**Artur GIELATA**[2], **Paweł RUSSEK**[1,2], **Kazimierz WIATR**[1,2]
[1] AKADEMICKIE CENTRUM KOMPUTEROWE „CYFRONET" AGH
[2] AKADEMIA GÓRNICZO-HUTNICZA KRAKÓW, KATEDRA ELEKTRONIKI

# Implementacja standardu szyfrowania AES w układzie FPGA dla potrzeb sprzętowej akceleracji obliczeń

**Artur GIELATA**

Student 5-go roku Elektroniki na Wydziale Elektrotechniki, Automatyki, Informatyki i Elektroniki Akademii Górniczo-Hutniczej w Krakowie. Zainteresowania naukowe: systemy i aplikacje oparte na układach reprogramowalnych z wykorzystaniem języka opisu sprzętu VHDL oraz środowiska EDK.

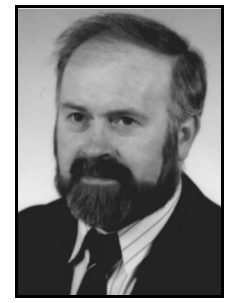
*e-mail: gielat@o2.pl*

**Dr inż. Paweł RUSSEK**

Ukończył studia na wydziale Elektrotechniki, Automatyki i Elektroniki AGH Kraków (1994), dr nauk technicznych (2003). Jest adiunktem w Katedrze Elektroniki AGH. Prowadzone prace badawcze dotyczą sprzętowej akceleracji obliczeń przy pomocy architektur dedykowanych, zagadnień realizacji obliczeń przy użyciu rekonfigurowalnego sprzętu oraz problemu przyśpieszania obliczeń w środowisku komputerów dużej mocy obliczeniowej.

*e-mail: russek@agh.edu.pl*

**Prof. dr hab. inż. Kazimierz WIATR**



Studia AGH Kraków (1980), dr nauk technicznych (1987), dr habilitowany (1999) i profesor (2002). Profesor zwyczajny na Akademii Górniczo-Hutniczej oraz Dyrektor Akademickiego Centrum Komputerowego Cyfronet AGH. Prowadzone prace badawcze dotyczą komputerowego sterowania procesami, systemów wizyjnych, systemów wieloprocesorowych, układów programowalnych, rekonfigurowalnych systemów obliczeniowych i sprzętowych metod akceleracji obliczeń.

*e-mail: wiatr@agh.edu.pl*

**Streszczenie**

Tematem artykułu jest implementacja standardu szyfrowania danych AES-128 w układach reprogramowalnych FPGA. W systemach, gdzie wymagana jest duża szybkość szyfrowania informacji implementacje programowe okazują się zbyt wolne. W związku z tym zachodzi konieczność sprzętowej akceleracji obliczeń, a idealnym rozwiązaniem jest wykorzystanie do tego celu możliwości, jakie dają układy reprogramowalne FPGA. Do implementacji w języku VHDL wybrana została podstawowa wersja algorytmu określonego w standardzie AES. W celu uzyskania maksymalnej szybkości szyfrowania zastosowana została architektura potokowa modułu.

**Słowa kluczowe**: Rijndael, AES, implementacja sprzętowa, FPGA.

## The AES ciper standard implementation on FPGA for hardware accelerated computing

**Abstract**

In this paper we investigate hardware implementation of AES-128 cipher standard on FPGA technology. In many network applications software implementations of cryptographic algorithms are slow and inefficient. To solve the problems custom architecture in reconfigurable hardware was used to speed up the performance and flexibility of Rijndael algorithm implementation. We aimed at achieving the maximum speed and efficiency of cipher process, therefore pipeline architecture of AES module was proposed. The investigations involved simulations and synthesis of VHDL code utilizing Virtex4 series of Xilinx.

