Rekonfigurowalny akcelerator kryptograficzny

Tomasz Kryjak

AGH Akademia Górniczo-Hutnicza, Wydział EAIiIB, Katedra Automatyki i Inżynierii Biomedycznej

Streszczenie: W artykule omówiono zastosowanie układów rekonfigurowalnych FPGA jako akceleratorów kryptograficznych – urządzeń, które mogą wykonywać operacje szyfrowania lub deszyfrowania danych szybciej i przy mniejszym zużyciu energii niż procesory ogólnego przeznaczenia, oferując jednocześnie dużą elastyczność oraz możliwość rozwoju i modyfikacji rozwiązania. W pierwszej części pracy przedstawiono budowę i zasoby dostępne we współczesnych układach FPGA, a w drugiej zaprezentowano implementację algorytmu kryptograficznego CLEFIA.

Słowa kluczowe: układy FPGA, kryptografia, algorytm CLEFIA

1. Wprowadzenie

Układy FPGA są komercyjnie dostępne od prawie 30 lat. W tym okresie wyodrębniła się grupa aplikacji, w których urządzenia te są chętnie stosowane. Należą do nich: prototypowanie układów ASIC, szeroko rozumiana akceleracja obliczeń (ang. *High Performance Computing*), sieci przewodowe i bezprzewodowe, przetwarzanie dźwięku, przetwarzanie obrazów (np. medycznych, satelitarnych) i sekwencji wideo (np. monitoring wizyjny, kompresja i dekompresja wideo), systemy bezpieczeństwa w samochodach, systemy wojskowe oraz kryptografia i kryptoanaliza.

Równocześnie należy stwierdzić, że układy FPGA stały się jedną z czterech głównych programowalnych platform obliczeniowych obok procesorów ogólnego przeznaczenia (CPU), procesorów sygnałowych (DSP) i kart graficznych (GPU). Wyróżnia je możliwość realizacji obliczeń w sposób zrównoleglony, duża swoboda w projektowaniu architektury obliczeniowej, różnorodność obsługiwanych interfejsów do urządzeń zewnętrznych i standardów napięciowych, niskie zużycie energii oraz wielokrotna reprogramowalność.

Kryptografia jest dziedziną nauki zajmującą się szeroko pojętym ukrywaniem informacji. Istnieje już ponad 2000 lat. Od swoich początków związana była z korespondencją dla potrzeb wojska i dyplomacji. Dopiero na przełomie XX i XXI wieku, wraz z rozwojem Internetu, nastąpiło gwałtowne upowszechnienie kryptografii „cywilnej". Obecnie zabezpieczane są wszelkie transakcje finansowe (przelewy, e-zakupy itp.), rozmowy i wideorozmowy, transfer danych, rozmaite dokumenty, wybrana korespondencja elektroniczna, nośniki multimedialne (ochrona przed nieautoryzowanym kopiowaniem – systemy DRM) i wiele innych.

Celowym zatem jest projektowanie akceleratorów kryptograficznych – modułów sprzętowych, które szyfrowanie i deszyfrowanie realizują szybciej i przy mniejszym zużyciu energii niż rozwiązania programowe. W artykule omówiono implementację algorytmu CLEFIA w układzie FPGA i budowę rekonfigurowalnego akceleratora kryptograficznego.

2. Układy FPGA

Pierwsze układy FPGA (ang. *Field Programmable Gate Array*) pojawiły się w 1985 r. i stanowiły kolejny etap rozwoju programowalnych układów logicznych – urządzeń, których końcowa funkcjonalność nie została określona na etapie produkcji. Podstawowym elementem układu FPGA są: matryca konfigurowalnych bloków logicznych *Configurable Logic Block* – CLB (nazewnictwo Xilinx), *Adaptive Logic Block* – ALB (nazewnictwo Altera) i konfigurowalne zasoby połączeniowe. Ponadto układy wyposażone są w wbudowane pamięci RAM, moduły do zarządzania sygnałem zegarowym, moduły DSP, interfejsy wejścia/wyjścia, kontrolery magistrali PCI Express i pamięci RAM DDR3. Szczegóły budowy nowoczesnych układów FPGA omówiono na przykładzie rozwiązań firm Xilinx [29] i Altera [1], które oferują najbardziej zaawansowane rozwiązania na rynku. Najnowsze serie układów FPGA firmy Xilinx to Virtex 7, Kintex 7 i Artix 7, a firmy Altera to Stratix V, Arria V i Cyclone V; wszystkie wykonane w technologii 28 nm. Dla obu producentów można zaobserwować podział na trzy kategorie produktów:

- układy o największych zasobach logicznych, osiągach i możliwościach: Virtex i Stratix,
- układy o korzystnym stosunku wydajności do ceny: Kintex i Arria,
- układy o najniższym zużyciu energii i niskiej cenie, przeznaczone do produkcji masowej: Artix i Cyclone.

2.1. Układy FPGA firmy Xilinx

Podstawowym elementem układu FPGA firmy Xilinx jest blok CLB. Jest on podzielony na dwa moduły *Slice*, z których każdy zawiera: cztery generatory funkcyjne, osiem elementów pamięciowych (przerzutniki D, ang. *Flip Flop* – FF), multipleksery oraz logikę przeniesienia. Generator

funkcyjny został zrealizowany jako LUT (ang. *Look-Up Table*) o 6 wejściach i dwóch niezależnych wyjściach. Istnieje możliwość łączenia dostępnych modułów LUT z wykorzystaniem multiplekserów. Generator funkcyjny może zostać także skonfigurowany jako:

– synchroniczna pamięć RAM, zwana pamięcią rozproszoną (ang. *Distributed RAM*) – o różnym rozmiarze i liczbie portów od 1 do 4, przy czym jeden port umożliwia synchroniczny zapis i asynchroniczny odczyt, a pozostałe asynchroniczny odczyt,
– 32-bitowy rejestr przesuwny wykorzystywany przy tworzeniu linii opóźniających.

Dedykowana logika przeniesienia zapewnia szybką realizację operacji dodawania i odejmowania w obrębie pojedynczego *Slice'a*. Za połączenia między zasobami logicznymi oraz modułami wejścia i wyjścia odpowiada sieć konfigurowalnych połączeń. Zestawienie zasobów dostępnych w układach FPGA firmy Xilinx zaprezentowano w tab. 1.

