Andrzej SKORUPSKI [1], Marek PAWŁOWSKI [2], Krzysztof GRACKI [2], Paweł KERNTOPF [2,3]
[1] WYŻSZA SZKOŁA MENEDŻERSKA, WYDZIAŁ INFORMATYKI STOSOWANEJ I TECHNIK BEZPIECZEŃSTWA, ul. Kawęczyńska 36, 03-772 Warszawa
[2] POLITECHNIKA WARSZAWSKA, WYDZIAŁ ELEKTRONIKI I TECHNIK INFORMACYJNYCH, ul. Nowowiejska 15/19, 00-665 Warszawa
[3] UNIWERSYTET ŁÓDZKI, WYDZIAŁ FIZYKI I INFORMATYKI STOSOWANEJ, ul. Pomorska 149/153, 90-236 Łódź

Modelowanie w FPGA szyfratorów implementowanych w logice odwracalnej

Dr inż. Andrzej SKORUPSKI

Docent w Wyższej Szkole Menedżerskiej w Warszawie. Autor wielu publikacji dotyczących projektowania układów cyfrowych i architektury komputerów. Brał udział w wielu projektach różnych urządzeń i systemów cyfrowych wykorzystywanych zarówno w dydaktyce, jak i w pracach badawczych.

e-mail: A.Skorupski@ii.pw.edu.pl

Mgr inż. Krzysztof GRACKI

Ukończył studia magisterskie na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. Obecnie pracuje jako wykładowca w Instytucie Informatyki na tym Wydziale. Interesuje się grafiką komputerową i projektowaniem układów cyfrowych.

e-mail: K.Gracki@ii.pw.edu.pl

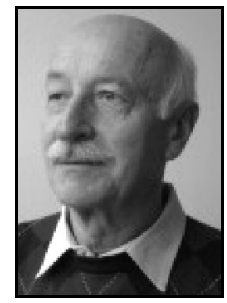
Mgr inż. Marek PAWŁOWSKI

Ukończył studia magisterskie na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. Obecnie pracuje jako wykładowca w Instytucie Informatyki na tym Wydziale. Interesuje się syntezą układów cyfrowych w strukturach FPGA oraz wspomaganiem komputerowym projektowania.

e-mail: M.Pawlowski@ii.pw.edu.pl

Dr hab. inż. Paweł KERNTOPF

Ukończył studia na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. Obecnie pracuje na stanowisku profesora nadzwyczajnego w Instytucie Informatyki na tym Wydziale i w Katedrze Fizyki Teoretycznej i Informatyki na Wydziale Fizyki i Informatyki Teoretycznej Uniwersytetu Łódzkiego. Jego zainteresowania naukowe to synteza układów logicznych, odwracalne układy logiczne, kwantowe układy logiczne, binarne i wielowartościowe diagramy decyzyjne.



e-mail: P.Kerntopf@ii.pw.edu.pl

Streszczenie

Idea projektowania cyfrowych układów w logice odwracalnej jest wykorzystywana do budowy układów małej mocy. Modelowanie takich układów stało się możliwe dzięki zastosowaniu współczesnych narzędzi symulacyjnych stosowanych do programowania układów FPGA. W niniejszym artykule pokazano wykorzystanie logiki odwracalnej do szyfrowania i przykładową implementację takiego układu. Dla zwiększenia złożoności szyfratora rozbudowano go o programowaną matrycę krosującą zmieniająca kolejność sygnałów wejściowych oraz o układ przekształcania klucza szyfrującego.

Słowa kluczowe: odwracalne układy logiczne, szyfrowanie, układy FPGA.

FPGA-based modeling of encryption systems implemented in reversible logic

Abstract

A circuit (gate) is called reversible if there is one-to-one correspondence between its inputs and outputs. Research on reversible logic circuits is motivated by advances in quantum computing, nanotechnology and low-power design. Therefore, reversible logic synthesis has been recently intensively studied. The attention is focused mainly on the synthesis of circuits built from the NCT library of gates, i.e. NOT, CNOT and Toffoli gates. Many developers work with design of classical digital devices like registers, adders, processors etc. using reversible circuits. Recently they have also tried to build more complex devices like for example an encryption devices [4, 5, 6, 7], however, only for saving energy. The other point of view, presented in this paper, is to use some features of reversible function. One of them is a big number of functions. For *n* variables there exist 2^n! different function. There are 24 reversible functions for 2 variables, 40320 functions for 3 variables and more than 20×10^{12} for 4 variables. Synthesis of circuits using 8 variable reversible function is too complicated. We use two cascades using 4 variable reversible function. We consider a 16-gates cascade. Depending on a given reversible function different cascade circuits will be obtained. These circuits correspond to a cryptographic key. Because we assume a 16-gates cascade and there exist 32 various gates we use 80-bit key for a 4-input cascade. Hence, for two cascades a cryptographic key will consist of 160 bits. Modern simulation tools based on FPGAs have enabled modeling of such circuits. In the paper we study application of reversible logic to developing encryption circuits. The results of FPGA-based simulation of a simple encryption circuit implemented built from reversible gates are also presented.

