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

Implementacja algorytmu szyfrującego AES-128 w układzie FPGA Spartan 3E z procesorami PicoBlaze

Paweł DĄBAL

Jest studentem IV roku Wydziału Elektroniki Wojskowej Akademii Technicznej, gdzie studiuje na specjalności Systemy Cyfrowe w trybie indywidualnym. Jego zainteresowania naukowe dotyczą projektowanie 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 wyniki badań dotyczących implementacji algorytmu szyfrującego AES-128 (Advanced Encryption Standard) w układzie FPGA (Field Programmable Gate Array) Spartan 3E. Podano opis kompletnego systemu cyfrowego, który umożliwia odbiór strumienia danych przez złącze szeregowe i ich szyfrowanie. W projekcie w istotny sposób zredukowano koszt i pobór mocy mikrosystemu dzięki zastosowaniu układu Spartan 3E z dwoma procesorami PicoBlaze. Przedstawiono wyniki badań eksperymentalnych zaimplementowanego algorytmu pod kątem szybkości przetwarzania strumienia danych.

Słowa kluczowe: szyfrowanie, algorytm AES, FPGA.

Implementation of ciphering algorithm AES-128 in FPGA Spartan 3E with PicoBlaze processors

Abstract

In this paper we present implementation of the AES-128 (Advanced Encryption Standard) ciphering algorithm in FPGA (Field Programmable Gate Array) Spartan 3E device. The complete digital microsystem is described, which receives a data stream by serial interface, and performs real-time encryption using the AES-128 ciphering algorithm. An important feature of the developed microsystem is significantly reduced size and power dissipation of the device. It has been obtained by optimized architecture of the encryption scheme, using look-up-tables and two PicoBlaze cores. Results of experimental tests focused on the maximum data throughput are also presented.

Keywords: data encryption, AES algorithm, FPGA.

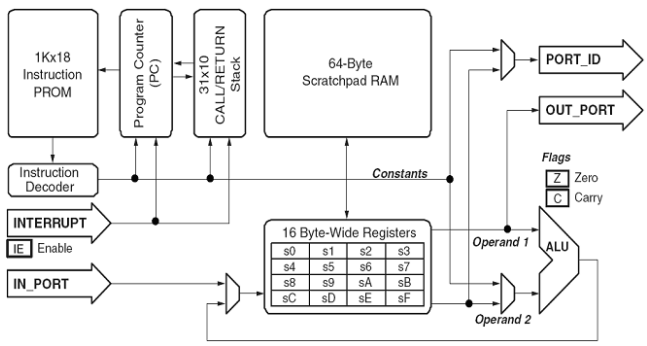
1. Wprowadzenie

Ze względu na coraz szersze wykorzystanie algorytmów szyfrujących w przenośnych systemach cyfrowych wzrasta zapotrzebowanie na zintegrowane mikrosystemy zdolne do realizacji tych algorytmów w niewielkich urządzeniach o małych gabarytach i niewielkim poborze mocy. Wraz z rozwojem technologii FPGA powstała atrakcyjna, alternatywna platforma do implementacji takich mikrosystemów w układach programowalnych zastępujących coraz częściej układy ASIC (Application Specific Integrated Circuit). W proponowanym rozwiązaniu zastosowano nowoczesny algorytm szyfrowania AES-128, znany w literaturze również pod nazwą Rijndael [1], [2]. Ze względu na spore wymagania algorytmów szyfrujących odnośnie mocy obliczeniowej, w praktyce implementowano je dotąd przy użyciu odpowiednio wydajnych procesorów sprzętowych lub programowych. W pracy [3] opisano na przykład porównanie wydajności kilku wybranych algorytmów szyfrujących (w tym AES-128) zaimplementowanych w układzie FPGA z 32-bitowym procesorem RISC(Reduced Instruction Set Computers) MicroBlaze.

