Andrzej SKORUPSKI [1], Marek PAWŁOWSKI [2], Krzysztof GRACKI [2], Paweł KERNTOPF [3]
[1] WYŻSZA SZKOŁA MENEDŻERSKA, WYDZIAŁ INFORMATYKI STOSOWANEJ I TECHNIK BEZPIECZEŃSTWA, ul. 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

Rekonfigurowanie funkcji odwracalnych modelowanych w układzie FPGA

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 starszy wykładowca w Instytucie Informatyki na tym Wydziale. Interesuje się grafiką komputerową i projektowaniem układów cyfrowych.

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

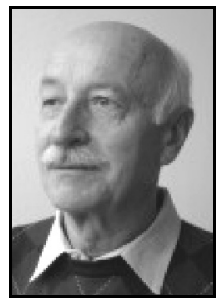
Mgr inż. Marek PAWŁOWSKI

Ukończył studia magisterskie na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. Obecnie pracuje jako starszy 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

Układy FPGA dobrze nadają się do modelowania układów odwracalnych, których implementacje sprzętowe są dopiero w stadium opracowywania. Układy odwracalne umożliwiają prostą realizację szyfratorów i deszyfratorów. W artykule rozpatrzono działanie dwóch szesnasto-bramkowych kaskad zbudowanych z cztero-wejściowych bramek odwracalnych NCT, aby uzyskać bajtowo zorientowany szyfrator. Zbiór bramek NCT o co najwyżej czterech wejściach zawiera 32 bramki, więc dla skonfigurowania jednej bramki potrzeba 5 bitów. Zatem kaskada może być określona przez 80-bitowe słowo, co dla dwóch kaskad daje 160-bitowy klucz. Po każdym wejściowym bajcie obie kaskady są rekonfigurowane za pomocą odpowiedniego przesuwania 80-bitowych słów. Sposoby przesuwania są określane przez dodatkowe bity klucza pomocniczego.

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

Reconfiguration of reversible functions using modeling of gates in FPGA

Abstract

FPGAs can be applied to modeling of reversible circuits because their practical realization is still under development. This technique enables implementing substitution ciphers. We try to build a byte-oriented stream cipher. Such a cipher uses two four-input and four-output cascades. Each of the cascades contains 16 reversible NCT gates. Because there exist 32 different NCT gates having at most four inputs we use 80 bits (16×5 bits) to determine one cascade so for two cascades 160 bits are needed. These bits are called the base key and are stored in the memory of a cipher. At the beginning of encryption the key is loaded to a circular shift register. After each input byte (a clock period) the contents of the shift register is shifted by a specified number of bits. The number of bits by which the register contents is shifted constitutes the second part of the cipher key. The shifting process causes changes in cascades after each input byte. If shifting the key is the same during both encryption and decryption, then the cipher will work correctly. In the paper, we present some methods of key shifting. If the register contents is shifted by 5 bits, then each gate is replaced by its predecessor (the first gate is replaced by the last one). The results of different shifting modes are presented showing that in all cases correct encryption/decryption is performed. For modeling and simulation of synthesis we used test-bench software ActiveHDL v 8.2 from ALDEC.

Keywords: reversible logic circuits, encryption, FPGA.

1. Wprowadzenie

Do prototypowania projektów urządzeń cyfrowych często stosuje się układy scalone FPGA. Jednym z takich zastosowań mogą być projekty układów odwracalnych, które umożliwiają redukcję energii wydzielanej w układach cyfrowych [10].

Układy odwracalne realizują jednoznaczne odwzorowanie sygnałów wejściowych na sygnały wyjściowe. Oznacza to, że każdemu sygnałowi wyjściowemu odpowiada tylko jeden sygnał wejściowy. Konsekwencją tego jest jednakowa liczba wejść i wyjść układu odwracalnego. Układy odwracalne rozwijane są nie tylko ze względu na zmniejszanie strat energii, ale także ze względu na zastosowanie do budowy komputerów kwantowych [10]. Niedostępność fizycznych realizacji bramek odwracalnych w chwili obecnej wymusza ich modelowanie za pomocą innych układów. Dobrym sposobem weryfikacji projektów z bramkami odwracalnymi jest ich modelowanie w układach FPGA. Jednym z urządzeń, które można zbudować z bramek odwracalnych jest szyfrator. Niedawno podjęto pierwsze próby opracowywania nowych algorytmów szyfrowania [1, 2, 3] i realizacji znanych algorytmów w logice odwracalnej [4, 5, 6, 7, 8]. W niniejszym artykule przedstawiono wstępne badania związane z nowym projektem bajtowo zorientowanego szyfratora strumieniowego.