Keywords: reversible logic circuits, encryption, FPGA.

1. Wprowadzenie

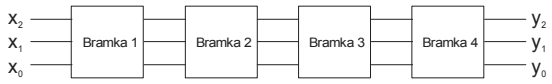
Bardzo mały pobór mocy przez układy odwracalne powoduje, że próbuje się je wykorzystywać w projektowaniu urządzeń stosowanych w różnych dziedzinach [1]. W wielu pracach wykorzystana jest ta własność, a autorzy koncentrują się na powielaniu standardowych układów cyfrowych (sumatory, rejestry, procesory itp.) w technice odwracalnej i zastosowaniu ich w różnych dziedzinach, w tym w kryptografii. Natomiast logika odwracalna ma drugą własność, polegającą na tym, że dla n zmiennych istnieje 2^n! różnych funkcji.

Tak duża liczba funkcji pozwala na wielką różnorodność konwersji słów wejściowych na słowa wyjściowe. Cecha ta predysponuje układy odwracalne do zastosowań kryptograficznych. W niniejszej pracy przedstawiono jedną z możliwych koncepcji wykorzystania logiki odwracalnej w takim zastosowaniu. Ponieważ fizyczna realizacja złożonych funkcji odwracalnych [1] jest w chwili obecnej trudna, to posłużono się modelem układów odwracalnych zaimplementowanych w strukturze FPGA.

Funkcje odwracalne n zmiennych, to zespół n funkcji boolowskich, które są funkcjami zrównoważonymi, tj. takimi, w których dla połowy kombinacji zmiennych wejściowych funkcja przyjmuje wartość 1, a dla drugiej połowy wartość 0. Można pokazać, że dla 2 zmiennych istnieją 24 (4!) różne funkcje odwracalne. Dla 3 zmiennych jest ich 40320 (8!), a dla 4 zmiennych ponad 20 bilionów (16!).

Każdą z funkcji odwracalnych można zrealizować wykorzystując specjalne bramki. Bramki te wykorzystują funktory XOR umieszczane na danej linii wejściowej. Jeśli bramka składa się tylko z tego funktora, którego argumentami są wybrane wejście oraz jedynka logiczną, to bramka odwracalna jest bramką negacji N. Jeśli drugie wejście funktora XOR dołączone jest do innego wejścia, to bramka jest nazywana bramką C. Jeśli do drugiego wejścia funktora XOR dołączony jest funktor AND stanowiący iloczyn wybranych wejść, to bramka jest nazywana bramką T. Dla trzech zmiennych jest 12 takich bramek. Dla 4 zmiennych są 32 takie bramki.

Implementacji dowolnej funkcji odwracalnej można dokonać za pomocą kaskadowego układu składającego się wyłącznie z wyżej omówionych bramek (rys. 1).



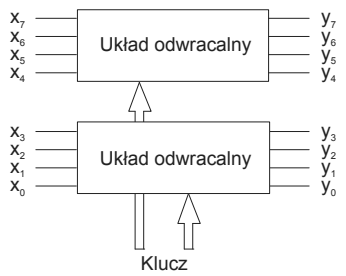
Rys. 1. Kaskadowy układ złożony z bramek odwracalnych
Fig. 1. Cascade circuit built from reversible gates

Każdą funkcję 3 zmiennych da się zrealizować za pomocą co najwyżej 8 bramek. Każdą funkcję 4 zmiennych da się zrealizować za pomocą co najwyżej 15 bramek [4].

2. Szyfrowanie podstawieniowe

Załóżmy, że postawione zadanie brzmi następująco: zaprojektować szyfrator strumieniowy, na którego wejścia podawana jest informacja w postaci bajtu. Jest to układ o 8 wejściach i 8 wyjściach. Synteza układów odwracalnych w takim przypadku jest bardzo złożona, bo liczba funkcji odwracalnych 8 zmiennych jest 256! Można ją uprościć dekomponując 8-wejściowy układ na dwa układy 4-wejściowe.