Tab. 1. Maksymalne zasoby dostępne w układach FPGA firmy Xilinx
Tab. 1. Maximal resources available in Xilinx FPGA devices

Zasoby	Artix-7	Kintex-7	Virtex-7
*Logic Cells**	360 k	478 k	1955 k
Block RAM	19 Mb	34 Mb	68 Mb
DSP48	1040	1920	3600
Wydajność DSP [GMAC/s]	1306	2845	5335
Transceivery	16	32	96
Maks. szybkość transceivera	6,6 Gb/s	12,5 Gb/s	28,05 Gb/s
Maks. przepustowość szeregowa	211 Gb/s	800 Gb/s	2784 Gb/s
Interfejs PCIe	× 4 Gen2	× 8 Gen2	× 8 Gen3
Interfejs RAM	1066 Mb/s	1866 Mb/s	1866 Mb/s
Wejścia/wyjścia	600	500	1200

*Porównywanie pojemności układów FPGA odbywa się za pomocą wielkości *Logic Cells*, która jest odpowiednikiem typowego zestawu: 4-bitowy LUT i przerzutnik. W układach Xilinx serii 7 przyjęto stosunek *Logic Cells* do liczby 6-bitowych LUT'ów jako 1,6:1.

2.1.1. Pozostałe zasoby dostępne w układzie FPGA firmy Xilinx

W układzie FPGA Xilinx serii 7 dostępne są następujące dodatkowe zasoby:

– CMT *Clock Managment Tiles* – bloki zarządzania sygnałem zegarowym, które zapewniają generację różnych częstotliwości, równomierną propagację sygnału oraz tłumienie zjawiska *jitter*,
– Block RAM – bloki dedykowanej dwuportowej pamięci RAM o pojemności 36 kb, które mogą zostać również skonfigurowane jako moduły FIFO (z niezależnym zegarem zapisu i odczytu),
– DSP48E1 - moduły z mnożarką 25 × 18 bitów oraz 48-bitowym akumulatorem. Mają jednostkę arytmetyczną, która działa w trybie SIMD (ang. *Single Instruction Multiple Data*), co umożliwia jednoczesne wykonywanie operacji na dwóch 24-bitowych danych lub czterech 12-bitowych,
– XADC – podwójny, 12-bitowy przetwornik analogowo-cyfrowy z częstotliwością próbkowania 1 MHz, umożliwiający obsługę do 17 wejść analogowych,
– GTX Transceivers – moduły nadawczo-odbiorcze umożliwiające transmisję szeregową z prędkością 500 Mb/s i 12,5 Gb/s. Wspierają one kilka trybów: PCI Express 1.1/2.0/3.0, 10GBASE-R, Interlaken i inne,
– Select IO – zasoby wejścia/wyjścia, podzielone na banki po 50 pinów (liczba banków zależy od typu, rozmiaru i obudowy układu). Mogą zostać skonfigurowane do pracy z wieloma standardami (w tym z różnicowymi): LVCMOS, LVTTL, HSTL, PCI, SSTL, LVDS i inne,
– Zintegrowany moduł PCI Express – wspiera transmisję z przepustowością 2,5 Gb/s i 5,0 Gb/s i liczbą linii 1, 2, 4, 8.

2.2. Układy FPGA firmy Altera

Podstawowy element układu FPGA firmy Altera stanowi moduł ALM (ang. *Adaptive Logic Module*). Jest on zbudowany z 8-wejściowego adaptacyjnego generatora funkcyjnego, dwóch sumatorów i czterech rejestrów. Generator funkcyjny został zrealizowany jako 8-wejściowy i 2-wyjściowy element LUT (składający się z dwóch 6-wejściowych elementów LUT). Wbudowane sumatory zapewniają realizację operacji dodawania 2- i 3-bitowego, bez konieczności wykorzystywania dodatkowych zasobów logicznych. Moduły ALM zestawione są w tablice bloków LAB (ang. *Logic Array Block*), które mogą stanowić 640-bitową dwuportową pamięć RAM.

Za połączenia między zasobami logicznymi oraz modułami wejścia i wyjścia odpowiada sieć konfigurowalnych połączeń. Zestawienie zasobów dostępnych w układach FPGA firmy Altera zaprezentowano w tab. 2.

Tab. 2. Maksymalne zasoby dostępne w układach FPGA firmy Altera
Tab. 2. Maximal resources available in Altera FPGA devices

Zasoby*	Cyclone V	Arria V	Stratix V
*Logic Cells**	301 k	504 k	952 k
M20K RAM	11 Mb	24 Mb	52 Mb
27 × 27 DSP	342	1139	352
Transceivery	12	36	66
Wejścia/wyjścia	488	668	840

*Podane wartości należy traktować orientacyjnie. W skład każdej serii układów wchodzą podserie dedykowane do innych zastosowań (np. DSP lub komunikacji) o zwiększonej liczbie wybranych zasobów.

2.2.1. Pozostałe zasoby dostępne w układzie FPGA firmy Altera

W układzie FPGA Altera serii V dostępne są następujące dodatkowe zasoby:

- fPLL – pętla sprzężenia fazowego (ang. *fractional phase locked loop*), moduł do precyzyjnego generowania częstotliwości zegara,
- VP DSP – moduły DSP o zmiennej precyzji (ang. *Variable Precision*), pojedynczy moduł umożliwia wykonanie do 3 mnożeń 9×9, dwóch 16×16 i jednego 27×27 (dane dla układów Stratix; dla Arria i Cyclone możliwa jest niezależna realizacja dwóch mnożeń 18×19),
- M20K - bloki dedykowanej dwuportowej pamięci RAM o pojemności 20 kb. Mogą zostać wykorzystane jako rejestry przesuwne lub moduły FIFO. Maksymalna częstotliwość pracy wynosi do 600 MHz. (w układach Arria i Cyclone pamięci blokowe są mniejsze – 10 kb i o niższej maksymalnej częstotliwości pracy – 400 MHz),
- Transcivers – moduły nadawczo-odbiorcze umożliwiające transmisję szeregową z szybkością od 14,1 Gb/s do 28,05 Gb/s (w układach Arria i Cyclone maksymalne prędkości wynoszą odpowiednio 10,0 Gb/s i 5,0 Gb/s),
- Moduły wejścia/wyjścia. Mogą zostać skonfigurowane do pracy z wieloma standardami (w tym z różnicowymi): LVCMOS, LVTTL, HSTL, PCI, SSTL, LVDS i inne,
- Zintegrowane moduły sprzętowe do protokołów Interlaken, 10 GbE, Serial Rapid IO, CPRI/OBSAI,
- Zintegrowany moduł PCI Express – wspiera działanie z przepustowością 2,5 Gb/s i 5,0 Gb/s, a dla układu Stratix z 8,0 Gb/s (Gen3),
- HPS – wbudowany procesor (ang. *Hard Processor System*). Występuje w wybranych układach serii Arria i Cyclone. Składa się z dwurdzeniowego procesora ARM Cortex A9, kontrolerów: pamięci SDRAM i DMA oraz modułów Ethernet, USB, I2C, UART, SPI.

