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

Implementacja algorytmu syntezy układów odwracalnych w strukturach FPGA

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.

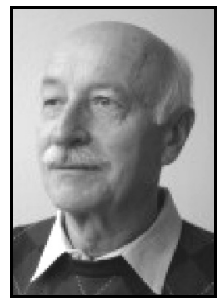
e-mail: M.Pawlowski@ii.pw.edu.pl

Dr 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

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

Synteza układów odwracalnych prowadząca do uzyskania układu optymalnego (składającego się z minimalnej liczby bramek) jest problemem bardzo trudnym. Dlatego często rezygnuje się z optymalności na rzecz prostszych metod projektowania. W niniejszym artykule przedstawiono wyniki prac związanych z możliwością implementacji uniwersalnego układu, który wykorzystuje pewien heurystyczny algorytm i pozwala na realizację dowolnej funkcji trzech zmiennych. Prowadzone prace wykorzystują układy FPGA i ich opisy w języku VHDL.

Słowa kluczowe: odwracalne układy logiczne, język VHDL, układy FPGA.

FPGA implementation of a reversible circuit synthesis algorithm

Abstract

Optimal synthesis of reversible circuit synthesis is a hard task. This why simpler algorithms are developed for finding suboptimal solutions. We show a simple heuristic algorithm implemented in a programmable FPGA circuit. In this paper the new algorithm and its hardware implementation in VHDL are described. The presented algorithm is based on some feature of reversible functions, namely, on the ordering of columns in the truth table for a given reversible function. We define the so called s-distance as a minimal length of gates cascade which is capable to order a column of the truth table, i.e. to transform a right side column to become identical to the corresponding left side column. It is possible to store s-distances for all possible columns. For every function the SF-distance is defined as the sum of all column s-distances. The proposed simple algorithm selects the gates which lead to the minimal SF-distance for the rest function (a rest function is the function to be still implemented after the given gate has been selected). The process is repeated until the consecutive rest function will become the identity function. The algorithm can be implemented using the FPGA circuit as the block scheme from Fig. 3. The description of this module using VHDL is presented and discussed.

Keywords: reversible logic circuits, VHDL language, FPGA circuits.

1. Wprowadzenie

W niniejszej pracy przedstawiono koncepcję sprzętowej realizacji algorytmu syntezy układów odwracalnych. Ze względu na złożoność problemu synteza zostanie zaprezentowana dla układów trzywejściowych.

Zadaniem algorytmu syntezy jest znalezienie kaskady bramek odwracalnych NCT [1-4], która składa się z jak najmniejszej liczby bramek. Opracowano wiele algorytmów projektowania układów odwracalnych [1-4], jednak nie są one zadowalające z punktu widzenia praktyki. Przedstawiony w pracy prosty heurystyczny algorytm, choć nie zawsze prowadzi do optymalnego rozwiązania, z powodzeniem może być implementowany w układach FPGA.

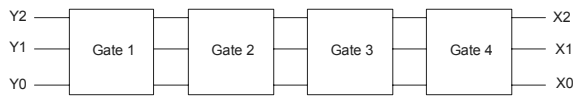
2. Porządkowanie zmiennych

Funkcja odwracalna trzech zmiennych może być przedstawiona w postaci tablicy prawdy. Przykładowa funkcja, której tablica prawdy pokazana jest na rys. 1, może być zapisana także jako permutacja wierszy funkcji jednostkowej, czyli lewej części tej tablicy, w ciąg wierszy prawej części tablicy <0,3,2,1,4,5,7,6>. Innym sposobem jej opisu jest ciąg liczb heksadecymalnych odpowiadających prawym kolumnom tablicy prawdy, tj. 0f6356.

Nr	X2	X1	X0	Y2	Y1	Y0
0	0	0	0	0	0	0
1	0	0	1	0	1	1
2	0	1	0	0	1	0
3	0	1	1	0	0	1
4	1	0	0	1	0	0
5	1	0	1	1	0	1
6	1	1	0	1	1	1
7	1	1	1	1	1	0

Rys. 1. Tablica prawdy przykładowej funkcji
Fig. 1. Truth table of an example function

Synteza polega na znalezieniu kaskady bramek realizującej daną funkcję. Możliwe są dwa podejścia: 1) wyznaczanie kolejnych bramek od wejścia do wyjścia, tj. od funkcji jednostkowej do zadanej funkcji, 2) wyznaczanie kolejnych bramek od wyjścia do wejścia, tj. od zadanej funkcji do funkcji jednostkowej. Tu przyjmiemy drugie z tych podejść, które pokazano na rys. 2.



