AUTOMATYKA • 2008 • Tom 12 • Zeszyt 3

Tomasz Kryjak*, Marek Gorgoń*

Akcelerator sprzętowy do szyfrowania strumienia danych**

1. Wprowadzenie

Wykorzystanie sprzętowej akceleracji w przetwarzaniu obrazów wydaje się w dzisiejszych czasach bardzo pożądane. Wynika to przede wszystkim z coraz większej ilości danych, które wymagają przetworzenia. Przykładowo wykonanie jednopunktowej operacji na wprowadzanym obecnie standardzie HDTV, o rozdzielczości 1920 × 1080 × 25 Hz wymusza przepustowość ponad 50 mln pikseli na sekundę. Przy założeniu, że reprezentacja pikseli jest ośmiobitowa, wymagany transfer wynosi ponad 1100 Mb/s. Obecnie dostępne komputery klasy PC z nowymi procesorami są w stanie przetwarzać takie ilości danych, jednak wiąże się to z wykorzystaniem praktycznie całej ich mocy obliczeniowej. Bardziej złożonych operacji, takich jak konwolucja, erozja, dylatacja kompresja najczęściej nie udaje się uruchomić w czasie rzeczywistym, tj. dla 25 klatek na sekundę w standardach europejskich [1].

Jednym z rozwiązań opisanego powyżej problemu jest użycie akceleratorów sprzętowych, w szczególności wykorzystujących układy FPGA (*Field Programmable Gate Array*). Układy te, dzięki możliwości zrównoleglenia obliczeń, przetwarzania potokowego i dopasowaniu architektury do konkretnego algorytmu są w stanie wykonywać operacje równie szybko jak procesory ogólnego przeznaczenia, pomimo iż te pracują z częstotliwością średnio o rząd wyższą. Dodatkowo rekonfigurowalność układów FPGA pozwala wykorzystywać ten sam układ do różnych zadań, np. filtracja, kompresja, szyfrowanie obrazu. Jeden akcelerator jest w stanie zastąpić kilka dedykowanych urządzeń.

Z budową akceleratorów wykorzystujących układy FPGA związany jest problem transferu danych pomiędzy pamięcią RAM komputera PC a płytą akceleratora. Dostępne zazwyczaj standardy komunikacyjne to port szeregowy, port równoległy, USB, Ethernet 1 Gb. Oferują one zbyt małą prędkość transferu. W konsekwencji, do szybko działającej logiki FPGA nie ma możliwości dostarczenia lub odebrania danych, co praktycznie unie-

* Katedra Automatyki, Akademia Górniczo-Hutnicza w Krakowie
** Pracę wykonano w ramach badań własnych (umowa AGH nr 10.10.120.783)

możliwia stworzenie użytecznego akceleratora i pokazania przewagi rozwiązania sprzętowego (wykorzystującego układ FPGA) nad programowym (zaimplementowanego na procesorze ogólnego zastosowania). Inną możliwością jest projekt i budowa własnej karty z układem FPGA oraz wydajnym interfejsem transmisji danych zbudowanym, np. przy użyciu światłowodów i technologii RocketIO firmy Xilinx [2]. Rozwiązaniem pośrednim, znacznie prostszym i wymagającym mniejszych nakładów, jest budowa akceleratora sprzętowego z wykorzystaniem gotowej, komercyjnej karty z układem FPGA, wyposażonej w szybki interfejs do komunikacji z komputerem PC np. XpressGen2V5 i PCI-X SYS V5 firmy PLDA [3], HTG-LX330T i PCIXSYS-V5 firmy HiTech Global [4], ADM-XRC-5T1, ADPe-XRC-4 i ADM-XP firmy Alpha Data [5] czy inne.

W niniejszej pracy opisano akcelerator sprzętowy, którego wydajność pozwala na szyfrowanie i deszyfrowanie strumienia danych z prędkością ok. 900 Mb/s, zbudowany z wykorzystaniem karty ADM-XP firmy Alpha Data z układem Virtex II Pro firmy Xilinx. Karta współpracuje z komputerem PC wyposażonym w magistralę PCI-64. Zaprezentowane rozwiązanie działa szybciej niż dostępne aplikacje programowe [6, 7], a maksymalną prędkość ogranicza przepustowość dysku twardego komputera PC i magistrali PCI-64.

2. Algorytm szyfrujący DES

Jako przykładową operację realizowaną przez akcelerator sprzętowy wybrano szyfrowanie danych algorytmem DES (*Data Encryption Standard*), głównie z powodu wcześniejszych prac nad jego implementacją na platformie FPGA.