**Keywords**: Rijndael, AES, hardware implementation, FPGA.

## 1. Wstęp

Wiele dzisiejszych systemów informatycznych oraz teleinformatycznych wymaga zapewnienia bezpieczeństwa danych oraz przesyłanych informacji. Najprostszym i najczęściej stosowanym sposobem jest programowa implementacja algorytmów szyfrowania. Jednak w przypadku, gdy wymagana jest duża szybkość szyfrowania danych okazuje się, że realizacje programowe stają się zbyt wolne, a z kolei zastosowanie coraz większych mocy obliczeniowych staje się mało efektywne. W związku z tym, zachodzi potrzeba sprzętowej akceleracji operacji kryptograficznych. Idealnym rozwiązaniem jest wykorzystanie do tego celu możliwości, jakie dają układy reprogramowalne FPGA. Gwarantuje to znaczący wzrost szybkości przetwarzania informacji przy jednoczesnym zwiększeniu bezpieczeństwa wynikającym z samej natury rozwiązań sprzętowych. W niniejszym artykule zaproponowano sprzętową realizację algorytmu Rijndael – zwycięzcy konkursu na nowy standard szyfrowania i jednocześnie następcy złamanego w 1997 roku algorytmu DES. Szyfr Rijndael [2] należy do kategorii iteracyjnych szyfrów blokowych, co oznacza, że blok wejściowy oraz klucz przechodzą wielokrotne rundy transformacji zanim wyprodukują wynik. Szyfr może operować na bloku o zmiennej długości, używając kluczy zmiennej długości. Oficjalna specyfikacja standardu dopuszcza użycie bloku danych o długości 128 bitów oraz kluczy o długościach 128, 192 lub 256 bitów.

W niniejszej pracy, do implementacji w układzie FPGA wybrana została podstawowa wersja algorytmu Rijndael z blokiem danych i kluczem o długości 128 bitów oznaczonej w specyfikacji standardu, jako AES-128.

## 2. Rijndael – standard szyfrowania AES

Szyfr Rijndael szybko osiągnął reputację bardzo szybkiego, ale i bardzo bezpiecznego algorytmu. Pomimo wielu lat pracy naukowej, nie wykryto żadnych prawdziwych problemów z bezpieczeństwem tego szyfru [6]. Według danych organizacji NIST - National Institutes of Standards and Technology komputer, który potrafił złamać algorytm DES o długości klucza wynoszącej 56 bitów w ciągu zaledwie jednej sekundy, potrzebowałby czasu rzędu miliardów lat na złamanie szyfru AES-128. Poziom bezpieczeństwa algorytmu dla kluczy długości 192 i 256 bitów jest na tyle wysoki, że został zaakceptowany przez amerykańską agencję do spraw bezpieczeństwa NSA - National Security Agency, do ochrony nawet najbardziej tajnych danych [1]. Stąd też do zabezpieczania wrażliwych informacji zarówno w sektorze cywilnym, dyplomatycznym jak i wojskowym wykorzystywany jest obecnie szyfr AES.

W algorytmie Rijndael [3] zarówno blok danych, jak i klucz reprezentowane są za pomocą macierzy, skonstruowanych według następujących zasad:

- elementem macierzy jest 8 bitów (bajt),
- liczba wierszy macierzy – Nb jest stała dla bloku danych oraz klucza i wynosi 4,
- liczba kolumn macierzy bloku danych – Ns jest równa długości bloku danych podzielonej przez wartość 32,
- liczba kolumn macierzy klucza – Nk jest równa długości klucza podzielonej przez wartość 32.

