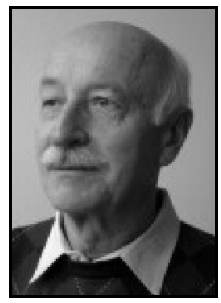
Paweł KERNTOPF
POLITECHNIKA WARSZAWSKA, INSTYTUT INFORMATYKI

Synteza odwracalnych układów logicznych

Dr hab. inż. Paweł KERNTOPF

Ukończył studia na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. Obecnie pracuje na stanowisku profesora PW w Instytucie Informatyki na tym Wydziale. 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

Opracowywanie metod syntezy binarnych odwracalnych układów logicznych rozpoczęto niedawno. Artykuł zawiera krótki przegląd publikacji na ten temat, w tym wyniki autora.

Słowa kluczowe: odwracalne bramki logiczne, algorytmy syntezy odwracalnych układów.

Synthesis of reversible logic circuits

Abstract

The development of synthesis methods for binary reversible logic circuits has started recently. The paper presents a brief survey of publications on the topic including the author's results.

Keywords: reversible logic gates, algorithms for reversible circuit synthesis.

1. Wprowadzenie

Układy logiczne nazywane są odwracalnymi (ang. reversible), jeśli nie następuje w nich strata informacji, tzn. jeśli na podstawie wektora wartości sygnałów wyjściowych można jednoznacznie odtworzyć wektor wartości sygnałów wejściowych: w tablicach prawdy takich układów każda kombinacja wartości sygnałów wyjściowych występuje dokładnie jeden raz.

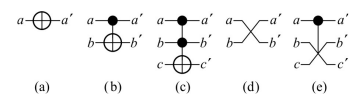
W ostatnich kilku latach opublikowano wiele prac na temat układów odwracalnych. To duże zainteresowanie wynika stąd, iż stosowanie takich układów prowadzi do zmniejszenia energii wydzielanej przez układ [1]. Zatem prace w tym kierunku mają duże znaczenie przy opracowywaniu przyszłych technologii komputerowych, w szczególności – nanotechnologii. Ponadto, wszystkie układy kwantowe mają z natury własność odwracalności. Pokazano, iż każdy kwantowy układ logiczny można podzielić na część kwantową i nie-kwantową, a następnie otrzymać optymalną postać całego układu poprzez dokonanie optymalizacji obu części oddzielnie, więc opracowanie efektywnych metod syntezy układów odwracalnych może pomóc w syntezie układów kwantowych. W niektórych przypadkach duża część układu kwantowego zawiera wyłącznie klasyczne bramki odwracalne. Warto dodać, iż w takich zadaniach obliczeniowych, jak cyfrowe przetwarzanie sygnałów, komunikacja, kryptografia i grafika komputerowa wymaga się, aby informacja nie była tracona w trakcie obliczeń, a więc są to zadania odwracalne [2]. Dodam także, iż opracowano i wykonano w technologii CMOS odwracalne procesory, a także odwracalne języki programowania i kompilatory dla nich [3].

Synteza układów odwracalnych różni się od syntezy układów konwencjonalnych. W układach odwracalnych linie sygnałowe nie mogą się rozgałęziać, więc liczby wejść i wyjść są takie same – są to wyłącznie układy kaskadowe. Aby taki układ mógł realizować funkcje nieodwracalne, niezbędne jest dodawanie wyjść, na których sygnały nie są wykorzystywane (ang. garbage outputs), jak również wejść, na które przykładane są sygnały stałe. To, jak dobierać zależności funkcjonalne pomiędzy wejściami i wyjściami, aby otrzymać układ o możliwie najmniejszej liczbie wejść i wyjść, jest wciąż problemem nierozwiązanym. W pracy przyjmujemy – podobnie jak w innych publikacjach – że ten etap został już dokonany.

2. Bramki odwracalne

Bramkę (układ) będziemy nazywali *odwracalną (odwracalnym)*, jeśli realizuje wzajemnie jednoznaczne odwzorowanie zbioru wektorów wejściowych na zbiór wektorów wyjściowych. Jeśli bramka (układ) ma n wejść i n wyjść, będziemy pisali, że jest to $n*n$ bramka (układ).