Wybór metody implementacji jest związany z rozważeniem dwóch podstawowych parametrów: ilością wykorzystywanych zasobów oraz szybkością szyfrowania. Celem pracy jest przedstawienie zależności między tymi parametrami, dla różnych konfiguracji systemu, przy implementowaniu algorytmu AES-128.

2. Procesor PicoBlaze

PicoBlaze (KCPSM3) jest prostym 8-bitowym mikroprocesorem RISC [4], dostępnym w postaci plików źródłowych VHDL/Verilog. Architektura mikroprocesora zawiera szesnaście 8-bitowych rejestrów ogólnego przeznaczenia, 64bajty wbudowanej pamięci RAM, pamięć stosu o wielkości 32 pozycji. Wszystkie instrukcje są 18-bitowe i wykonywane w dwóch cyklach zegarowych. Procesor może obsłużyć do 256 bajtowych portów wejścia/wyjścia. W prosty sposób można przyłączać peryferia takie jak kontrolery pamięci, jednostki MAC (Multiply and Accumulate) i interfejsy komunikacyjne. Architektura procesora PicoBlaze została przedstawiona na rys. 1.



Rys. 1. Schemat blokowy mikroprocesora PicoBlaze [4]
Fig. 1. PicoBlaze microprocessor block diagram [4]

2.1. Współpraca dwóch procesorów PicoBlaze

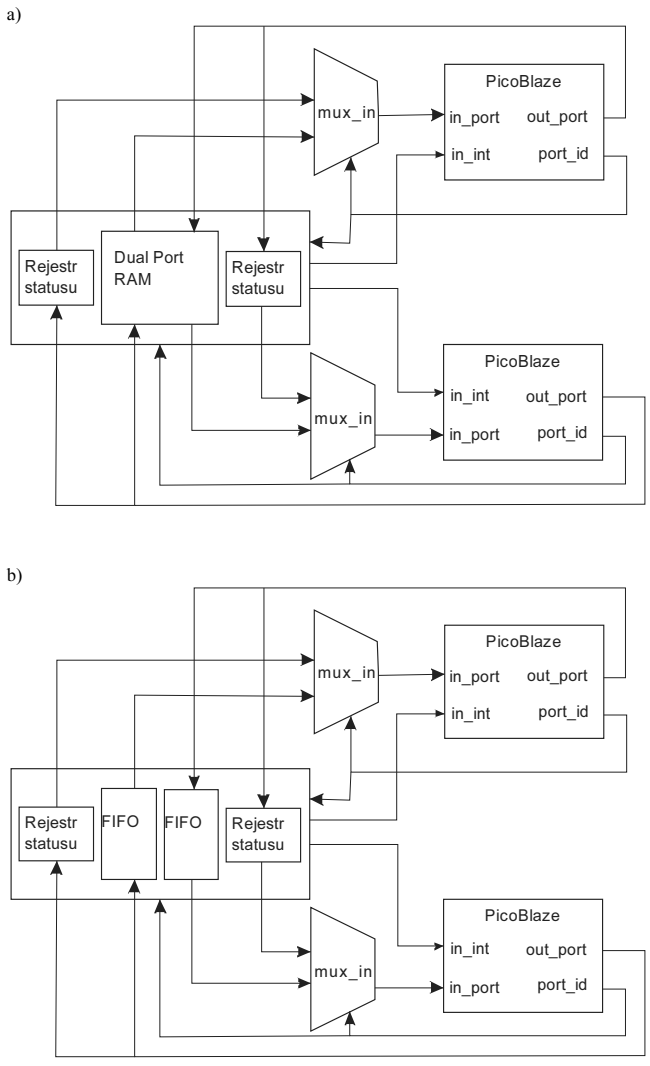
W celu zwiększenia wydajności obliczeniowej projektowanego mikrosystemu wybrano architekturę z dwoma współpracującymi rdzeniami procesorów PicoBlaze, w której procesor wykonujący algorytm szyfrowania, może być traktowany przez inny układ jako czarna skrzynka, reprezentowana przez zbiór rejestrów. Istnieje kilka wariantów komunikacji dwóch procesorów. W niniejszej pracy przeanalizowano dwa z nich: z wykorzystaniem pamięci dwuportowej oraz z pamięcią typu FIFO (First In First Out). Do nadzoru wymiany danych pomiędzy układami zastosować można jeden z dwóch mechanizmów. Pierwszy wykorzystuje wejście