3. Projektowanie logiki układów FPGA

Logikę układów FPGA można projektować za pomocą kilku narzędzi. Podstawowym sposobem jest opis w tzw. języku opisu sprzętu (ang. *Hardware Description Language* – HDL). Przykładami są popularne VHDL i Verilog. W odróżnieniu od języków programowania (C/C++/C#, Java), nie opisuje się ciągu instrukcji, który zostanie wykonany na zadanej architekturze sprzętowej, a elementy obliczeniowe i połączenia między nimi.

Wspomniane języki HDL zalicza się do narzędzi niskopoziomowych i sprawne posługiwanie się nimi wymaga znajomości zagadnień elektroniki cyfrowej oraz pewnego doświadczenia. Stanowi to przeszkodę w upowszechnianiu technologii FPGA, szczególnie jako akceleratorów obliczeniowych. Aby ją wyeliminować powstało szereg narzędzi wysokiego poziomu, wśród których można wymienić: Catapult-C [4], Mitrion-C [20], rozwiązania firmy Maxeler [19], System Generator i HDL Coder (powiązane z popularnym pakietem MATLAB/Simulink), rozszerzenia pakietu LabVIEW do współpracy z układami FPGA [18] i inne.

4. Zastosowania układów FPGA

Układ FPGA może pełnić rolę akceleratora w systemie obliczeniowym, realizującym konkretną fukcjonalność (np. przy współpracy z komputerem PC lub superkomputerem) lub być samodzielnym urządzeniem (tzw. systemem wbudowanym, ang. *embedded system*). W obu przypadkach warto wskazać grupę algorytmów (i powiązanych zastosowań), których implementacja w zasobach układu FPGA jest opłacalna, tj. uzyskuje się znaczne przyspieszenie obliczeń i/lub niższe zużycie energii. Drugi aspekt jest szczególnie istotny w przypadku systemów superkomputerowych, gdyż ma bardzo wyraźne przełożenie na koszty prowadzenia obliczeń. Również w przypadku urządzeń mobilnych, niskie zużycie energii jest bardzo istotne. Szczególnie predestynowane do akceleracji z wykorzystaniem FPGA są algorytmy spełniające następujące warunki:

- składają się z szeregu względnie prostych operacji powtarzanych na wielkiej liczbie danych (tzw. algorytmy zdominowane przez dane), np. filtracja kontekstowa obrazów,
- mają duże możliwości zrównoleglenia typu: SIMD (*Single Instruction Multiple Data* - pojedyncza instrukcja, wiele danych), MIMD (*Multiple Instruction Multiple Data,*
- wiele instrukcji, wiele danych) oraz MISD (*Multiple Instruction Single Data* – wiele instrukcji pojedyncze dane),
- operują na liczbach stałoprzecinkowych. Układy FPGA nie mają bezpośredniego sprzętowego wsparcia dla operacji zmiennoprzecinkowych, zatem realizacja tego typu obliczeń, jakkolwiek możliwa wymaga znacznych zasobów logicznych,
- charakteryzować się uporządkowanym dostępem do danych (np. przetwarzanie obrazu linia po linii).

Główne grupy zastosowań układów FPGA to przede wszystkim przetwarzanie i analiza obrazów i sekwencji wideo, wykorzystywane w przemyśle wojskowym, samochodowym, urządzeniach medycznych, elektronice użytkowej (kamery, telewizory), systemach monitoringu i pokrewnych. Przykładowe aplikacje to detekcja twarzy [12] oraz monitoring wizyjny [17]. Szersze omówienie tematyki odnaleźć można w pracy [10].

Kolejną grupą zastosowań jest przetwarzanie innych sygnałów cyfrowych: dźwięku (w tym sygnałów pozyskanych z sonaru [13]) oraz zarejestrowanych fal elektromagnetycznych (np. radar) [30]. Prowadzone są również prace nad ich wykorzystaniem w zadaniach sterowania [15].

Układy FPGA, z uwagi na swoją elastyczność i wielość możliwych do zrealizowania interfejsów wejścia/wyjścia stanowią istotny element w sieciach bezprzewodowych (stacje BTS) i przewodowych (zaawansowane routery) [22].

Następną grupą zastosowań jest akceleracja algorytmów obliczeniowych wykorzystywanych w takich dziedzinach jak fizyka, chemia, bioinformatyka, inżynieria materiałowa, budownictwo, ekonomia, finanse [3, 11]. Uzyskuje się w ten sposób przyspieszenie obliczeń, przy jednoczesnym obniżeniu ich kosztów (dzięki znacznie mniejszemu zużyciu energii niż procesor lub karta graficzna).

Układy FPGA mogą również być wykorzystywane jako akceleratory kryptograficzne. Tematyka ta zostanie szerzej omówiona w dalszej części artykułu.

5. Kryptografia i szyfr CLEFIA

Szyfry można, w zależności od zastosowania, podzielić na: asymetryczne lub symetryczne, a także blokowe i strumieniowe (zagadnienie szerzej omówione w [21]). W artykule skupiono się na szyfrach symetrycznych, blokowych, gdyż do takich należy analizowany i implementowany algorytm CLEFIA.