Na rys. 2 pokazano układ szyfrujący bajty składający się z dwóch czterowejściowych układów odwracalnych. Załóżmy, że oba układy czterowejściowe realizują różne funkcje odwracalne i że składają się z 16 bramek. Istnieją 32. różne bramki 4 zmiennych. Do zakodowania typu bramki potrzeba więc co najmniej 5 bitów.



Rys. 2. Dekompozycja układu 8-wejsciowego
Fig. 2. Decomposition of an 8-input circuit

Do określenia całego układu 4-wejściowego potrzeba zatem co najmniej 80 bitów (dla 16. bramek). Zatem długość klucza szyfrującego powinna wynosić 160 bity (dwa niezależne klucze po jednym dla każdego z układów). Ponieważ każdy z układów może realizować $20x10^{12}$ funkcji, to cały układ może realizować $400x10^{24}$ funkcji.

Oczywiście niektóre funkcje należy pominąć jako słabo nadające się do szyfrowania. Jednak funkcji afinicznych, które są łatwe do łamania jest zaledwie 322560 [3].

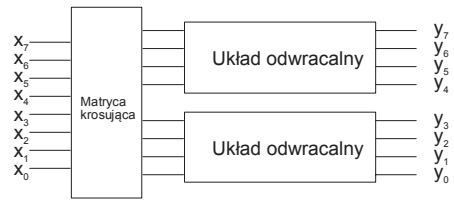
3. Zwiększanie złożoności szyfrowania

Przedstawiony wyżej sposób szyfrowania oparty jest na koncepcji szyfru podstawieniowego [7]. Złożoność szyfrowania można zwiększyć na kilka sposobów. Dwa z nich przedstawiono poniżej.

3.1. Mieszanie bitów

Jedną z prostych możliwości zwiększenia złożoności szyfrowania może być zastosowanie dwóch matryc krosujących na wejściu i wyjściu układu jak pokazano na rysunku 3. Matryca krosująca występuje na wejściu szyfratora i wyjściu deszyfratora.

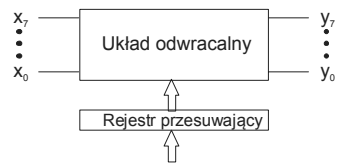
Takie rozwiązanie zwiększa długość klucza o 16 bitów (liczba możliwych krosowań wynosi 8!, bo tyle jest różnych możliwości połączeń wejść i wyjść matrycy krosującej).



Rys. 3. Układ 8-wejsciowy z programowanym krosowaniem
Fig. 3. An 8-input circuit with programmed crossing

3.2. Dynamiczna zmiana funkcji

Złożoność szyfrowania można zwiększyć projektując układ ze zmiennymi funkcjami. Mogą one zmieniać się dynamicznie w czasie szyfrowania. Takie rozwiązanie utrudnia złamanie szyfru nawet po stracie klucza. Na rys. 4 przedstawiono schematycznie jedną z możliwości dynamicznej zmiany klucza. Klucz jest zapisywany do *Rejestru klucza*, a następnie przepisywany do *Rejestru przesuwającego* w momencie rozpoczęcia szyfrowania wiadomości. Dla każdego bajtu wejściowego rejestr przesuwający z rotacją zawartości zmienia układ bramek, czyli realizowaną funkcję odwracalną.



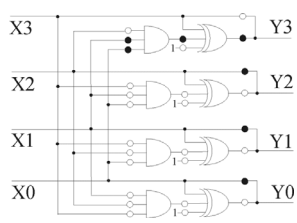
Rys. 4. Układ 8-wejściowy z dynamiczną zmianą funkcji
Fig. 4. An 8-input circuit with dynamic change of functioning

4. Implementacja szyfratora

Szyfrator (i deszyfrator) zaprojektowano w układzie FPGA (firmy Altera). W szyfratorze bramki są ustawione w kolejności od numeru 0 do 15 a deszyfratorze od 15 do 0.