Algorytm DES powstał na początku lat 70. XX w., a jako standard został wprowadzony w 1977 r. Jest jednym z najlepiej zbadanych algorytmów kryptograficznych w historii. W użyciu pozostawał przez ponad 20 lat. W tym czasie był wielokrotnie implementowany programowo i sprzętowo. Powstało też wiele prac na temat jego bezpieczeństwa i kryptoanalizy. DES został wycofany na początku XXI w., ponieważ rosnąca moc obliczeniowa komputerów i konstrukcja specjalizowanych urządzeń, takich jak [8], umożliwiła złamanie klucza w dość krótkim czasie, przy stosunkowo niewielkich nakładach finansowych. Do dzisiaj w użyciu pozostaje wersja 3DES, która wykorzystuje trzykrotne szyfrowanie DES. Zabieg ten zwiększa efektywną długość klucza z 56 do 112 bitów.

Algorytm DES [9] jest blokowym, symetrycznym szyfrem z 64-bitowymi blokami danych i 64-bitowym kluczem. Można podzielić go na dwa główne elementy funkcjonalne: 16 rund szyfrowania oraz moduł generacji podkluczy. Operacje wchodzące w skład wymienionych elementów to:

- permutacje (podstawienia),
- suma logiczna modulo 2 (XOR),
- nieliniowe podstawienia (SBOX) zrealizowane jako LUT (*Look-Up Table*),
- obrót wektora bitowego w lewo lub prawo (ROL, ROR).

Wymienione operacje można prosto zrealizować za pomocą zasobów logicznych układu FPGA. Wynika to z faktu, że jednym z wymagań stawianych algorytmowi kryptograficznemu, który miał zostać standardem szyfrowania, była jego prosta implementacja sprzętowa.

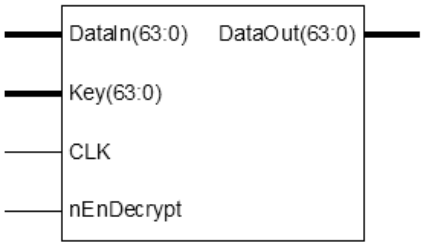
W najprostszym trybie pracy ECB (*Electronic Code Book*) szyfrowanie DES można zrealizować w sposób potokowy. Podstawową zaletą tego trybu jest duża prędkość, a główną wadą fakt, że dla danego klucza identyczne dane wejściowe zostaną zaszyfrowane identycznie, co obniża bezpieczeństwo. Trybu nie posiadającego tej wady – CBC (*Cipher Block Chaining*) – nie da się zaimplementować w sposób potokowy. Tryby pracy szyfrów symetrycznych zostały dokładnie omówione w dokumencie [10].

3. Wyniki implementacji DES

W wyniku prac nad implementacją algorytmu DES na platformie FPGA powstał potokowo działający moduł szyfrujący. Interfejs został przedstawiony na rysunku 1. Wejście stanowią sygnały:

DataIn – dane do zaszyfrowania/odszyfrowania,
Key – klucz,
CLK – zegar (moduł jest synchroniczny),
nEnDecrypt – wybór operacji (szyfrowanie lub deszyfrowanie).

Wyjściem z modułu jest sygnał DataOut – dane zaszyfrowane lub odszyfrowane. Zasoby FPGA, jakie zużywa moduł, ilustruje tabela 1.



Rys. 1. Interfejs modułu DES

Tabela 1
Zasoby FPGA zużywane przez moduł DES

LUT4	4217 (2%)*
FF	1912 (4%)*
SLICE	2492 (5%)*

* w nawiasie podano procent wykorzystania zasobów układu Virtex II Pro (2VP100)

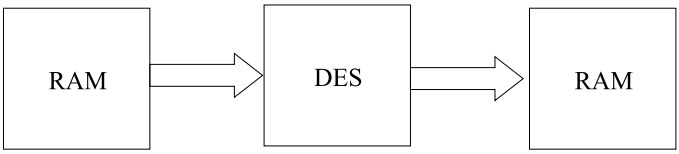
Maksymalna częstotliwość pracy modułu DES wyniosła ok. 80 MHz, przy 16-elementowym potoku. Prędkość pracy, mimo iż stosunkowo niewielka w porównaniu z najszybszymi opublikowanymi – 237 MHz [11] i 333 MHz [12] – okazała się wystarczająca, bowiem dla omawianej konfiguracji akceleratora nie było możliwe szybsze doprowadzenie strumienia danych do układu FPGA.

Moduł DES został zrealizowany w dwóch wersjach: za pomocą języka VHDL oraz w środowisku System Generator [13]. Poprawne działanie obu modułów zostało zweryfikowane symulacyjnie oraz poprzez testy w sprzęcie.