Kaskada bramek odwracalnych może stanowić układ szyfratora podstawieniowego [3]. Dla funkcji czterech zmiennych istnieją 32 różne bramki odwracalne NCT [9]. Zatem podczas modelowania układu odwracalnego potrzebnych jest 5 bitów, aby wybrać rodzaj bramki. Dla kaskady bramek o długości 16 trzeba zastosować 80-bitowe (16×5) słowo sterujące, stanowiące klucz bazowy. Aby dokonać rekonfiguracji kaskady trzeba zmienić wartość tego klucza, a co za tym idzie zmienić bramki w kaskadzie.

W niniejszym artykule przedstawiono wyniki badania kaskady 16 bramek odwracalnych sterowanych 80-bitowym kluczem bazowym. Rekonfiguracja kaskady została zrealizowana w układzie FPGA poprzez zapisanie klucza do rejestru przesuwającego, który może być przesuwany o n bitów. Złożenie dwóch takich kaskad pozwala na budowę bajtowo zorientowanego szyfratora strumieniowego.

W zależności od liczby sposobów przesuwania rejestru trzeba uzupełnić klucz bazowy o klucz pomocniczy. Przedstawione niżej badanie polegało na sprawdzeniu kilku możliwości przesuwania rejestru. Oto niektóre z nich:

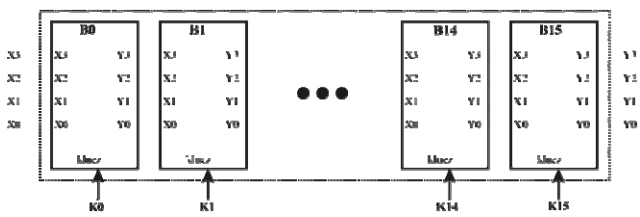
1) Przesuwanie klucza bazowego o 10 bitów w prawo po każdym znaku wiadomości wejściowej.
2) Przesuwanie klucza bazowego o 1 bit w lewo po każdym znaku wiadomości wejściowej.
3) Przesuwanie klucza o 2 bity w lewo po każdym znaku wiadomości wejściowej.

Kaskada złożona z 16 bramek odwracalnych pozwala na implementację dowolnej funkcji odwracalnej czterech zmiennych [11]. Przesuwanie klucza bazowego o wielokrotność 5 powoduje rotację bramek w kaskadzie. Przesuwanie klucza bazowego o inną wartość powoduje, że każda bramka jest zastępowana przez inną bramkę wynikającą z aktualnego 5-bitowego kodu odpowiadającego danej pozycji.

Do syntezy projektu i jego symulacji z wykorzystaniem testbencha wykorzystano oprogramowanie ActiveHDL v 8.2 firmy ALDEC.

2. Szyfrowanie i deszyfrowanie

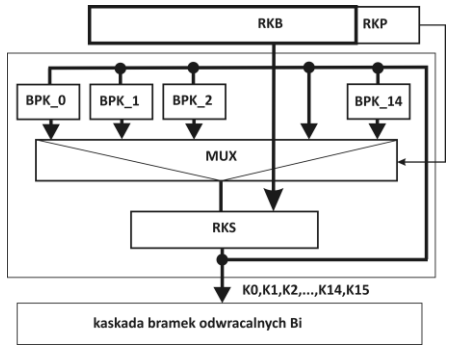
Układ szyfratora składa się z dwóch kaskad pokazanych na rysunku 1. Dane wejściowe podawane są na bramkę B0, a wynik szyfrowania pojawia się na wyjściu bramki B15. Wybór jednego z 32 typów bramki Bi zależy od wartości podklucza Ki. Klucz bazowy tworzą podklucze K0, K1, K2, …, K14, K15.



Rys. 1. Kaskada bramek odwracalnych jako szyfrator
Fig. 1. Cascade of reversible gates as an encryption block

Układ deszyfratora składa się także z 16-tu bramek odwracalnych Bi połączonych w odwrotnej kolejności. Dane wejściowe podawane są na bramkę B15, a wynik deszyfrowania pojawia się na wyjściu bramki B0.

Modyfikację klucza bazowego na 15 różnych sposobów można zrealizować w układzie pokazanym na rysunku 2. Do układu szyfratora/deszyfratora dostarcza się klucz składający się z dwóch części: klucz bazowy KB i klucz pomocniczy KP. Są one zapisywane odpowiednio do rejestrów RKB i RKP.



