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

Synteza układów odwracalnych metodą różnicową

Dr inż. Andrzej SKORUPSKI

Docent w Wyższej Szkole Menedżerskiej w Warszawie. Autor publikacji z dziedziny projektowania układów cyfrowych i architektury komputerów. Projektant wielu urządzeń i systemów cyfrowych wykorzystywanych zarówno w dydaktyce, pracach badawczych, jak i w innych dziedzinach techniki.

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

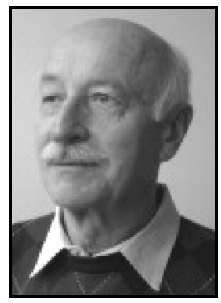
Mgr inż. Marek PAWŁOWSKI

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

W niniejszej pracy przedstawiony jest prosty algorytm projektowania układów odwracalnych. Proponowany algorytm polega na wyznaczeniu dla danej funkcji zbioru bramek (nazywanego zbiorem bramek pierwszych), które mogą znajdować się na początku układu kaskadowego realizującego zadaną funkcję. Po wyznaczeniu takiego zbioru można wybrać jeden z jego elementów, a następnie powtórzyć algorytm dla tzw. funkcji resztowej. Postępuje się tak, aż do momentu, gdy funkcja resztowa stanie się funkcją identycznościową. Liczba iteracji algorytmu jest równa liczbie bramek projektowanej kaskady.

Słowa kluczowe: odwracalne układy logiczne, projektowanie układów.

Difference method of reversible circuits synthesis

Abstract

Research on reversible logic circuits is motivated by advances in quantum computing, nanotechnology and low-power design. Implementation of such functions is realized by special gates. These gates always form a cascade circuit. Minimization of such circuits is a very difficult problem. In this paper a novel concept of synthesis of reversible logic is presented. For simplicity, the method is described for three variables only but it is scalable for more variables. The proposed method is based on XOR function applied to input and output sides of the truth table of a function to be synthesized. The result of applying XOR function indicates bits in the truth table which have to be changed by reversible gates. Due to this property the number of analyzed gates is small. We present the comparison of three variants of the difference method. Each of them leads to different numbers of 3-variable functions for which exact optimal circuits have been found.

Keywords: reversible logic circuits, circuit synthesis.

1. Wprowadzenie

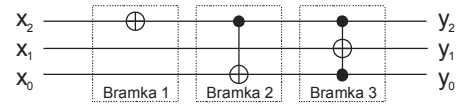
Zainteresowanie projektowaniem układów odwracalnych ma swoje źródło w możliwościach realizacji komputerów kwantowych, które pozwoliłyby na znaczne przyspieszenie obliczeń w porównaniu z obecnymi komputerami, jak również w możliwościach zmniejszania energii pobieranej przez układy cyfrowe (potencjalnie do zera) [1]. Metody projektowania układów odwracalnych skierowane są na zmniejszenie złożoności układów, m.in. na zmniejszanie liczby bramek. Niniejsza praca prezentuje nowy algorytm minimalizacji układów odwracalnych.

Funkcja boolowska o n argumentach i n wartościach (w skrócie $n*n$ funkcja) jest nazywana odwracalną, jeśli jest przekształceniem wzajemnie jednoznacznym. Dla n zmiennych istnieje $2^n!$ funkcji odwracalnych. Zatem, dla dwóch zmiennych istnieją 24 (4!) takie funkcje, dla trzech zmiennych jest ich 40320 (8!), a dla czterech zmiennych - ponad 20 bilionów (16!).