Cała procedura szyfrowania algorytmem Rijndael składa się z trzech faz operacyjnych. Pierwsza polega na wykonaniu operacji XOR wejściowego bloku danych z kluczem pierwotnym oznaczonej jako transformacja AddRoundKey. Po każdej rundzie powstaje

pośredni wynik szyfrowania, oznaczony jako *State*. Wynik ten następnie poddawany jest wielokrotnym rundom transformacji, których liczba – Nr zależna jest od wersji algorytmu, a dokładniej od długości klucza. W fazie drugiej wykonywanych jest Nr-1 identycznych rund, polegających na odpowiednich transformacjach zrealizowanych na macierzy *State.* Na każdą rundę składają się następujące cztery transformacje. Dla przypadku szyfrowania są to operacje:

- SubBytes
- ShiftRows
- MixColumns
- AddRoundKey

W ostatniej, trzeciej fazie szyfrowania wykonywana jest runda finałowa, która rożni się od poprzednich rund jedynie tym, że macierz *State* nie poddawana jest transformacji MixColumns. Deszyfrowanie przebiega natomiast według odwróconej kolejności przy użyciu dokładnie tych samych kluczy dla poszczególnych rund. Każda runda składa się z odpowiednich transformacji odwrotnych do operacji szyfrujących za wyjątkiem operacji tożsamościowej AddRoundKey, która w obu przypadkach jest identyczna.

Odwrócona jest również kolejność wykonywania operacji w każdej rundzie na następującą:

- AddRoundKey
- InvMixColumns
- InvShiftRows
- InvSubBytes

Dla przypadku deszyfrowania w pierwszej fazie wykonywana jest jedna runda bez operacji InvMixColumns. Druga faza składa się z Nr-1 pełnych rund, a w fazie ostatniej wykonywana jest tylko transformacja AddRoundKey. Parametry Ns, Nk oraz Nr dla trzech różnych wersji szyfru zapisanych w specyfikacji standardu AES przedstawia tab. 1. W praktyce możliwe są różne kombinacje długości bloku wejściowego i klucza głównego, jednak ze względu na to, iż jest to standard rządowy USA wszelkie komercyjne implementacje ograniczają się do zestandaryzowanych wersji szyfru Rijndael.

Tab. 1. Parametry algorytmu dla trzech wersji standardu AES
Tab. 1. Parameters of standard AES algorithm versions

| Wersja algorytmu | Parametry | | |
|---|---|---|---|
| | Ns | Nk | Nr |
| AES-128 | 4 | 4 | 10 |
| AES-192 | 6 | 6 | 12 |
| AES-256 | 8 | 8 | 14 |

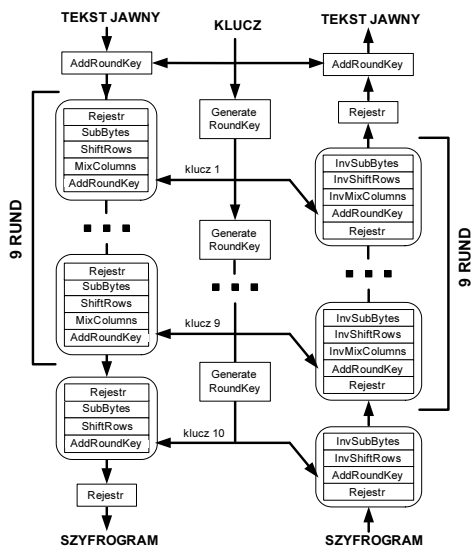
Alg. 1. Przykładowy kod rozszerzenia klucza
Alg. 1. Pseudo code for Key Expantion

```
KeyExpantion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
    word  temp
  i = 0
  while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
  end while
  i = Nk
  while (i< Nb*(Nr+1) )
     temp = w=[i-1]
      if(i mod Nk=0)
          temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if(Nk > 6 and i mod Nk=4)
          temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i+1
  end while
end
```

