfiguracji przedstawione zostały w tabeli 2. Okazuje się, że wydajność rozwiązania bazującego na procesorach PicoBlaze jest większa od oferowanego przez rozbudowany procesor jakim jest MicroBlaze. Szybkość realizowania deszyfrowania zależy silnie od tego, czy stosujemy wcześniej przygotowane podklucze. Jest ona zawsze mniejsza od szybkości procedury szyfrowania.

Porównanie z innymi rozwiązaniami zorientowanymi na niewielkie wymagania zasobów logicznych zostało przedstawione w tabeli 3.

Tab. 2. Szybkość przetwarzania dla różnych trybów pracy i konfiguracji
Tab. 2. Speed of data processing for different operating modes and configurations

Konfiguracja	Tryb I		Tryb II		Tryb III	
	Liczba cykli/blok	b/s	Liczba cykli/blok	b/s	Liczba cykli/blok	b/s
MB-A	21 196	603888	53 410	239655	74 862	170981
MB-B	11 248	2275960	33 482	764589	44 986	284533
MB-C	8 964	2855868	9 044	2830606	9 300	1376344

Tab. 3. Wydajność szyfrowania w porównaniu z innymi rozwiązaniami
Tab. 3. Encryption throughput compared to other solutions

Procesor	Przepustowość [Mbps]	Liczba bloków slice
IP-core @ 100MHz	2,85	656
Chodowiec & Gaj @ 60MHz [9]	69	222
ASIP @ 72,3MHz [8]	2,18	122
PicoBlaze @ 100MHz [3]	1,12	149
MicroBlaze @ 100MHz	0,6	1385

5. Wnioski

Dokonano porównania wydajności zaproponowanego IP-core z innymi znanymi implementacjami realizującymi algorytm Rijndael. Proponowane rozwiązanie zapewnia wsparcie w realizacji procesu szyfrowania, deszyfrowania oraz zamiany kluczy dla procesorów ogólnego przeznaczenia, do których strumień danych doprowadzony jest za pośrednictwem magistrali PLB. Potencjalne zastosowanie przedstawionego układu obejmuje zintegrowane systemy cyfrowe, w których konieczne jest wdrożenie algorytmów szyfrujących przy niewielkiej ilości wymaganych zasobów. Istotną zaletą przedstawionego rozwiązania jest elastyczność konfiguracji umożliwiająca dostosowanie układu do indywidualnych potrzeb.

6. Literatura

[1] J. Daemen, V. Rijmen: The Design of Rijndael, AES - The Advanced Encryption Standard, Springer-Verlag, 2002.

[2] I. Gonzalez, F. J. Gomez-Arribas: Ciphering algorithms in MicroBlazebased embedded systems, IEE Proc.-Comput. Digit. Tech., Vol. 153, No. 2, March 2006, pp. 87-92.

[3] P. Dąbal, R. Pełka: Implementacja algorytmu szyfrującego AES-128 w układzie FPGA Spartan 3E z procesorami PicoBlaze. Pomiary, Automatyka, Kontrola, Vol. 54, no 8/2008, pp. 520-522.

[4] Xilinx Inc.: UG 081 - MicroBlaze Processor Reference Guide, January 2008. http://www.xilinx.com

[5] Xilinx Inc.: DS 531 - Processor Local Bus (PLB) v4.6, June 2008. http://www.xilinx.com

[6] Xilinx Inc.: UG 129 - PicoBlaze 8-bit Embedded Microcontroller User Guide, June 2008. http://www.xilinx.com

[7] K. Malbrain: A byte oriented higher-performance AES http://www.geocities.com/malbrain/aestable2_c.html

[8] T.Good, M.Benaissa: Very Small FPGA Application-Specific Instruction Processor for AES, IEEE Ttransactions on circuits and systems, vol. 53, no. 7, July 2006.

[9] P. Chodowiec, K. Gaj: Very Compact FPGA Implementation of the AES Algorithm," in Proc. LNCS'03, 2003, vol. 2779, pp. 319–333.

Artykuł recenzowany