Alexander A. BARKALOV
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Projektowanie skończonego automatu stanów z wyjściami typu Mealy'ego z transformacją kodów obiektów w systemie funkcji wzbudzeń

Prof. dr hab. inż. Alexander A. BARKALOV

Prof. Alexander A. Barkalov ukończył studia magisterskie (1976) o specjalności Komputery na Politechnice Donieckiej. W latach 1980-83 odbył studia doktoranckie w Instytucie Mechaniki i Optyki w Leningradzie, gdzie obronił rozprawę doktorską z zakresu informatyki. W 1995 roku obronił rozprawę habilitacyjną w Instytucie Cybernetyki im. V.M. Glushkova w Kijowie i uzyskał tytuł doktora habilitowanego ze specjalnością informatyka. Od 2004 pracuje jako profesor na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego.

e-mail: a.barkalov@iie.uz.zgora.pl

Streszczenie

W artykule została zaprezentowana metoda optymalizacji kosztu układu logicznego skończonego automatu stanów z wyjściami typu Mealy'ego. Metoda opiera się na transformacji kodów obiektów w systemie funkcji wzbudzeń przerzutników. Jako obiekt automatu Mealy'ego rozumie się wewnętrzny stan automatu i zbiór mikrooperacji. Głównym założeniem jest aby stan wewnętrzny był reprezentowany jako funkcja zbioru mikrooperacji i identyfikatorów. Zastosowanie metody łączy się z wykorzystaniem specjalnego konwertera kodu w układzie logicznym automatu stanów.

Słowa kluczowe: Skończony automat stanów, jednostka sterująca, układ logiczny, układy programowalne.

Design of Mealy FSM with transformation of Objects Codes in the System of Excitation Functions

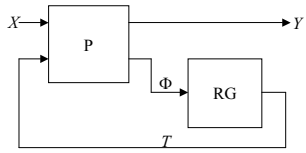
Abstract

The method of optimization of the cost of logical circuit of Mealy finite-state-machine is proposed. Method is based on the transformation of the objects codes in the system of excitation functions of flip-flops. The objects of Mealy FSM are internal states and sets of microoperations. The main idea is to express the states as some functions of the sets of microoperations and identifications. The application of the method is connected with applying of special code converter in the logic circuit of the logic circuit of FSM.

Keywords: FSM, control unit, Logic circuit, programmable logic device.

1. Wstęp

Jednostka sterująca każdego sytemu cyfrowego może zostać zaprojektowana jako skończony automat stanów [1]. Aktualnie programowalne układy cyfrowe są bardzo często stosowane do realizacji układu logicznego takiego automatu [7]. Układy tej klasy (CPLD, FPGA) są jednak wciąż kosztowne co prowadzi do wysokiego kosztu realizacji jednostki sterującej [4]. Tak więc jednym z głównych problemów projektowania takich układów jest znalezienie kompromisu pomiędzy ceną a wydajnością układu [3]. Jednopoziomowy układ automatu skończonego (rys. 1), zwany automatem-P, prowadzi do najszybszego układu logicznego jednak jego realizacja wiąże się również z najwyższymi kosztami [2].



Rys. 1. Schemat blokowy jednopoziomowego automatu Mealy'ego
Fig. 1. Structural diagram of single-level circuit of Mealy FSM

W rozwiązaniu tym kombinacyjny układ P implementuje systemy:

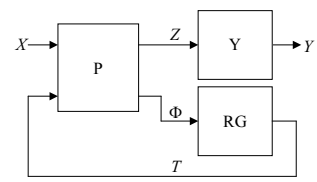
$$Y = Y(T, X), \tag{1}$$

$$\Phi = \Phi(T, X), \tag{2}$$

gdzie $Y = \{y_1, ..., y_N\}$ jest zbiorem mikrooperacji; $X = \{x_1, ..., x_L\}$ jest zbiorem warunków logicznych; $T = \{T_1, ..., T_R\}$ jest zbiorem zmiennych wewnętrznych kodujących stany $a_m \in A$, gdzie $A = \{a_1, ..., a_M\}$ jest zbiorem stanów automatu, $R =]\log_2 M[$; $\Phi = \{\Phi_1, \ldots, \Phi_R\}$ jest zbiorem funkcji wzbudzeń przerzutników. Rejestr RG służy do zapamiętania kodu $K(a_m)$ stanu $a_m \in A$.