Rys. 2. Kaskada bramek odwracalnych
Fig. 2. A cascade of reversible gates

Można pokazać, że dla funkcji boolowskich *n* zmiennych istnieje różnych kolumn w prawej części tablic prawdy. W przypadku trzech zmiennych różnych kolumn jest 70. Każdą z tych 70 kolumn można uporządkować za pomocą ciągu bramek odwracalnych. Dla trzech zmiennych najdłuższy ciąg zawiera cztery bramki. Jedna z 70 kolumn, która jest uporządkowana jak w funkcji jednostkowej, nie wymaga stosowania bramek (dla najbardziej znaczącej zmiennej jest to kombinacja 0f). Cztery inne kolumny można uporządkować jedną bramką, bo tyle jest różnych bramek na danej linii. Dwóch bramek wymaga 12 kolumn, trzech bramek wymaga 29 kolumn i czterech bramek wymaga 24 kolumny.

Definicja: Najmniejszą długość ciągu porządkującego daną kolumnę nazywać będziemy *s-odległością*. Sumę s-odległości dla wszystkich trzech kolumn nazywać będziemy *SF-odległością*.

Założenie: Jako pierwszą bramkę w kaskadzie należy przyjąć taką, dla której funkcja wyjściowa będzie miała najmniejszą SF-odległość.

Dla funkcji trzech zmiennych można zapamiętać *s-odległości* dla każdej z 70 kolumn. Potrzebna jest zatem pamięć o pojemności 210 słów (po 70 dla każdej zmiennej).

Algorytm:

1. Dla danej funkcji odwracalnej *F* i dla każdej z 12 bramek odwracalnych obliczyć funkcję resztową (tzn. funkcję odwracalną, którą trzeba zrealizować, aby połączenie kaskadowe rozpatrywanej bramki z układem realizującym funkcje resztową stanowiło implementację funkcji *F*).
2. Dla każdej z funkcji resztowych odczytać *s-odległość* i obliczyć *SF-odległość*.
3. Wybrać bramkę dla której *SF-odległość* jest najmniejsza.
4. Powtarzać algorytm dla każdej funkcji resztowej, aż do momentu, gdy kolejna taka funkcja stanie się funkcją jednostkową, tj. 0f3355.

W ten sposób w każdym kroku iteracji otrzymuje się kolejne bramki kaskady. Można pokazać, że algorytm jest zbieżny.

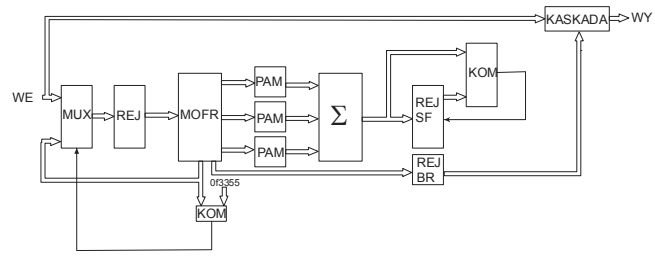
3. Układ realizujący proponowany algorytm

Powyższy algorytm można zrealizować sprzętowo wykorzystując układ FPGA. Z pierwszego punktu algorytmu wynika, że projektowany układ musi zawierać moduł obliczania funkcji resztowej. Jest to prosta operacja zamiany wierszy w tablicy prawdy [2]. Przykładowo, bramka C3-1 zamienia wiersze pierwszy z piątym i trzeci z siódmym, więc funkcja resztowa dla funkcji wejściowej z rys. 1 będzie funkcją <0,5,2,6,4,3,7,1>, lub inaczej 5a3647.

Z drugiego punktu algorytmu wynika, że kolejnym modułem powinna być pamięć *s-odległości*. Składa się ona z trzech pamięci zawierających po 70 dwubitowych słów (kolumny, dla których nie są potrzebne bramki, można pominąć). Ponieważ *s-odległości* należy sumować, to układ musi zawierać 3-wejściowy sumator z 4-bitowym wyjściem.

Trzeci punkt algorytmu narzuca konieczność pamiętania ostatnio obliczonej *SF-odległości*, aby porównać ją z następną. W tym celu można zastosować układ rejestru wraz z komparatorem. W rejestrze pamiętana będzie aktualna najmniejsza wartość *SF-odległości* wraz z numerem bramki.