Rys. 2. Układ przekształcania klucza szyfrującego/deszyfrującego
Fig. 2. Block diagram of the circuit for transforming the key

Sesję szyfrowania/deszyfrowania rozpoczyna się od załadowania 80-bitowego klucza bazowego do rejestru RKS, którego zawartość określa bramki w kaskadzie. Zawartość rejestru RKP wskazuje sposób modyfikacji klucza podczas procesu szyfrowania/deszyfrowania dalszych słów wejściowych. W tym przypadku liczba bitów klucza pomocniczego wynosi 4. W ten sposób można wybrać jeden z 15 bloków BPK_i, a więc jeden z 15 sposobów modyfikacji klucza bazowego.

Sposób realizacji kaskady bramek odwracalnych został przedstawiony w [3] i zrealizowany w projekcie szyfr4b_T.vhd. Układ przekształcania klucza zrealizowano w postaci test-bench'a szyfr_tb_T.vhd, którego fragmenty omówiono w rozdziale 3. W artykule przedstawiono wyniki tylko trzech wyżej wymienionych sposobów modyfikacji klucza bazowego.

3. Badanie metod modyfikacji klucza

Procedurę testowania metod przekształcania klucza szyfrującego opisano w procesie *testowanie* przedstawionym poniżej. Przyjęto, że dana wejściowa nie zmienia się i ma wartość szesnastkową A.

```
testowanie : process (TB_RESET,TB_GWE,TB_DWYS,LWE) is
type tablica_wynikow is array (natural range <>)
                        of std_logic_vector ( 3 downto 0);
                     -- zakodowany stan wyjść
  variable TWY : tablica_wynikow(0 to 32);
begin
  TB_DWES<="1010";
-- wyznaczenie kodów bramek kaskady szyfratora
  for i in 0 to 15 loop
    TB_KL_B(i)<=TB_KL_S((i*5)+4 downto (i*5));
  end loop;
-- układ przekształcania klucza
  if TB_RESET/='1' then LWE<=0;  LSH<=0;  NR_TEST<=0;
                        TB_KL_S<=TAB_KLA(0).KLS;
  elsif falling_edge(TB_GWE) then
    if LWE<32
        -- modyfikacja sygnału wejściowego
      then LWE<=LWE+1;
        -- modyfikacja sygnału wejściowego
      else LWE<=0;
        if LSH<31 then
          LSH<=LSH+1;
          if    TAB_KLA(NR_TEST).CSL then
        -- przesunięcie w lewo o wskazaną liczbę pozycji
TB_KL_S<=(TB_KL_S((79-TAB_KLA(NR_TEST).POZ) downto 0) &
          TB_KL_S(79 downto (80-TAB_KLA(NR_TEST).POZ)));
          elsif TAB_KLA(NR_TEST).CSR then
        -- przesunięcie w prawo o wskazaną liczbę pozycji
TB_KL_S<=(TB_KL_S((TAB_KLA(NR_TEST).POZ -1) downto 0) &
          TB_KL_S(79 downto (   TAB_KLA(NR_TEST).POZ)));
          end if;
        else
        -- kolejny test?
         if NR_TEST=NR_TEST_max
         then report "koniec symulacji" severity FAILURE;
         else NR_TEST<=NR_TEST+1; LSH<=0; LWE<=0;
              TB_KL_S<=TAB_KLA(NR_TEST+1).KLS;
         end if;
        end if;
      end if;
   end if;
end process testowanie;
```

Klucz szyfrujący oraz metodę jego modyfikacji opisano stałą `TAB_KLA` typu `tablica_kluczy` zdefiniowaną rekordem `opis_testu`, co pokazano poniżej.