4. Uruchomienie algorytmu na platformie docelowej ADM-XP

Dostarczanie i odbieranie danych z układu FPGA stanowi spory problem i często jest przeszkodą w wykazaniu realnej przewagi implementacji sprzętowej nad programową. W trakcie wykonanych prac sprawdzano kilka kart z układami FPGA pod kątem ich przydatności do budowy akceleratora sprzętowego.

Karty typowo edukacyjne takie jak RC203 i RC300 firmy Celoxica (obecnie Agility [14]) nie sprawdzają się w roli akceleratorów. Nie dysponują dedykowanymi, wydajnymi modułami do transferu danych – najszybszy dostępny to Ethernet 1 Gb. Mogą stanowić zatem jedynie środowisko testowe. Metodologię testu modułu DES zaprezentowano na rysunku 2.



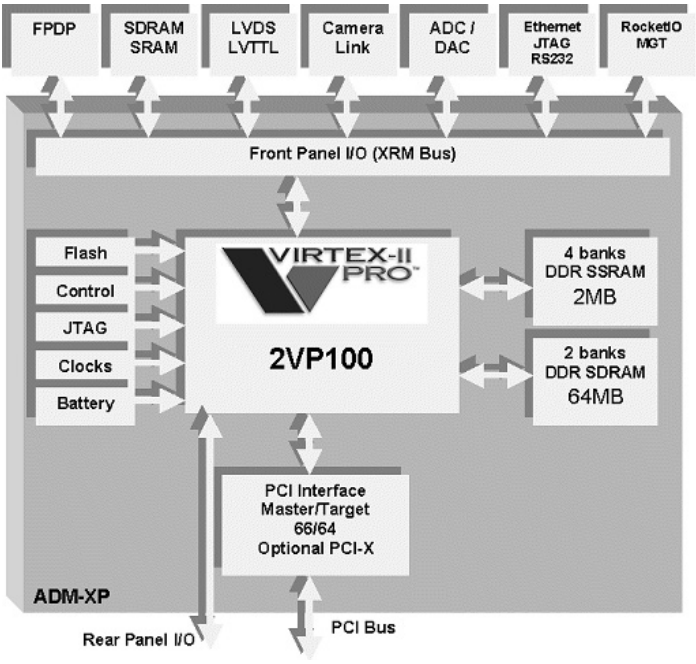
Rys. 2. Metodologia testu modułu DES

Dane do modułu DES dostarczane były z banku pamięci RAM i po przetworzeniu zapisywane były pod innym adresem lub w innym banku pamięci. Rozwiązanie to pozwoliło zweryfikować działanie modułu DES z prędkościami ok. 80 MHz – z taką częstotliwością poprawnie pracował kontroler pamięci RAM.

Osobnym problem stanowiło wgranie danych do RAM-u z komputera PC i odczytanie ich w celu weryfikacji poprawności algorytmu. W przypadku karty RC203 komunikacja odbywała się za pomocą portu szeregowego. Rozwiązanie to było proste w implementacji, ale dość podatne na błędy. Aby zapewnić spójność danych, należało wprowadzić sumy kontrolne, co dodatkowo spowolniło komunikację. Dostęp do banków pamięci RAM karty RC300 jest możliwy za pomocą USB, z poziomu dedykowanego programu dostarczanego przez producenta. Stanowi to znaczne ułatwienie dla programisty – niewielkim nakładem pracy można przetestować moduł w sprzęcie – nie jest konieczne tworzenie logiki do komunikacji z komputerem PC oraz odpowiedniej aplikacji programowej.

Obie wymienione karty są typowo edukacyjne – zawierają sporo modułów wejścia i wyjścia, głównie multimedialnych: DVI, VGA, S-Video, Audio. Pozwalają na przetestowanie algorytmów na platformie FPGA, ale stworzenie akceleratora sprzętowego z ich wykorzystaniem wydaje się zadaniem trudnym i nieefektywnym.

Kartą, która może stanowić podstawę akceleratora sprzętowego, jest ADM-XP firmy Alpha-Data. Urządzenie zawiera układ FPGA (Virtex II Pro 2VP100 firmy Xilinx [15]), banki pamięci DDR SRAM i DDR SDRAM, interfejs PCI oraz przedni i tylny moduł wejść/wyjść, które umożliwiają podłączenie dodatkowych modułów RAM, przetworników AC/CA oraz interfejsu Camera Link [16]. Architektura przedstawiona została na rysunku 3. Karta posiada jeszcze dwie dodatkowe cechy, które ułatwiają budowę akceleratora: możliwość transferu danych poprzez DMA (*Direct Memory Access*) oraz rozbudowane wsparcie programowe w postaci pakietu SDK (*Software Development Kit*). W jego skład wchodzą: biblioteka do obsługi karty z poziomu języka C/C++ oraz liczne przykładowe aplikacje.