Symetrycznymi określa się algorytmy, w których podczas szyfrowania i deszyfrowania danych wykorzystuje się ten sam klucz (tj. informację umożliwiającą wykonanie czynności kryptograficznej), a realizowane operacje są bardzo podobne. Blokowymi określa się rozwiązania, w których dane wejściowej dzielone są na równe porcje (tj. bloki). Do popularnych symetrycznych szyfrów blokowych należą: DES, 3DES, AES, IDEA, Blowfish, a także CLEFIA.

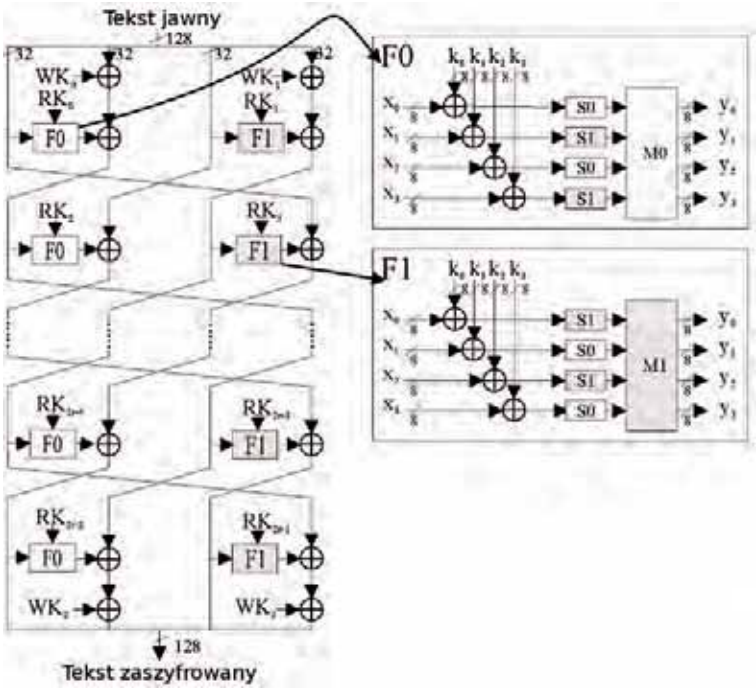
Szyfry asymetryczne (np. RSA) charakteryzują się występowaniem dwóch rodzajów kluczy: publicznych i prywatnych oraz tym, że proces szyfrowania i deszyfrowania przebiega w odmienny sposób [24].

5.1. Kryptografia w układach FPGA

Tematyka implementacji algorytmów kryptograficznych w układach FPGA jest obecna w badaniach naukowych od ponad 15 lat. Wpływ na to mają:

- łatwość implementacji większości operacji wykorzystywanych w algorytmach kryptograficznych w zasobach logicznych dostępnych w FPGA,
- możliwości, jakie oferuje rekonfigurowalność, tj. opcję zmiany algorytmu lub podmiany implementacji na wydajniejszą lub bezpieczniejszą (np. odporną na ataki kryptoanalityczne z wykorzystaniem różnicowej analizy mocy).

Jako pierwsze zostały zaimplementowane szyfry blokowe DES [26, 16], IDEA [2], AES [9], następnie strumieniowe [14] i asymetryczne (RSA) [8].



Rys. 1. Przetwarzanie danych w algorytmie CLEFIA [6]
Fig. 1. Data processing in CLEFIA algorithm [6]

Warto podkreślić, że moduły szyfrujące i deszyfrujące są sprzedawane w postaci tzw. IP Core (ang. *Intellectual Property Core*), czyli gotowych, sprawdzonych elementów, które można użyć we własnym projekcie. Ich zestawienia znajdują się na stronach internetowych firm Xilinx [29] i Altera [1]. Dostępne są implementacje algorytmów 3DES, AES, RSA, SHA-1, SHA-256, MD5.

Układy FPGA znajdują również zastosowanie jako podstawa do budowy maszyn kryptoanalitycznych. Przykładem jest COPACOBANA [7], zbudowana z 200 układów Spartan 3 (lub w nowszej wersji ze 128 Virtex 4) zdolna do łamania szyfru DES w czasie krótszym niż tydzień.

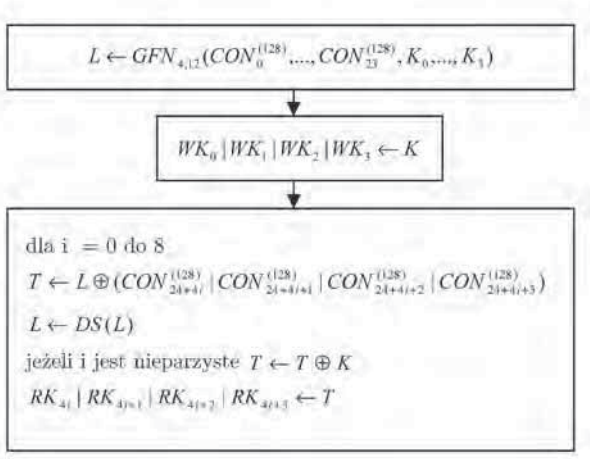
5.2. Szyfr CLEFIA

CLEFIA jest 128-bitowym szyfrem blokowym z kluczem o długości 128, 192, 256 bitów stworzonym przez firmę SONY [6]. W 2012 r. algorytm został dołączony do standardu ISO/IEC 29192-2, w zakresie kompaktowych szyfrów blokowych, które to znajdują zastosowanie w urządzeniach o ograniczonych zasobach, przykładowo w inteligentnych kartach (ang. *smart cards*), identyfikatorach RFID (ang. *Radio-Frequency Identification*), sieciach czujników oraz urządzeniach medycznych.

Szyfr CLEFIA można podzielić na dwa funkcjonalne moduły: ścieżkę przetwarzania danych oraz generator podkluczy. Podstawą budowy ścieżki przetwarzania danych jest uogólniona sieć Fiestela (ang. *General Fiestel Network*, $GFN_{a,b}$ gdzie a – liczba gałęzi, b – liczba rund[1]). Wejściowy 128-bitowy blok danych dzielony jest na cztery 32-bitowe fragmenty. GFN wykorzystuje dwie tzw. F-funkcje: F0 i F1. Zbudowane są one w oparciu o operacje XOR (oznaczone jako $\oplus$), dwa 8-bitowe bloki podstawieniowe (S-Box) - S0 i S1 oraz mnożenie przez macierze rozpraszające M0 i M1. Przekształcenie S0 i S1 zdefiniowano za pomocą arytmetyki w ciele skończonym $GF(2^4)$ i $GF(2^8)$ oraz operacji afinicznych. Szczegóły zawarto w dokumencie [6]. Schematycznie przetwarzanie danych przedstawiono na rys. 1.

