STUDIA INFORMATICA 2014
Volume 35 Number 4 (118)

Andrzej SKORUPSKI, Krzysztof GRACKI, Marek PAWŁOWSKI, Paweł KERNTOPF
Politechnika Warszawska, Instytut Informatyki; Uniwersytet Łódzki, Zakład Informatyki Stosowanej

HEURYSTYCZNY ALGORYTM SYNTEZY QUASI-OPTYMALNYCH UKŁADÓW ODWRACALNYCH

Streszczenie. W artykule przedstawiono nową metodę syntezy układów odwracalnych. Polega ona na stopniowym porządkowaniu bitów w kolejnych kolumnach części wyjściowej tablicy prawdy odwracalnej funkcji boolowskiej, aż do momentu zrównania części wyjściowej tablicy z jej częścią wejściową. Długość szacunkowej sekwencji bramek, która uporządkowałaby bity we wszystkich kolumnach wyjściowych, stanowi kryterium wyboru bramki w każdym kroku iteracji proponowanego algorytmu. Dla ponad 80% funkcji odwracalnych trzech zmiennych algorytm ten generuje układy optymalne.

Słowa kluczowe: zrównoważone funkcje boolowskie, funkcje odwracalne, bramki odwracalne, układy odwracalne, synteza układów logicznych

HEURISTIC ALGORITHM OF QUASI-OPTIMAL REVERSIBLE CIRCUITS SYNTHESIS

Summary. In this paper a new method of reversible circuits synthesis is presented. The method is based on iterative ordering of bits in subsequent output columns of the truth table of a reversible function until the output part of the truth table becomes identical with the input part. The length of the estimated shortest sequence which would guarantee the proper orderof bits in all output columns is a criterion for choosing a gate at each step of the proposed algorithm. For over 80% of 3-variable reversible functions the algorithm generates optimal circuits.

Keywords: balanced Boolean function, reversible functions, reversible gates, reversible circuits, logic circuit synthesis

1. Wprowadzenie

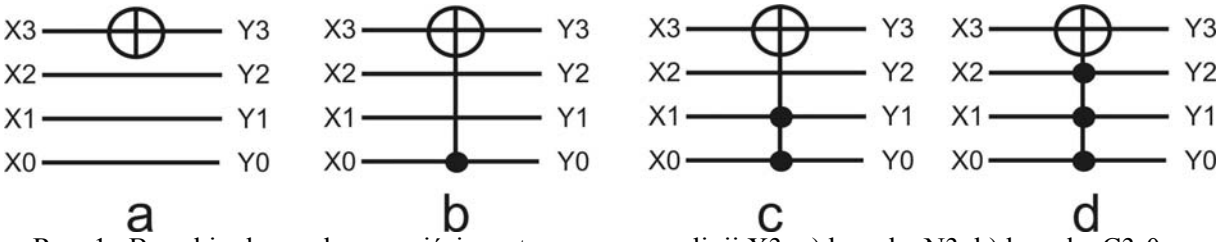
Jedną z dziedzin współczesnej informatyki są tzw. obliczenia odwracalne (ang. *reversible computing*). Dziedzina ta stwarza nadzieje na konstruowanie układów o bardzo małym pobo-

rze mocy. W pracy [3] udowodniono, że urządzenia, w których nie jest tracona informacja, mogą przetwarzać dane bez poboru energii. Są nimi tzw. układy odwracalne, które realizują wzajemnie jednoznaczne odwzorowanie wektorów wejściowych na wektory wyjściowe (w układach odwracalnych liczba wyjść jest równa liczbie wejść).

Funkcją odwracalną *n* zmiennych nazywać będziemy taki zbiór *n* funkcji *n* zmiennych, że każdemu z 2^n wektorów wejściowych odpowiada inny wektor wyjściowy. Można wykazać, że każda z tych *n* funkcji składowych jest tzw. funkcją boolowską zrównoważoną, tj. dla 2^{n-1} wektorów wejściowych przyjmuje wartość „1" i dla pozostałych 2^{n-1} wektorów wejściowych przyjmuje wartość „0". Funkcja odwracalna odpowiada permutacji na zbiorze 2^n wektorów binarnych, więc dla *n* zmiennych istnieje $2^n!$ różnych funkcji odwracalnych. Dla dwóch zmiennych jest ich 24, dla trzech zmiennych – 40320, a dla 4 zmiennych – ponad 20×10^{12}.