przerwania mikroprocesora i zapewnia małe opóźnienia odpowiedzi. Wymaga on jednak dodatkowego bloku sterującego przerwaniami. Drugą metodą jest cykliczne sprawdzanie rejestru statusu pamięci (informacji o rodzaju zawartości). W końcowym projekcie został zastosowany pierwszy mechanizm nadzoru.

Dwuportowa pamięć RAM obsługuje wymianę danych, natomiast dwa rejestry jednokierunkowe służą do przekazywania informacji sterujących (rys. 2a). Adresowanie pamięci wymaga multipleksowania naprzemiennie adresu i danych, co powoduje większe opóźnienia w odczycie/zapisie danych. Zaletą tego rozwiązania jest swobodny dostęp do dowolnej komórki pamięci. Aby zapewnić pełną dostępność przestrzeni adresowej, wybrano architekturę bez multipleksowania adresu. Należy jednak dodać, że rozwiązanie to nie jest zalecane, jeżeli chcemy uzyskać możliwość łatwego zaimplementowania rozwiązania w przypadku innych aplikacji.

Drugie rozwiązanie wykorzystuje parę pamięci typu FIFO oraz dwa rejestry przechowujące informacje sterujące, do komunikacji jednokierunkowej (rys. 2b). Zasada pracy jest zbliżona do pracy interfejsu Fast Simplex Link (FSL) [5], który jest dedykowany dla współpracy z układami MicroBlaze. Do komunikacji w tym przypadku wystarczą tylko cztery adresy (rejestry), po dwa na wejście i wyjście.

Sposób połączenia układów zaprezentowany został na rys. 2.



Rys. 2. Warianty komunikacji między procesorami PicoBlaze przy użyciu pamięci: a) dwuportowej RAM, b) FIFO

Fig. 2. Two configurations of dual-core PicoBlaze system, using: a) dual-port RAM, b) FIFO memory

2.2. Metoda projektowa i wykorzystane narzędzia

Do zaprojektowania systemu szyfrującego wykorzystano środowisko ISE, które pozwala na korzystanie z konfigurowalnych elementów zawartych w dołączonych bibliotekach. Oprogramowanie opracowano z użyciem środowiska pBlazeIDE [6]. Składa się ono z edytora, kompilatora oraz zawansowanego debugera, umożliwiającego symulację działania portów I/O. Programy dla procesorów napisane zostały w języku asemblera procesora PicoBlaze.

3. System szyfrujący z procesorem PicoBlaze

Głównym celem pracy jest przedstawienie wpływu architektury systemu na szybkość wykonania algorytmu szyfrującego AES-128. Pomiary wykonano w celu ukazania zależności pomiędzy złożonością układu (ilością wykorzystywanych zasobów), a czasem wykonania szyfrowania.

Pierwsza z kilku przebadanych konfiguracji zawiera pojedynczy mikroprocesor. Szyfrowanie jest realizowane całkowicie metodą programową. Wykorzystano minimalną ilość zasobów. Drugi wariant to również rozwiązanie jednoprocesorowe. Do procesora dołączona została pamięć ROM wspomagającą operacje podmiany bajtów SubBytes [1]. Obie te wersje mają swoje odpowiedniki w konfiguracji dwuprocesorowej, każda w dwóch wariantach współpracy (pamięć dwuportowa, FIFO). Nadane zostały im odpowiednio nazwy od PBlaze-A do PBlaze-G. Zestawienie wszystkich zbadanych konfiguracji podano w tabeli 1.