W algorytmie Rijndael na podstawie głównego klucza szyfrowania wykonywana jest procedura rozszerzenia klucza generująca tablicę kluczy (unikalnych dla każdej rundy), przy czym dla wersji AES-256 została ona nieznacznie zmodyfikowana. Zarówno w przypadku szyfrowania, jak i deszyfrowania klucze mają takie same wartości, z tym, że kolejność ich użycia w obu procesach jest odwrotna. Macierz klucza rundy ma wymiar odpowiadający macierzy *State* niezależnie od długości klucza głównego. Zatem dla każdej wersji standardu klucz rundy ma długość 128 bitów. Procedura rozszerzenia klucza dla wersji AES-128 oraz AES-192 przebiega według algorytmu 1. Funkcja SubWord dokonuje podstawienia 4-bajtowego słowa zgodnie z tablicą S-box określoną w transformacji SubBytes. Funkcja RotWord wykonuje cykliczną rotację bajtów w ramach pojedynczego słowa, natomiast Rcon jest rejestrem zawierającym ściśle zdefiniowane wartości dla kolejnych cykli generacji klucza.

## 3. Implementacja sprzętowa

Jednym z podstawowych wymagań konkursu na standard szyfrowania AES była prostota konstrukcji algorytmu, a więc założenie, iż szyfrowanie ma być łatwe w implementacji zarówno programowej, jak i sprzętowej. Twórcy algorytmu [4] zaprojektowali go z myślą o tym, by w jak najbardziej wydajny sposób mógł być zaimplementowany na różnego typu procesorach, w układach reprogramowalnych [8] oraz w układach specjalnego przeznaczenia typu ASIC . Mimo, iż Rijndael z założenia należy do kategorii iteracyjnych szyfrów blokowych bardzo dobrze nadaje się również do rozwiązań równoległych czy też potokowych [9]. Aby uzyskać jak największą przepustowość szyfrowania do realizacji modułu AES-128 wybrana została architektura w pełni potokowa przedstawiona na rys. 2.




Rys. 2. Architektura potokowa modułu szyfrowania i deszyfrowania
Fig. 2. Pipeline architecture of encrypt and decrypt module

Przed każdą rundą tymczasowy stan *State* zatrzaskiwany jest do rejestru 128-bitowego. W związku z tym wartość zaszyfrowanej danej wejściowej pojawia się w rejestrze wyjściowym po 10 taktach zegarowych. Łącznie daje to czas opóźnienia *latency* dla tej architektury wynoszący 11 CLK. Procedura rozszerzenia klucza zrealizowana została całkowicie, jako logika kombinacyjna. Było to możliwe przy założeniu, że czas potrzebny na wyliczenie klucza dla kolejnej rundy jest mniejszy, niż czas trwania pojedynczej rundy szyfrowania. W związku z tym szybkość szyfrowania, a tym samym przepustowość układu zależy wyłącznie od czasu potrzebnego na wykonanie pojedynczej pełnej rundy algorytmu. Maksymalna częstotliwość taktowania modułu jest w tym przypadku w przybliżeniu odwrotnością czasu trwania pełnej rundy szyfrowania z dołączonym rejestrem.

Każda z transformacji wykonywana jest na bloku 128-bitowym czyli pełnej macierzy *State* i została zaimplementowana jako logika kombinacyjna pod kątem minimalizacji czasu potrzebnego na jej wykonanie. Prosta konstrukcja samego algorytmu Rijndael pozwala na bardzo efektywną realizację poszczególnych transformacji w układzie FPGA. Operacja SubBytes sprowadza się bowiem do implementacji 16 identycznych tablic podstawień S-box, według których realizowane jest podstawienie każdego bajtu

macierzy stanu. Transformacja ShiftRows w przeciwieństwie do aplikacji programowych nie wprowadza w sprzęcie żadnego opóźnienia, gdyż jest to jedynie zmiana kolejności odpowiednich bajtów macierzy *State*. Operacja MixColumns polega na mnożeniu macierzy stanu przez odpowiedni wielomian *a(x)* w ciele $GF(2^8)$.