Bramka o n wejściach i n wyjściach (w skrócie $n*n$ bramka) jest nazywana odwracalną, jeśli realizuje odwracalną $n*n$ funkcję. Taką samą konwencję stosujemy do układów. W układach odwracalnych rozgałęzienia sygnałów są niedopuszczalne, zatem $n*n$ układ odwracalny jest kaskadą $k*k$ bramek odwracalnych, gdzie $k \le n$. Zbiór bramek, z których mogą być budowane układy, nazywany jest biblioteką bramek. W literaturze najczęściej rozpatrywana jest biblioteka NCT [2-5], która w przypadku 3*3 układów zawiera 1*1 bramki negacji NOT, 2*2 bramki sterowane CNOT i 3*3 bramki Toffoliego, np. dla zmiennej x_0 (nazywanej linią 1) bramki te są definiowane następująco ($\oplus$ oznacza XOR):

1. bramka negacji ***N***1 wykonuje operację $x_0 \oplus 1$,
2. bramka ***C***1-2 wykonuje operację $x_0 \oplus x_1$,
3. bramka ***C***1-3 wykonuje operację $x_0 \oplus x_2$,
4. bramka ***T***1 wykonuje operację $x_0 \oplus x_1 x_2$.

Analogicznie oznaczamy bramki dla pozostałych zmiennych (dla pozostałych linii). Układy o trzech wejściach można budować z 12 takich bramek (trzech bramek N, sześciu bramek C i trzech bramek T). Na rys. 1 przedstawiono kaskadę złożoną z bramek N3, C1-3 i T2.



Rys. 1. Układ złożony z trzech bramek odwracalnych
Fig. 1. A circuit consisting of three reversible gates

Można wykazać, że każdą funkcję trzech zmiennych da się zrealizować za pomocą co najwyżej 8 bramek. Układ złożony

z minimalnej liczby bramek NCT nazywany jest układem optymalnym [4].

2. Metoda różnicowa syntezy układu odwracalnego

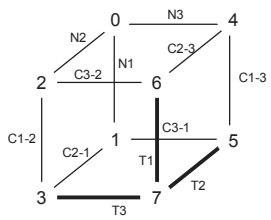
2.1. Wyznaczanie zbioru bramek pierwszych

Metoda różnicowa polega na wykorzystaniu tzw. funkcji XOR do ograniczenia liczności zbioru bramek, które należy rozpatrywać w każdym kroku algorytmu. Przedstawimy to na przykładzie. Niech będzie dana funkcja odwracalna w postaci tablicy prawdy, której lewa strona jest uporządkowanym zbiorem wektorów wejściowych, a prawa strona - zbiorem wektorów wyjściowych odpowiadających wektorom wejściowym funkcji. W tab. 1 pokazano funkcję, dla której wektory wyjściowe, zapisane w postaci dziesiętnej, występują w następującej kolejności (czytane od góry do dołu): 0,3,1,7,4,2,6,5 (skrótowy zapis tej funkcji to 1b5671[1]).

Tab. 1. Tablica prawdy funkcji odwracalnej 1b5671
Tab. 1. Truth table for the function 1b5671

L.p.	wektor wejściowy $x_2x_1x_0$	wektor wyjściowy $y_2y_1y_0$	Funkcja XOR
0	000	000	000
1	001	011	010
2	010	001	011
3	011	111	100
4	100	100	000
5	101	010	111
6	110	110	000
7	111	101	010

Ostatnia kolumna tablicy stanowi wynik zastosowania funkcji XOR do wektora wejściowego i odpowiadającego mu wektora wyjściowego funkcji odwracalnej. Jedynki funkcji XOR pokazują które bity wektora wejściowego należy zmienić, aby otrzymać wektory wyjściowe. Zmiany te można osiągnąć wykorzystując odpowiednie bramki.



Rys. 2. Sześcian z bramkami przyporządkowanymi poszczególnym krawędziom
Fig. 2. A cube with appropriate gates attached to edges

Działanie bramek może zostać przedstawione na sześcianie (rys. 2), którego wierzchołki oznaczone są numerami wierszy tablicy prawdy. Na krawędziach sześcianu podane są bramki, których należy użyć, aby uzyskać odpowiednie przejście między wartościami funkcji. Np. jeśli chcemy zastąpić wartość 0 wartością 4, to musimy użyć bramki N3, co spowoduje także zamianę wartości 1 i 5, 2 i 6 oraz 3 i 7. Jeśli należy zmienić wartości 1 i 5 bez zmiany 0 i 4, to należy użyć bramki C3-1. Należy pamiętać, że wtedy zostaną zmienione także wartości 3 i 7. Funkcję odwracalną można więc przedstawić za pomocą opisanego wyżej sześcianu, w którego wierzchołkach umieszczone zostaną odpowiednio wektory wyjściowe funkcji odwracalnej. Poszukiwanie układu odwracalnego realizującego daną funkcję odwracalną polega na takich zamianach wartości w wierzchołkach, aby otrzymać tzw. funkcję identycznościową czyli uporządkowanie jak na rys. 2.

