ZESZYTY
NAUKOWE
POLITECHNIKI
ŚLĄSKIEJ

P. 3341 /77

AUTOMATYKA

Z. 39
GLIWICE
1977

P. 3341 / 77

BIBLIOTEKA GŁÓWNA

POLITECHNIKA ŚLĄSKA

ZESZYTY NAUKOWE

Nr 526

FERDYNAND WAGNER

PROJEKTOWANIE KRÓTKICH REJESTRÓW LICZĄCYCH

PL ISSN 0434-0760

GLIWICE 1977

OPINIODAWCY

Prof. dr inż. Jerzy Siwiński

Doc. dr hab. inż. Zygmunt Szwaja

Doc. dr hab. inż. Wiesław Traczyk

REDAKTOR NACZELNY WYDAWNICTW UCZELNIANYCH
POLITECHNIKI ŚLĄSKIEJ

Jan Bandrowski

REDAKTOR DZIAŁU

Anna Skrzywan-Kosek

SEKRETARZ REDAKCJI

Jan Znamirowski

Wydano za zgodą
Rektora Politechniki Śląskiej

Dział Wydawnictw Politechniki Śląskiej
Gliwice, ul. Kujawska 2

Nakł. 150+65 Ark wyd. 3,07 Ark. druk. 3 Papier offsetowy kl. III. 70x100, 70 g
Oddano do druku 27 6.1977 Podpis. do druku 17. 9. 1977 Druk ukończ. we wrześniu 1977
Zam. 948/77 Cena zł 8,—

Skład, fotokopie, druk i oprawę
wykonano w Zakładzie Graficznym Politechniki Śląskiej w Gliwicach

P64/78

SPIS TREŚCI

Str.

1. POSTAWIENIE PROBLEMU 5

2. OPIS DZIAŁANIA REJESTRU LICZĄCEGO 7

3. LICZBA s-STANOWYCH SEKWENCJI BINARNYCH 11

3.1. Grupa generująca i generator sekwencji binarnej 11
3.2. Liczba grup generujących 12
3.3. Algorytm poszukiwania generatorów sekwencji 13
3.4. Ocena algorytmu i wyników 19

4. CIĄGI KODOWE 20

4.1. Obliczanie ciągów kodowych, realizujących daną sekwencję binarną metodą dopisywania 20
4.2. Obliczanie ciągów kodowych, realizujących daną sekwencję binarną metodą "bazy" 20
4.3. Obliczanie ciągów kodowych generowanych przez rejestr o długości n 22
4.4. Ciągi kodowe 25

5. PROJEKTOWANIE KRÓTKICH REJESTRÓW LICZĄCYCH 29

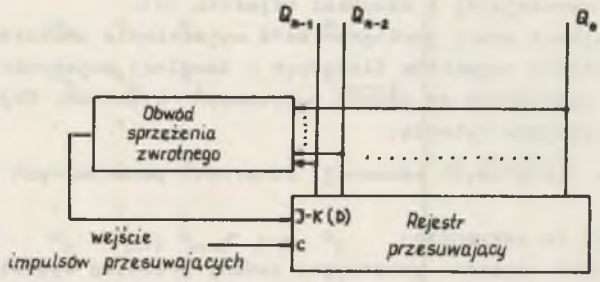
5.1. Projektowanie rejestru liczącego. Samokorekcja 29
5.2. Rejestry liczące jako liczniki 32
5.3. Generowanie przebiegów czasowych. Generatory słów 34
5.4. Uwagi dotyczące konstrukcji licznika pierścieniowego 38
5.5. Generatory sekwencji pseudoprzypadkowych 40

6. PODSUMOWANIE 42

LITERATURA 44

1. POSTAWIENIE PROBLEMU

Rejestr liczący to rejestr przesuwający z obwodem sprzężenia zwrotnego przedstawiony na rys. 1.1. Sygnał wyjściowy obwodu sprzężenia zwrotnego, będący funkcją wyjść Q_{n-1}, Q_{n-2} ... Q_0 rejestru, stanowi wejście reje-



Rys. 1.1. Schemat blokowy rejestru liczącego

stru. Wejście rejestru może być wejściem informacyjnym (programującym) J-K lub D, w zależności od typu przerzutników zastosowanych w rejestrze przesuwającym.

Rejestr liczący nazywa się również rejestrem przesuwającym ze sprzężeniem zwrotnym.

Problem sekwencji generowanych przez rejestry liczące jest znany i dyskutowany w literaturze technicznej od wielu lat. Chyba najbardziej wyczerpująco przedstawiono to zagadnienie w [3], która to monografia - mimo że upłynęło 10 lat od jej wydania - ciągle zachowuje charakter podstawowego podręcznika z tego zakresu.

Charakterystycznym tematem dla rejestrów liczących, rozwiązywanym w szeregu publikacji, jest dobór sprzężenia zwrotnego, które pozwala na otrzymanie rejestru generującego ciąg o określonych właściwościach.

Typem najczęściej rozpatrywanym w tym temacie rejestru liczącego jest tzw. rejestr liniowy, co wiąże się z pseudoprzypadkowym charakterem generowanej przez niego sekwencji zer i jedynek na wyjściu [1, 3, 7, 11, 12]. Pseudoprzypadkowy charakter mają nie tylko sekwencje zer i jedynek na dowolnym wyjściu rejestru liniowego, ale również sekwencje stanów rozumianych jako liczby dwójkowe utworzone z sygnałów wyjściowych rejestru. Rejestry liczące jako źródło sekwencji liczb pseudoprzypadkowych są również często analizowane w literaturze, np. w [3, 4, 5, 8, 10, 11].

Innym typowym tematem dyskutowanym przy rozpatrywaniu rejestrów liczących jest ich zastosowanie do realizacji kodów wykrywających i korygujących błędy binarne, jakie mogą występować przy przekazywaniu informacji cyfrowych.

Rozpatrując możliwość wykorzystania rejestrów liczących do realizacji zadań wykonywanych przez liczniki (liczenie impulsów, generowanie żądanych sekwencji sygnałów) mówi się zwykle o konstrukcji licznika pierścieniowego, czy licznika pseudopierścieniowego (Johnsona), zwracając uwagę na problem samokorekcji układu. Ponadto wiadomo [3], że za pomocą rejestru przesuwającego, drogą doboru odpowiedniego obwodu sprzężenia zwrotnego, można otrzymać rejestr liczący o dowolnej liczbie stanów, nie przekraczającej wartości 2^n, wynikającej z długości rejestru (n).

Celem niniejszej pracy jest gruntowne wyjaśnienie problemów występujących przy tworzeniu rejestrów liczących o dowolnej pojemności (nie większej od 2^n), budowanych za pomocą n-bitowych rejestrów. Objęło to odpowiedzi na następujące pytania:

- ile istnieje s-stanowych sekwencji binarnych generowanych przez n-bitowy rejestr;
- jak generować te sekwencje;
- jak zrealizować rejestr, generujący zadany przebieg sygnału w czasie;
- jak uwzględnić w procesie projektowania samokorekcję sekwencji generowanej przez rejestr.

Przedstawiony w pracy sposób podejścia pozwala odpowiedzieć na powyższe pytania. Ponadto uzyskano kilka interesujących wyników dotyczących działania i budowy rejestrów liczących.

Ograniczenia stosowania przedstawionych w pracy metod związane są z wydłużeniem się czasu potrzebnego do obliczeń przy rosnącej długości rejestrów i liczbie stanów sekwencji. Stąd ograniczenie tematu do krótkich rejestrów zawarte w tytule. Ocena długości rejestrów i sekwencji, które tymi metodami można analizować, jest zawarta w pracy.