Dla celów porównawczych opracowano siódmy wariant składający się z pojedynczego procesora i IP-core realizującego sprzętowo szyfrowanie, nazwany PBlaze-Core.

Do procesora nadrzędnego (master), przyłączony został układ do transmisji szeregowej UART.

Testy przeprowadzono z użyciem płyty uruchomieniowej firmy AVNET ADS-XLX-SP3E-EVL100, zbudowanej z wykorzystaniem układu XC3S100E Spartan-3E. System pracował z zegarem 100MHz. Do implementacji posłużyło środowisko ISE ver. 9.2.

Tab. 1. Przebadane architektury systemu szyfrującego i zużyte zasoby

Tab. 1. Architectures of encryption system and used resources

Konfiguracja	Liczba procesorów	Typ komunikacji	S_Box	IP-core	FPGA slices	Bloki BRAM
PBlaze-A	1	-	-	-	148	1
PBlaze-B	1	-	Tak	-	149	2
PBlaze-C	2	Dual Port	-	-	263	2
PBlaze-D	2	Dual Port	Tak	-	272	4
PBlaze-F	2	FIFO	-	-	285	1
PBlaze-G	2	FIFO	Tak	-	293	3
PBlaze-Core	1	-	-	Tak	1065	3

3.1. Implementacja algorytmu AES-128

Z analizy algorytmu szyfrującego wynika, że najbardziej złożonymi operacjami są generowanie klucza dla kolejnej rundy NextRoundKey oraz podmiana bajtów SubBytes [1]. Obie wymagają czasochłonnego mnożenia macierzy. Spośród możliwych sposobów przyspieszenia wykonania programu wskazać należy: zaimplementowanie sprzętowego układu mnożącego macierze [7] oraz zastosowanie pamięci ROM, zawierającej wyniki możliwych

przekształceń. Metoda druga jest najczęściej stosowana w rozwiązaniach szyfrujących.

3.2. System PicoBlaze ze sprzętowym blokiem IP-core

Jako sprzętowy IP-core wykorzystany został opis [8], zawierający moduł szyfrujący wraz z generatorem klucza. Wybrano go ze względu na 8-bitową magistralę danych, co ułatwia znacznie współpracę z mikroprocesorem, oraz na mniejsze zapotrzebowaniem na zasoby w porównaniu do innych rozwiązań. Z racji ograniczonych zasobów układu docelowego, nie udało się go zaimplementować, wyniki dla tej konfiguracji zostały oszacowane na podstawie dokumentacji [8]. Poprawność tych oszacowań została jednak zweryfikowana eksperymentalnie w innym, dysponującym większymi zasobami, układzie z rodziny Spartan3.

4. Metodologia pomiarowa

Do pomiarów szybkości szyfrowania wykorzystane zostały dwie metody. Pierwsza (Test_1) polega na przesłaniu poprzez interfejs UART z komputera do układu, 1040 bajtowego bloku danych, z którego pierwsze 16 bajtów stanowi klucz. Wynik przetwarzania jest odbierany na komputerze. Test ten ma na celu uwidocznienie zysku z pracy wieloprocesorowej. Druga metoda (Test_2) polega na zaszyfrowaniu podanym kluczem jednego 128-bitowego bloku danych, zapisanego w pamięci procesora komunikacyjnego. W obu przypadkach mierzona była liczba cykli zegara koniecznych do wykonania operacji szyfrowania i transmisji danych.

5. Wyniki badań

Wyniki pomiarów dla różnych konfiguracji zestawione zostały w tabeli 2. Zastosowanie dedykowanego IP-cora daje najlepsze rezultaty. Systemy w których szyfrowanie odbywało się w pełni programowo wypadły najsłabiej. Ich jedyną zaletą jest bardzo mała ilość potrzebnych zasobów. Dużą poprawę w szybkości działania uzyskano dzięki przyłączenie układu pamięci ROM, zawierającą tablice S_Box.