Każdą z funkcji odwracalnych realizuje się za pomocą układu kaskadowego zbudowanego z bramek odwracalnych. Dla danej funkcji układ, który składa się z najmniejszej liczby bramek, nazywać będziemy optymalnym. Dla większości funkcji istnieje wiele układów optymalnych. Znalezienie ich jest złożonym procesem obliczeniowym. W niniejszym artykule przedstawiono prosty algorytm, który dla większości funkcji znajduje jedno z rozwiązań optymalnych.

Istnieje wiele bibliotek funkcjonalnie pełnych bramek odwracalnych. Tutaj posługiwać się będziemy tzw. zbiorem NCT, składającym się z bramek, które pokazano na rys. 1.



Rys. 1. Bramki odwracalne z wyjściem sterowanym na linii X3: a) bramka N3, b) bramka C3-0, c) bramka C3-10, d) bramka T3

Fig. 1. Reversible gates with controlled output at line X3: a) gate N3, b) gate C3-0, c) gate C3-10, d) gate T3

Na rys. 1a pokazano bramkę N3, która neguje wartość sygnału na wejściu X3. Na rys. 1b pokazano bramkę C3-0, tj. bramkę, która neguje sygnał z wejścia X3 wtedy i tylko wtedy, gdy sygnał sterujący X0 jest równy 1. Na rys. 1c pokazano bramkę C3-10, tj. bramkę, która neguje sygnał z wejścia X3 wtedy i tylko wtedy, gdy obydwa sygnały sterujące X0 i X1 są równe 1. Na rys. 1d pokazano bramkę T3, tj. bramkę, która neguje sygnał z wejścia X3 wtedy i tylko wtedy, gdy trzy sygnały sterujące X0, X1 i X2 są równe 1. Liczba bramek z wyjściem sterowanym na linii 3 wynosi osiem: bramka N3, bramka T3, trzy bramki C z pojedynczą linią sterującą i trzy bramki C z dwoma liniami sterującymi. Zatem, łączna liczba bramek biblioteki NCT dla 4 zmiennych wynosi 32.

2. Funkcje odwracalne

Funkcje odwracalne, tak jak zwykłe funkcje boolowskie, można opisać na wiele różnych sposobów. Tutaj przedstawione zostaną dwa z nich: tablica prawdy i permutacja wierszy.

Tablica 1

Tablica prawdy przykładowej funkcji odwracalnej trzech zmiennych

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

W tablicy 1 podano tablicę prawdy przykładowej funkcji odwracalnej trzech zmiennych. Kolumna po lewej stronie zawiera dziesiętny numer wiersza, będącego wektorem wartości argumentów tej funkcji (nazywanego dalej „wektorem wejściowym"). Następne trzy kolumny zawierają właśnie wartości argumentów X2, X1 i X0, przy czym wektory binarnych wartości tych argumentów (tzn. wektory wejściowe) są uporządkowane leksykograficznie od 000 do 111. Trzy kolejne kolumny definiują trzy zrównoważone funkcje boolowskie (wiersze tej części tablicy prawdy nazywane będą „wektorami wyjściowymi"). Równocześnie spełniony jest warunek, że poszczególnym wartościom trzybitowego wektora wejściowego odpowiada inny trzybitowy wektor wyjściowy (wzajemna jednoznaczność). Ostatnia kolumna zawiera dziesiętne odpowiedniki wektorów wyjściowych. Drugim (równoważnym) opisem funkcji jest zapis permutacji wierszy tablicy prawdy. Przykładową funkcję z tabl. 1 można zapisać jako permutację wierszy <0,3,2,1,4,5,7,6>. Jest to lista dziesiętnych odpowiedników wektorów wyjściowych danej funkcji odwracalnej w kolejności od górnego do dolnego wiersza wejściowego.