Zadaniem generatora podkluczy jest wygenerowanie, na podstawie klucza szyfrowania, dwóch rodzajów podkluczy używanych w algorytmie CLEFIA: określonych jako „wybielające" (ang. *Whitening Key*, WK), które wykorzystywane są na początku i końcu przetwarzania (operacja XOR z danymi, por. rys. 1) oraz tzw. kluczy rundy (ang. *Round Key*, RK), które sterują pracą każdej z F-funkcji. Składa się on z dwóch modułów. W pierwszym, na podstawie klucza szyfrowania K, za pomocą GFN, wyliczany jest klucz pośredni L. W przypadku klucza 128-bitowego wykorzystuje się sieć $GFN_{4,12}$, dla kluczy 192- i 256-bitowych $GFN_{8,10}$. W drugim etapie na podstawie K i L wyznaczane są wartości WK i RK. W module używane są też 32-bitowe wartości stałe $CON_i(k)$. Ich liczba,

1) Szyfr symetryczny podzielony jest na szereg wielokrotnie powtarzających się operacji – rund szyfrowania. W przypadku CLEFIA pojedyncza runda składa się z modułów F0 i F1 oraz dwóch operacji XOR.



Rys. 2. Generacja podkluczy dla klucza 128-bitowego
Fig. 2. 128-bit key-scheduling block

w zależności od rozmiaru klucza, wynosi 60, 84 lub 92. Schemat blokowy algorytmu generatora dla klucza 128-bitowego przedstawiono na rys. 2. Symbol DS to funkcja zamiany bitów w 128-bitowym wektorze, a | oznacza konkatenację wektorów. Szerszy opis algorytmu dostępny jest w dokumencie [6], a referencyjna implementacja zamieszczona jest na stronie internetowej producenta [25].

6. Sprzętowa implementacja algorytmu CLEFIA

Opisany w punkcie 5.2 szyfr CLEFIA zrealizowano w sposób w pełni potokowy, wykorzystując zasoby rekonfigurowalne układu FPGA. Oznacza to możliwość płynnego przetwarzania kolejno dostarczanych 128-bitowych danych wejściowych oraz niewielkie opóźnienie (kilkanaście taktów zegara) między pierwszą próbką danych jawnych a rezultatem szyfrowania. Podejście to wymaga zaimplementowania logiki dla wszystkich rund szyfrowania (tj. 18 w przypadku wersji z kluczem 128-bitowym).

Alternatywnym sposobem jest wykonanie modułu iteracyjnego. W tym przypadku konieczne jest zaimplementowanie logiki tylko jednej rundy. Wiąże się to jednak z N-krotnym spadkiem wydajności (N – liczba rund). W tym przypadku szyfrowanie każdego bloku danych wymaga, co najmniej N taktów zegara.

Najbardziej złożonym elementem pojedynczej rundy są bloki S-Box S0 i S1 – sposób ich implementacji w znacznym stopniu decyduje o całkowitym opóźnieniu i maksymalnej częstotliwości pracy modułu sprzętowego. Pozostałe operacje, tj. XOR, bardzo dobrze implementują się w układzie FPGA.

6.1. Implementacja S0 i S1

Najprostsza implementacja bloków S-Box S0 i S1 polega na wykorzystaniu operacji LUT (ang. *Look-Up Table*). Wartości podstawienia zapisane są w pamięci ROM opartej na zasobach *Distributed* RAM lub *Block* RAM układu FPGA. Takie rozwiązanie wymaga stosunkowo dużej ilości pamięci $16 \times 16 \times 8$ bitów = 2048 bitów. Pojedynczy S-Box wykorzystuje 128 elementów LUT układu FPGA i wprowadza opóźnienie ok. 1,75 ns (wyniki prezentowane w tym rozdziale dotyczą technologii FPGA Xilinx Virtex II Pro).

Innym rozwiązaniem może być implementacja bloków S-Box wprost na podstawie definicji (podejście z pracy [5]). W tym przypadku moduł S0 wykorzystuje 24 elementy LUT i wprowadza opóźnienie ok. 2,3 ns, a S1 56 elementów LUT oraz wprowadza opóźnienie ok. 5,7 ns. Różnica wynika ze stopnia skomplikowania obliczeń. W pierwszym przypadku konieczne jest zaimplementowanie mnożenia w ciele GF(2^4) i czterech niewielkich elementów LUT, a w drugim obliczania odwrotności w ciele GF(2^8).

6.2. Implementacja macierzy rozpraszających M0 i M1

Moduły M0 i M1 zaimplementowano wykorzystując schemat zaproponowany w pracy [5]. Pozwala on zrealizować obie operacje za pomocą mnożenia przez {01, 02, 04, 08} i XOR. Implementacja M0 wykorzystuje 54 elementy LUT i wprowadza opóźnienie ok. 2,2 ns, a M1 – 57 LUT i opóźnienie ok. 2,5 ns.

6.3. Implementacja F-funkcji F0 i F1

F-funkcje F0 i F1 składają się z operacji XOR, S-Box S0 i S1 oraz mnożenia przez macierze M0 i M1. Podstawą każdej F-funkcji i jednocześnie najbardziej złożonym elementem są bloki podstawieniowe. Bazując na wstępnych wynikach (por. punkt 6.1) zebranych w tab. 3 zdecydowano się wykonać trzy implementacje F-funkcji: z S-Box jako *look-up table* (pobranie wartości podstawienia z tablicy), z S-Box zrealizowanymi na podstawie definicji (każdorazowe wyliczanie wartości podstawienia zgodnie z definicją) oraz konfigurację mieszaną (*mixed*), w której S0 implementowane jest na podstawie definicji, a S1 jako *look-up table*.