W rozwiązaniu tym układ kombinacyjny P realizuje maksymalnie największą liczbę funkcji $t(\mathrm{P}) = N + R$ zależnych od warunków logicznych i zmiennych wewnętrznych. Do obniżenia kosztu automatu-P można zastosować różne metody dekompozycji funkcji logicznych [5].

Jednak jeżeli wydajność układu nie jest parametrem krytycznym koszt układu może zostać również zmniejszony poprzez zastosowanie dekompozycji strukturalnej układu logicznego [2]. Jednym z możliwych rozwiązań jest zastosowanie kodowania zbiorów mikrooperacji [2]. Prowadzi to do powstania dwupoziomowej struktury (rys. 2).



Rys. 2. Schemat blokowy dwupoziomowego automatu Mealy'ego
Fig. 2. Structural diagram of double-level circuit of Mealy FSM

W strukturze tej układ P realizuje system (2) i:

$$Z = Z(T, X), \tag{3}$$

gdzie $Z = \{z_1, ..., z_G\}$ jest zbiorem zmiennych służących do kodowania zbiorów mikrooperacji $Y_q \subseteq Y$. Wartość parametru G zależy od obranej metody kodowania. Układ Y realizuje system:

$$Y = Y(Z). \tag{4}$$

W przypadku maksymalnego kodowania zbiorów mikrooperacji [2] układ Y implementowany jest jako pamięć ROM i $G =]\log_2 Q[$, gdzie Q jest liczbą rożnych zbiorów mikrooperacji występujących w algorytmie sterowania (w sieci działań lub tablicy przejść-wyjść automatu).

W przypadku kodowania klas kompatybilnych mikrooperacji układ Y jest zbiorem I dekoderów $\mathrm{DC}_1, ..., \mathrm{DC}_I$, gdzie I jest liczbą klas kompatybilnych mikrooperacji [3] i $G = G_1 + ... + G_I$, gdzie G_i jest liczbą zmiennych potrzebnych do zakodowania mikrooperacji z i-tej klasy ($i = 1, ..., I$).

Oczywiste jest, że układ P realizuje $t(\mathrm{PY}) = G + R < t(\mathrm{P})$ funkcji. Jednocześnie badania [7] wykazują, że cena układu logicznego jest proporcjonalna do liczby funkcji realizowanych przez układ P. Tak więc zmniejszenie tej liczby prowadzi do obniżenia kosztów automatu.

2. Metoda z transformacją kodów

Proponowana metoda prowadzi do zmniejszenia liczby funkcji realizowanych przez układ kombinacyjny P. Bazuje ona na trans-

formacji kodów pewnych obiektów. Jako obiekt rozumiany jest wewnętrzny stan automatu i zbiór mikrooperacji.

Niech automat Mealy'ego będzie opisany tablicą przejść-wyjść automatu [1, 5] z kolumnami: a_m, $K(a_m)$, a_s, $K(a_s)$, X_h, Y_h, Φ_h, h, gdzie a_m jest początkowym stanem automatu, $a_m \in A$ gdzie; $K(a_m)$ jest kodem stanu; a_s jest następnym stanem automatu; $K(a_s)$ jest kodem stanu $a_s \in A$; X_h jest koniunkcją pewnych zmiennych $x_e \in X$ wymuszając przejście $\langle a_m , a_s \rangle$; Y_h jest sygnałem wyjściowym formowanym podczas przejęcia $\langle a_m , a_s \rangle$, $Y_q \subseteq Y$; Φ_h jest zbiorem funkcji wzbudzeń, które są równe 1 w celu przełączenia pamięć automatu z kodu $K(a_m)$ na kod $K(a_s)$, $\Phi_h \subseteq \Phi$; $h = 1, ..., H$ jest numerem wiersza tablicy przejść-wyjść. Na podstawie tej tablicy można wyprowadzić systemy (1) i (2).

Rozważmy przykład automatu S_1 (tab. 1) o następującej charakterystyce: $A = \{a_1, ..., a_5\}$, $X = \{x_1, ..., x_4\}$, $Y = \{y_1, ..., y_7\}$, $M = 5$, $R = 3$, $L = 4$, $N = 7$, $H = 12$.