Przy projektowaniu układów istotne jest ustalenie, które bramki są uniwersalne. Prawie wszystkie bramki odwracalne mają taką własność. Autor niniejszej pracy w [4] obliczył, że spośród wszystkich 8! = 40320 3*3 binarnych bramek tylko 1344 (3,33 %) nie jest uniwersalnych, oraz pokazał, że spośród wszystkich 16! 4*4 binarnych bramek co najwyżej 0,0000028 % nie jest uniwersalnych (ze wzrostem n ich procent maleje), co potwierdziły wzory podane później w [5] (wynik ten uogólniono też na bramki wielowartościowe [6]).



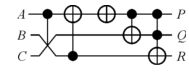
Rys. 1. Graficzne oznaczenia bramek odwracalnych: a) NOT, b) CNOT, c) Toffoliego, d) SWAP, e) Fredkina
Fig. 1. Pictorial representations of reversible gates: a) NOT, b) CNOT, c) Toffoli, d) SWAP, e) Fredkin

W literaturze najczęściej rozpatrywanych jest pięć bramek: 1*1 NOT, 2*2 CNOT, 3*3 Toffoliego, 2*2 SWAP i 3*3 Fredkina (w skrócie: N, C, T, S i F). Zbiór trzech pierwszych bramek jest nazywany biblioteką NCT, czterech pierwszych – biblioteką NCTS, wszystkich pięciu – biblioteką NCTSF. Oznaczenia graficzne tych bramek podane są na rys. 1. Oto ich definicje:

bramka NOT: $a' = 1 \oplus a$;
bramka CNOT: $a' = a$, $b' = a \oplus b$;
bramka Toffoliego: $a' = a$, $b' = b$, $c' = c \oplus ab$;
bramka SWAP: $a' = b$, $b' = a$;
bramka Fredkina: $a' = a$,
jeśli $a = 0$, to $b' = b$, $c' = c$
jeśli $a = 1$, to $b' = c$, $c' = b$

Wejścia a i b w bramce Toffoliego oraz wejście a w bramkach CNOT i Fredkina nazywane są wejściami sterującymi (na rys. 1 oznaczone czarnymi kółkami), natomiast pozostałe wejścia tych bramek – sterowanymi.

Przydatne są skrótowe oznaczenia bramek z rys. 1 jako NOT(a), CNOT(a;b), TOF(a;b,c), SWAP(a,b) i FRE(a;b,c), gdzie średnikami oddzielone są wejścia sterujące od sterowanych. Podobnie będziemy oznaczali bramki, w których nastąpiła permutacja wejść i wyjść, np. w układzie podanym na rys. 2 drugą od lewej bramkę będziemy opisywali jako CNOT(C;A).



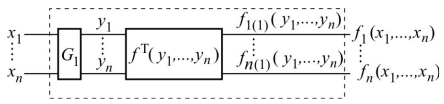
Rys. 2. Optymalny układ dla funkcji podanej w Tab. 1
Fig. 2. Optimal circuit for the function shown in Tab. 1

3. Przegląd algorytmów

Najwcześniejsza koncepcja systematycznego algorytmu syntezy oparta była na możliwości bezpośredniego przekształcania opisu funkcji w postaci wyrażenia Reeda-Mullera (ang. PPRM –

positive polarity Reed-Muller) na układ odwracalny [7]. W tym celu dla dowolnego n zdefiniowano $n*n$ bramki, będące uogólnieniami bramek Toffoliego. Taka biblioteka nie była przez nikogo więcej rozpatrywana, dlatego nie będziemy omawiać tego podejścia szczegółowo. Warto jednak podkreślić, iż była to jedyna metoda, w której podano realizację funkcji nie w pełni określonych.

W pracy omówimy opracowane w ostatnich latach metody inkrementacyjne, w których w każdym kroku wykonywania algorytmu wybiera się jedną bramkę (rys. 3) lub kilka bramek naraz, a potem oblicza się funkcję resztową f^T, tj. taką, która pozostała jeszcze do realizacji po dokonanym wyborze bramek w danym kroku. Cykl ten powtarzany jest aż do momentu, gdy funkcja resztowa będzie funkcją tożsamościową, co oznacza, iż zbudowany układ realizuje zadaną funkcję odwracalną. Wyboru bramek można dokonywać od lewej strony układu do prawej (rys. 3), w kierunku odwrotnym lub w obydwu kierunkach.