Tab. 2. Wyniki pomiarów czasu wykonania szyfrowania (odniesione do wariantu PBlaze-A)
Tab. 2. Time of data encrypion (related to the PBlaze-A architecture)

	Test_1			Test_2	
Konfiguracja	Liczba cykli	Czas wykonania [s]	Przyspieszenie	Liczba cykli	Przepustowość [kbps]
PBlaze-A	171191498	1,711915	1	2674202	4,67
PBlaze-B	754386	0,007544	226,9283	10920	1144,69
PBlaze-C	171167730	1,711677	1,0001	2674560	4,67
PBlaze-D	727990	0,00728	235,1564	11562	1081,13
PBlaze-F	171164658	1,711647	1,0002	2674512	4,67
PBlaze-G	723058	0,007231	236,7604	11362	1100,16
PBlaze-Core	105546	0,001055	1621,961	980	12755,1

Zastosowanie pamięci FIFO do komunikacji pomiędzy układami zapewnia szybszą wymianę danych, w porównaniu do zastosowania pamięci dwu portowej. To przyspieszenie, przy przetwarzaniu strumieni danych skraca czas szyfrowania, proporcjonalnie do długości kodowanej informacji. Szybkość interfejsu szeregowego jest wystarczająca i nie powoduje przestojów w przetwarzaniu.

Jak wynika z tablicy 1, systemy oparte na mikroprocesorach ogólnego przeznaczenia nie dorównują pod względem wydajności wyspecjalizowanym układom IP-core. Cechują się łatwością implementacji i krótkim czasem wdrożenia projektu.

W tablicy 3 zestawione zostały wyniki badań dla rozwiązań opartych o inne procesory ogólnego przeznaczenia.

Tab. 3. Wydajność szyfrowania dla innych rdzeni procesorów [9]
Tab. 3. Throughput of the encryption algorithm for other processors [9]

Procesor	Przepustowość [Mbps]
PicoBlaze @ 100MHz	1,12
MicroBlaze @ 50MHz[3]	8,75
LEON @ 50MHz[9]	9,10
Xtensa @ 50MHz[9]	4,57

6. Wnioski

Zbadano efektywność algorytmu AES-128, zaimplementowanego w układzie Spartan 3E w kilku wariantach architektury opartej na układzie PicoBlaze. Podano wyniki porównawcze badań oraz określono najkorzystniejszą z punktu widzenia wydajności konfigurację układu szyfrującego. Przebadane architektury są kompletnymi rozwiązaniami, dedykowanymi do zastosowań wymagających małej zajętości zasobów. Potencjalne zastosowanie to szyfrowanie rozmów telefonicznych, transakcji bezgotówkowych, wymiany krótkich wiadomości tekstowych itp. Ważną zaletą przedstawionego rozwiązania jest niski koszt implementacji w porównaniu do ceny komercyjnych bloków IP-core.

7. Literatura

[1] Advanced Encryption Standard (AES), National Institute of Standards and Technology, Federal Information Processing Standards (FIPS) Pub. 197, Nov. 2001.

[2] J. Daemen, V. Rijmen: The design of Rijndael: AES – the advanced encryption standard, Springer Verlag, ISBN: 3-540-42580-2, 2002

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

[4] Xilinx Inc.: PicoBlaze User Guide, Oct.2005. http://www.xilinx.com

[5] Xilinx Inc.: Fast Simplex Link (FSL) Bus (v2.11a) Data Sheet, July 2007. http://www.xilinx.com

[6] Mediatronix Inc.: pBlazeIDE, http://www.mediatronix.com/pBlazeIDE.htm

[7] Xilinx Inc.: CryptoBlaze: 8-Bit Security Microcontroller, June 2006. http://www.xilinx.com

[8] J. Gbur : AES Core Modules, http://www.opencores.org

[9] A. Hodjat, and I. Verbauwhede.: Interfacing a high speed crypto accelerator to an embedded CPU, Proc. 38th Asilomar Conf. on Signals, Systems and Computers, 2004, vol. 1, pp. 488–492

Artykuł recenzowany