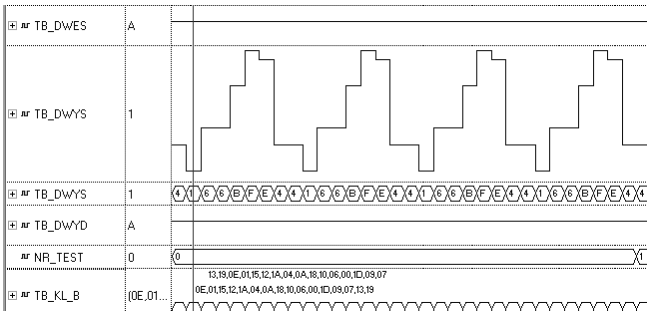
```
type opis_testu is  -- pulse description
  record
   KLS : std_logic_vector(79 downto 0); -- klucz szyfrujący
   CSL : boolean;  -- cykliczne przesunięcie klucza w lewo
   CSR : boolean;  -- cykliczne przesunięcie klucza w prawo
                   -- liczba pozycji przesunięcia w bitach na cykl
   POZ : integer range 1 to 15;
end record opis_testu;

type tablica_kluczy is array (natural range <>) of opis_testu;
constant TAB_KLA : tablica_kluczy(0 to 4) :=  (
((C3_0   &T2_30 &T3_21 &T0_31 &C1_3   &C3_2   &T2_10 &T0_321&C0_1
&N1    &T1_20 &C1_0  &T2_31 &T3_20 &C2_0  &C0_2), false,true,10),
 ((C3_0   &T2_30 &T3_21 &T0_31 &C1_3   &C3_2   &T2_10 &T0_321&C0_1
&N1    &T1_20 &C1_0  &T2_31 &T3_20 &C2_0  &C0_2),true,false,1),
 ((C3_0   &T2_30 &T3_21 &T0_31 &C1_3   &C3_2   &T2_10 &T0_321&C0_1
&N1    &T1_20 &C1_0  &T2_31 &T3_20 &C2_0  &C0_2),true,false,2),
 ((C3_0   &T2_30 &T3_21 &T0_31 &C1_3   &C3_2   &T2_10 &T0_321&C0_1
&N1    &T1_20 &C1_0  &T2_31 &T3_20 &C2_0  &C0_2), false,true,1),
 ((C3_0   &T2_30 &T3_21 &T0_31 &C1_3   &C3_2   &T2_10 &T0_321&C0_1
&N1    &T1_20 &C1_0  &T2_31 &T3_20 &C2_0  &C0_2 ), false,true,2));
```

Poszczególne podklucze opisano nazwami bramek z biblioteki NCT. Dla przedstawienia wpływu zmiany klucza szyfrującego na daną jawną na wejście szyfratora `TB_DWES` podano stałą wartość szesnastkową 0xA. Dana zaszyfrowana z wyjścia `TB_DWYS`

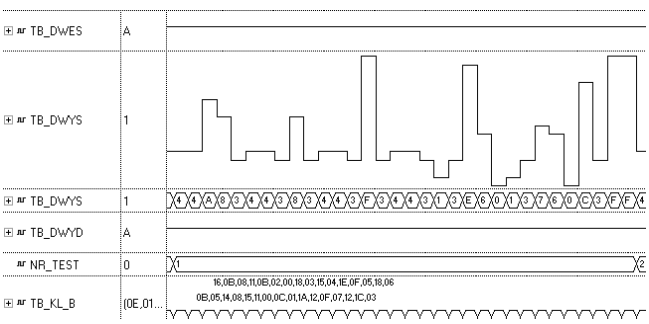
została wprowadzona na wejście deszyfratora, a jego wyjście jest obserwowane jako sygnał TB_DWYD.

W przypadku testu 0 (stała TAB_KLA(0)) klucz jest przesuwany o 10 bitów w prawo, co odpowiada przesunięciu o dwa podklucze. Do szyfrowania wykorzystywane są te same bramki, choć w innej kolejności. W wyniku uzyskujemy powtórzenie sekwencji danych wyjściowych co 8 cykli (rys. 3), gdyż taki jest okres powtarzania się wartości klucza bazowego.



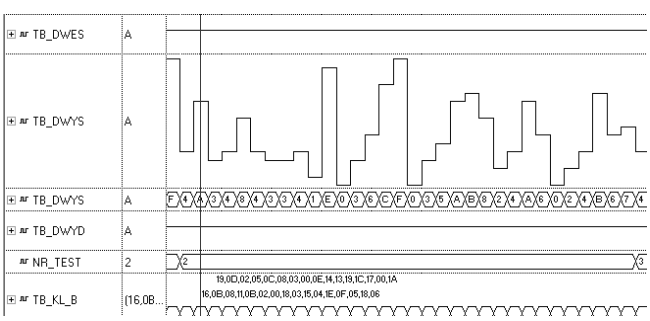
Rys. 3. Wynik działania szyfratora dla testu 0
Fig. 3. Result of encryption for test 0

W przypadku testu 1 klucz jest przesuwany o 1 bit w lewo, co powoduje zmianę typów bramek w kaskadzie (patrz podane wartości sygnału TB_KL_B na rys. 4). W dolnym wierszu komentarza podano wartości podkluczy przed rekonfiguracją, a w górnym - po rekonfiguracji. Taka rekonfiguracja powoduje powtarzanie sekwencji danych wyjściowych co 80 cykli. Natomiast dla stałej wartości na wejściu TB_DWES wartości uzyskane na wyjściu szyfratora są różne, co pokazano w wierszu TB_DWYS.