2. OPIS DZIAŁANIA REJESTRU LICZĄCEGO

Wprowadza się [14] t a b l i c ę s t a n ó w rejestru liczącego, która opisuje jego działanie. Tablica ta wygląda następująco:

$$\left.\begin{array}{ccccc} a_1 & a_s & a_{s-1} & \cdots & a_{s-n+2} \\ a_2 & a_1 & a_s & \cdots & a_{s-n+3} \\ a_3 & a_2 & a_1 & \cdots & a_{s-n+4} \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ a_n & a_{n-1} & a_{n-2} & \cdots & a_1 \\ a_{n+1} & a_n & a_{n-1} & \cdots & a_2 \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ a_s & a_{s-1} & a_{s-2} & \cdots & a_{s-n+1} \end{array}\right\} \text{s wierszy} \qquad (2.1)$$

$$\underbrace{\qquad\qquad\qquad\qquad}_{\text{n kolumn}}$$

W tablicy tej symbole a_i oznaczają stany wyjść przerzutników rejestru, a więc

$$a_i = 0 \quad \text{lub} \quad a_i = 1.$$

Kolumny tablicy przedstawiają ciągi 0 i 1 na wyjściach rejestru. Pierwsza kolumna z lewej odpowiada wyjściu Q_{n-1}, druga - wyjściu Q_{n-2}, ostatnia (pierwsza z prawej) - wyjściu Q_o. Tablica zawiera więc n kolumn, gdzie n jest liczbą wyjść (przerzutników) rejestru.

Wiersze tablicy określają stany x_k rejestru liczącego; liczba ich wynosi s. S t a n r e j e s t r u określa się wartością dziesiętną liczby dwójkowej utworzonej z wyrazów k-tego wiersza tablicy, tj.:

$$x_k = (a_k a_{k-1} \ldots a_o)_2$$

Ciąg stanów rejestru liczącego nazywa się również c i ą g i e m k o d o w y m rejestru liczącego.

Tablica stanów rejestru liczącego jest t a b l i c ą p r a w i d ł o w ą, gdy

$$x_i \neq x_j \quad \text{dla} \quad i \neq j, \tag{2.2}$$

a odpowiadający jej ciąg jest c i ą g i e m r z e c z y w i s t y m (można go zrealizować za pomocą rejestru przesuwającego).

Tablica stanów rejestru liczącego jest ważna dla

$$s \geqslant n \tag{2.3}$$

Dla rejestru, w którym $s = n$, tablica ta wygląda następująco:

$$\begin{matrix} a_1 & a_n & \dots & a_2 \\ a_2 & a_1 & \dots & a_3 \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ a_n & a_{n-1} & \dots & a_1 \end{matrix} \tag{2.4}$$

Dopisanie kolejnej kolumny oznaczałoby powtórzenie pierwszej. Z technicznego punktu widzenia oznaczałoby to wydłużenie rejestru o jeden przerzutnik, którego stan nie miałby już wpływu na kod liczenia. Przerzutnik ten powielałby przebieg wyjściowy przerzutnika wejściowego rejestru. Ciągi kodowe, dla których $s < n$, nazywane będą c i ą g a m i p o z o r n y m i.

Również dla $s \geqslant n$ tablica może definiować ciąg pozorny. Sprawdzenie czy dany ciąg jest ciągiem pozornym polega na zmniejszeniu tablicy o jedną skrajną kolumnę; jeżeli tablica pozostała tablicą prawidłową, oznacza to, że ciąg był pozorny.

Ciąg zdefiniowany tablicą prawidłową, której nie można już zmniejszyć, nazywa się c i ą g i e m z a s a d n i c z y m.

Ciągi pozorne wyklucza się z rozważań, gdyż ich realizacja techniczna jest nieciekawa i polega na wydłużeniu rejestru realizującego odpowiadający im ciąg zasadniczy bez zmiany struktury obwodu sprzężenia zwrotnego.

Związek pomiędzy stanami x_k rejestru a stanami wyjść rejestru można w oparciu o tablicę (2.1) ująć w następującej formie układu równań:

$$
\begin{cases}
2^{n-1}a_1 + 2^{n-2}a_s + 2^{n-3}a_{s-1} + \ldots + 2^0 a_{s-n+2} = x_1 \\
2^{n-1}a_2 + 2^{n-2}a_1 + 2^{n-3}a_s + \ldots + 2^0 a_{s-n+3} = x_2 \\
2^{n+1}a_3 + 2^{n-2}a_2 + 2^{n-3}a_1 + \ldots + 2^0 a_{s-n+4} = x_3 \\
\vdots \\
2^{n-1}a_n + 2^{n-2}a_{n-1} + 2^{n-3}a_{n-2} + \ldots + 2^0 a_1 = x_n \\
2^{n-1}a_{n+1} + 2^{n-2}a_n + 2^{n-3}a_{n-1} + \ldots + 2^0 a_2 = x_{n+1} \\
\vdots \\
2^{n-1}a_s + 2^{n-2}a_{s-1} + 2^{n-3}a_{s-2} + \ldots + 2^0 a_{s-n+1} = x_s
\end{cases}
\tag{2.5}
$$

Ten układ równań - po dokonaniu zmian w kolejności zmiennych - można zapisać w postaci macierzowej:

$$B \cdot A = X, \tag{2.6}$$

gdzie B jest macierzą kwadratową o stopniu s

$$
B = \begin{bmatrix}
2^{n-1} & 0 & 0 & \ldots & 2^{n-2} \\
2^{n-2} & 2^{n-1} & 0 & \ldots & 2^{n-3} \\
2^{n-3} & 2^{n-2} & 2^{n-1} & \ldots & 2^{n-4} \\
\cdot & \cdot & \cdot & & \cdot \\
\cdot & \cdot & \cdot & & \cdot \\
\cdot & \cdot & \cdot & & \cdot \\
2^0 & 2^1 & 2^2 & \ldots & 0 \\
0 & 2^0 & 2^1 & \ldots & 0 \\
\cdot & \cdot & \cdot & & \cdot \\
\cdot & \cdot & \cdot & & \cdot \\
\cdot & \cdot & \cdot & & \cdot \\
0 & 0 & 0 & \ldots & 2^{n-1}
\end{bmatrix}
\tag{2.7}
$$

natomiast A oraz X to macierze kolumnowe

$$A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \cdot \\ \cdot \\ \cdot \\ a_n \\ a_{n+1} \\ \cdot \\ \cdot \\ \cdot \\ a_s \end{bmatrix} \quad (2.8) \qquad X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ \cdot \\ \cdot \\ x_n \\ x_{n+1} \\ \cdot \\ \cdot \\ \cdot \\ x_s \end{bmatrix} \quad (2.9)$$

Elementy a_i macierzy A przyjmują wartość 0 lub 1.
W równaniu macierzowym (2.6)

- macierz B jest związana ze schematem rejestru przesuwającego; jej wymiary są określone długością s ciągu kodowego, a elementy - liczbą n stopni (wyjść) rejestru,
- wektor A określa ciąg zerojedynkowy na wyjściu Q_{n-1} rejestru,
- wektor X przedstawia ciąg kodowy rejestru liczącego.

Wprowadzono więc dwie formy opisu działania rejestru liczącego. Pierwszą formą jest tablica stanów rejestru liczącego (2.1), drugą - równanie macierzowe (2.6). Oba sposoby będą w dalszym ciągu wykorzystywane.

3. LICZBA S-STANOWYCH SEKWENCJI BINARNYCH

3.1. Grupa generująca i generator sekwencji binarnej

Sekwencja binarna składa się z 0 i 1. Liczba wyrazów tego ciągu wynosi s. Przyjmuje się, że sekwencja powstająca z sekwencji oryginalnej przez zmianę 0 na 1 i 1 na 0 jest równoważna oryginałowi.

Sekwencja binarna przedstawia liczbę dwójkową. Ze względu na cykliczność sekwencji istnieje wiele liczb dwójkowych określających daną sekwencję. Grupę liczb określających s-stanową sekwencję binarną, obejmującą 2 s lub w szczególnym przypadku s liczb, nazywać będziemy g r u p ą g e n e r u j ą c ą sekwencji binarnej.