Rys. 3. Ogólny schemat wykonania jednego kroku algorytmu inkrementacyjnego
Fig. 3. General scheme of one step of an incremental algorithm

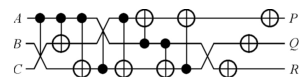
Trzy algorytmy, które niżej omówimy, będziemy nazywali – od pierwszych liter nazwisk ich autorów – algorytmami MMD, AJG i K. Pierwszy z nich ([10, 11]) oparty jest na stopniowej modyfikacji tablicy prawdy. W każdym kroku algorytmu przegląda się w porządku leksykograficznym wiersze tablicy prawdy dla funkcji resztowej (w kroku zerowym – zadanej funkcji), aż do znalezienia pierwszego wiersza, w którym wektor wyjściowy różni się od wejściowego. Po jego znalezieniu wybiera się z biblioteki taki zestaw bramek, aby po ich dodaniu do układu (od prawej strony do lewej) nastąpiła w tablicy dla funkcji resztowej zgodność wektorów w danym wierszu, a przy tym w wierszach poprzednich nic się nie zmieniło (celem jest zmniejszenie liczby pozycji, na których w tablicy prawdy część wyjściowa różni się od części wejściowej). Następnie powtarza się przeglądanie tablicy, aby znaleźć kolejny wiersz, w którym występuje niezgodność wektorów, aż dojdziemy do tablicy funkcji tożsamościowej. Otrzymany układ może zawierać znacznie więcej bramek niż układ optymalny. Dlatego autorzy algorytmu po wygenerowaniu układu stosują jego upraszczanie na podstawie tzw. wzorców (ang. template), które są parami równoważnych podukładów o różnej liczbie bramek. Wymaga to iteracyjnego przeglądania redukowanych układów, gdyż czasami zastosowanie jednego z wzorców umożliwia przeprowadzenie dalszej redukcji na podstawie innego wzorca.

Tab. 1 ilustruje wykonywanie algorytmu MMD dla funkcji zdefiniowanej przez pierwsze dwie kolumny tej tablicy (kolejność zmiennych od lewej: w części wejściowej – C, B, A, w części wyjściowej – R, Q, P). Niezgodność wytłuszczonych wektorów w pierwszym wierszu usuwamy w Kroku 0 przez wybranie trzech bramek negacji NOT(A), NOT(B), NOT(C). W Kroku 1 znajdujemy różnicę (**001**,**110**), która powstała w wierszu 2 po wykonaniu Kroku 0. Można ją usunąć dodając bramkę SWAP(B,C), a następnie bramkę CNOT(C;A), która spowoduje zanegowanie wartości zmiennej C w tych wierszach, w których A=1.

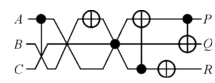
Analogicznie w Kroku 2, aby usunąć różnicę w wierszu trzecim (**010**,**111**) dodajemy bramki CNOT(B;C) i CNOT(B;A). W Kroku 3 znajdujemy różnicę w wierszu czwartym (**011**, **100**) i usuwamy ją dzięki bramkom CNOT(A;C) i FRE(C;A,B). W kroku 4 usuwamy różnicę w wierszu piątym (**100**,**111**), dodając bramki CNOT(A;C) i CNOT(A;B). Wreszcie, w Kroku 5 usuwamy różnicę w wierszu szóstym (**101**,**110**) przez dodanie bramki FRE(A;B,C) i uzyskujemy kolumnę wyjściową identyczną z kolumną wejściową, co kończy obliczenia. Otrzymany układ, pokazany na rys. 4, liczy 12 bramek. Po zastosowaniu redukcji za pomocą wzorców liczba bramek zmniejszyła się do 7 (optymalny układ na rys. 2 zawiera 5 bramek).