Synteza układów realizujących zadaną funkcję polega na znalezieniu układu odwracalnego (kaskady bramek) realizującego tę funkcję. Poszukiwany układ odwracalny powinien składać się z jak najmniejszej liczby bramek. Dla trzech i czterech zmiennych opracowano algorytmy syntezy układów optymalnych lub quasi-optymalnych, które jednak są złożone obliczeniowo lub zajmują dużą pojemność pamięci [1, 4-18]. W niniejszej pracy zaproponowany został prosty algorytm heurystyczny, który dla większości funkcji prowadzi do optymalnego rozwiązania.

3. Porządkowanie bitów w kolumnach

Ponieważ każda z funkcji boolowskich n zmiennych, składająca się na funkcję odwracalną, jest funkcją zrównoważoną, więc istnieje ich $\binom{2^n}{2^{n-1}}$. Dla trzech zmiennych jest ich 70, dla czterech zmiennych – 12870, a dla pięciu zmiennych – 601080390. Problem syntezy funkcji odwracalnej można zdekomponować na n problemów poszukiwania kaskady bramek porządkujących bity tylko w jednej z n zrównoważonych funkcji boolowskich.

Rozpatrzmy przypadek funkcji trzech zmiennych. Można wykazać, że 70 różnych funkcji zrównoważonych można podzielić na 5 grup:

1. funkcje, dla których liczba bramek porządkujących wynosi 0,
2. funkcje, dla których liczba bramek porządkujących wynosi 1,
3. funkcje, dla których liczba bramek porządkujących wynosi 2,
4. funkcje, dla których liczba bramek porządkujących wynosi 3,
5. funkcje, dla których liczba bramek porządkujących wynosi 4.

Liczność tych grup jest następująca:

1. dla pierwszej grupy wynosi 1,
2. dla drugiej grupy wynosi 4,
3. dla trzeciej grupy wynosi 12,
4. dla czwartej grupy wynosi 29,
5. dla piątej grupy wynosi 24.

Zatem, dla funkcji trzech zmiennych największa liczba bramek potrzebna dla uporządkowania funkcji zrównoważonej wynosi 4, czyli dla uporządkowania wszystkich trzech funkcji potrzeba co najwyżej 12 bramek. Z drugiej strony wykazano, że każda funkcja odwracalna trzech zmiennych ma optymalną realizację nie dłuższą niż osiem bramek [2]. Dlatego, oprócz porządkowania każdej z funkcji oddzielnie, należy uwzględniać także wzajemne zależności pomiędzy funkcjami.

Problem dla czterech zmiennych jest bardziej złożony, bowiem istnieje 8 grup funkcji zrównoważonych. Kolumna w tablicy prawdy odpowiadająca każdej z 12870 funkcji może zostać uporządkowana przez układ o co najwyżej siedmiu bramkach.

Rozpatrzmy także liczności funkcji odwracalnych związane z tymi grupami. Dla każdej funkcji zrównoważonej istnieje ich $\binom{2^{n-1}}{2^{n-2}}^2$ funkcji odwracalnych, w których występuje dana funkcja. Dla trzech zmiennych liczba ta wynosi 36. Podobnie, dla każdej kombinacji dwóch funkcji zrównoważonych istnieje $\binom{2^{n-2}}{2^{n-3}}$ funkcji odwracalnych, mających takie funkcje składowe. Dla trzech zmiennych liczba ta wynosi 16. Stąd wynika, że dla każdej funkcji zrówno-

ważonej istnieje 36×16=576 funkcji odwracalnych, w których jedną z funkcji składowych jest dana funkcja zrównoważona. Zatem, wszystkie funkcje odwracalne trzech zmiennych po podzieleniu na pięć grup będą miały liczności:

1. 1×576=576 funkcji odwracalnych nie wymaga bramek,
2. 4×576=2304 funkcji odwracalnych wymaga jednej bramki,
3. 12×576=6912 funkcji odwracalnych wymaga dwóch bramek,
4. 29×576=16704 funkcji odwracalnych wymaga trzech bramek,
5. 24×576=13824 funkcji odwracalnych wymaga czterech bramek,

Analogicznie można obliczyć tego typu liczności dla funkcji odwracalnych czterech zmiennych.