Tab. 1. Tablica przejść-wyjść automatu S_1
Tab. 1. A DST table of FSM S_1

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Y_h	Φ_h	h
a_1	000	a_2	010	x_1	$y_1 y_2$	D_2	1
		a_3	011	$\sim x_1$	y_3	$D_2 D_3$	2
a_2	010	a_2	010	x_2	$y_1 y_2$	D_2	3
		a_3	011	$\sim x_2 x_3$	y_4	$D_2 D_3$	4
		a_4	100	$\sim x_2 \sim x_3$	$y_1 y_2$	D_1	5
a_3	011	a_4	100	x_1	$y_2 y_5$	D_1	6
		a_5	101	$\sim x_1$	y_6	$D_1 D_3$	7
a_4	100	a_5	101	1	$y_3 y_7$	$D_1 D_3$	8
a_5	101	a_5	010	$x_2 x_3$	$y_1 y_2$	D_2	9
		a_3	011	$x_2 \sim x_3$	y_3	$D_2 D_3$	10
		a_5	101	$\sim x_2 x_4$	$y_3 y_7$	$D_1 D_3$	11
		a_1	000	$\sim x_2 \sim x_4$	-	-	12

Rozważmy metodę maksymalnego kodowania zbiorów mikrooperacji. Zakodujmy każdy zbiór $Y_q \subseteq Y$ binarnym kodem $K(Y_q)$ z wykorzystaniem zmiennych ze zbioru $Z = \{z_1, ..., z_G\}$. W przypadku automatu S_1 występuje $Q = 7$ różnych zbiorów mikrooperacji $Y_1 = \varnothing$, $Y_2 = \{y_1, y_2\}$, $Y_3 = \{y_3\}$, $Y_4 = \{y_4\}$, $Y_5 = \{y_2, y_5\}$, $Y_6 = \{y_6\}$, $Y_7 = \{y_3, y_7\}$. W tym przypadku $G = 3$ i $Z = \{z_1, z_2, z_3\}$. Przyjmijmy następujące kodowanie: $K(Y_1) = 000$, ..., $K(Y_7) = 110$. Utwórzmy przekształconą tablicę przejść-wyjść dla automatu S_1 poprzez zastąpienie kolumny Y_h kolumną Z_h, zawierającą zmienne $z_g \in Z$, które są równe 1 w kodzie $K(Y_q)$ (tab. 2). Tablica ta jest podstawą do implementacji systemów (2) i (3) automatu-PY.

Tab. 2. Tablica przejść-wyjść automatu-PY S_1
Tab. 2. A DST table of PY FSM S_1

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Z_h	Φ_h	h
a_1	000	a_2	010	x_1	z_3	D_2	1
		a_3	011	$\sim x_1$	z_2	$D_2 D_3$	2
a_2	010	a_2	010	x_2	z_3	D_2	3
		a_3	011	$\sim x_2 x_3$	$z_2 z_3$	$D_2 D_3$	4
		a_4	100	$\sim x_2 \sim x_3$	z_3	D_1	5
a_3	011	a_4	100	x_1	z_1	D_1	6
		a_5	101	$\sim x_1$	$z_1 z_3$	$D_1 D_3$	7
a_4	100	a_5	101	1	$z_1 z_2$	$D_1 D_3$	8
a_5	101	a_5	010	$x_2 x_3$	z_3	D_2	9
		a_3	011	$x_2 \sim x_3$	z_2	$D_2 D_3$	10
		a_5	101	$\sim x_2 x_4$	$z_1 z_2$	$D_1 D_3$	11
		a_1	000	$\sim x_2 \sim x_4$	-	-	12

Nazwijmy stany automatu $a_m \in A$ obiektami pierwszego typu a zbiory mikrooperacji $Y_q \subseteq Y$ obiektami drugiego typu. Przedstawmy obiekty pierwszego typu jako funkcję obiektów drugiego typu.