Liczbę o najmniejszej wartości w grupie generującej oznaczamy przez G i nazywamy g e n e r a t o r e m s e k w e n c j i. Generator sekwencji jest liczbą, która mnożono przez 2 modulo $(2^s - 1)$ przechodzi w siebie po s operacjach mnożenia.

Znając generator sekwencji określenie pozostałych liczb, wchodzących w skład grupy generującej, polega na cyklicznym przesuwaniu bitów liczby. Przyjmujemy, że przesuwanie następuje w lewo. Uwzględniając cykliczność przesuwania oznacza to, że grupa generująca obejmuje liczby równe

$$(2^n G) \bmod (2^s-1), \tag{3.1}$$

gdzie $n = 0,1,\ldots s-1$

oraz uwzględniając sekwencje równoważne liczby

$$2^s-1-(2^n G) \bmod (2^s-1), \tag{3.2}$$

gdzie $n = 0,1 \ldots s-1$

Jeżeli wszystkie liczby otrzymane wg zależności (3.1) i (3.2) są różne, otrzymuje się grupę generującą, liczącą 2s liczb. W szczególnym przypadku może się zdarzyć, że

$$(2^n G) \bmod (2^s-1) = 2^s-1-(2^m G) \bmod (2^s-1), \tag{3.3}$$

tzn. że sekwencja równoważna pokrywa się z oryginałem. Wtedy grupa generująca obejmuje jedynie s liczb.

U w a g a:

Najmłodszy bit generatora G sekwencji równa się 1. Gdyby najmłodszy bit generatora był równy 0, to istniałaby w grupie generującej liczba od niego mniejsza. Otrzymywałoby się ją przez przesunięcie generatora o jedną pozycję w prawo.

W n i o s e k:

Wartość dziesiętna generatora sekwencji jest liczbą nieparzystą.

3.2. Liczba grup generujących

Spróbowano oszacować liczbę różnych s-stanowych sekwencji binarnych. Wykorzystano w tym celu wprowadzone w poprzednim punkcie pojęcie grup generujących.

Istnieje 2^s s-bitowych liczb dwójkowych. Z punktu widzenia dyskutowanych sekwencji binarnych wiele z nich jest sobie równoważnych. Jedna sekwencja odpowiada całej grupie liczb, które zostały nazwane grupą generującą.

Pewne liczby z całego zbioru 2^s liczb nie wchodzą w skład żadnej grupy generującej. Są to:

a) liczby 0 i 2^s-1,

b) tzw. liczby symetryczne, które określają sekwencje binarne, będące złożeniem sekwencji o mniejszej wartości s, np. sekwencja 010101 stanowi złożenie trzech sekwencji 01.

Ponieważ ilość liczb symetrycznych wydaje się być trudna do ujęcia zależnością matematyczną, można zaproponować następujący wzór, szacujący liczbę k grup generujących o s stanach:

$$k \cong \text{int}\,\frac{2^s-2}{2s} \qquad (3.4)$$

We wzorze tym zmniejszono ogólną ilość liczb 2^s o dwa (liczby 0 i 2^s-1) oraz przyjęto, że grupy generujące obejmują 2s liczb. Dokładność tego oszacowania zależy więc od tego, ile liczb nie wchodzi w skład grup generujących oraz ile istnieje grup generujących, które obejmują tylko s liczb.

Tablica 3.1 przedstawia liczby grup generujących dla $s \leqslant 20$ obliczonych wg powyższego wzoru oraz rzeczywistą liczbę grup generujących. Rzeczywistą liczbę grup generujących obliczono wg algorytmu poszukiwania generatorów sekwencji omówionego w punkcie 3.3.

Tablica 3.1

Liczba grup generujących

k_s	$k_{rzeczywiste}$	$k_{oszacowane}$
2	1	0
3	1	1
4	2	1
5	3	3
6	5	5
7	9	9
8	16	15
9	28	28
10	51	51
11	93	93
12	170	170
13	315	315
14	585	585
15	1091	1092
16	2048	2047
17	3855	3855
18	7280	7281
19	13797	13797
20	26214	26214

3.3. Algorytm poszukiwania generatorów sekwencji

Koncepcja algorytmu poszukiwania generatorów sekwencji polega na sprawdzaniu, czy w wyniku cyklicznego przesunięcia postaci dwójkowej liczby nie otrzyma się liczby mniejszej. Jeżeli sprawdzanie rozpoczyna się od liczby 1, to otrzyma się wszystkie generatory. W trakcie sprawdzania przesuwa się najpierw liczbę a później jej uzupełnienie do 1, co wynika z przyjęcia założenia o równoważności tych dwóch postaci liczby.

Jeżeli oznaczyć przez:

s - długość ciągu,
B - aktualnie sprawdzaną liczbę

i wprowadzić niżej określony parametr bieżący A, to można opisać I etap algorytmu, obejmujący przesuwanie liczby w następujący sposób:

a) wprowadzić liczbę stanów s sekwencji,
b) przyjąć pierwszą liczbę B = 1,

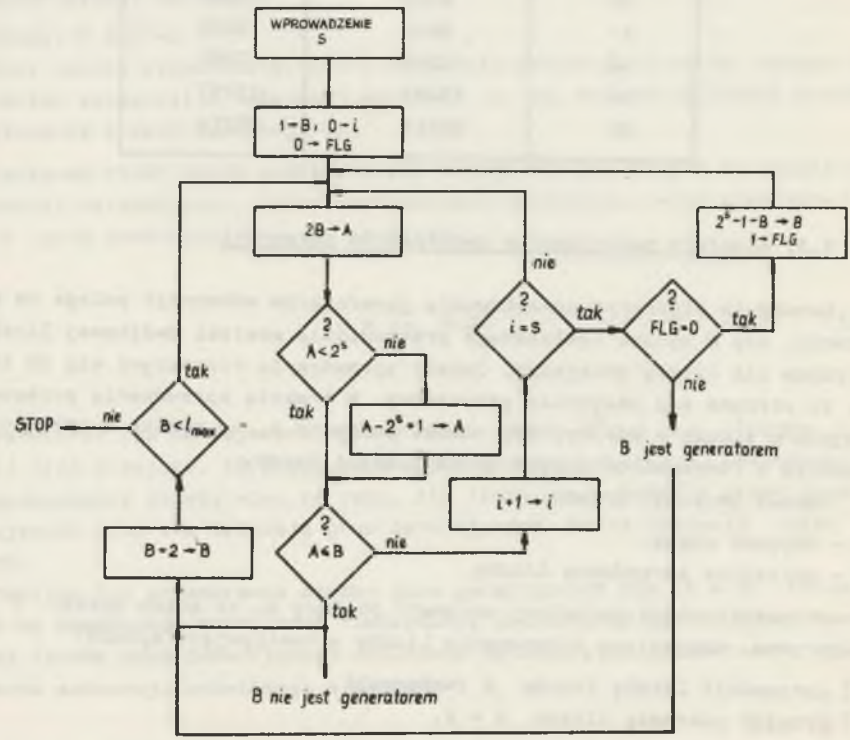
c) pomnożyć B przez 2, co odpowiada przesunięciu postaci dwójkowej B w lewo o jeden bit,

d) porównać wartość wyniku A = 2 B z wartością 2^s, co odpowiada sprawdzeniu, czy nie powinno nastąpić cykliczne przepisanie 1 z najstarszego bitu postaci dwójkowej liczby B na bit najmłodszy. Jeżeli $A < 2^s$, należy przejść do kroku e. Jeżeli $A \geqslant 2^s$, należy zmienić wartość A na: $A-2^s+1$ i przejść do kroku e,

e) porównać A z B.
Jeżeli $A \leqslant B$, zakończyć proces sprawdzania, wnioskując, że liczba B nie stanowi generatora sekwencji i przejść do kroku f (równość eliminuje liczby symetryczne). Jeżeli $A > B$, kontynuować obliczenia, wracając do kroku c,

f) powiększyć B o 2 i wrócić do kroku c.