4. Algorytm syntezy

Wprowadźmy nowe pojęcia:

Definicja 1: S-odległością funkcji zrównoważonej (kolumny w tablicy prawdy) nazywać będziemy liczbę bramek potrzebnych dla uporządkowania bitów w kolumnie tablicy prawdy, która odpowiada danej funkcji zrównoważonej.

Definicja 2: SF-odległością nazywać będziemy liczbę bramek będących sumą wszystkich S-odległości dla poszczególnych funkcji składowych danej funkcji odwracalnej.

SF-odległość będzie stanowić kryterium proponowanego algorytmu heurystycznego. W każdym kroku algorytmu należy wybrać jedną bramkę poszukiwanego układu. Dla trzech zmiennych można rozpatrywać i porównywać układy dla każdej z 12 bramek, ale wówczas złożoność algorytmu staje się bardzo duża. Proponowany w tym artykule algorytm przewiduje branie pod uwagę tylko tych bramek, dla których SF-odległość jest minimalna. Dla każdej funkcji zrównoważonej można obliczyć S-odległość albo można wszystkie S-odległości umieścić w S-tablicy, co zdecydowanie przyśpiesza znalezienie SF-odległości.

Algorytm dla trzech zmiennych zaczyna się od pierwszej iteracji, w której dla każdej z 12 bramek sprawdza się SF-odległość funkcji wyjściowej tej bramki. Funkcje odwracalne, które trzeba będzie jeszcze zrealizować po wybraniu danej bramki jako pierwszej, nazywane są funkcjami resztowymi. Jako pierwszą bramkę w kaskadzie wybiera się tę, dla której funkcja resztowa ma minimalną SF-odległość.W przypadku gdy istnieje więcej niż jedna taka funkcja, można rozpatrywać wiele równoważnych rozwiązań. Dalsze iteracje polegają na wybieraniu następnych bramek na podstawie wartości SF-odległości dla poszczególnych funkcji resztowych.

Algorytm:

Krok 1. Dla danej funkcji odwracalnej obliczyć funkcje resztowe po każdej z bramek (dla 3 zmiennych jest 12 bramek).

Krok 2. Dla każdej z funkcji resztowych obliczyć (lub odczytać z S-tablicy) S-odległości, a następnie SF-odległość.

Krok 3. Wybrać bramkę (bramki), dla której funkcja resztowa miała minimalną SF-odległość.

Krok 4. Jeśli funkcja resztowa nie jest funkcją identycznościową, to powtórzyć dla niej kroki1, 2 i 3.

5. Przykład

Niech będzie dana funkcja trzech zmiennych <1,0,2,4,6,7,3,5>, której tablica prawdy jest podana w tablicy. 2.

Tablica 2
Tablica prawdy przykładowej funkcji <1,0,2,4,6,7,3,5>

X2X1X0	Y2Y1Y0
0 0 0	0 0 1
0 0 1	0 0 0
0 1 0	0 1 0
0 1 1	1 0 0
1 0 0	1 1 0
1 0 1	1 1 1
1 1 0	0 1 1
1 1 1	1 0 1

Na podstawie bazy wszystkich optymalnych układów odwracalnych realizujących funkcje trzech zmiennych [2] wiadomo, że dla powyższej funkcji istnieją trzy optymalne układy 5-bramkowe:

1. T0, T2, C1-2, C0-1, N0.
2. T0, T2, C1-2, N0, C0-1.
3. T0, T2, N0, C1-2, C0-1.

Prześledźmy działanie podanego algorytmu. Funkcje resztowe dla każdej z 12 bramek pokazano w tablicy. 3, w której dwie ostatnie kolumny zawierają S-odległości i SF-odległości.

W pierwszej iteracji algorytmu wybrane zostaną bramki C0-2 i T0, gdyż ich funkcje resztowe mają najmniejszą wartość SF-odległości równą 6.