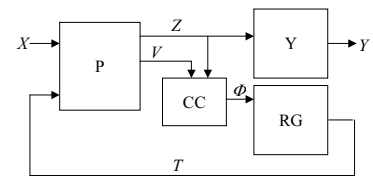
Niech stan przejścia będzie reprezentowany jako funkcja zbiorów mikrooperacji. Z analizy wynika, że nie musi występować zależność jeden do jednego pomiędzy stanami a zbiorami mikrooperacji. W prezentowanym przykładzie zbiór $Y_1 \subseteq Y$ koresponduje ze stanami a_2 (wiersze 1, 3 i 9) i a_4 (wiersz 5). Dlatego wymagane jest utworzenie etykiet do wyrażenia stanu przejścia (funkcji wzbudzeń przerzutników) jako funkcji zbiorów mikrooperacji. Niech $I = \{I_1, ..., I_K\}$ będzie zbiorem etykiet. W tym przypadku stan $a_m \in A$ może być reprezentowany jako funkcja:

$$A = A(Z, I). \tag{5}$$

Jeżeli etykiety $I_k \in I$ są kodowane binarnym kodem $K(I_k)$ pryz użyciu zmiennych $v_b \in V = \{v_1, ..., v_B\}$, gdzie $B =]\log_2 K[$, wtedy funkcje wzbudzeń przerzutników $\Phi_r \in \Phi$ mogą być reprezentowane jako:

$$\Phi = \Phi(Z, V). \tag{6}$$

Prowadzi to do powstania automatu Mealy'ego o strukturze $P_Y A$ (rys. 3).



Rys. 3. Schemat blokowy automatu-$P_Y A$
Fig. 3. Structural diagram of Mealy $P_Y A$ FSM

W strukturze tej układ kombinacyjny P implementuje system (3) i:

$$V = V(T, X). \tag{7}$$

Układ konwertera kodu CC realizuje system (6) a układ Y system (4).

Proponowane podejście prowadzi do zmniejszenia liczby wyjść kombinacyjnego układu P do $t(P_Y A) = G + B < t(PY)$. Zapewnia to zmniejszeni kosztów realizacji układu logicznego automatu w porównaniu ze strukturą PY. Oczywiście należy tutaj uwzględnić koszt realizacji dodatkowego układu konwertera kodu CC.

3. Metoda syntezy automatu-$P_Y A$

Niech automat Mealy'ego będzie reprezentowany przez tablice przejść-wyjść z maksymalnym kodowaniem zbiorów mikrooperacji (tab. 2). W takim przypadku proponowana metoda syntezy automatu-$P_Y A$ składa się z następujących kroków:

1. *Etykietowane stanów*. Niech $A(Y_q)$ będzie zbiorem stanów, takich że zbiór $Y_q \subseteq Y$ jest formowany podczas przejścia do stanu $a_m \in A(Y_q)$. Wystarczające jest więc zastosowanie $m_q = |A(Y_q)|$ etykiet do poetykietowania stanów $a_m \in A(Y_q)$. Tak więc wystarczające jest $K = \max(m_1, ..., m_Q)$ etykiet do identyfikacji dowolnego stanu $a_m \in A$. Etykiet te utworzą zbiór I. W prezentowanym przykładzie: $A(Y_1) = \{a_1\}$, $A(Y_2) = \{a_2, a_4\}$, $A(Y_3) = \{a_3\}$, $A(Y_4) = \{a_3\}$, $A(Y_5) = \{a_4\}$, $A(Y_6) = \{a_5\}$, $A(Y_7) = \{a_5\}$, $m_1 = m_3 = ... = m_7 = 1$, $m_2 = 2$, $K = 2$, $I = \{I_1, I_2\}$.
2. *Kodowanie etykiet*. Zakodujmy etykiety $I_k \in I$ binarnym kodem $K(I_k)$ na $B =]\log_2 K[$ bitach. Zmienne ze zbioru $V = \{v_1, ..., v_B\}$ posłużą do reprezentacji tego kodu. Utwórzmy pary $\alpha_q^s = \langle I_k, Y_q \rangle$ dla wszystkich elementów zbirów $A(Y_q) \subseteq A$ odpowiadające stanom $a_s \in A(Y_q)$. Jeżeli $m_q = 1$ wtedy $I_k = \varnothing$, co będzie odpowiadało $K(I_k) = *$ (wartość nieokreślona). W tym przypadku kod $C(a_s)^q$ odpowiada konkatencji:

$$C(a_s)^q = K(Y_q) * K(I_k). \tag{8}$$

W prezentowanym przykładzie: $B = 1$, $V = \{v_1\}$. Niech $K(I_1) = 0$, $K(I_2) = 1$. $\alpha_1^1 = \langle \varnothing, Y_1 \rangle$, $\alpha_2^2 = \langle I_1, Y_2 \rangle$, $\alpha_2^4 = \langle I_2, Y_2 \rangle$, $\alpha_3^3 = \langle \varnothing, Y_3 \rangle$, $\alpha_4^3 = \langle \varnothing, Y_4 \rangle$, $\alpha_5^4 = \langle \varnothing, Y_5 \rangle$, $\alpha_6^5 = \langle \varnothing, Y_6 \rangle$, $\alpha_7^5 = \langle \varnothing, Y_7 \rangle$. Kody $C(a_s)^q$ wyprowadzone w oparciu o (8)

przedstawione są w tabeli 3. Pierwsze trzy pozycje kolumny $C(a_s)^q$ zawierają kod $K(Y_q)$ natomiast ostatnia pozycja odpowiada kodowi $K(I_k)$.

Tab. 3. Kodowanie stanów automatu-P_YA S_1
Tab. 3. Encoding of states of P_YA FSM S_1

a_s	$C(a_s)^q$	α_q^s	h
a_1	000∗	α_1^1	1
a_2	0010	α_2^2	2
a_3	010∗	α_3^3	3
a_3	011∗	α_4^3	4
a_4	100∗	α_5^4	5
a_4	0011	α_2^4	6
a_5	101∗	α_6^5	7
a_5	110∗	α_7^5	8

3. *Utworzenie tablicy przejść-wyjść automatu-P_YA*. Tablica ta jest podstawą do utworzenia systemów (3) i (7).
Konstruuje się ją poprzez zastąpienie kolumn a_s, $K(a_s)$, Φ_h z tablicy przejść-wyjść automatu-PY kolumną V_h. Kolumna V_h zawiera zmienne $v_b \in V$, które są równe 1 w kodzie $K(I_k)$ odpowiadającemu parze α_q^s z h-tego wiersza tablicy.
Dla prezentowanego przykładu tablica przejść-wyjść automatu-P_YA przedstawiona jest w tabeli 4. Symbol '-' oznacza, że $v_1 = 0$ (w ogólnym przypadku żadna zmienna v_b nie przyjmuje wartości 1) a symbol '∗' oznacza, że zmienna v_1 przyjmuje wartość nieokreśloną.

Tab. 4. Tablica przejść-wyjść automatu-P_YA S_1
Tab. 4. A DST table of P_YA FSM S_1

a_m	$K(a_m)$	X_h	Z_h	V_h	h
a_1	000	x_1	z_3	-	1
		$\sim x_1$	z_2	∗	2
a_2	010	x_2	z_3	∗	3
		$\sim x_2 x_3$	$z_2 z_3$	∗	4
		$\sim x_2 \sim x_3$	z_3	v_1	5
a_3	011	x_1	z_1	∗	6
		$\sim x_1$	$z_1 z_3$	∗	7
a_4	100	1	$z_1 z_2$	∗	8
a_5	101	$x_2 x_3$	z_3	-	9
		$x_2 \sim x_3$	z_2	∗	10
		$\sim x_2 x_4$	$z_1 z_2$	∗	11
		$\sim x_2 \sim x_4$	-	∗	12

4. *Utworzenie tablicy konwertera kodu*. Tablica ta jest podstawą do utworzenia systemu (6). Posiada ona kolumny Y_q, $K(Y_q)$, I_k, $K(I_k)$, a_s, $K(a_s)$, Φ_h, h. Tabela ta posiada $H_0 = m_1 + ... + m_q$ wierszy, z których każdy odpowiada jednej parze α_q^s. Symbol '∗' w kolumnie $K(I_k)$ oznacza że odpowiednia zmienna v_b przyjmuje wartość nieokreśloną i w tym przypadku wartość zmiennych D_r w kolumnie Φ_h dla tego samego wiersza są nie zależne od tej zmiennej v_b.
Dla prezentowanego przykładu tablica konwertera kodu przedstawiona jest w tabeli 5.