Rys. 3. Schemat budowy karty ADM-XP [5]

Kartę ADM-XP podłącza się do komputera PC za pomocą odpowiedniego adaptora. W opisywanych pracach wykorzystywany był moduł ADC-PCM. Urządzenie, dzięki mostkowi PCI-PCI Intel 21154 ma możliwość pracy w standardzie PCI 2.2. w trybie 64 bity/66 MHz (transfer 533 MB/s). Wspierany jest też tryb 32 bity/33 MHz (transfer 133 MB/s). Kartę ADM-XP można podłączyć także do adaptora ADC-EMC pracującego w standardzie PCI-X. Więcej informacji w dokumentacji [5].

5. DMA – podstawa budowy akceleratora sprzętowego

Analiza funkcjonalności ADM-XP pokazała, że do przesyłania danych między kartą a komputerem PC można wykorzystać DMA (*Direct Memory Access*) – bezpośredni dostęp urządzeń pracujących na magistrali PCI do pamięci systemowej RAM. Znacznym ułatwieniem była przykładowa aplikacja, wchodząca w skład SDK, realizująca transfer DMA.

Przykładowa aplikacja DMA składa się z dwóch części:

1) programowej – kod C,
2) sprzętowej – kod VHDL, Verilog oraz pliki konfiguracyjne „bit".

Działa ona według następującego schematu: dane odczytywane są przez kartę z bufora aplikacji w pamięci komputera PC i niezmienione zapisywane w innym buforze. Do odczytu wykorzystywany jest kanał 0 DMA, a do zapisu kanał 1 DMA. W układzie FPGA dane buforowane są w kolejce FIFO.

Aplikacja programowa wykorzystuje dwa wątki – pierwszy wysyła dane, a drugi je odbiera. Ponadto pokazana jest szybka konfiguracja układu FPGA poprzez DMA – zmierzony czas to ok. 110 ms dla pliku konfiguracyjnego o wielkości 4 268 670 bajtów. Istnieje także możliwość pomiaru prędkości transferu oraz weryfikacji poprawności danych.

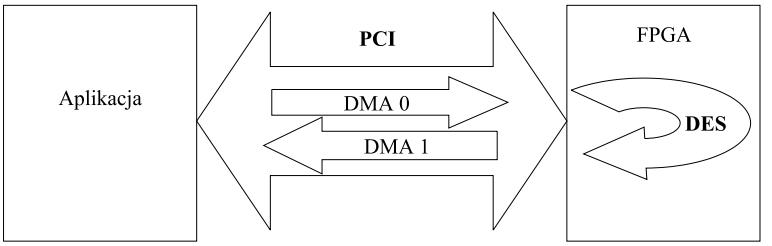
Przeprowadzone zostały testy transferu. Wyniki dotyczą transferu dwukierunkowego, czyli sumy danych wysłanych i odebranych.

– tryb 32 bity, zegar lokalnej magistrali 33 MHz – 113 MB/s – 904 Mb/s,
– tryb 32 bity, zegar lokalnej magistrali 80 MHz – 213 MB/s – 1704 Mb/s,
– tryb 64 bity, zegar lokalnej magistrali 80 MHz – 228 MB/s – 1824 Mb/s.

Największy uzyskany transfer to ok. 230 MB/s. Nie udało się ściśle ustalić, dlaczego nie jest możliwa praca z maksymalną prędkością (533 MB/s). Prawdopodobna przyczyna to nieoptymalność aplikacji FPGA.

6. Akcelerator sprzętowy

Koncepcja akceleratora sprzętowego do szyfrowania danych została przedstawiona na rysunku 4. Składa się on z części programowej (C++) oraz sprzętowej (FPGA, VHDL).