W drugiej iteracji powtarza się procedurę dla funkcji <1,0,2,4,7,6,5,3> i <1,0,2,4,6,7,5,3> jako funkcji resztowych odpowiednio dla bramek C0-2 i T0. Rozważymy tu jedynie funkcję <1,0,2,4,6,7,5,3>. W tym celu trzeba zbudować analogiczną tablicę do tablicy 3. W ten sposób można pokazać, że minimalna SF-odległość równa 4 będzie dla bramki T2, dla której funkcja resztowa ma postać <1,0,2,3,6,7,5,4>.

W trzeciej iteracji otrzymuje się minimalną SF-odległość (równą 2) dla bramki C1-2. Funkcja resztowa ma postać <1,0,2,3,5,4,6,7>. Po tej iteracji mamy już trzy bramki poszuki-

wanego układu: T0, T2 i C1-2. Ponieważ funkcja resztowa nie jest funkcją identycznościową, więc przeprowadza się dalsze iteracje.

Dla funkcji resztowej <1,0,2,3,5,4,6,7> istnieją dwie bramki C0-1 i N0, dla których SF-odległości są najmniejsze i wynoszą 1. Dla bramki C0-1 funkcja resztowa ma postać <1,0,3,2,5,4,7,6>, a dla N0 – <0,1,3,2,4,5,7,6>.

W następnej iteracji biorąc funkcję <1,0,3,2,5,4,7,6> otrzymuje się SF-odległość równą zero dla bramki N0, a funkcja resztowa będzie identycznościowa. Natomiast biorąc funkcję <0,1,3,2,4,5,7,6> otrzymuje się taki sam wynik dla bramki C0-1. W ten sposób wyznaczone zostają dwa układy odwracalne realizujące początkową funkcję <1,0,2,4,6,7,3,5>:

1. T0, T2, C1-2, C0-1, N0.
2. T0, T2, C1-2, N0, C0-1.

Trzeci optymalny układ T0, T2, N0, C1-2, C0-1 nie został znaleziony, gdyż po wyborze dwóch pierwszych bramek T0 i T2, dla funkcji resztowej po bramce N0 wyznaczona SF-odległość jest większa niż dla funkcji resztowej po bramce C1-2.

Wykonaliśmy obliczenia dla wszystkich funkcji trzech zmiennych. Dla 33477 funkcji (ponad 83%) otrzymaliśmy układy optymalne. Dla pozostałych 6843 funkcji otrzymaliśmy 6372 układy dłuższe o jedną bramkę, 444 układy dłuższe o 2 bramki i 27 układów dłuższych o 3 bramki.

Tablica 3

S-odległości funkcji resztowych

Bramka	Funkcja resztowa	S-odległości	SF-odległość
N0	<0,1,4,2,7,6,5,3>	3,4,1	8
C0-2	<1,0,2,4,7,6,5,3>	1,2,3	6
C0-1	<1,0,4,2,6,7,5,3>	3,4,1	8
T0	<1,0,2,4,6,7,5,3>	1,2,3	6
N1	<2,4,1,0,3,5,6,7>	3,4,4	11
C1-2	<1,0,2,4,3,5,6,7>	3,3,4	10
C1-0	<1,4,2,0,6,5,3,7>	3,3,3	9
T1	<1,0,2,4,6,5,3,7>	2,3,3	8
N2	<6,7,3,5,1,0,2,4>	3,4,3	10
C2-1	<1,0,3,5,6,7,2,4>	2,3,3	8
C2-0	<1,7,2,5,6,0,3,4>	2,3,4	9
T2	<1,0,2,5,6,7,3,4>	2,3,3	8

6. Wyniki dla wybranych funkcji czterech zmiennych

Przedstawiony algorytm został zastosowany do wybranych funkcji testowych (ang. *benchmarks*) [11]. Wyniki zostały podane w tabl. 4.

W opisywanym eksperymencie dla 18 spośród 37 funkcji testowych (benchmarków) otrzymano optymalne układy. Układy znalezione przez nasz algorytm są średnio tylko o 1,22 bramki (o 12,4%) dłuższe od optymalnych.