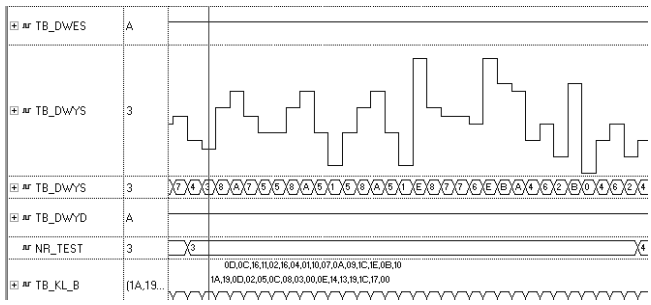
Rys. 4. Wynik działania szyfratora dla testu 1
Fig. 4. Result of encryption for test 1

W przypadku testu 2 klucz jest przesuwany o 2 bity w lewo i podobnie jak poprzednio wskazane przez nie bramki w kaskadzie szyfrującej różnią się od bramek klucza bazowego (rys. 5). W wyniku uzyskano powtórzenie sekwencji danych wyjściowych co 40 cykli.



Rys. 5. Wynik działania szyfratora dla testu 2
Fig. 5. Result of encryption for test 2

Test 3 różni się od testu 1 kierunkiem przesuwania klucza szyfrującego. Wyniki symulacji pokazane na rys. 6 są podobne do przedstawionych powyżej.



Rys. 6. Wynik działania szyfratora dla testu 3
Fig. 6. Result of encryption for test 3

Z wykonanych symulacji wynika, że zmiana klucza szyfrującego dla kolejnych danych jawnych pozwala uzyskać praktycznie wszystkie wartości od 0x0 do 0xF na wyjściu szyfratora.

4. Podsumowanie

Podstawowym celem pracy było badanie efektywności rekonfiguracji bramek odwracalnych kaskady modelowanych w układzie FPGA. Implementacja dwóch rekonfigurowanych kaskad 16-sto bramkowych zajmuje ok. 3000 elementów LUT i działa z częstotliwością ok. 250MHz, co pozwala sądzić, że w niewielkim układzie FPGA da się zamodelować szybki układ odwracalny. Przedstawione wyniki badań modyfikacji klucza bazowego mogą stanowić podstawę do dalszych prac związanych z projektem szyfratora wykorzystującego logikę odwracalną. Pokazano, że jest możliwa dynamiczna modyfikacja klucza w czasie szyfrowania. Dalsze prace dotyczyć będą zarówno zwiększania liczby sposobów modyfikacji, co umożliwi wydłużenie klucza pomocniczego, jak i szybkości rekonfiguracji kaskady.

5. Literatura

[1] Thapliyal H., Zwolinski M.: Reversible logic to cryptographic hardware: a new paradigm. Proc. 49th International Midwest Conference on Circuits and Systems, s. 342-346, 2006.
[2] Banerjee A.: Reversible cryptographic hardware with optimized quantum cost and delay. Annual IEEE India Conference, s. 1-4, 2010.
[3] Skorupski A., Pawłowski M., Gracki K., Kerntopf P.: Modelowanie w FPGA szyfratorów implementowanych w logice odwracalnej. Pomiary Automatyka Kontrola, vol. 58, s. 620-622, 2012.
[4] 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, s. 49-56, 2009.
[5] Zhang Y., Guan Z., Nie Z.: Function modular design of the DES encryption system based on reversible logic gates. Proc. International Conference on Multimedia Communications, s. 104-107, 2010.
[6] Datta K.,Sengupta I.: Applications of reversible logic in cryptography and coding theory (Tutorial). Proc. 26th International Conference on VLSI Design (VLSID), 2013.
[7] Datta K., Shrivastav V. ; Sengupta I. ; Rahaman H.: Reversible logic implementation of AES algorithm. Proc. 8th International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS), s. 140-144, 2013.
[8] Nuthan.A.C, Nagaraj.C, Havyas.V.B.: Implementation of Data Encryption Standard Using Reversible Gate Logic. International Journal of Soft Computing and Engineering, vol. 3, no. 3, s. 270-272, 2013.
[9] Skorupski A., Szyprowski M., Kerntopf P.: Algorytm syntezy kombinacyjnych układów odwracalnych. Pomiary Automatyka Kontrola, vol. 57, s. 858-860, 2011.
[10] De Vos A.: Reversible Computing. Fundamentals, Quantum Computing, and Applications, Wiley-VCH, Berlin 2010.
[11] Golubitsky O., Maslov D.: A study of optimal 4-bit reversible Toffoli circuits and their synthesis. IEEE Transactions on Computers, vol. 61, no. 9, s. 1341-1353, 2012.

otrzymano / received: 12.04.2014
przyjęto do druku / accepted: 02.06.2014 *artykuł recenzowany / revised paper*