Tab. 3. Porównanie różnych implementacji S-Box S0 i S1
Tab. 3. S0 and S1 S-Box different implementations comparison

	S0		S1	
	look-up table	definicja	*look-up table*	definicja
LUT	128	24	128	56
Opóźnienie	1,75 ns	2,3 ns	1,75 ns	5,7 ns

Implementacja S-Box S0 na podstawie definicji wykorzystuje o 104 elementy LUT mniej i wprowadza opóźnienie większe o 0,55 ns niż wersja *look-up table*. Można zatem stwierdzić, że opłaca się ją stosować. W przypadku S-Box S1 implementując na podstawie definicji zyskuje się 72 elementy LUT, ale kosztem aż 3,95 ns opóźnienia. W aplikacjach zorientowanych na szybkość taka implementacja S1 może spowodować spadek wydajności całego modułu. Opracowany tryb *mixed* jest zatem kompromisem pomiędzy dużą szybkością działania implementacji typu *look-up table* (S-Box S1) a małym zużyciem zasobów wersji S-Box'a na podstawie definicji (S0).

2) W implementacji zastosowano opisane w dokumencie [5] rozwiązanie polegające na transformacji ciała GF(2^8) w ciało GF((2^4)^2). Wyniki zaprezentowane w pracy [23] (dla szyfru AES) pokazują, że takie postępowanie pozwala zmniejszyć zapotrzebowanie na zasoby logiczne.

Dodatkowo zaimplementowano obie F-funkcje z wykorzystaniem techniki tzw. T-Box. Polega ona na połączeniu układów S-Box z następującą po nich operacją liniową (M0 i M1), celem skrócenia ścieżki krytycznej. Oryginalnie cztery typy układów T-Box zostały zaproponowane w [5] dla implementacji programowych (dla procesorów ogólnego przeznaczenia).

6.4. Implementacja pojedynczej rundy szyfrowania

Wyniki implementacji pojedynczej rundy szyfrowania zaprezentowano w tab. 4. Analiza rezultatów pokazuje, że runda z wykorzystaniem S-Box jako *look-up table* wykorzystuje najwięcej zasobów logicznych przy przewidywanej częstotliwości pracy ok. 140 MHz (*Post Place & Route Static Timing*, układ Virtex II Pro 100). Wersja, w której S-Box są zaimplementowane na podstawie definicji wykorzystuje zdecydowanie mniej zasobów, przy częstotliwości pracy o ok. 20 MHz niższej. Wersje *mixed* i T-Box osiągają podobną złożoność zasobową oraz częstotliwość pracy ok. 140 MHz.

Tab. 4. Rezultaty implementacji pojedynczej rundy szyfrowania
Tab. 4. Single round implementation result

	look-up table	definicja	*mixed*	T-Box
Slices	949	462	648	669
LUT	1675	748	1099	1133
FF	282	256	267	269
Maks. częstotliwość [MHz]	142	119	144	142

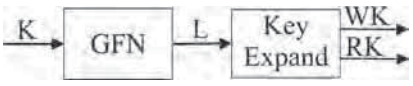
6.5. Implementacja generatora podkluczy

Schemat blokowy zaproponowanego rozwiązania przedstawiono na rys. 3. Iteracyjny moduł GFN, na podstawie klucza K oraz stałych CON, generuje klucz pośredni L. Składa się z pojedynczej rundy szyfrowania oraz kontrolera, który dostarcza odpowiednie stałe (w zależności od numeru iteracji). Implementacja wprowadza opóźnienie 10 lub 12 cykli zegara, w zależności od długości klucza.

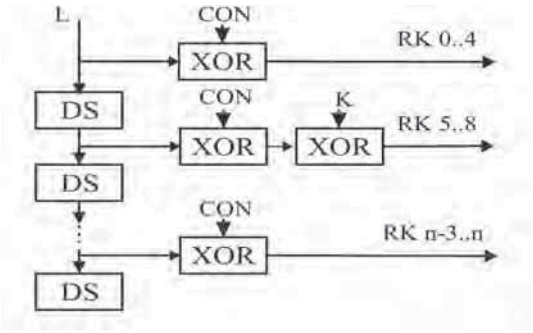
Moduł *Key Expand* generuje klucze WK i RK na podstawie klucza pośredniego L i stałych CON. Schematycznie jego budowę przedstawiono na rys. 4, a użycie zasobów FPGA (dla układu Virtex II Pro 100) dla różnych długości klucza zestawiono w tab. 5.

6.6. Implementacja 18 rund szyfrowania

W dalszych badaniach skupiono się na module szyfrującym zbudowanym z 18 rund (w wersji dla klucza 128-bitowego). Dokonano porównania wyników implementacji czterech zaproponowanych wersji (*look-up table*, definicja, *mixed*, T-Box) dla u kładów FPGAVitrex II Pro 100 oraz Virtex



Rys. 3. Moduł generacji podkluczy
Fig. 3. Key scheduler block



Rys. 4. Moduł Key Expand
Fig. 4. Key Expand Module

4 LX 200. Do syntezy oraz implementacji wykorzystano narzędzie ISE firmy Xilinx. Maksymalna częstotliwość pracy podana została na podstawie *Post Place & Route Static Timing*. Wyniki zaprezentowano w tab. 6 i 7.

Tab. 5. Rezultaty implementacji generatora podkluczy
Tab. 5. Key scheduler implementation results

	128	192	256
Slices	823	1400	1462
LUT	1491	2411	2545
FF	299	569	569
Maks. częstotliwość [MHz]	118	103	74

Tab. 6. Moduł przetwarzania danych – Virtex II Pro 100
Tab. 6. Data processing block – Virtex II Pro 100

	look-up table	definicja	*mixed*	T-Box
Slices	11924	5822	8334	8527
LUT	23654	11271	16568	16761
FF	2345	2319	2342	2308
Maks. częstotliwość [MHz]	80	84	115	124

Tab. 7. Moduł przetwarzania danych – Virtex 4 LX 200
Tab. 7. Data processing block – Virtex 4 LX 200

	look-up table	definicja	*mixed*	T-Box
Slices	14069	6018	9566	9896
LUT	23659	11502	16304	16713
FF	4779	2396	3755	3836
Maks. częstotliwość. [MHz]	146	152	167	167

W tab. 8 przedstawiono porównanie wyników implementacji FPGA z implementacją ASIC opisaną w pracy [27]. Zaproponowana potokowa wersja algorytmu jest ok. 3,8-krotnie wydajniejsza przy 4,5-krotnie niższej częstotliwości pracy.