Tablica 4

Porównanie układów optymalnych z układami znalezionymi przez prezentowany algorytm

Nazwa funkcji testowej	Długość optymalnego układu [11]	Długość układu znalezionego przez algorytm	Różnica
4_49	12	14	2
4_49+hwb4	12	15	3
4b15g_1	15	15	0
4b15g_2	15	17	2
4b15g_3	15	15	0
4b15g_4	15	15	0
4b15g_5	15	15	0
f1	9	9	0
f2	8	9	1
g1	3	3	0
g2	4	4	0
g3	7	7	0
g4	9	9	0
g5	8	8	0
g6	8	9	1
g7	7	7	0
g8	9	9	0
g9	9	11	2
g10	9	11	2
g11	9	11	2
g12	9	15	6
oc5	11	14	3
oc6	12	12	0
oc7	13	14	1
oc8	12	13	1
mod10_171	9	9	0
mod10_176	7	11	4
gyang	10	10	0
hwb4	11	13	2
imark	7	7	0
mini_alu	6	6	0
mperk	9	13	4
nth_prime4_inc	11	12	1
primes4	10	12	2
App2.11	9	11	2
decode42	10	14	4
dmasl	9	9	0
			Śred=1,22

7. Podsumowanie

Programowa implementacja algorytmu ma niewielką złożoność i w praktycznych zastosowaniach może być przydatnym narzędziem w procesie syntezy układów odwracalnych. Nie wymaga też dużej pojemności pamięci i drogich komputerów.

BIBLIOGRAFIA

1. Golubitsky O. , Maslov D.: A study of optimal 4-bit reversible Toffoli circuits and their synthesis. *IEEE Trans. on Computers,* 61, (2012), p. 1341÷1353.
2. Kerntopf P.: Some remarks on reversible logic synthesis. *Proc. 8th International Workshop on Applications of the Reed-Muller Expansion in Circuit Design and Representations and Methodology of Future Computing Technology*, Oslo, Norway, May 2007, p. 51÷57.
3. Landauer R.: Irreversibility and heat generation in the computing process. *IBM J. Res. Dev. 5*, 183 (1961).
4. Maslov D., Dueck G.W., Miller D. M.: Techniques for the synthesis of reversible Toffoli networks, *ACM Transactions on Design Automation of Electronic Systems*, 12, 4 (Sept. 2007), article 42: p. 1÷28.
5. Miller D. M., Maslov D., Dueck G. W.: A transformation based algorithm for reversible logic synthesis. *Proc. Design Automation Conference*, Anaheim, CA, June 2003, p. 318÷323.
6. Mishchenko A., Perkowski M.: Logic synthesis of reversible wave cascades. *Proc. 11th IEEE/ACM International Workshop on Logic Synthesis*, New Orleans, LA, USA, June 2002, p. 197÷202.
7. Patel K.N., Markov I.L., Hayes J.P.: Optimal synthesis of linear reversible circuits. *Quantum Information and Computation*, 8, 3&4 (2008), p. 282÷294.
8. Prasad A.K., Shende V.V., Patel K.N., Markov I.L., Hayes J.P.: Data structures and algorithms for simplifying reversible circuits, *ACM Journal on Emerging Technologies in Computing Systems*, 2, 4 (2006), p. 277÷93.
9. Shende V.V., Prasad A.K., Markov I.L., Hayes J.P.: Reversible logic circuit synthesis. *IEEE Trans. on CAD*, 22, 6 (2003), p. 710÷22.
10. Skorupski A., Gracki K., Pawłowski M., Kerntopf P.: Synteza układów odwracalnych metodą różnicową. *Pomiary Automatyka Kontrola,* 2013, nr 8, p.784÷786.
11. Szyprowski M., Kerntopf P.: Reducing quantum cost in reversible Toffoli circuits. Proc. 10th Reed-Muller Workshop, Tuusula, Finland, May 2011, p. 127÷136; poprawiona wersja: http://arxiv.org/abs/1105.5831