Liczba pętli programowych (przejście z e do c), odpowiadająca liczbie przesunięć cyklicznych postaci dwójkowej liczby B, powinna być równa liczbie stanów s. Po pierwszym etapie sprawdzenia następuje obliczenie uzupełnienie do 1 liczby B i ponowne sprawdzenie wg algorytmu analogicznego do wyżej opisanego. Jeżeli również w II etapie sprawdzania program wykona s pętli, to po ostatnim wyniku $A < 2^s$ w kroku d następuje wydruk liczby B jako generatora sekwencji.

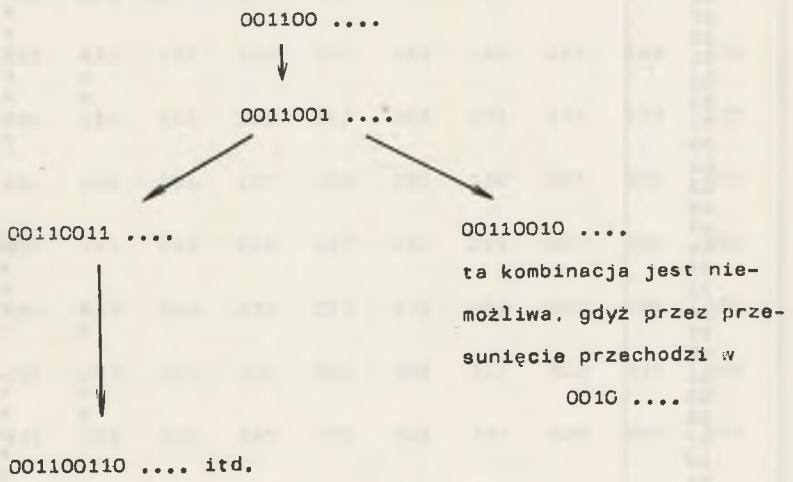


Rys. 3.1. Schemat blokowy programu obliczania generatorów sekwencji

Rys. 3.1 przedstawia schemat blokowy programu realizującego wyżej przedstawiony algorytm. Poza oznaczeniami użytymi w tekście na rys. 3.1 wprowadzono następujące symbole: i - który określa liczbę pętli programowych, FLG - który jest warunkiem pozwalającym odróżnić oba etapy obliczeń, lmax - będący maksymalnym generatorem, określonym poniżej.

Nie ma potrzeby sprawdzania wszystkich 2^s-1 liczb. Po pierwsze, zgodnie z wnioskiem z punktu 3.1, sprawdza się tylko liczby nieparzyste. Po drugie, z przyjętej równoważności liczb i ich uzupełnień do 1, wynika, że wszystkie generatory są zawarte w pierwszej połowie całego zbioru liczb 2^s-1, tj. wystarcza sprawdzenie liczb mniejszych od 2^{s-1}.

Można w dalszym ciągu zmniejszyć zakres sprawdzanych liczb drogą następującego oszacowania. Jeśli przyjąć, że największa sprawdzana liczba zaczyna się kombinacją 01 ..., to byłaby nią liczba 010101 ... 01, która jest liczbą symetryczną. Wobec tego ta największa liczba musi posiadać na początku kombinację 001 ... . Można jeszcze spróbować ocenić wartość następnego bitu. Gdyby tym bitem było 1, to początek liczby musiałby wyglądać następująco: 00110 ... . Kolejny, szósty bit nie może być 1, gdyż układ 001101 ... można drogą inwersji i przesunięć sprowadzić do 0010 ..., co daje liczbę mniejszą. Szósty bit nie może być również 0, gdyż układ 001100 prowadzi w dalszym ciągu do następujących liczb:



Kontynuując to rozumowanie otrzymuje się dla parzystej ilości bitów liczbę symetryczną 0011001100 ... 110011, a dla nieparzystej liczbę 0011001100 ... 11001, która drogą przesunięcia przechodzi w 0010 .... Wynika z tego, że kombinacja 0011 ... na początku jest niemożliwa, a więc musi to być kombinacja 0010 .... Dla dużych s stosunek maksymalnego ge-

Tablica 3.2

Generatory sekwencji mniejsze od 100

s liczba	1	2	3	4	5	6	7	8	9	10	11	12
1	x	x	x	x	x	x	x	x	x	x	x	x
3				x	x	x	x	x	x	x	x	x
5					x	x	x	x	x	x	x	x
7						x	x	x	x	x	x	x
9							x	x	x	x	x	x
11						x	x	x	x	x	x	x
13							x	x	x	x	x	x
15								x	x	x	x	x
17									x	x	x	x
19							x	x	x	x	x	x
21							x	x	x	x	x	x
23								x	x	x	x	x
25								x	x	x	x	x
27								x	x	x	x	x
29									x	x	x	x
31										x	x	x
33											x	x
35									x	x	x	x
37								x	x	x	x	x
39									x	x	x	x
41									x	x	x	x
43								x	x	x	x	x
45								x	x	x	x	x
47										x	x	x
49										x	x	x
51									x	x	x	x
53									x	x	x	x
55									x	x	x	x
57										x	x	x
59										x	x	x
61											x	x
63												x
65												
67											x	x
69										x	x	x
71										x	x	x
73										x	x	x
75									x	x	x	x
77									x	x	x	x
79											x	x
81											x	x
83									x	x	x	x
85									x	x	x	x
87										x	x	x
89										x	x	x
91										x	x	x
93										x	x	x
95												x
97												x
99											x	x

cd. w tablicy 3.3

Tablica 3.3

Generatory sekwencji większe od 100

s	Generatory										
10	101	103	105	107	109	147	149	155	171	173	179
11	101	103	105	107	109	111	113	115	117	119	137
	139	141	147	149	151	153	155	157	163	165	167
	169	171	173	179	181	183	185	199	201	203	205
	211	213	215	217	219	293	299	301	307	309	331
	333	339	341								
12	101	103	105	107	109	111	113	115	117	119	121
	123	133	135	137	139	141	143	145	147	149	151
	153	155	157	163	165	167	169	171	173	175	177
	179	181	183	185	187	189	197	199	201	203	205
	207	209	211	213	215	217	219	221	227	229	231
	233	235	237	239	275	277	279	281	283	285	291
	293	295	297	299	301	307	309	311	313	315	327
	329	331	333	339	341	343	345	347	349	355	357
	359	361	363	365	371	397	403	405	407	409	411
	421	423	425	427	429	435	437	439	441	587	589
	595	597	603	613	619	661	683	685	691	693	717

Tablica 3.4

Maksymalne generatory sekwencji s-stanowych

s	Maksymalny generator lmax		$\frac{lmax}{2^s}$
	dziesiętnie	dwojkowo	
1	1	1	0,5
2	1	0 1	0.25
3	1	0 0 1	0,125
4	3	0 0 1 1	0,188
5	5	0 0 1 0 1	0,156
6	11	0 0 1 0 1 1	0,172
7	21	0 0 1 0 1 0 1	0.164
8	45	0 0 1 0 1 1 0 1	0,176
9	85	0 0 1 0 1 0 1 0 1	0,166
10	179	0 0 1 0 1 1 0 0 1 1	0,175
11	341	0 0 1 0 1 0 1 0 1 0 1	0,167
12	717	0 0 1 0 1 1 0 0 1 1 0 1	0,175
13	1365	0 0 1 0 1 0 1 0 1 1 0 1 0 1	0,1666
14	2867	0 0 1 0 1 1 0 0 1 1 0 0 1 1	0,175
15	5461	0 0 1 0 1 0 1 0 1 0 1 0 1 0 1	0,1666
16	11475	0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 1	0,175

neratora do maksymalnej wartości s-bitowej liczby dwójkowej jest ograniczony z góry następującą wartością:

$$\lim_{s\to\infty} \frac{2^{s-3} + 2^{s-4} - 1}{2^s - 1} = \lim_{s\to\infty} \frac{1 + \frac{1}{2} + \frac{1}{2^{s-3}}}{2^3 - \frac{1}{2^{s-3}}} \to \frac{3}{16} = 0{,}1875$$

Dzięki temu oszacowaniu wartości maksymalnej generatora sekwencji można zmniejszyć zakres rozpatrywanych liczb do 18,75% wszystkich liczb s-bitowych.