Tab. 5. Tablica konwertera kodu automatu-P_YA S_1
Tab. 5. A table of code converter of P_YA FSM S_1

Y_q	$K(Y_q)$	I_k	$K(I_k)$	a_s	$K(a_s)$	Φ_h	h
Y_1	000	-	∗	a_1	000	-	1
Y_2	001	I_1	0	a_2	010	D_2	2
		I_2	1	a_4	100	D_1	3
Y_3	010	-	∗	a_3	011	$D_2 D_3$	4
Y_4	011	-	∗	a_3	011	$D_2 D_3$	5
Y_5	100	-	∗	a_4	100	D_1	6
Y_6	101	-	∗	a_5	101	$D_1 D_3$	7
Y_7	110	-	∗	a_5	101	$D_1 D_3$	8

5. *Utworzenie tablicy mikrooperacji*. Tablica ta jest podstawą do utworzenia systemu (4). Posiada ona kolumny Y_q, $K(Y_q)$, $y_1, ..., y_N$, q. Każdy wiersz tej tablicy odpowiada jednemu zbiorowi mikrooperacji $Y_q \subseteq Y$. Opisuje ona pamięć ROM o wejściu adresowym Z i wyjściu danych Y.
Dla prezentowanego przykładu tablica mikrooperacji przedstawiona jest w tabeli 6.

Tab. 6. Tablica mikrooperacji automatu-P_YA S_1
Tab. 6. A table of microoperations of P_YA FSM S_1

Y_q	$K(Y_q)$	y_1	y_2	y_3	y_4	y_5	y_6	y_7	q
Y_1	000	0	0	0	0	0	0	0	1
Y_2	001	1	1	0	0	0	0	0	2
Y_3	010	0	0	1	0	0	0	0	3
Y_4	011	0	0	0	1	0	0	0	4
Y_5	100	0	1	0	0	1	0	0	5
Y_6	101	0	0	0	0	0	1	0	6
Y_7	110	0	0	1	0	0	0	1	7

6. *Zaprojektowanie układu logicznego automatu*. Układ taki projektuje się wykorzystując systemy (3) i (7) do realizacji układu kombinacyjnego P, system (6) do realizacji konwertera kodu CC i system (4) do realizacji układu Y.
Układy P i CC mogą zostać zaimplementowane z wykorzystaniem struktur PLD natomiast układ Y, ponieważ zawiera ponad 50% możliwych termów [3], z wykorzystaniem pamięci PROM.
W przypadku zastosowania układu FPGA struktura P_YA daje efektywne wykorzystanie zasobów sprzętowych jeżeli układy P i CC implementowane są z wykorzystaniem tablic LUT natomiast układ Y z wykorzystaniem osadzonych bloków pamięci [6].

4. Podsumowanie

Proponowana metoda implementacji skończonego automatu stanów z wyjściami typu Mealy'ego z transformacją kodów obiektów w systemie funkcji wzbudzeń pozwala na zmniejszenie kosztów realizacji układu logicznego jednostki sterującej w porównaniu z klasycznymi rozwiązaniami. Przeprowadzone badania analityczne wykazały, że proponowana metoda obniża koszt do 15%.

Zysk ten może zostać zwiększony poprzez zastosowanie różnych metod funkcjonalnej dekompozycji funkcji boolowskich [5].

5. Literatura

[1] Baranov S.: Logic Synthesis for Control Automata, Kluwer Academic Publishers, 1994
[2] Barkalov A. A.: Development of Formal Methods of Structural Synthesis of Compositional Automata, DonTSY, Donieck, 1994 (po rosyjsku).
[3] Barkalov A. A.: Synthesis of Operational Units, DonNTU, Donieck, 2004 (po rosyjsku).
[4] Grushnitsky R., Mursaev A., Ugrjumov E.: Development of Systems on Chips with Programmable Logic, SPb: BHV, Petersburg, 2002 (po rosyjsku).
[5] Łuba T. (ed.): Synteza układów cyfrowych, WKŁ, Warszawa 2003.
[6] Salcic Z.: VHDL and FPLDs in Digital Systems Design, Prototyping and Customization, Kluwer Academic Publishers, 1998.
[7] Solovjev V.V.: Design of Digital Systems Using the Programmable Logic Integrate Circuits, Hot Line-Telecom, Moskwa, 2001 (po rosyjsku).

Artykuł recenzowany