$$a(x) = '03'*x^3 + '01'*x^2 + '01'*x + '02' \quad (1)$$

Wymaga to wykonania operacji mnożenia przez wartości tylko dwóch współczynników, co sprzętowo zostało zrealizowane w prosty sposób, z zastosowaniem operacji przesuwania bitowego oraz funkcji XOR. Podobnie transformacja AddRoundKey w układzie FPGA wymaga jedynie użycia operacji XOR na dwóch 128-bitowych danych. W ten sposób uzyskano zoptymalizowaną pod kątem czasowym sprzętową realizację pełnej rundy szyfrowania. Ostatnim elementem składowym algorytmu jest moduł rozszerzenia klucza realizujący funkcję z rys. 1. Kolejny klucz wyliczany jest wyłącznie na podstawie klucza poprzedniej rundy, co pozwala na użycie logiki kombinacyjnej, a sama funkcja GenerateRoundKey zaimplementowana została jedynie z użyciem tablic S-box oraz operacji XOR. Ten sam blok rozszerzenia klucza, jak również operacja AddRoundKey wykorzystane zostały przy realizacji modułu deszyfrującego (rys. 2). Pozostałe transformacje odwrotne składające się na pełną rundę procesu deszyfrowania zachowują bardzo podobną konstrukcję. W operacji InvSubBytes zastąpiona została jedynie tablica S-box przez odpowiednią do niej, odwrotną tablicę InvS-box. Transformacja InvShiftRows analogicznie, jak jej odwrotność nie zajmuje żadnej logiki z punktu widzenia sprzętowego. Istotnym zmianom podlega natomiast implementacja operacji InvMixColumns, co wynika z własności wielomianu odwrotnego *b(x)* w ciele $GF(2^8)$.

$$b(x) = '0B'*x^3 + '0D'*x^2 + '09'*x + '0E' \quad (2)$$

W tym przypadku wykonywane jest mnożenie przez każdy z czterech współczynników. W stosunku do transformacji MixColumns, przekłada się to bezpośrednio na wydłużenie czasu trwania operacji, jak również wzrost wymaganej liczby zasobów układu FPGA. Tym samym odpowiednio maleje również maksymalna częstotliwość taktowania zegara modułu deszyfrowania.

## 4. Podsumowanie

Efektywność implementacji sprzętowych w układach reprogramowalnych [5] oceniana jest zwykle na podstawie czterech miarodajnych kryteriów:

- ilość zajmowanych zasobów,
- maksymalna częstotliwość taktowania zegara,
- czas opóźnienia *latency*,
- przepustowość *throughput*.

Moduł szyfrowania AES-128 został opisany w języku opisu sprzętu VHDL [7], a następnie zweryfikowany pod kątem poprawności wyników przy użyciu symulatora Active-HDL firmy ALDEC. Dla oceny projektu przedstawione zostały wyniki raportów wygenerowanych przez program Xilinx XST po przeprowadzeniu procesu syntezy poszczególnych bloków. Osobno poddano syntezie moduły szyfrowania oraz deszyfrowania (10 pełnych rund w architekturze potokowej z rys. 2) bez bloków ekspansji klucza głównego, co dało możliwość przeanalizowania wykorzystanych zasobów potrzebnych na pojedynczą rundę obu procedur. Zastosowanie architektury potokowej pozwala również wyliczyć minimalny czas trwania pełnej rundy. Odwrotność tego okresu wyznacza z kolei maksymalną częstotliwość zegara, z jaką teoretycznie może pracować układ. Szczegółowe wykorzystanie zasobów układu Virtex4 serii 4vlx200ff1513-10 zostało przedstawione w tab. 2. Z kolei parametry czasowe układu zaprezentowane zostały w tab. 3. Wykonanie operacji szyfrowania/deszyfrowania wymaga 11 cykli zegarowych. Każda z dziesięciu rund jest wykonywana w jednym takcie zegara. Dodatkowy cykl to operacja odczytania wyników z rejestru wyjściowego. Blok realizujący generowanie kluczy pracuje równolegle z blokiem algorytmu w przypadku szyfrowania, natomiast deszyfrowanie może rozpocząć się dopiero po czasie rzędu 46 ns (czas uzyskany w raporcie po syntezie modułu rozszerzenia klucza) niezbędnym na wyliczenie kluczy wszystkich rund.