Przy tych założeniach przeprowadzono obliczenia generatorów sekwencji dla $s \leqslant 12$, które zostały zebrane w tablicach 3.2 i 3.3.

Dla potwierdzenia jakości dokonanego wyżej oszacowania obliczono również maksymalne generatory sewkencji dla $s \leqslant 16$, co zostało ujęte w tablicy 3.4. Obliczony w tej tablicy stosunek $l_{max}/2^s-1$ potwierdza, że przyjęte w celu skrócenia czasu obliczeń oszacowanie maksymalnej wartości jest wystarczające.

3.4. Ocena algorytmu i wyników

Przeprowadzone w rozdziale rozważania pozwalają na obliczenie liczby sekwencji binarnych, które można utworzyć z s stanów oraz wszystkich generatorów określających te sekwencje.

Można wysunąć przypuszczenie, że wzór 3.1, szacujący ilość sekwencji binarnych, jest prawdziwy dla dowolnego s.

Zaprezentowany algorytm poszukiwania generatorów sekwencji został wykorzystany do ułożenia programów na kalkulator HP 9820, wynikiem obliczeń są tablice 3.1, 3.2 i 3.3. Programy na HP 9820 można eksploatować dla $s \leqslant 16$; przy przejściu na komputer ODRA 1325 - dla $s \leqslant 23$. Ograniczeniem dla HP 9820 jest zbyt długi czas obliczeń, a dla ODRY 1325 - przekroczenie zakresu dla obliczeń ze stałym przecinkiem.

4. CIĄGI KODOWE

4.1. Obliczanie ciągów kodowych, realizujących daną sekwencję binarną metodą dopisywania

Do obliczania ciągów kodowych odpowiadających określonej sekwencji binarnej (s - bitowej) można wykorzystać tablicę 2.1, określającą działania rejestru liczącego.

W tym celu tworzy się tablicę prawidłową w sposób następujący:

1. Zapisać s - bitową liczbę dwójkową (sekwencję binarną).
2. Dopisać do niej liczbę powstałą przez cykliczne przesunięcie o jedną pozycję liczby wg 1, otrzymując w kroku pierwszym

$$
\begin{array}{cc}
a_1 & a_s \\
a_2 & a_1 \\
a_3 & a_2 \\
\cdot & \cdot \\
\cdot & \cdot \\
\cdot & \cdot \\
\cdot & \cdot \\
a_s & a_{s-1}
\end{array}
$$

Po każdym dopisaniu sprawdzić ciąg kodowy, Jeżeli jest zasadniczy (tablica jest prawidłowa), proces dopisywania zostaje zakończony. Otrzymany ciąg kodowy jest ciągiem zasadniczym, a więc odpowiadający mu rejestr będzie miał najmniejszą długość potrzebną do zrealizowania rozważanej sekwencji binarnej.

4.2. Obliczanie ciągów kodowych, realizujących daną sekwencję binarną metodą "bazy"

Wprowadzimy pojęcie bazy jako tablicy stanów rejestru określonej generatorem sekwencji, w której $n = s$. Jest to więc tablica, w której liczba kolumn jest równa liczbie wierszy i ze względu na sposób powstawania tej tablicy odpowiadające sobie wiersze i kolumny są sobie równe.

Twierdzenie 4.1

Baza jest tablicą prawidłową.

D o w ó d:

Prawdziwość twierdzenia wynika z definicji generatora jako liczby, która przechodzi w siebie dopiero po s operacjach mnożenia przez 2 modulo (2^s-1). Oznacza to więc, że kolumny (wiersze) tablicy są różne i określają połowę (w szczególnym przypadku całą) grupy generującej.

Baza określa ciąg rzeczywisty ale niekoniecznie zasadniczy. Poszukiwanie ciągu zasadniczego polega na zmniejszaniu liczby kolumn bazy aż do momentu, gdy tablica przestaje być prawidłowa. Tablica o najmniejszej liczbie kolumn, która jest jeszcze prawidłowa, określa ciąg zasadniczy.

Na tej podstawie można sformułować następujący algorytm obliczania ciągu zasadniczego, realizującego daną sekwencję binarną:

1. Utworzyć bazę, która jest ciągiem s liczb, powstającym przez (s-1)-krotne mnożenie generatora sekwencji przez 2 modulo (2^s-1).
2. Dzielić liczby bazy przez 2 (bez reszty) tak długo, aż nastąpią powtórzenia w ciągu. Ostatni ciąg, w którym nie było powtórzeń, jest ciągiem zasadniczym, stanowiącym minimalną realizację zadanego ciągu binarnego.
3. Przy przyjętym sposobie zdefiniowania bazy otrzymane tą drogą rozwiązanie stanowi uzupełnienie do 1 ciągu otrzymanego przy użyciu metody dopisywania odczytywanego w odwrotnej kolejności.

Obie metody zostaną zilustrowane poniższym przykładem.

Przykład 4.1

Obliczyć ciąg kodowy generowany przez sekwencję binarną 110100 oraz długość rejestru liczącego, generującego tę sewkencję.

a) rozwiązanie metodą "dopisywania"

Dopisując do kolumny utworzonej z zadanej sekwencji binarnej kolumnę przesuniętą cyklicznie o jeden bit, otrzymuje się:

1	0	2
1	1	3
0	1	1
1	0	2
0	1	1
0	0	0

Ponieważ otrzymany ciąg kodowy zawiera powtórzenia (liczby 2 oraz 1), otrzymana tablica stanów nie jest tablicą regularną i proces dopisywania należy kontynuować.

W kroku drugim otrzymuje się:

1	0	0	4
1	1	0	6
0	1	1	3
1	0	1	5
0	1	0	2
0	0	1	1

Otrzymana tablica stanów jest tablicą regularną, gdyż ciąg kodowy nie zawiera powtórzeń. Rozwiązaniem zadania jest więc ciąg kodowy: 4-6-3-5-2-1, generowany przez 3-bitowy rejestr przesuwający.

b) rozwiązanie metodą "bazy"

Tworzy się "bazę", która jest zbiorem 6 liczb powstających przez kolejne mnożenie liczby (sekwencji) binarnej przez 2 modulo $2^6-1=63$. Otrzymuje się bazę: 52, 41, 19, 38, 13, 26.

Następnie dzieli się liczby bazy przez 2 bez reszty. Proces dzielenia kontynuuje się do momentu wystąpienia powtórzeń.

Kolejne dzielenia dają następujące zbiory:

26,20,9,19,6,13

13,10,4,9,3,6

6,5,2,4,1,3

Ponieważ następne dzielenie daje powtórzenia, ostatni zbiór odczytywany od tyłu (3,1,4,2,5,6) stanowi rozwiązanie w postaci uzupełnienia do 1. Przepisując ten ciąg w postaci prostej, uzyskuje się ostateczne rozwiązanie: 4-6-3-5-2-1.

4.3. Obliczanie ciągów kodowych generowanych przez rejestr o długości n

Struktura rejestru liczącego uniemożliwia otrzymywanie dowolnego ciągu kodowego. Wprowadzony w punkcie 2 sposób zapisu działania rejestru liczącego pozwala jednak na obliczenie s-stanowych ciągów kodowych, które potrafi wygenerować rejestr o określonej długości n. Wykorzystuje się w tym celu równanie macierzowe (2.6).