Tab. 1. Przykład podstawowego algorytmu syntezy MMD
Tab. 1. An example of the basic MMD synthesis algorithm

We	Wy	Krok 0	Krok 1	Krok 2	Krok 3	Krok 4	Krok 5
000	**111**	*000*	*000*	*000*	*000*	*000*	*000*
001	001	**110**	***001***	***001***	***001***	***001***	***001***
010	100	011	**111**	*010*	*010*	*010*	*010*
011	011	100	100	**100**	*011*	*011*	*011*
100	000	111	011	110	**111**	*100*	*100*
101	010	101	110	011	101	**110**	*101*
110	110	001	010	111	110	101	*110*
111	101	010	101	101	100	111	*111*



Rys. 4. Układ odwracalny otrzymany za pomocą modyfikacji tablicy prawdy (Tab. 1)
Fig. 4. Reversible circuit designed by modification of the given truth table (Tab. 1)



Rys. 5. Układ z Rys. 2 po uproszczeniach
Fig. 5. Circuit from Fig. 2 after simplification

Opis funkcji w postaci PPRM zastosowany został w algorytmie AJG [8]. Przedstawimy podstawową wersję algorytmu posługując się przykładem. Załóżmy, że 3*3 układ opisany jest następującymi równaniami, w których a, b, c oznaczają wejścia, zaś a', b', c' - wyjścia:

$$a' = a \oplus 1, \quad b' = b \oplus c \oplus ac, \quad c' = b \oplus ab \oplus ac. \qquad (1)$$

Jako miarę złożoności układu przyjęto łączną liczbę wyrazów po prawej stronie tych równań, która dla (1) wynosi 2+3+3 = 8. Kolejne bramki wybierane są w taki sposób, aby złożoność funkcji resztowej monotonicznie malała. Dla obliczenia równań opisujących funkcję resztową podstawia się wyrażenie PPRM odpowiadające bramce (N, C lub T) w miejsce zmiennej a, b lub c w równaniach definiujących zmienne wyjściowe zgodnie z poniższymi regułami:

1) zamiast zmiennej a podstawiamy
   $a \oplus 1$, jeśli wybraną bramką jest N(a),
   $a \oplus b$, jeśli wybraną bramką jest CNOT(b;a)
   $a \oplus c$, jeśli wybraną bramką jest CNOT(c;a)
   $a \oplus bc$, jeśli wybraną bramką jest TOF(b,c;a)
2) zamiast zmiennej b podstawiamy
   $b \oplus 1$, jeśli wybraną bramką jest N(b),
   $b \oplus a$, jeśli wybraną bramką jest CNOT(a;b)
   $b \oplus c$, jeśli wybraną bramką jest CNOT(c;b)
   $b \oplus ac$, jeśli wybraną bramką jest TOF(a,c;b)
3) zamiast zmiennej c podstawiamy
   $c \oplus 1$, jeśli wybraną bramką jest N(c),
   c $\oplus$ a, jeśli wybraną bramką jest CNOT(a;c)
   $c \oplus b$, jeśli wybraną bramką jest CNOT(b;c)
   c $\oplus$ ab, jeśli wybraną bramką jest TOF(a,b;c)

W wersji podstawowej algorytmu najpierw przegląda się wyrażenia PPRM definiujące układ. Jeśli w wyrażeniu dla zmiennej wyjściowej x_i' występuje zmienna wejściowa x_i oraz wyraz t nie zawierający tej zmiennej, to dla każdego takiego przypadku oblicza się złożoność po podstawieniu $x_i = x_i \oplus t$. Do dalszych obliczeń (w formie drzewa poszukiwań) bierze się podstawienia, które prowadzą do zmniejszenia złożoności. Zastosujemy te reguły do równań (1). Po prawej stronie równania dla a' występuje a i wyraz 1, w którym nie występuje zmienna a. Zatem pierwszym podstawieniem do sprawdzenia jest $a = a \oplus 1$. W równaniu dla b' występuje b oraz dwa wyrazy nie zawierające b (tj. c, ac). Zatem mamy do sprawdzenia jeszcze dwa podstawienia: $b = b \oplus c$ i $b = b \oplus ac$. W równaniu dla c' nie występuje c, więc nie jest spełniony warunek generowania

podstawienia. Pierwsze z wygenerowanych podstawień prowadzi do funkcji resztowej

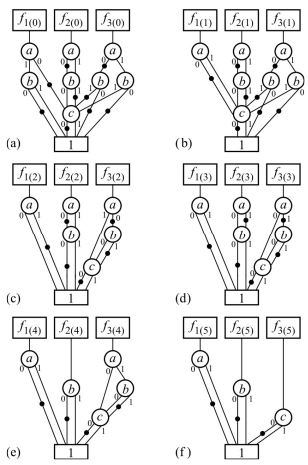
$$a' = a, \quad b' = b \oplus ac, \quad c' = c \oplus ab \oplus ac. \tag{2}$$