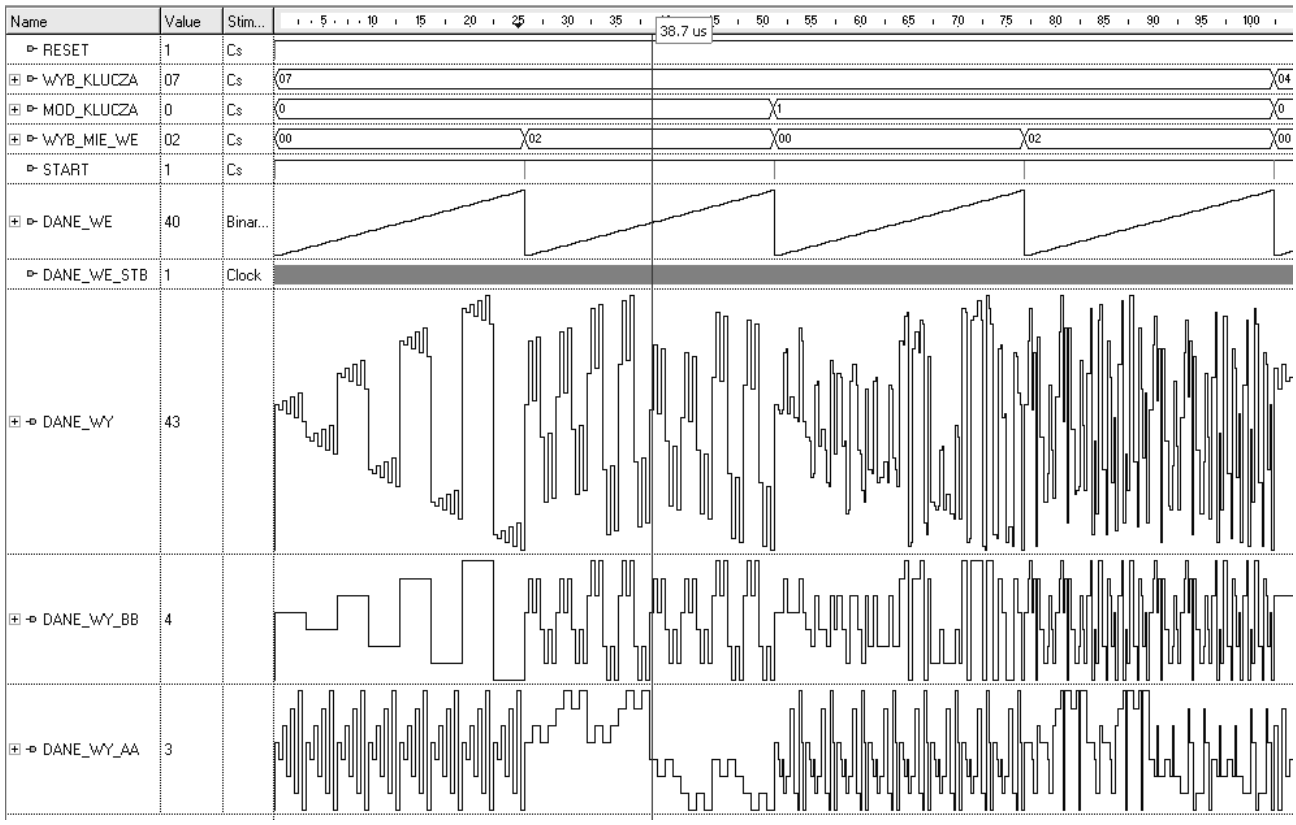
Schemat bramki w układach szyfratora i deszyfratora pokazano na rys. 5. Do realizacji bramki odwracalnej w układzie FPGA wykorzystane są funktory XOR i AND oraz programowane kluczem zwory konfiguracyjne zaznaczone kółkami. na rys. 5.

Zwora konfiguratora zaznaczona na schemacie pustym kółkiem oznacza brak połączenia pomiędzy lewą a prawą częścią ścieżki sygnałowej.



Rys. 5. Konfiguracja układu dla realizacji bramki C3-1,0
Fig. 5. Configuration of the circuit for implementation of gate C3-1,0

Na rys. 5 pokazano sposób skonfigurowania układu dla realizacji bramki, która realizuje funkcję Y3 = X3ρX0X1. Bramka tego typu przepuszcza na wyjścia Y2, Y1 i Y0 odpowiednio stany wejść X2, X1 i X0. Aby było to możliwe, zwory konfiguracyjne łączą wyjścia Yi z wejściami Xi dla i=0,1,2 pozostawiając otwarte połączenia wyjść Yi z wyjściami przypisanych im bramek XOR. W wypadku wyjścia Y3 zostaje ono połączone z wyjściem bramki XOR, dla której wejście środkowe jest połączone z wyjściem bramki AND. Bramka AND ma realizować funkcję X1•X0, dlatego do jej wejść zostają dołączone sygnały wejściowe X1 i X0. Pozostałe połączenia są nieaktywne.



Rys. 6. Wyniki symulacji dla układu 6-bitowego szyfratora
Fig. 6. Simulation results for a 6-bit encryption circuit

5. Prezentacja wyników

Przedstawiony wyżej projekt został opisany w języku VHDL. Celem lepszej czytelności przedstawienia wyników symulacji funkcjonalnej układu został on uproszczony i zawiera dwa identyczne układy przekształcające 3-bitową daną wejściową w 3-bitową daną wyjściową. Symulację projektu wykonano w systemie EDA ActiveHDL, gdyż umożliwia on szybkie wprowadzanie zmian i szybką symulację oraz analogowe przedstawianie przebiegów wybranych sygnałów. W wyniku symulacji otrzymano przebiegi pokazane na rysunku 6.