Rys. 4. Schemat działania akceleratora sprzętowego do szyfrowania

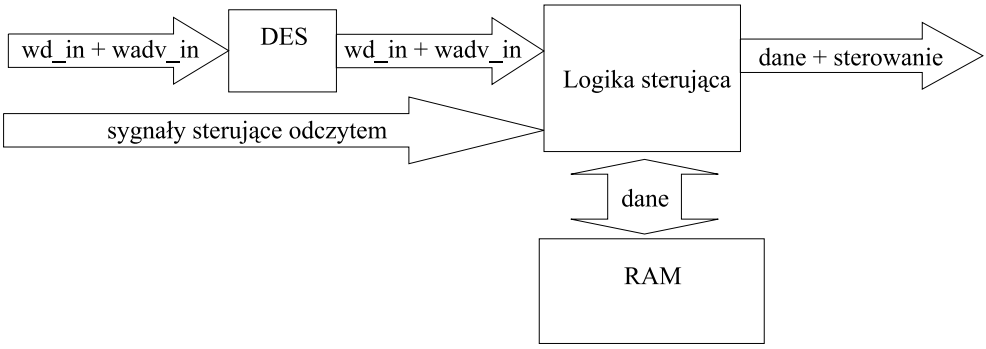
6.1. Aplikacja C++

Aplikacja C++, nazwana „CryptoCoProcessor", napisana została w środowisku Microsoft Visual Studio 2005, jako okienkowa aplikacja MFC (*Microsoft Foundation Classes*). Stworzona została na podstawie części programowej przykładu DMA z SDK (*Software Development Kit*) do karty ADM-XP [5], przy czym poczynione zostały znaczne modyfikacje. Zamiast oryginalnie użytego języka C, zastosowano C++ oraz technologię obiektową i interfejs graficzny. Usunięto zbędne funkcje oraz dodano nowe, potrzebne funkcjonalności: wczytywanie danych z plików i obrazów w formacie bmp.

6.2. Aplikacja FPGA

Sprzętowa część prezentowanego akceleratora powstała z połączenia opisanego w rozdziale 3 modułu DES oraz kodu VHDL przykładowej aplikacji do transferu DMA. Integracja polegała na wstawieniu modułu DES w istniejącą ścieżkę danych DMA -> FIFO -> DMA. Przeanalizowano kilka rozwiązań i ostatecznie zdecydowano się na wymagające najmniejszych modyfikacji w istniejącym kodzie – umieszczenie modułu DES wewnątrz modułu FIFO.

Na rysunku 5 zaprezentowano sposób, w jaki wstawiono moduł DES w istniejącą logikę FIFO. Wykorzystano fakt, że istnieją tylko dwa sygnały wejściowe sterujące zapisem: wd_in (dane wejściowe) oraz wadv_in (sygnał sterujący „zapisz i zwiększ wskaźnik" kolejki). Dane zostały wprowadzone na wejście modułu DES, a sygnał wadv_in został opóźniony o 16 taktów zegara. Następnie zaszyfrowane dane i opóźniony wadv_in trafiają do logiki sterującej FIFO. W ten sposób moduł DES w żaden sposób nie wypływa na istniejącą logikę. W trakcie testów okazało się, że z uwagi na 16-taktowe opóźnienie konieczne jest zwiększenie rozmiaru kolejki do 1024 (oryginalny rozmiar to 512). Przedstawione rozwiązanie pozwoliło w prosty i szybki sposób połączyć własny moduł szyfrujący DES z istniejącą logiką przykładu DMA.



Rys. 5. Schemat modułu FIFO wraz z modułem DES

6.3. Konfiguracja modułu DES

Ostatnim problemem, jaki należało rozwiązać, było wprowadzanie danych konfiguracyjnych do modułu. W przypadku algorytmu DES konieczne było przekazanie klucza oraz informacji o tym, czy moduł ma szyfrować, czy deszyfrować, a także rozmiaru szyfrowanych danych.

Zdecydowano się na stworzenie prostego protokołu komunikacyjnego. Pojedynczy pakiet, będący jednocześnie całym buforem DMA, przedstawiono na rysunku 6. Wszystkie pola są 64-bitowe. Pakiet zaczyna się od sekwencji startowej: '1' dla szyfrowania i '2' dla deszyfrowania, następne pola to klucz, rozmiar danych, pusty blok oraz *n*-bloków właściwych danych. W logice FPGA zrealizowano maszynę stanową, która na podstawie kolejnych bloków z pakietu danych odpowiednio steruje procesem szyfrowania lub deszyfrowania.

Rys. 6. „Pakiet" danych – konfiguracja buforu DMA

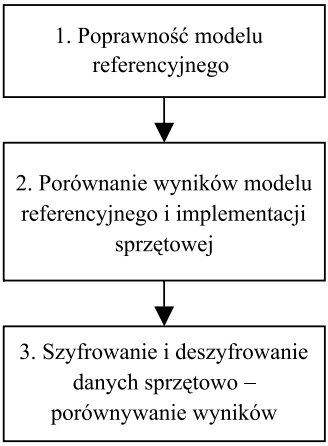
7. Testy rozwiązania

Testowane były następujące aspekty zaprezentowanego rozwiązania:

- poprawność implementacji,
- szybkość działania,
- sposób szyfrowania obrazów.