o złożoności 6, natomiast pozostałe podstawienia dają złożoność 7. Dlatego najpierw badamy nowe podstawienia dla równań (2). Postępując podobnie jak poprzednio, otrzymujemy dwa podstawienia: $b = b \oplus ac$ i $c = c \oplus ab$. Pierwsze z nich daje równania o złożoności 4:

$$a' = a, \quad b' = b, \quad c' = c \oplus ab, \tag{3}$$

podczas gdy drugie – o złożoności 7. Równania (3) generują tylko podstawienie $c = c \oplus ab$, które prowadzi do równań tożsamościowych. Zatem w ten sposób znaleźliśmy układ, będący sekwencją bramek TOF(a,b;c), TOF(a,c;b) i N(a) – w kolejności od lewej do prawej.

W pierwotnej wersji algorytmu AJG reguły wyboru nie obejmowały wszystkich możliwych podstawień, prowadzących do zmniejszenia złożoności, co niekiedy uniemożliwiało znalezienie rozwiązania. Dlatego autorzy opracowali nową wersję algorytmu [9], w której podstawienia są generowane także w takich przypadkach, gdy w równaniu dla zmiennej wyjściowej x_i' zmienna wejściowa x_i nie występuje po prawej stronie równania. Nie opracowali jednak rozszerzenia algorytmu na inne biblioteki bramek niż NCT.



Rys. 6. Diagramy CSBDD dla funkcji resztowych obliczonych przez algorytm K
Fig. 6. CSBDDs for the reminder functions calculated by the algorithm K

Algorytmy MMD i AJG nie mogą być stosowane dla dowolnych bibliotek bramek i dowolnych funkcji kosztów bramek. Nie ma takiego ograniczenia algorytm K [12-14]. Po licznych eksperymentach wybrano dla niego następującą nową miarę: $D(f) = s(f) - n$, gdzie $s(f)$ oznacza rozmiar połączonego binarnego diagramu decyzyjnego z negowanymi gałęziami (ang. shared binary decision diagram with complemented edges, w skrócie CSBDD) dla funkcji f o n wejściach i wyjściach. Wyraz $-n$ we wzorze na $D(f)$ ma służyć normalizacji tej miary, tzn. temu, by funkcja tożsamościowa miała złożoność równą zeru. W każdym kroku wykonywania algorytmu K po kolei rozpatruje się wszystkie bramki, obliczając dla nich złożoność funkcji resztowych, a następnie wybiera się do dalszych obliczeń bramkę lub bramki, które umożliwiły największą redukcję złożoności. Na Rys. 6 pokazano diagramy CSBDD dla kolejnych etapów konstruowania układu realizującego funkcję, która była rozpatrywana wyżej w przykładzie stosowania algorytmu MMD. Ich rozmiary wynoszą kolejno 5, 4, 3, 3, 2, 0, malejąc monotonicznie (efektem jest układ optymalny z Rys. 2).

Przedstawione algorytmy nie wyczerpują propozycji opisanych w literaturze, jednak są według autora najbardziej interesujące dla dalszego rozwoju dziedziny (obszerną bibliografię tematu zawiera [15]). Czynione są próby projektowania regularnych struktur o własnościach analogicznych do FPGA, jak np. Reversible Programmable Gate Array [16]. Warto też wspomnieć, że układy FPGA stosowane są do emulacji układów kwantowych (np. [17, 18]).

4. Podsumowanie

Pierwsze systematyczne algorytmy syntezy odwracalnych układów logicznych opracowane zostały dopiero w ostatnich latach. W pracy przedstawiony został przegląd niektórych z nich, z uwzględnieniem wyników autora. Pozwalają one na projektowanie optymalnych układów o niewielkiej liczbie wejść i wyjść. Dziedzina ta wymaga dalszych badań, aby rozwinąć algorytmy skalowalne. Analizy i otwarte problemy przedstawione są w [15]. Pomocą dla pracujących w tej dziedzinie jest strona internetowa [19], która powstała w 2004 r. i zawiera m.in. najlepsze znalezione do tej pory realizacje typowych układów.