Przyjęcie parametrów n oraz s określa macierz B. Uwzględniając to, że elementy macierzy A przyjmują tylko wartości 0 lub 1 otrzymuje się 2^s różnych macierzy A. Przez pomnożenie macierzy B przez macierze A można więc obliczyć 2^s różnych ciągów kodowych, określonych macierzą X. Te z nich, które spełniają warunek (2.2), określający prawidłową tablicę (2.1) stanów, są rozwiązaniem, tzn. stanowią ciąg kodowy rejestru liczącego.

Liczbę operacji mnożenia B A można zmniejszyć dzięki następującemu twierdzeniu.

Twierdzenie 4.2

Ciągi kodowe rejestru liczącego występują parami, przy czym suma odpowiadających sobie elementów obu ciągów wynosi (2^n-1).

D o w ó d:

Ze względu na zerojedynkowe wartości elementów macierzy A obliczanie macierzy X polega na sumowaniu odpowiednich kolumn macierzy B. O tym, które kolumny są sumowane, decyduje wartość elementów macierzy A: kolumna "i" jest sumowana, jeżeli $a_i = 1$.

Można zauważyć, że przemnożenie macierzy B przez wektor A, zawierający same jedynki, co odpowiada dodaniu do siebie wszystkich kolumn macierzy B, daje wektor składający się z tych samych wyrazów o wartości (2^n-1). Jeżeli więc macierz B zostanie przemnożona przez określony wektor A i otrzyma się:

$$B \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ \cdot \\ a_s \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_s \end{bmatrix} \tag{4.1}$$

to przemnożenie tej macierzy B przez wektor A, zawierający elementy będące negacjami elementów macierzy A, da w wyniku:

$$B \begin{bmatrix} \bar{a}_1 \\ \bar{a}_2 \\ \cdot \\ \cdot \\ \cdot \\ \bar{a}_s \end{bmatrix} = \begin{bmatrix} 2^n - 1 - x_1 \\ 2^n - 1 - x_2 \\ \cdot \\ \cdot \\ \cdot \\ 2^n - 1 - x_s \end{bmatrix} \tag{4.2}$$

c.b.d.o.

U w a g a:

Interpretacja techniczna tego twierdzenia jest następująca. Stan rejestru jest określony przez sygnały na wyjściach przerzutników przyjętych jako wyjścia proste. Natomiast wyjścia zanegowane przerzutników określają uzupełnienie do 1 tego stanu, tj. ich suma wynosi $2^n - 1$. Oznacza to

więc, że oba ciągi kodowe określone są przez ten sam rejestr liczący - jeden przez sygnały na wyjściach prostych przerzutników, drugi - przez sygnały na wyjściach zanegowanych.

Tym samym zostało uzasadnione przyjęte w punkcie 2 założenie o równoważności liczby i jej uzupełnienia do 1. Przeprowadzone w punkcie 3 rozważania pozwoliły na określenie tzw. generatorów sekwencji, które odpowiadają macierzom A i które były zawarte w przybliżeniu w pierwszych 18,75% zbioru liczb od 1 do $2^s - 1$ i obejmowały wyłącznie liczby nieparzyste. Do obliczania ciągów kodowych należy więc wykorzystać jako macierz A generatory sekwencji określone w punkcie 3

Do znalezionych w czasie tego mnożenia ciągów kodowych można zawsze obliczyć związane z nimi ciągi, wykorzystując prawą stronę zależności (4.2). Z praktycznego punktu widzenia znajomość tego drugiego kodu jest nieinteresująca. Do zaprojektowania rejestru wystarcza jeden kod, a drugi stanowi uzupełnienie do 1 stanów ciągu pierwszego.

W szczególnym przypadku następuje zlanie się obu ciągów kodowych w jeden. Występuje to dla niektórych wektorów A, zawierających symetryczny układ zer i jedynek, a więc może wystąpić jedynie dla parzystych s. Ciąg taki będzie nazywany ciągiem kodowym pojedynczym. Rejestr liczący, realizujący ciąg pojedynczy, realizuje ten ciąg zarówno na wyjściach prostych jak i zanegowanych, z tym, że są one przesunięte względem siebie. Przesunięcie to równa się s/2 impulsów przesuwających.

U w a g a:

Nie każdy ciąg, którego ciąg odwrotny składa się z tych samych liczb, jest ciągiem pojedynczym. Ilustracją tego jest ciąg w przykładzie 4.2 oraz większość ciągów o długości $s = 2^n$.

Niektóre obliczone tą metodą ciągi kodowe są ciągami pozornymi. Dlatego każdy z ciągów musi być sprawdzony czy jest ciągiem prawidłowym. Jeżeli zauważyć, że sprawdzanie czy ciąg jest prawidłowy, polegające na (por. punkt 4.2) zmniejszeniu tablicy stanów, odpowiada dzieleniu (bez reszty) wyrazów ciągu kodowego przez 2, to otrzymuje się prosty algorytm:

a) podzielić (bez reszty) wyrazy ciągu kodowego przez 2,
b) jeżeli nowo otrzymany ciąg jest rzeczywisty, ciąg pierwotny jest pozorny. Jeżeli nowo otrzymany ciąg nie jest ciągiem rzeczywistym, to ciąg pierwotny jest zasadniczy.

Przykład 4.2

Dla n = 4 i s = 6 obliczono ciąg kodowy 9-12-6-11-5-2 jako rezultat następującego działania:

$$\begin{bmatrix} 8 & 0 & 0 & 1 & 2 & 4 \\ 4 & 8 & 0 & 0 & 1 & 2 \\ 2 & 4 & 8 & 0 & 0 & 1 \\ 1 & 2 & 4 & 8 & 0 & 0 \\ 0 & 1 & 2 & 4 & 8 & 0 \\ 0 & 0 & 1 & 2 & 4 & 8 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 9 \\ 12 \\ 6 \\ 11 \\ 5 \\ 2 \end{bmatrix}$$

Po podzieleniu wyrazów ciągu przez 2 (bez reszty) otrzymuje się ciąg: 4-6-3-5-2-1, który jest również ciągiem rzeczywistym. Stąd wniosek, że ciąg pierwotny 9-12-6-11-5-2 jest ciągiem pozornym, a sekwencją binarną opisaną macierzą A (1-1-0-1-0-0) można zrealizować za pomocą rejestru złożonego z trzech (n = 3) przerzutników. Nie ma potrzeby sprawdzania czy istnieje realizacja tego przebiegu za pomocą rejestru złożonego z dwóch przerzutników, za pomocą których można otrzymać tylko cztery stany.

4.4. Ciągi kodowe

Z przedstawionych poprzednio metod generowania ciągów kodowych metoda dopisywania jest najwygodniejsza przy ręcznych obliczeniach. Wykorzystywanie zapisu macierzowego opisującego rejestr daje szybkie rezultaty, jeżeli chce się uzyskać ciągi kodowe realizowane przez n-bitowy rejestr. Natomiast metoda "bazy" jest nejlepsza w przypadku poszukiwania realizacji kodów dla ustalonej długości s sekwencji (ile trzeba przerzutników dla zrealizowania s-stanowej sekwencji zdefiniowanej danym generatorem).

Posługując się metodą "bazy" obliczono realizację ciągów kodowych dla sekwencji o długościach nie większych od 16. Wyniki tych obliczeń przedstawiono w tablicach 4.1-4.5.

W tablicy 4.1 pokazano ile istnieje kodów (generatorów) o określonej długości s realizowanych na rejestrze liczącym o n przerzutnikach. Szczególnie interesujące są realizacje minimalne. Zebrane one zostały w tablicach 4.2-4.4, które przedstawiają generatory sekwencji o długości s realizowanych na najkrótszych rejestrach liczących.

Natomiast w tablicy 4.5 zebrano część informacji, które posłużyły do sporządzenia poprzednich tablic. Są w niej określone długości n rejestrów liczących, potrzebnych do realizacji generatorów sekwencji mniejszych od 100 dla sekwencji nie dłuższych niż s = 10. Dla s = 10 określono również te długości dla generatorów większych od 100, przekazując w ten sposób pełną informację o sekwencjach o s = 10.