Czwarty punkt algorytmu nakazuje konieczność iteracji, co oznacza, że wyżej wymienione moduły mają znajdować się w pętli sprzężenia zwrotnego, z której wychodzi się gdy funkcja resztowa wyniesie 0f3355. W przeciwnym przypadku funkcję resztową podaje się na wejście układu i powtarza się kroki algorytmu. Schemat blokowy opisanego układu został pokazany na rys. 3, który jest umieszczony na końcu artykułu.



Rys. 3. Schemat blokowy zaproponowanego układu
Fig. 3. Block diagram of the proposed circuit

```
Begin
 if rising_edge(CLK) then
  case stan_MOFR is
   when 0 => ERROR<='0'; L_BR:=0; REJ_SF:=22;
             TYP_BR:="0000"; L_LEV:=0; REJ_BR:=NOP;
             R_FUN_WE:=WE_FUN;
               --czy funkcja wejściowa jest gotowa?
             if FUN_WE_OK='0'
               then stan_MOFR:=0;
               else BR_OK<='0'; stan_MOFR:=1;
             end if;
   when 1 =>   -- inicjalizacja rejestru REJ BR
              for i in BR_max downto 0 loop
                                BR_TYP(i)<=NOP; end loop;
             stan_MOFR:=2;
   when 2 =>   -- wyznaczenie bramki
             TYP_BR:=conv_std_logic_vector(L_BR,4);
               -- wyznaczenie funkcji resztowej
             for i in 0 to 7 loop
                L_bzw:=bramka_zw(TYP_BR)(i);
                R_FUN_WY(i*3+2 downto i*3):=
                      R_FUN_WE(L_bzw*3+2 downto L_bzw*3);
             end loop;
               -- uporządkowanie kolumn funkcji resztowej
             for i in 7 downto 0 loop
                L0(7-i):=R_FUN_WY(i*3);
                L1(7-i):=R_FUN_WY(i*3+1);
                L2(7-i):=R_FUN_WY(i*3+2);
              end loop;
               -- obliczenie SF-odległości
             PAM_SUM:=PAM_L2(L2)+PAM_L1(L1)+PAM_L0(L0);
                -- zapamiętanie najmniejszej SF-odległości
              if PAM_SUM<REJ_SF
               then REJ_SF:=PAM_SUM; REJ_BR:=TYP_BR;
                    R_FUN_TMP:=R_FUN_WY;
             end if;
                -- przejście do kolejnej bramki
             if L_BR<11
               then L_BR:=L_BR+1; stan_MOFR:=2;
               else stan_MOFR:=3;
             end if;
   when 3 =>  -- zapamiętanie wyniku i przejście do
              -- wyznaczenia kolejnej bramki kaskady
             R_FUN_WE:=R_FUN_TMP; BR_TYP(L_LEV)<=REJ_BR;
             if L_LEV<BR_max
               then L_LEV:=L_LEV+1; stan_MOFR:=4;
               else stan_MOFR:=0;
             end if;
   when 4 => FUN_COD:=conv_fun(R_FUN_WE); L_BR:=0;
             REJ_SF:=22; TYP_BR:="0000"; REJ_BR:=NOP;
               -- czy funkcja resztowa jest jednostkową?
             if FUN_COD=x"0f3355"
               then stan_MOFR:=5;
               else stan_MOFR:=2;
             end if;
   when 5 =>   -- potwierdzenie wyznaczenia kaskady
             BR_OK<='1';
             if FUN_WE_OK='1'
               then stan_MOFR:=5;
               else stan_MOFR:=0;
             end if;
  end case;
 end if;
```

Rys. 4. Opis układu w języku VHDL
Fig. 4. VHDL description of the circuit

Na rys. 4 pokazano opis w języku VHDL układu, którego schemat podany jest na rys. 3 [5]. W opisie tym nie uwzględniono jedynie bloku funkcjonalnego KASKADA, będącego wynikiem algorytmu syntezy. Algorytm został zaimplementowany w postaci automatu stan_MOFR.