5. Literatura

[1] C.H. Bennett, Notes on the history of reversible computation, IBM Journal of Research and Development, 1988, vol. 30, pp. 16-23

[2] V.V. Shende, A.K. Prasad, I.L.Markov, J.P. Hayes, Synthesis of reversible logic circuits, IEEE Trans. on Computer-Aided Design, 2003, vol. 22, no 6, pp. 710-722

[3] M.P. Frank, Reversibility for efficient computing, Ph.D. Thesis, Massachussets Institute of Technology, Cambridge, MA, USA, 1999

[4] P. Kerntopf, On universality of binary reversible logic gates, Proc. 5th Int. Workshop on Boolean Problems, Freiberg, Niemcy, 2002, pp. 47-52

[5] A. De Vos, L. Storme, All non-linear reversible logic gates are r-universal, Proc. 6th Int. Workshop on Boolean Problems, Freiberg, Niemcy, 2004, pp. 25-31

[6] P. Kerntopf, M.A. Perkowski, M.H.A. Khan, On universality of general reversible multiple-valued logic gates, Journal of Multiple-Valued Logic and Soft Computing, 2006, vol. 12, no 5-6, pp. 717-429

[7] A. Mishchenko, M. Perkowski, Logic synthesis of reversible wave cascades, Proc. 11th IEEE/ACM Workshop on Logic and Synthesis, New Orleans, USA, 2002, pp. 197-202

[8] A. Agrawal, N.K. Jha, Synthesis of reversible logic, Proc. Design Automation & Test in Europe Conference, Feb. 2004, pp. 21384-21385

[9] P. Gupta, A. Agrawal, N.K. Jha, An algorithm for synthesis of reversible logic circuits, IEEE Trans. on Computer-Aided Design, 2006, vol. 25, pp. 2317-2330

[10] D.M. Miller, D. Maslov, G.W. Dueck, A transformation based algorithm for reversible logic synthesis, Proc. 40th Design Automation Conference, ACM, Anaheim, CA, USA, 2003, pp. 318-323

[11] D. Maslov, G.W. Dueck, D.M. Miller, Fredkin/Tofoli templates for reversible logic synthesis, Proc. Int. Conference on Computer-Aided Design, ACM, San Jose, CA, USA, 2003, pp. 256-261

[12] P. Kerntopf, Reversible logic circuit synthesis based on a new complexity measure, Proc. 13th IEEE/ACM Workshop on Logic and Synthesis, Temecula, USA, 2004, pp. 106-113

[13] P. Kerntopf, A new heuristic algorithm for reversible logic circuit synthesis, Proc. 41st Design Automation Conference, ACM, San Diego, USA, 2004, pp. 834-837

[14] P. Kerntopf, An approach to synthesis of multiple-valued reversible logic circuits, Proc. 6th Int. Workshop on Boolean Problems, Freiberg, Niemcy, 2004, pp. 33-40

[15] P. Kerntopf, Some remarks on reversible logic synthesis, Proc. 8th Int. Workshop on Representations and Methodology of Future Computer Technology, Oslo, 2007

[16] M. Perkowski, P. Kerntopf, A. Coppola, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, Xiaoyu Song, A. Al-Rabadi, L. Jóźwiak, B. Massey, Regularity and symmetry as a base for efficient realization of reversible logic circuits, Proc. 10th IEEE/ACM Workshop on Logic and Synthesis, Granlibakken, CA, USA, 2001, pp. 90-95

[17] G. Negovetovic, M. Perkowski, M. Lukac, A. Buller, Evolving quantum circuits and an FPGA-based quantum computing emulator, Proc. 5th Int. Workshop on Boolean Problems, Freiberg, Niemcy, 2002, pp. 15-22

[18] A.U. Khalid, Z. Zilic, K. Radecka, FPGA emulation of quantum circuits, Proc. IEEE Int. Conference on Computer Design, 2004, pp. 310-315

[19] D. Maslov, G. Dueck, N. Scott, Reversible logic synthesis benchmarks page, http://www.cs.uvic.ca/~dmaslov/

Artykuł recenzowany