Interesującym wnioskiem wynikającym z tablicy 4.1 jest stwierdzenie, że każdy s-bitowy ciąg (zasadniczy) jest realizowany na co najwyżej n = = s-1 przerzutnikach. Uogólnienie tego na dowolne s jest rzeczą trywialną - wystarczy w tym celu metodą dopisywania poszukiwać ciąg zasadniczy o sekwencji 100 ... 0.

Tablica 4.1

Liczba ciągów kodowych (generatorów)
dla danej długości sekwencji s i ich rozkład w funkcji
długości rejestru n

s \ n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	$\sum$ *
1																
2	1															1
3		1														1
4		1	1													2
5			1	2												3
6			2	2	1											5
7			2	2	2	3										9
8			1	6	4	3	2									16
9				7	9	7	2	3								28
10				9	21	8	6	5	2							51
11				7	32	23	15	7	4	5						93
12				8	50	53	27	19	8	3	2					170
13				6	71	97	67	35	19	9	5	6				315
14				11	95	184	153	66	37	18	10	8	3			585
15				16	134	322	279	162	85	53	20	13	3	4		1091
16				8	199	562	556	368	174	88	46	26	10	7	4	2048

* $\sum$ odpowiada liczbie grup generujących zebranych w tabl. 3.1.

Tablica 4.2

Generatory do minimalnych realizacji
na rejestrach 2-bitowych

s	Generator
3	1
4	3

Tablica 4.3

Generatory dla minimalnych realizacji na rejestrach 3-bitowych

s	Generator
5	3
6	7,11
7	11,13
8	23

Tablica 4.4

Generatory dla minimalnych realizacji na rejestrach 4-bitowych

s	Generator
9	19, 23, 25, 29, 39, 45, 55
10	39, 45, 47, 55, 57, 59, 83, 93, 101
11	79, 83, 93, 101, 111, 167, 215
12	167, 189, 215, 229, 235, 311, 315, 441
13	311, 315, 335, 431, 441, 473
14	623, 635, 889, 1239, 1259, 1335, 1437, 1447, 1485, 1687, 1721
15	1239, 1259, 1335, 1339, 1437, 1447, 1485, 1491, 1629, 1687, 1721, 1765, 1837, 1867, 1881, 1893
16	2479, 2539, 2671, 2877, 2895, 3021, 3375, 3449

Tablica 4.5

Długości rejestrów (n) potrzebnych
do realizacji określonego generatora o długości e

Generator e	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	43	45	47	49	51	53	55	57	59	61	63	65	67	69	71	73	75	77	79	81	83	85	87	89	91	93	95	97	99	101	103	105	107	109	.	.	.	.	.	147	149	155	171	173	179
2	①																																																																			
3	②																																																																		
4	3	②																																																																	
5	4	③	/	4																																																															
6	5	4		4	③	-		③																																																										
7	6	5		4	4	6		③	③			5		6																																																			
8	7	6		5	5	6		4	4	4	-	4		⑥	③		4	5					7					5	4																																							
9	8	7		6	6	6		5	5	5	8	④		6	④		4	5		④			6	5	④	5	5	④			7	5	④						-	6	6			5	8																						
10	9	8		7	7	6		6	6	6	8	5		6	5		5	5		5	-		5	5	④	5	5	④	④	5	7	5	④	④	④		8		6	9	5	5		④	8	5	5	6	④			-	4	5	5	5	6				8	7	5	7	5	7

Uwagi:

a) "-" oznacza sekwencję symetryczną.

b) linia łamana ogranicza zakres możliwych generatorów dla danej długości sekwencji e.

c) w kółku oznaczono rozwiązania o minimalnej długości rejestru.

5. PROJEKTOWANIE KRÓTKICH REJESTRÓW LICZĄCYCH

5.1. Projektowanie rejestru liczącego. Samokorekcja

Synteza rejestru liczącego polega na określeniu liczby przerzutników rejestru oraz schematu obwodu sprzężenia zwrotnego.

Punktem wyjścia jest ciąg kodowy rejestru otrzymany sposobami przedyskutowanymi w poprzednich punktach. Określenie liczby przerzutników rejestru jest rzeczą oczywistą. Liczba ich jest ograniczona zależnością:

$$2^n \geqslant s, \qquad (5.1)$$

gdzie:

n - ilość przerzutników rejestru,

s - ilość stanów ciągu kodowego.

Określenie schematu obwodu sprzężenia zwrotnego polega na znalezieniu funkcji przełączających sygnału wejściowego rejestru, przy czym wejście może być wejściem informacyjnym przerzutnika D lub J - K. Dla określenia tych funkcji należy utworzyć tablicę zależności, w której dla poszczególnych stanów rejestru i odpowiadających im stanów na wyjściu Q_{n-1} rejestru określa się sygnał informacyjny D lub J - K w zależności od zastosowanych w układzie przerzutników. W przypadku stosowania przerzutników J - K należy określić sygnał T, a dopiero na tej podstawie sygnały J i K zgodnie z metodą określoną w [13]. Otrzymuje się wtedy minimalne postacie funkcji przełączających.

Tak prowadzona synteza umożliwia równocześnie otrzymywanie układów samokorekcyjnych. Uwzględnienie tego problemu przy tworzeniu obwodu sprzężenia zwrotnego rejestru daje z reguły prostsze rozwiązanie, aniżeli dekodowanie niedozwolonych stanów i ustawienie rejestru w określony stan prawidłowy.

Problem samokorekcji w rejestrach liczących należy rozwiązać przez właściwą interpretację stanów obojętnych. Przyczyna wystąpienia stanu obojętnego w układach kombinacyjnych jest inna, aniżeli w układach sekwencyjnych. W układach kombinacyjnych stan obojętny nie może wystąpić ze względu na określone warunki, jakie spełniają urządzenia generujące sygnały wejściowe. W układach sekwencyjnych należy być bardziej ostrożnym przy wykorzystywaniu stanów obojętnych. Wynika to stąd, że są one często wynikiem przyjętego sposobu kodowania elementów pamięci. Istnieje więc zawsze możli-

wość taka, że układ przejdzie pod wpływem zakłóceń względnie po załączeniu napięcia zasilającego urządzenia w stan nieprzewidziany przyjętym kodowaniem elementów pamięci. Specyficznym przypadkiem układów synchronicznych, w których ten problem występuje bardzo wyraźnie, są rejestry liczące. W rejestrach liczących stany obojętne są wyłącznie niewykorzystanymi stanami elementów pamięci.

Uwzględnianie problemu samokorekcji rejestrów liczących polega na analizie wszystkich stanów obojętnych w siatkach Karnaugha, określających sygnały D lub T. Wynikiem tej analizy jest stwierdzenie czy dany stan obojętny można przy określaniu funkcji przełączającej sygnału D względnie T uważać za obojętny, czy też należy mu przypisać ściśle określoną wartość 0 lub 1. W efekcie tej analizy otrzymuje się skorygowane siatki Karnaugha sygnału D względnie T, określające sprzężenia zwrotne samokorekcyjnych rejestrów liczących. Praktycznie, przeprowadzenie takiej analizy jest kłopotliwe i dlatego wydaje się celowe następujące postępowanie:

- po przepisaniu zawartości tablicy zależności do siatki Karnaugha należy uzupełnić ją jedynie o prawidłowe wartości sygnałów D względnie T dla stanów 00...0 i 11...1 o ile nie są wykorzystywane w ciągach kodowych. Wyjście rejestru liczącego z tych dwóch stanów jest możliwe jedynie przez wpisanie 1 do rejestru w przypadku pierwszym, a 0 w przypadku drugim, innymi słowy musi nastąpić zmiana stanu przerzutnika wejściowego rejestru. Oznacza to więc, że w siatkach, określających sygnał D, należy wpisać dla stanu 00...0 wartość 1, a dla stanu 11...1 - wartość 0, a w siatkach dla sygnału T należy wpisać dla obu rozpatrywanych stanów sygnał 1;
- wypisać funkcje przełączające dla sygnałów informacyjnych przerzutnika wejściowego rejestru i narysować diagram przejść zrealizowanego w ten sposób rejestru liczącego;
- w przypadku wystąpienia w diagramie przejść cykli nieprawidłowych skorygować siatkę Karnaugha tak, by uzyskać samokorekcję układu, tzn. uzyskać przejście z wszystkich stanów (cykli) nieprawidłowych do ciągu kodowego rejestru liczącego;
- w oparciu o skorygowane siatki Karnaugha wypisać ostateczne funkcje przełączające, określające sygnały informacyjne rejestru liczącego, który jest układem samokorekcyjnym.