7. Akcelerator kryptograficzny

Zrealizowany moduł szyfrujący algorytmu CLEFIA zdecydowano się przetestować na dwóch kartach z układami FPGA: ADM-XP (Virtex II Pro 100) oraz RASC RC 100 (Virtex 4 LX 200).

Tab. 8. Porównanie częstotliwości pracy i przepustowości
Tab. 8. Frequency and throughput comparison

	V2P (0,13 µm)	V4 (90 nm)	ASIC (90 nm)
Maks. częstotliwość [MHz]	124	167	746,27
Przepustowość [MB/s]	1892,09	2548,22	663,35

7.2. Testowanie na karcie ADM-XP

Karta ADM-XP firmy Alpha-Data współpracuje z komputerem PC (hostem) poprzez magistralę PCI-X 64b/66 MHz. Podczas prowadzonych wcześniej prac, w oparciu o tę konfigurację stworzono działający akcelerator kryptograficzny dla algorytmu DES. Udało się wówczas osiągnąć maksymalny transfer danych na poziomie ok. 115 MB/s przy częstotliwości pracy ok. 80 MHz [16].

Stworzenie podobnego systemu w przypadku szyfru CLEFIA okazało się bardzo nieefektywne z powodu 128-bitowej magistrali danych (dla DES miała ona 64 bity). Z uwagi na konieczność buforowania danych (sklejania dwóch słów 64-bitowych w jedno słowo 128-bitowe) efektywny transfer i częstotliwość pracy zmalałaby dwukrotnie. Dla celów testowych w pamięci ROM umieszczono 32 wektory testowe, które poddano szyfrowaniu, a rezultat operacji został odczytany przez hosta. Karta ADM -XP umożliwia płynne ustalanie częstotliwości pracy zegara w zakresie od 6 MHz – 160 MHz. Maksymalne częstotliwości, przy których logika szyfrująca pracowała poprawnie wyniosły 110 MHz dla trybu *definition* i ok. 140 MHz dla pozostałych.

7.3. Testowanie na karcie RC 100

Karta RC 100 firmy SGI współpracuje z superkomputerem Altix 4700 umożliwiając tworzenie aplikacji programowo-sprzętowych, w których cześć obliczeń przeprowadzana jest w zasobach rekonfigurowalnych.

Logika na karcie RC 100 może pracować z programowalnymi częstotliwościami zegara 50 MHz, 66 MHz, 100 MHz i 200 MHz. Analiza danych zaprezentowanych w tab. 7 pokazuje, że wszystkie wersje algorytmu powinny dobrze działać przy częstotliwości 100 MHz. Tezę tę udało się potwierdzić eksperymentalnie, uruchamiając algorytm na karcie RC 100 i przeprowadzając testy na losowym zbiorze danych. Podobne testy przeprowadzone dla 200 MHz pokazały, że żadna z wersji nie działa poprawnie, co również jest zgodne z wynikami otrzymanymi w symulacji.

Maksymalny uzyskany transfer danych (1,2 GB/s) jest niższy od maksymalnej przepustowości modułu szyfrującego CLEFIA, pracującego przy częstotliwości 100 MHz (1,5 GB/s). Zatem, w przypadku omawianej aplikacji, częstotliwość 100 MHz wydaje się wystarczająca. W warunkach występowania górnego limitu transferu (1,2 GB/s), kiedy nie jest wymagane zastosowanie rozwiązania pracującego z najwyższą możliwą częstotliwością, wskazane jest użycie opracowanego modułu szyfrowania w wersji „definicja", ponieważ wymaga ono najmniejszych zasobów logicznych.

Zaprezentowany akcelerator kryptograficzny wykonuje szyfrowanie ponad 6 razy szybciej niż opisana w dokumencie [5] aplikacja programowa działająca na procesorze ogólnego przeznaczenia (190 MB/s na Athlon 4000+). Świadczy to o realnej przewadze rozwiązania rekonfigurowalnego nad programowym.

Ponieważ środowisko HPC karty RC 100 umożliwia transfer danych z kilkakrotnie większą prędkością niż popularne standardy (PCI, PCI-X, podstawowe wersje PCI-E) bardzo dobrze nadaje się do testowania wysokowydajnych algorytmów w układach FPGA.

8. Podsumowanie

W pierwszej części artykułu omówiono budowę współczesnych układów FPGA firm Xilinx i Altera, metody ich programowania i potencjalne zastosowania. Zasadnicza część artykułu dotyczy wykorzystania układów FPGA jako akceleratorów kryptograficznych. Zagadnienie to przedstawiono na przykładzie algorytmu CLEFIA. Dokonano analizy sposobu budowy pojedynczej rundy szyfrowania, w szczególności modułów S-Box. Zaproponowano nową wersję budowy F-funkcji – „*mixed*" łączącą zalety implementacji podstawienia S-Box zrealizowanego na podstawie definicji i przy pomocy operacji *look-up table*.

Porównanie szybkości działania zrealizowanego rozwiązania i opisanej w literaturze implementacji w układach ASIC pokazuje, że udało się osiągnąć znacznie większą wydajność (*throughput*) szyfrowania przy niższej częstotliwości pracy.

Moduł szyfrujący algorytmu CLEFIA uruchomiono na dwóch kartach z układami FPGA. Przeprowadzone eksperymenty pokazują, że w oparciu o kartę RC 100 współpracującą z superkomputerem SGI Altix 4700 możliwe jest tworzenie bardzo efektywnych aplikacji rekonfigurowalnych (akceleratorów sprzętowych), które ze względu na bardzo dobre parametry transferu danych umożliwiają w pełni wykorzystać potencjał obliczeniowy układów FPGA.

Podziękowania

Praca finansowana ze środków AGH (umowa AGH nr 11.11.120.612). Autor dziękuje pracownikom ACK CYFRONET AGH za udostępnienie platformy sprzętowej do badań.