7.1. Poprawność implementacji

Poprawność działania algorytmu DES została zweryfikowana w trzech etapach, przedstawionych na rysunku 7. Na początku stworzono model referencyjny w języku C++ i porównano jego działanie z dostępnymi w literaturze wektorami testowymi. Następnie szyfrowano te same dane wersją programową i sprzętową – porównywano wyniki działania. Ostatni etap to szyfrowanie i deszyfrowanie danych sprzętowo i porównywanie wyników – dane wyjściowe powinny być identyczne jak przed szyfrowaniem. Moduł DES przeszedł podane powyżej testy pomyślne dla kilku różnych zestawów kluczy i plików, co wskazuje na poprawność działania wykonanej implementacji.



Rys. 7. Etapy weryfikacji sprzętowej implementacji algorytmu DES

7.2. Prędkość implementacji

Do sprawdzenia prędkości działania wykorzystano mechanizm zaproponowany w wykorzystanej, przykładowej aplikacji DMA. Używa on funkcji systemowej *GetTickCount()*, która zwraca liczbę milisekund, jakie upłynęły od uruchomienia systemu operacyjnego.

Operacje szyfrowania lub deszyfrowania danych można podzielić na trzy etapy:

1. Wczytanie danych z dysku.
2. Właściwe szyfrowanie lub deszyfrowanie oraz transfery poprzez DMA pomiędzy hostem a kartą ADM-XP.
3. Zapisanie danych na dysku.

Maksymalny transfer danych pomiędzy dyskiem a pamięcią RAM zależy głównie od prędkości dysku i dla współczesnych urządzeń wynosi ok. 560÷640 Mb/s (70÷80 MB/s). Wartość tę potwierdziły przeprowadzone testy.

Z uwagi na sposób integracji modułu DES ze sprzętową częścią aplikacji DMA (podrozdział 6.2) wykonanie transferu DMA połączone jest z operacjami szyfrowania lub deszyfrowania. Zatem prędkość działania (ok. 900 Mb/s) jest ograniczona przez prędkość transferu DMA. Taką też prędkość jest w stanie osiągnąć akcelerator kryptograficzny, przy założeniu, że dane przechowywane są tylko w pamięci RAM hosta.

W aplikacji C++ zaimplementowane zostały dwa różne tryby obsługi plików. W pierwszym dane wczytywane są do pamięci RAM, następnie wykonywane jest szyfrowanie poprzez transmisję DMA, a po jej zakończeniu wynik zapisywany jest na dysk. W trybie tym nie jest możliwe szyfrowanie dużych plików, z uwagi na ograniczony rozmiar bufora DMA. Średnie wartości prędkości działania akceleratora dla tego trybu, zaprezentowane zostały w tabeli 2.

Tabela 2
Średnia prędkość działania akceleratora sprzętowego do szyfrowania – pierwszy tryb obsługi plików

		Średnia prędkość [Mb/s]			
Plik	Rozmiar [B]	Odczyt*	DMA**	Zapis***	Łączna****
[1]	4286670	695.84	930.43	695.84	253.23
[2]	13500416	600.65	859.52	598.83	267.64

* HDD – RAM, **RAM – ADM-XP – RAM, *** RAM – HDD, **** HDD – ADM-XP – HDD

W drugim trybie obsługi plików wykorzystano współbieżnie wykonywane wątki. Pierwszy wątek odpowiada za odczytanie danych z dysku i wysyłanie ich kanałem DMA, a drugi za odbiór danych z kanału DMA i zapis ich na dysk. Rozmiar bufora DMA został ustalony na 64 kB. Rozwiązanie to pozwala na szyfrowanie i deszyfrowanie plików o praktycznie nieograniczonym rozmiarze. W tabeli 3 zaprezentowano średnie prędkości działania akceleratora dla tego trybu.

Tabela 3
Średnia prędkość działania akceleratora sprzętowego
do szyfrowania – drugi tryb obsługi plików

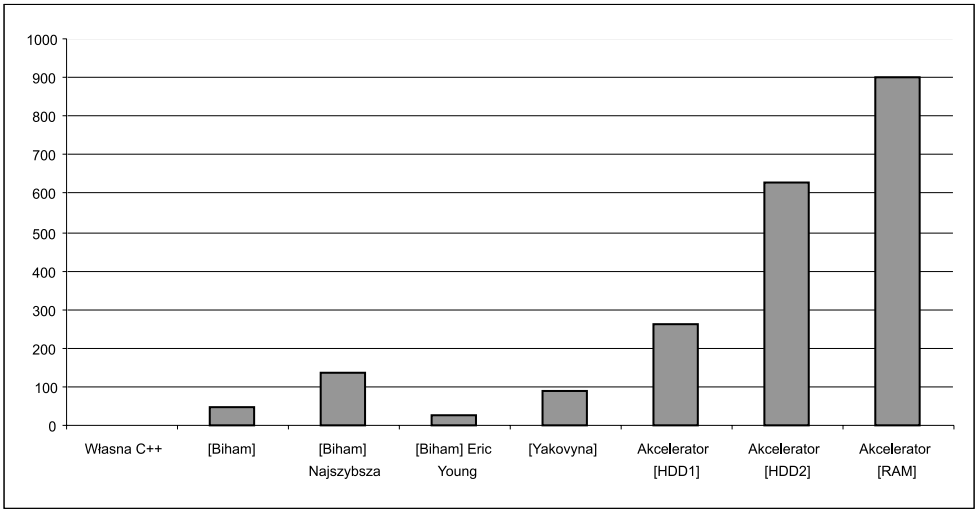
Plik	Rozmiar [B]	Średnia prędkość [Mb/s]
[1]	4286670	620
[2]	13500416	600
[3]	102553405	630
[4]	196044536	625