Przy pomocy sygnału WYB_KLUCZA (wartość 07) wybrano zestaw bramek realizujących pewną funkcję odwracalną. Wykonano cztery cykle symulacji. Dla każdego cyklu podano na wejścia układu DANE_WE kolejno wszystkie kombinacje 6-bitowej zmiennej wejściowej począwszy od 0 do wartości 63 (na rysunku daje to liniowo narastający przebieg). Sygnał MOD_KLUCZA wybiera sposób modyfikacji klucza podczas szyfrowania. Dla wartości 0 klucz wybrany sygnałem WYB_KLUCZA nie ulega zmianie podczas szyfrowania. Dla wartości 1 wybrany klucz bazowy zapisywany jest do rejestru klucza, którego zawartość jest przesuwana w lewo o dwie bramki (10 bitów) dla każdej kolejnej danej wejściowej. Wartość sygnału WYB_MIE_WE decyduje o sposobie mieszania bitów danej wejściowej DANA_WE na wejściu szyfratora. Wartość 0 tego sygnału oznacza podanie na wejście szyfratora danej wejściowej bez mieszania kolejności bitów. Wartość 2 powoduje, że na szyfrator podano wektor (DANE_WE(0), DANE_WE(4), DANE_WE(2), DANE_WE(3), DANE_WE(1), DANE_WE(5)). Cykl pierwszy symulacji (chwile czasowe od 0 do 25ns), to cykl bez modyfikacji klucza i bez mieszania wejść. Cykl drugi, to cykl z mieszaniem wejść. Cykl trzeci, to cykl z przesuwaniem klucza. Cykl czwarty to cykl zarówno z mieszaniem wejść, jak i z przesuwaniem klucza. Z rysunku można zauważyć, że w pierwszym cyklu da się zaobserwować zależność pomiędzy przebiegiem wejściowym i wyjściowym. Natomiast w każdym następnym cyklu staje się to coraz trudniejsze, co wskazuje na coraz lepsze szyfrowanie. Linie DANE_WY_BB i DANE_WY_AA reprezentują sygnały wyjściowe szyfratora, odpowiednio bardziej znaczącą część słowa wyniku i mniej znaczącą część słowa wyniku.

6. Podsumowanie

W artykule opisana została propozycja implementacji algorytmu szyfrowania wiadomości za pomocą układów odwracalnych. Przedstawiono koncepcje budowy takiego szyfratora za pomocą układów FPGA. Dokonano symulacji układów o 6 wejściach, a wyniki przedstawiono za pomocą modelu o trzech wejściach i trzech wyjściach.

Praca była wykonana w ramach realizacji grantu MNiSzW nr 4810/B/T02/2010/38.

7. Literatura

[1] De Vos A.: Reversible Computing. Fundamentals, Quantum Computing and Applications, Wiley-VCH, Berlin 2010.
[2] Skorupski A., Szyprowski M., Kerntopf P.: Algorytm syntezy kombinacyjnych układów odwracalnych, Pomiary-Automatyka-Kontrola, vol. 57, nr 8, 2011, ss. 858-860.
[3] Golubitsky O., Maslov D.: A Study of Optimal 4-bit Reversible Toffoli Circuits and Their Synthesis, arXiv:1103.2686v2, 2011.
[4] Thapliyal H., Zwolinski M.: Reversible logic to Cryptographic hardware: a new paradigm, Proc. 49th Int'l Midwest Conf. on Circuits and Systems, 2006, pp. 342-346.
[5] Nayeem N. M., Jamal L., Babu H. M. H.: Efficient reversible Montgomery multiplier and its application to hardware cryptography, Journal of Computer Science, vol. 5, no. 1, 2009, pp. 49-56.
[6] Banerjee A.: Reversible cryptographic hardware with optimized quantum cost and delay, Annual IEEE India Conference, 2010, 4 pages.
[7] Zhang Y., Guan Z., Nie Z.: Function modular design of the DES encryption system based on reversible logic gates, Proc. Int'l Conf. on Multimedia Communications, 2010, pp. 104-107.
[8] Menezes A. J., van Oorschot P. C., Vanstone S. A.: Kryptografia stosowana, WNT, Warszawa 2005.

otrzymano / received: 11.04.2012
przyjęto do druku / accepted: 01.06.2012 *artykuł recenzowany / revised paper*