Grube krawędzie sześcianu przedstawiają zmiany, które zachodzą podczas użycia dowolnej bramki na danej linii, np. zamiana wartości 7 i 5 zachodzi podczas użycia bramek N2, C2-3, C2-1 i T2. Warto podkreślić, że użycie dowolnej bramki zawsze zmienia wartość wierzchołka 7.

Każdej jedynce funkcji XOR odpowiada zatem zbiór bramek, których należy użyć, aby zamienić odpowiadający tej jedynce bit danej funkcji. Zbiór ten można wyznaczyć posługując się kostką przedstawioną na rys. 2, z zaznaczonymi wszystkimi bramkami z biblioteki NCT. W tab. 2 przedstawiono zbiory bramek umieszczanych na odpowiednich liniach zmieniające bity funkcji XOR. Każda bramka zamienia odpowiednie wiersze tablicy prawdy, więc zadaniem procesu syntezy jest znalezienie takich zamian wierszy, które prowadzą do układu implementującego zadaną funkcję.

Tab. 2. Tablica bramek odpowiadających jedynkom funkcji XOR
Tab. 2. Table of gates corresponding to 1's in the XOR function

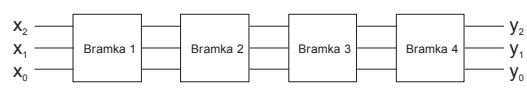
wiersz	Bramki na linii 3	Bramki na linii 2	Bramki na linii 1
0	N3	N2	N1
1	C3-1	C2-1	N1
2	C3-2	N2	C1-2
3	N3,C3-2,C3-1,T3	C2-1	C1-2
4	N3	C2-3	C1-3
5	C3-1	N2,C2-3, C2-1,T2	C1-3
6	C3-2	C2-3	N1, C1-3,C1-2,T1
7	N3,C3-2,C3-1,T3	N2,C2-3, C2-1,T2	N1, C1-3,C1-2,T1

Z tab. 2 wynika, że wiersz 0 zmieniają tylko bramki N*i*. Wiersze 1, 2 i 4 są zmieniane przez odpowiednie bramki N*i* i C*i*. Wiersze 3, 5 i 6 są zmieniane przez odpowiednie bramki N*i*, C*i* i T*i*. Wiersz 7 jest zmieniany przez każdą bramkę.

Posługując się tab. 2 można wyznaczyć zbiór bramek, które należy zastosować dla danej funkcji. Zadaniem proponowanego algorytmu jest zawężenie zbioru rozpatrywanych bramek do tych, które występują najliczniej w funkcji XOR.

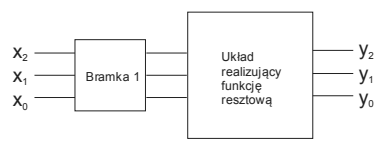
Analizując przykładową funkcję (tab. 1) można zauważyć, że w wierszu nr 1 jedynka funkcji XOR występuje na linii drugiej. Oznacza to, że do zbioru bramek pierwszych należy dodać bramkę C2-1 (tab. 2). Z analizy wiersza nr 2 wynika, że należy dodać bramki N2 i C1-2 (jedynki w funkcji XOR na liniach 2 i 1). Postępując tak dalej i analizując kolejne wiersze tabeli, należy zliczać częstość wystąpień każdej z bramek.

Implementacji dowolnej funkcji odwracalnej można dokonać za pomocą kaskadowego układu składającego się wyłącznie z wyżej omówionych bramek. Na rysunku 3 jest pokazany schematycznie taki układ, w którym wyjścia każdej bramki sterują wejściami następnej, a linie nie krzyżują się.