Analiza wyników zaprezentowanych w tabelach 2 i 3 pokazuje, że przy operacjach na danych zapisanych na dysku większą prędkość szyfrowania uzyskuje się w drugim trybie obsługi plików. Wynosi ona ok. 600 Mb/s, tj. ponad 70 MB/s. Przy szyfrowaniu z wykorzystaniem pamięci RAM uzyskana prędkość to ok. 900 Mb/s, tj. ok. 110 MB/s. Niewielkie różnice w zaprezentowanych wynikach wynikają z metodologii pomiaru oraz sposobu działania wielozadaniowego systemu operacyjnego.

Na rysunku 8 zaprezentowano porównanie prędkości działania implementacji programowych oraz opisywanego akceleratora kryptograficznego. Na podstawie jego analizy można stwierdzić, że akcelerator (wersja Akcelerator [HDD2]) jest ok. 350 razy szybszy od zaproponowanej własnej implementacji programowej oraz ok. 4,5 razy szybszy od najszybszej implementacji programowej (zrealizowanej z wykorzystaniem specjalistycznego procesora) opisanej w literaturze [6]. W obu przypadkach brano pod uwagę szyfrowanie z odczytem i zapisem danych z dysku.

Wynik uzyskany dla szyfrowania: RAM – akcelerator – RAM (Akcelerator [RAM]) ilustruje duże możliwości akceleratorów z układami FPGA. Praktyczne wykorzystanie

mocy obliczeniowych oferowanych przez urządzenie, uzależnione jest od możliwości współpracy z pamięcią masową komputera nadrzędnego. Rozwiązanie oferujące większy stopień akceleracji mogłoby powstać jako dedykowany system do szyfrowania, poza środowiskiem komputera klasy PC, z użyciem dużych zasobów szybkiej pamięci RAM lub bardzo szybkich interfejsów komunikacyjnych.



Rys. 8. Porównanie prędkości implementacji programowych i akceleratora kryptograficznego

W tabeli 4 zamieszczono prędkości przykładowych implementacji programowych oraz uzyskane, dzięki akceleratorowi sprzętowemu przyspieszenie. Analiza pokazuje, że zaproponowane rozwiązanie jest 4,4 razy szybsze od najszybszego programowego – wykorzystującego procesor o specjalnej architekturze [6] oraz 6,8 razy szybsze od rozwiązania wykorzystującego popularny szybki procesor [7].

Tabela 4
Prędkości implementacji programowych algorytmu DES oraz uzyskane przyspieszenie

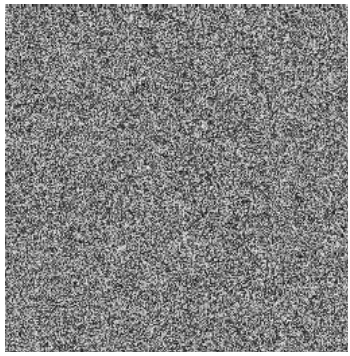
		Uzyskane przyspieszenie (*speed up*)*		
Implementacja	Prędkość [Mb/s]	HDD1	HDD2	RAM
Własna programowa	1,7	× 153	× 353	× 530
[6] Standardowa	46	× 5,7	× 13,0	× 19,6
[6] Najszybsza	137	× 1,9	× 4,4	× 6,6
[6] Eric Young	28	× 9,3	× 21,4	× 32,1
[7]	88,8	× 2,9	× 6,8	× 10,1

* prędkość implementacji sprzętowej / prędkość implementacji programowej

7.3. Przykładowa aplikacja – szyfrowanie obrazów

Zasadniczo szyfrowanie obrazów nie różni się od szyfrowania innych danych. Konieczne jest tylko „pozostawienie" oryginalnego nagłówka pliku „bmp", tak aby możliwe było graficzne przeglądanie wyników szyfrowania. Do testu wykorzystano obraz „Lena.bmp". Na rysunku 9 zaprezentowano oryginał, a na rysunku 10 wersję zaszyfrowaną (klucz: 133457799BBCDFF5 (hex)).