12. Van Rentergem Y., De Vos A.: Synthesis and optimization of reversible circuits. *Proc. 8th International Workshop on Applications of the Reed-Muller Expansion in Circuit Design and Representations and Methodology of Future Computing Technology*, Oslo, Norway, May 2007, p. 67÷65.
13. Van Rentergem Y., De Vos A., De Keyser K.: Six synthesis methods for reversible logic. *Open Systems and Information Dynamics*, 14, 1 (2007), p. 91÷116.
14. Wille R., Grosse D.: Fast exact Toffoli network synthesis of reversible logic, *Proc. International Conference on CAD*, San Jose, CA, USA, Nov. 2007, p. 60÷64.
15. Wille R., Le H.M., Dueck G.W., Grosse D.: Quantified synthesis of reversible logic, *Proc. Design, Automation and Test in Europe Conference*, Munich, Germany, March 2008.
16. Yang G., Song X., Hung W.N.N., Perkowski M.A.: Fast synthesis of exact minimal reversible circuits using group theory. *Proc. 10th Asia and South Pacific Design Automation Conference*, Shanghai, China, 2005, p. 1002÷1005.
17. Yang G., Xie F., Song X., Hung W.N.N., Perkowski, M.: A constructive algorithm for reversible logic synthesis. *Proc. IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada, July 2006, p. 2416÷2421.
18. Yang G., Song X., Hung W.N.N., Xie F., Perkowski M.A.: Group theory based synthesis of binary reversible circuits. *Proc. 3rd International Conference on Theory and Applications of Models of Computation,* Lecture Notes in Computer Science, 3959, 2006, p. 365÷374.

Abstract

In this paper a new method of synthesis of reversible function is presented. The truth table of a reversible function has 2^n rows. To each *n*-bit input vector corresponds a unique *n*-bit output vector. Each of the output functions is balanced, i.e. the number of rows with output "1" is equal to the number of rows with output "0" (Tab. 1). For *n* variables there exist $2^n!$ different reversible functions. There are 24 reversible functions for 2 variables, 40320 functions for 3 variables and more than 20×10^{12} for 4 variables.

Reversible function can be implemented by a cascade of reversible gates (Fig. 1). The cascade circuits with the minimal number of gates are called optimal circuits. There are many computer-aided algorithms finding an optimal or a suboptimal circuit for a given reversible function [1, 4-17]. These algorithms usually require complex calculations or very big capacity of memory. In this work we propose a simple heuristic algorithm which generates an optimal or quasi-optimal solutions. The method is based on ordering of bits in each output column of the truth table of the function. The length of the shortest sequence of reversible gates which

leads to exact order of bits in a given column is the criterion of choosing a subsequent gate during every iteration step of the proposed synthesis algorithm.

An example of synthesis for a 3-variable reversible function defined by tab. 2 is presented. Tab. 3 shows the result of first iteration step of algorithm. In this example two optimal solutions have been obtained. Results of experimental calculations for 37 selected 4-variable benchmark functions are presented in tab. 4. In this experiment the average overhead of the circuits generated by the proposed algorithm is only 12.4% in the comparison with the optimal circuits.

Adresy

Andrzej SKORUPSKI, Politechnika Warszawska, Instytut Informatyki, ul. Nowowiejska 15/19, 00-665 Warszawa, Polska, ask@ii.pw.edu.pl
Krzysztof GRACKI, Politechnika Warszawska, Instytut Informatyki, ul. Nowowiejska 15/19, 00-665 Warszawa, Polska, kgr@ii.pw.edu.pl
Marek PAWŁOWSKI, Politechnika Warszawska, Instytut Informatyki, ul. Nowowiejska 15/19, 00-665 Warszawa, Polska, mpw@ii.pw.edu.pl
Paweł KERNTOPF, Uniwersytet Łódzki, Wydział Fizyki i Informatyki Stosowanej, Zakład Informatyki Stosowanej, ul. Pomorska 149/153, 90-236, Łódź, Polska, pke@ii.pw.edu.pl