Rys. 3. Kaskadowy układ złożony z bramek odwracalnych
Fig. 3. A cascade circuit consisting of reversible gates

Zadaniem procesu projektowania układu odwracalnego realizującego zadaną funkcję jest znalezienie minimalnej sekwencji bramek realizującej daną tablicę prawdy.



Rys. 4. Początkowa dekompozycja układu
Fig. 4. Initial decomposition of the circuit

Proponowana metoda różnicowa polega na wyznaczeniu dla danej funkcji zbioru bramek, które mogą być umieszczone na początku kaskady (nazywanego zbiorem bramek pierwszych). Po

[1] Skrócony, kolumnowy zapis funkcji w postaci szesnastkowej

wyznaczeniu takiego zbioru można wybrać jedną z należących do niego bramek i poszukiwany układ przedstawić w postaci dekompozycji pokazanej na rys. 4.

Po wyborze bramki 1 oblicza się tzw. funkcję resztową, dla której należy powtórzyć procedurę i znowu wyznaczyć zbiór bramek pierwszych. Tak postępuje się aż do momentu, gdy uzyskana funkcja resztowa jest funkcją identycznościową. Podstawowym problemem jest więc wybór odpowiedniej bramki 1 dla dowolnej funkcji odwracalnej. W wielu metodach jako bramkę 1 wybiera się dowolną bramkę ze zbioru 12 bramek NCT. Natomiast w metodzie różnicowej korzysta się ze zbiorów o mniejszej liczności, który otrzymywany jest z tab. 2.

2.2. Wersja 1 algorytmu

Podstawowa wersja algorytmu korzysta z tab. 2 do wyznaczenia zbioru bramek pierwszych. Jedynki funkcji XOR wskazują komórki tab. 2, w których znajdują się bramki służące do wyznaczenia bramki 1.

Krok 1. Dla danej funkcji odwracalnej wyznaczyć funkcję XOR.
Krok 2. Z tab. 2 wyznaczyć zbiór bramek odpowiadający każdej jedynce (bez wiersza 7) funkcji XOR.
Krok 3. Znaleźć zbiór bramek realizujących największą liczbę jedynek funkcji XOR.
Krok 4. Dla każdej bramki z tego zbioru wyznaczyć funkcję resztową i wybrać rozwiązanie składające się z najmniejszej liczby bramek (przez porównanie).

Wynik:

Dla wszystkich (40320) funkcji trzech zmiennych otrzymano 33639 rozwiązań dłuższych od optymalnych, a średnia długość rozwiązań wynosi 7,88 (dla optymalnych 5,87). Średnia liczba rozpatrywanych bramek wynosi 2,92, a średnia liczba iteracji dla wszystkich funkcji wynosi 3896.

2.3. Wersja 2 algorytmu

Duża liczba nieoptymalnych rozwiązań otrzymanych w wersji 1 algorytmu skłoniła nas do zwiększenia zbioru bramek branych do wyznaczenia bramki 1 (dodanie bramek T*i* przy spełnieniu dodatkowego warunku).

Krok 1. Dla danej funkcji odwracalnej wyznaczyć funkcję XOR.
Krok 2. Z tab. 2 wyznaczyć zbiór bramek odpowiadający każdej jedynce (bez wiersza 7) funkcji XOR.
Krok 3. Znaleźć zbiór bramek realizujących największą liczbę jedynek funkcji XOR.
Krok 4. Uzupełnić go o bramki T*i*, jeśli są one jedynymi w danej kolumnie.
Krok 5. Dla każdej bramki z tego zbioru wyznaczyć funkcję resztową i wybrać rozwiązanie składające się z najmniejszej liczby bramek (przez porównanie).

Wynik:

Dla wszystkich funkcji trzech zmiennych otrzymano 28862 rozwiązań dłuższych od optymalnych, a średnia długość rozwiązań wynosi 7,18. Dla poszczególnych funkcji średnia liczba rozpatrywanych bramek wynosi 2,96, a średnia liczba iteracji dla wszystkich funkcji wynosi 5814.