Rys. 9. Lena.bmp – oryginał

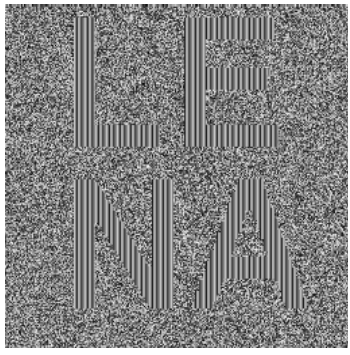


Rys. 10. Lena.bmp – zaszyfrowane

Graficzna prezentacja danych pozwala lepiej zaobserwować wadę trybu szyfrowania ECB.

Na rysunku 11 zaprezentowano zmodyfikowaną wersję obrazka „Lena.bmp" z dodanym napisem LENA, a rysunku 12 wersję zaszyfrowaną.

Rys. 11. LenaMod.bmp – oryginał



Rys. 12. LenaMod.bmp – zaszyfrowane

W trybie ECB, jeżeli identyczne dane wejściowe – np. piksele o tej samej wartości jasności – zaszyfrowane zostaną takim samym kluczem, to również w wynikowym szyfrogramie będzie to odzwierciedlone. Dlatego na rysunku 12 można zaobserwować widoczny kontur napisu LENA.

8. Wnioski

Celem opisanych w artykule prac było stworzenie akceleratora sprzętowego do szyfrowania strumienia danych. Użyto karty FPGA ADM-XP wyposażonej w układ Virtex II Pro (2VP100), komunikującej się z hostem poprzez magistralę PCI-64. W rezultacie badań powstał działający sprzętowy moduł DES oraz aplikacja programowo-sprzętowa (C++, FPGA) zapewniająca szyfrowanie lub deszyfrowanie danych przy odczycie i zapisie z dysku twardego z prędkością ok. 600 Mb/s, a przy odczycie i zapisie z pamięci RAM z prędkością ok. 900 Mb/s. W stosunku do najszybszej implementacji programowej, uzyskano ponad czterokrotne (odczyt z dysku) oraz ponad sześciokrotne (odczyt z RAM) przyspieszenie działania. Tym samym pokazano realną przewagę rozwiązania programowo-sprzętowego nad programowymi.

Warto podkreślić, że stworzony szablon można wykorzystać do budowy dowolnego innego akceleratora sprzętowego dla algorytmów potokowych. Zastosowanie karty FPGA z magistralą PCI-64 eliminuje „wąskie gardło", jakim często jest sam transfer danych między komputerem PC a układem FPGA. Karta ADM-XP zawiera dość nowoczesny układ FPGA, o całkiem sporej pojemności, oraz spore zasoby pamięci RAM. Powinno to umożliwić w przyszłości akcelerację bardziej złożonych algorytmów.

Literatura

[1] Gorgoń M., *Architektury rekonfigurowalne do przetwarzania i analizy obrazu oraz dekodowania cyfrowego sygnału wideo*. Kraków, UWND AGH 2007.
[2] Hao Xu, Zhan'an Liu, Yunpeng Lu, Lu Li, Dixin Zhao, and Ya'nan Guo, *FPGA Based High Speed Data Transmission with Optical Fiber in Trigger System of BES II.* IEEE Nuclear Science Symposium Conference Record, Vol. 1, 2007, 818–821.
[3] http://www.plda.com/.
[4] http://www.hitechglobal.com/.
[5] http://www.alpha-data.com/.
[6] Eli Biham – *A Fast New DES Implementation in Software*. 1997.
[7] Yakovyna V., Fedasyuk D., Seniv M., *Software Realization and Performance Testing of DES Cryptographic Algorithm on the .NET Platform*. 2007.
[8] http://www.copacobana.org/.
[9] FIPS PUB 46-3 *Data Encryption Standard (DES)*, 1999 http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.
[10] FIPS PUB 81 *DES modes of operation*, 1980.
[11] Pasham V., Trimberger S., *High – Speed DES and Triple DES Encryptor/Decryptor*. 2001.
[12] Rouvroy G., Standaert F.X., Quisquater J.-J., Legat J.-D., *Efficient Uses of FPGAs for Implementations of DES and Its Experimental Linear Cryptanalysis.* 2003.
[13] http://www.xilinx.com/ise/optional_prod/system_generator.htm.
[14] http://www.agilityds.com/.
[15] http://www.xilinx.com/.
[16] http://www.alacron.com/downloads/vncl98076xz/CameraLinkSPEC.pdf.