Bibliografia

1. Altera, [www.altera.com] (dostęp 26.02.2012 r.).
2. Beuchat J.-L., *Modular multiplication for FPGA implementation of the IDEA block cipher*, IEEE International

Conf. on Application-Specific Systems, Architectures and Processors. Proceedings of: 412–422, 24–26 June 2003.
3. Boukerche A., Correa J.M., Melo A., Jacobi R.P., *A Hardware Accelerator for the Fast Retrieval of DIALIGN Biological Sequence Alignments in Linear Space*, IEEE Transactions on Computers, vol. 59, no. 6, 808–821, June 2010.
4. CatapultC, [www.calypto.com] (dostęp 26.02.2012 r.).
5. Clefia Evaluations, *Sony Corporation*, *The 128-bit Block Cipher CLEFIA Security and Performance Evaluations*, 2007.
6. Clefia Specification, *Sony Corporation, The 128-bit Block Cipher CLEFIA Algorithm Specification*, 2007.
7. Copacobana, [www.copacobana.org] (dostęp 26.02.2012 r.)
8. Fournaris A.P., Koufopavlou O., *Efficient CRT RSA with SCA Countermeasures*, 14th Euromicro Conference on Digital System Design (DSD), 593–599, 2011.
9. Gielata A., Russek P., Wiatr K., *Implementacja standardu szyfrowania AES w układzie FPGA dla potrzeb sprzętowej akceleracji obliczeń*, „Pomiary Automatyka, Kontrola", Stowarzyszenie Inżynierów i Techników Mechaników Polskich, vol. 53, nr 5, 48–50, 2007.
10. Gorgoń M., *Architektury rekonfigurowalne do przetwarzania i analizy obrazu oraz dekodowania cyfrowego sygnału wideo*, Uczelniane Wydawnictwa Naukowo-Dydaktyczne AGH, Kraków, 2007.
11. Jamro E., Wiatr K., Wielgosz, M., *FPGA Implementation of 64-Bit Exponential Function for HPC*, International Conference on Field Programmable Logic and Applications, FPL 2007, 718–721, 27–29 Aug. 2007.
12. Jin S., Kim D., Nguyen T.T., Kim D., Kim M., Jeon J.W., *Design and Implementation of a Pipelined Datapath for High-Speed Face Detection Using FPGA*, IEEE Transactions on Industrial Informatics, vol. 8, no. 1, 158–167, Feb. 2012.
13. Karabchevsky S., Kahana D., Ben-Harush O., Guterman H., *FPGA-Based Adaptive Speckle Suppression Filter for Underwater Imaging Sonar*, IEEE Journal of Oceanic Engineering, vol. 36, no. 4, 646–657, Oct. 2011.
14. Kitsos P., Sklavos N., Skodras A. N., *An FPGA Implementation of the ZUC Stream Cipher*, 14th Euromicro Conf. on Digital System Design (DSD), 814–817, 2011.
15. Kołek K., *FPGA-based PLC-like controller,* 12th IEEE international conference on Methods and Models in Automation and Robotics, 2006.
16. Kryjak T., Gorgoń M., *Akcelerator sprzętowy do szyfrowania strumienia danych*, Automatyka: półrocznik Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie, t. 12, z. 3, 695–707, 2008.
17. Kryjak T., Komorkiewicz M., Gorgoń M., *Real-time moving object detection for video surveillance system in FPGA,* The 2011 Conference on Design & Architecturesfor Signal and Image Processing (DASIP), 209–216, 2011.
18. LabVIEW, [www.ni.com/fpga] (dostęp 26.02.2012r.).
19. Maxeler, [www.maxeler.com/content/frontpage] (dostęp 26.02.2012 r.).
20. Mitrion C., [www.mitrionics.com] (dostęp 26.02.2012r.).
21. Ogiela M.R., *Systemy utajniania informacji*, Uczelniane Wyd. Naukowo-Dydaktyczne AGH, Kraków, 2003.
22. Petrovic M., Smiljanic A., Blagojevic M., *Design of thwitching Controller for the High-Capacity Non-Blocking Internet Router*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 17, no. 8, 1157–1161, Aug. 2009.
23. Rudra A., Dubey P. K., Julta C.S., Kumar V., Rao J. R., Rohatgi P., *Efficient Implementation of Rijandel Encryption with Composite Field Arithmetic*, CHES 2001, LNCS 2162, 171–184, Springer-Verlag, 2001.
24. Schneier B., *Kryptografia dla praktyków*, WNT, 1995
25. SONY, [www.sony.net/Products/cryptography/clefia/index.html], (dostęp 26.02.2012 r.).
26. Standaert F.-X., Rouvroy G., Quisquater, J.-J., *FPGA Implementations of the DES and Triple-DES Masked Against Power Analysis Attacks*, International Conference on Field Programmable Logic and Applications, 2006. FPL '06, 1–4, 28–30 Aug. 2006.
27. Sugawara T., Homma N., Aoki T., Satoh A., *High-performance ASIC Implementations of the 128-bit Block Cipher CLEFIA*, IEEE International Symposium on Circuits and Systems, 2008.
28. Wielgosz M., Jamro E., Wiatr K., *Feasibility of achieving high calculation speeds on the RASC platform*, ICSES 2008 International Conference On Signals and Electronic Systems, Krakow, September 14–17, 2008.
29. Xilinx, [www.xilinx.com] (dostęp 26.02.2012 r.).
30. Yunqiang Y., Fathy A.E., *Development and Implementation of a Real-Time See-Through-Wall Radar System Based on FPGA*, IEEE Transactions on Geoscience and Remote Sensing, vol. 47, no. 5, 1270–1280, May 2009.

Reconfigurable cryptographic accelerator

Abstract: This paper discusses the use of FPGA devices as cryptographic accelerators, which are able to perform the encryption or decryption operation faster and using less power than general-purpose processors while offering great flexibility and the ability to further develop and modify the design. In the first part, the structure and resources available in modern FPGAs are presented and in the second the implementation of the cryptographic algorithm CLEFIA is discussed.

Keywords: FPGA devices , cryptography, CLEFIA cipher

dr inż. Tomasz Kryjak

Jest asystentem w Laboratorium Biocybernetyki Katedry Automatyki i Inżynierii Biomedycznej AGH. Interesuje się przetwarzaniem i analizą obrazów oraz akceleracją algorytmów wizyjnych i kryptograficznych z wykorzystaniem układów FPGA.

e-mail: kryjak@agh.edu.pl