Tab. 2. Wykorzystanie zasobów układu Virtex4
Tab. 2. Resources utilization of Virtex4

| | Rodzaje wykorzystanych zasobów | | |
|---|---|---|---|
| | Slices | Slice Flip Flops | 4 input LUTs |
| Pełna runda szyfrowania | 1209 [1,4%] | 128 [0,1%] | 2403 [1,4%] |
| Pełna runda deszyfrowania | 1934 [2,2%] | 128 [0,1%] | 3584 [2%] |
| Moduł rozszerzenia klucza | 3296 [2,2%] | 0 [0%] | 6400 [2,2%] |

Tab. 3. Parametry czasowe modułu AES
Tab. 3. Time based parameters of AES module

| | Moduł szyfrowania | Moduł deszyfrowania |
|---|---|---|
| Min. okres cyklu rundy | 6 ns | 7,5 ns |
| Max. częstotliwość zegara | 165 MHz | 130 MHz |
| Czas opóźnienia *Latency* | 11 CLK | 11 CLK |
| Przepustowość *Throughput* | 21,2 Gbit/s | 16,6 Gbit/s |

Zrealizowaną implementację w celu oceny szybkości szyfrowania można porównać z realizacjami programowymi w języku ANSI C [10]. Przykładowo na platformie Pentium II 450 MHz uzyskano szybkość szyfrowania 77 Mbit/s (deszyfrowania 74 Mbit/s). Akceleracja algorytmu w układach FPGA znacząco zwiększa jego przepustowość. Pozwala to realnie myśleć o praktycznych zastosowaniach w szyfrowaniu informacji w bardzo szybkich sieciach komputerowych, takich jak GigaEthernet. Dodatkowo oceniając przydatność technologii układów FPGA w zastosowaniach kryptograficznych należy wymienić również takie zalety, jak bezpieczeństwo (fizycznie nie ma możliwości odczytania klucza kryptograficznego „ukrytego" w strukturze układu), rekonfigurowalność (pozwala na zachowanie elastyczności porównywalnej z implementacjami programowymi) oraz możliwość realizacji wielu algorytmów w strukturze jednego układu reprogramowalnego.

## 5. Literatura


[1] Daemen J., Rijmen. V.: The design of Rijndael, 2002.
[2] Advanced Encryption Standard Development Effort, http://www.nist.gov/aes
[3] FIPS PUB 197, Advanced Encryption Standard (AES), National Institute of Standards and Technology, U.S. Department of Commerce, November 2001
[4] Daeman J., Rijmen V.: AES Proposal: Rijndael, Proceedings of the First Advanced Encryption Standard Candidate Conference, Ventura, California, NIST, August 1998
[5] Gaj K., Chodowiec P.: Comparison of the hardware performance of the AES candidates using reconfigurable hardware, Proc. 3rd Advanced Encryption Standard (AES) Candidate Conference, New York, April 13-14, 2000
[6] Ferguson N., Schneider B.: Kryptografia w praktyce, 2004
[7] Zwoliński M.: Projektowanie układów cyfrowych z wykorzystaniem języka VHDL, 2002
[8] Gaj K., Chodowiec P.: Hardware performance of the AES finalists – survey and analysis of results, http://ece.gmu.edu/crypto/AES_survey.pdf
[9] Gaj K., Chodowiec P.: Fast implementation and fair comparison of the final candidates for Advanced Encryption Standard using Field Programmable Gate Arrays, San Francisco, CA, April 8-12, 2001
[10] Lawrence E. Bassham III: Efficiency Testing of ANSI C Implementations of Round1 Candidate Algorithms for the Advanced Encryption Standard, October 13, 1999


*Artykuł recenzowany*