W stanie 0 automatu do zmiennej R_FUN_WE (blok REJ z rys. 3) jest podstawiane 24-bitowe słowo reprezentujące zadaną funkcję. Wczytanie tego słowa jest potwierdzane sygnałem FUN_WE_OK. Po inicjalizacji rejestru BR_TYP, konfigurującego układ KASKADA, następuje przejście do stanu 2 automatu. W stanie tym sprawdzane są SF-odległości dla wszystkich bramek ze zbioru NCT, w celu wybrania bramki o najmniejszej SF-odległości. Bieżąca bramka w kaskadzie jest wybierana jako jedno z 12 rozwiązań funkcji bramka_zw, które określa sposób zamiany wierszy w funkcji wejściowej podczas tworzenia funkcji resztowej R_FUN_WY. Kolejną operacją jest wyznaczenie indeksów Li dla kolumn funkcji resztowej. Indeksy te wskazują pozycje w pamięciach PAM z rys. 3, a w opisie VHDL używane są jako argument wejściowy funkcji PAM_Li wyznaczających s-odległości. Suma s-odległości jest reprezentowana zmienną PAM_SUM, która następnie jest porównywana ze zmienną REJ_SF. Jeżeli PAM_SUM jest mniejsza od REJ_SF, to zostaje zmieniona zawartość REJ_SF i do rejestru REJ_BR zapisywany jest kod wybranej bramki, a do rejestru REJ_FUN_TMP zapisywana jest aktualnie wyliczona funkcja resztowa. Opisane wyżej działania są powtarzane dla wszystkich bramek, aby wybrać bramkę o najmniejszej SF-odległości. Wtedy automat przechodzi do stanu 3, w którym zapamiętuje wybraną, odpowiednią bramkę kaskady i funkcję resztową. Zmienna L_LEV, określająca numer pozycji w kaskadzie, jest w tym stanie inkrementowana. W stanie 4 sprawdzany jest warunek zakończenia generacji układu bramek odwracalnych w kaskadzie, po osiągnięciu funkcji resztowej równej funkcji identycznościowej.

Synteza opisanego wyżej projektu wymagała użycia do jego realizacji 1062 elementów logicznych i 150 rejestrów w układzie z rodziny Cyclone IV [6].

4. Podsumowanie

Stosując opisany wyżej algorytm zaprojektowano układy dla wszystkich 40320 funkcji trzech zmiennych. W tab. 1 podano wyniki dla wybranych funkcji.

Funkcja nr 12 jest funkcją przedstawioną na początku artykułu. Funkcja nr 11 jest funkcją tożsamościową, która nie wymaga bramek (podane rozwiązanie stanowią dwie identyczne bramki, co jest równoważne układowi bez bramek). Trzy spośród przedstawionych funkcji mają rozwiązania nieoptymalne (funkcja 391e74 - dłuższe o jedną bramkę, funkcja e8a372 - dłuższe o trzy bramki, funkcja 17b1d2 - dłuższe o pięć bramek).

Symulacja pokazała, że 22605 funkcji miało optymalne rozwiązania (56% wszystkich funkcji), 13631 funkcji miało rozwiązania o 1 bramkę dłuższe od optymalnych, 3432 o dwie, 588 o trzy, 51 o cztery i 13 funkcji o 5 bramek.

Tab. 1. Wyniki stosowania algorytmu syntezy dla wybranych funkcji odwracalnych
Tab. 1. Results of using the synthesis algorithm to selected reversible functions

Nr	funkcja	numery bramek w kaskadzie
0	0f3366	7
1	0f3553	484
2	1b2d78	2327
3	3ccc99	0a7
4	0f335a	6
5	391e74	023268
6	4d7166	17501
7	0f3a66	735
8	2b1766	7050
9	17b1d2	30123681789a
10	e8a372	05703b502a
11	0f3355	00
12	0f6356	845

W dalszych pracach zostaną poddane testom inne kryteria heurystyczne, które zdaniem autorów mogą prowadzić do jeszcze lepszych rozwiązań.

5. 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, s. 858-860, 2011.
[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] 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.
[5] Pawłowski M., Skorupski A.: Projektowanie złożonych układów cyfrowych, WKŁ Warszawa 2010.
[6] Cyclone IV Device Handbook, Altera Corporation 2013.

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

INFORMACJE

Informacja redakcji dotycząca artykułów współautorskich

W miesięczniku PAK od numeru 06/2010 w nagłówkach artykułów współautorskich wskazywany jest autor korespondujący (Corresponding Author), tj. ten z którym redakcja prowadzi wszelkie uzgodnienia na etapie przygotowania artykułu do publikacji. Jego nazwisko jest wyróżnione drukiem pogrubionym. Takie oznaczenie nie odnosi się do faktycznego udziału współautora w opracowaniu artykułu. Ponadto w nagłówku artykułu podawane są adresy korespondencyjne wszystkich współautorów.

Wprowadzona procedura wynika z międzynarodowych standardów wydawniczych.

Redakcja