2.4. Wersja 3 algorytmu

Dalsze zmniejszenie liczby nieoptymalnych rozwiązań otrzymanych w wersji 1 i 2 algorytmu może nastąpić poprzez zwiększenie liczności zbioru bramek spośród których wybierana jest bramka 1. W tej wersji zwiększono priorytet dla wierzchołka 0, gdyż teraz zmiana jego wartości może nastąpić tylko dzięki użyciu bramki N*i*.

Krok 1. Dla danej funkcji odwracalnej wyznaczyć funkcję XOR.
Krok 2. Z tab. 2 wyznaczyć zbiór bramek odpowiadający każdej jedynce (bez wiersza 7) funkcji XOR.
Krok 3. Znaleźć zbiór bramek realizujących największą liczbę jedynek funkcji XOR i uzupełnić go o bramki T*i*, jeśli są one jedynymi w danej kolumnie.
Krok 4. Nadać większy priorytet bramkom realizującym zamianę wierszy o wartościach 0.
Krok 5. Dla każdej bramki z tego zbioru wyznaczyć funkcję resztową i wybrać rozwiązanie składające się z najmniejszej liczby bramek (przez porównanie).

Wynik:

Dla wszystkich funkcji trzech zmiennych otrzymano 24039 rozwiązań dłuższych od optymalnych, a średnia długość rozwiązań wynosi 6,21. Dla poszczególnych funkcji średnia liczba rozpatrywanych bramek wynosi 4,11, a średnia liczba iteracji dla wszystkich funkcji wynosi 12053.

3. Podsumowanie

W pracy przedstawiono algorytm syntezy układów odwracalnych i dla trzech zmiennych podano wyniki syntezy tym algorytmem dla trzech wariantów. Algorytm nie prowadzi do otrzymania optymalnych układów dla wszystkich funkcji. Pokazano, że zbliżanie się do rozwiązań optymalnych wymaga zwiększenia złożoności obliczeniowej. Wynika to z liczby bramek branych pod uwagę przy rozpatrywaniu zbioru bramek pierwszych. W tab. 3 porównano wyniki działania trzech wariantów algorytmu, zarówno pod kątem optymalności uzyskanych rozwiązań, jak i ich złożoności obliczeniowej. Pokazuje ona zależność długości otrzymanego układu od liczności zbioru bramek pierwszych.

Tab. 3. Porównanie wyników dla trzech wariantów algorytmu różnicowego
Tab. 3. Comparison of the results for three variants of the difference algorithm

	Optimum	Wersja 1	Wersja 2	Wersja 3
Liczba układów nieoptymalnych	-	33639	28862	24039
Maksymalna dł. układu	8	17	16	11
Średnia dł. rozwiązania	5,87	7,88	7,18	6,21
Średnia liczność zbioru bramek pierwszych	12	2,92	2,96	4,11
Średnia liczba iteracji	-	3896	5814	12053

4. Literatura

[1] De Vos A.: Reversible Computing. Fundamentals, Quantum Computing and Applications, Wiley-VCH, Berlin 2010.
[2] Saeedi M., Markov I.L.: Synthesis and optimization of reversible circuits – a survey, ACM Computing Surveys, 2013, także w: arXiv:1110.2574.
[3] Skorupski A., Szyprowski M., Kerntopf P.: Algorytm syntezy kombinacyjnych układów odwracalnych, Pomiary Automatyka Kontrola, vol. 57, nr 8, 2011, ss. 858-860.
[4] Szyprowski M., Kerntopf P.: Metody konstrukcji optymalnych układów odwracalnych, Pomiary Automatyka Kontrola, vol. 58, nr 7, 2012, ss. 647-649.
[5] Kerntopf P.: Some remarks on reversible logic synthesis, Proc.-8th International Reed-Muller Workshop, 2007, ss. 51-57.

otrzymano / received: 09.05.2013
przyjęto do druku / accepted: 03.07.2013 *artykuł recenzowany / revised paper*