<u>Przykład 5.1</u>

Zaprojektować rejestr liczący na 4 przerzutnikach, generujący 9-stanową sekwencję, odpowiadającą generatorowi 19 (z tablicy 4.5 wynika, że można ją zrealizować na 4 przerzutnikach).

Generatorowi 19 odpowiada następujący ciąg kodowy:

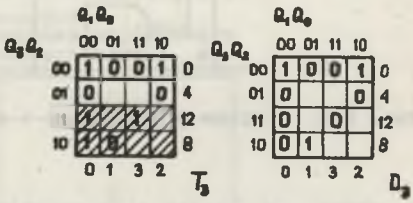
Q_3				
0	1	1	0	6
0	0	1	1	3
0	0	0	1	1
0	0	0	0	0
1	0	0	0	8
0	1	0	0	4
0	0	1	0	2
1	0	0	1	9
1	1	0	0	12

Ciągowi temu odpowiada tablica zależności i siatki Karnaugha, przedstawione na rys. 5.1.

a).

	Q_3	T_3	D_3
6	0	0	0
3	0	0	0
1	0	0	0
0	0	1	1
8	1	1	0
4	0	0	0
2	0	1	1
9	1	0	1
12	1	1	0

b).



Rys. 5.1. Tablica zależności (a) i siatki Karnaugha (b) do przykładu 5.1

W siatkach Karnaugha stany sygnałów T_3 i D_3 dla $Q_3Q_2Q_1Q_0$ = 1111 zostały wpisane dla uzyskania samokorekcji.

Na podstawie siatek Karnaugha można wypisać następujące funkcje przełączające:

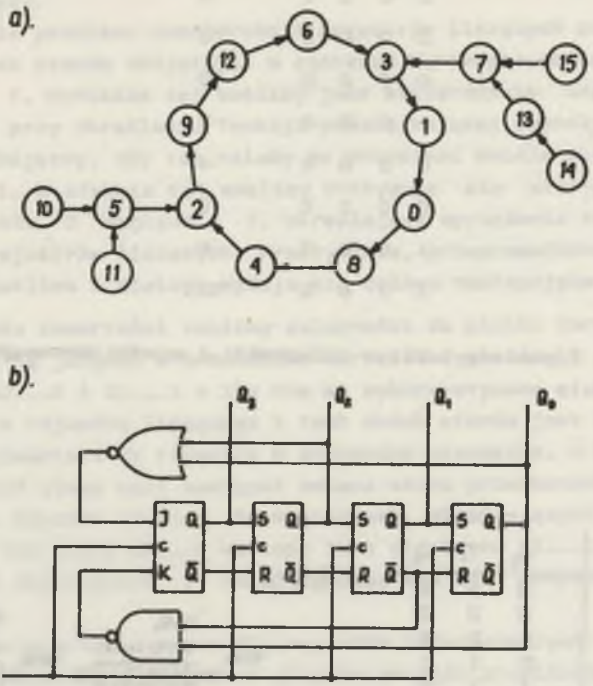
$$J_3 = \bar{Q}_2\bar{Q}_0$$

$$K_3 = Q_1 + \bar{Q}_0$$

oraz

$$D_3 = \bar{Q}_3\bar{Q}_2\bar{Q}_0 + Q_3\bar{Q}_2Q_0$$

Rejestr liczący, posiadający na wejściu przerzutnik J - K,którego wejścia informacyjne J i K pobudzane są wyżej wypisanymi sygnałami,posiada diagram przejść przedstawiony na rys. 5.2.



Rys. 5.2. Diagram przejść (a) i schemat rejestru liczącego (b) do przykładu 5.1

Widać, że rejestr jest samokorekcyjny; jego schemat jest pokazany na tym samym rysunku. Nieuwzględnienie w siatce Karnaugha dla T_3 stanu 1 dla $Q_3Q_2Q_1Q_0$ = 1111 pozwala na uproszczenie funkcji przełączającej, określającej sygnał K_3, która redukuje się do wyrażenie $K_3 = \bar{Q}_0$. Rejestr z tym sprzężeniem nie ma jednak zapewnionej samokorekcji.

Podobnie można określić schemat i diagram przejść w przypadku,gdy przerzutnikiem wejściowym rejestru jest przerzutnik typu D. Schemat ten jest bardziej złożony.

5.2. Rejestry liczące jako liczniki

W wielu przypadkach nie jest potrzebne stosowanie liczników liczących modulo s i wykorzystujących pierwszych s liczb naturalnego kodu dwój-

kowego. W szczególności nie jest to potrzebne przy współpracy licznika z dekoderem, kiedy to właściwe zdekodowanie otrzymuje się przez odpowiednie oznaczenie wyjść dekodera.

W tego rodzaju zastosowaniach, w których kod liczenia jest obojętny, wykorzystanie rejestrów liczących jako liczników może się okazać najwłaściwszym rozwiązaniem. Otrzymywane schematy tych liczników są proste i pozwalają na uzyskiwanie maksymalnych częstotliwości liczenia, co wynika z równoległej (synchronicznej) budowy rejestrów liczących.

Dla ilustracji możliwości w tym zakresie przeliczono wszystkie możliwe warianty rejestrów liczących o pojemnościach od 3 do 16, konstruowanych na możliwie najmniejszej liczbie przerzutników. Dane dotyczące sygnałów informacyjnych J i K pierwszego przerzutnika rejestru, wymagającego do realizacji najmniejszej liczby elementów, zostały zebrane w tablicach 5.1, 5.2, 5.3. Podane w tych tablicach dane określają struktury samokorekcyjnych rejestrów liczących. Usunięcie z funkcji określających sygnał K dla s = 13, 14, 15 wyrażeń w nawiasach daje prostsze struktury, lecz liczniki nie mają wtedy właściwości samokorekcji.

Jeżeli porównać zebrane w tablicach 5.1-5.3 funkcje przełączające ze strukturami odpowiednich liczników synchronicznych [1] o analogicznych pojemnościach, to okazuje się, że układ w postaci rejestru liczącego jest z reguły tańszy, tj. jego realizacja wymaga zastosowania mniejszej liczby bramek przełączających.

Tablica 5.1

Sygnały informacyjne przerzutnika wejściowego 2-bitowych liczników liczących modulo s

s	J	K	Generator
3	$\bar{Q}_0$	1	1
4	$\bar{Q}_0$	Q_0	3

Tablica 5.2

Sygnały informacyjne przerzutnika wejściowego 3-bitowych liczników liczących modulo s

s	J	K	Generator
5	$\bar{Q}_0$	Q_1	3
6	$\bar{Q}_0$	Q_0	7
7	$\bar{Q}_0$	$Q_1 + \bar{Q}_0$	11
8	$\bar{Q}_0$	$Q_1 \oplus Q_0$	23

Tablica 5.3

Sygnały informacyjne przerzutnika wejściowego 4-bitowych liczników liczących modulo s

s	J	K	Generator
9	$\bar{Q}_0$	$Q_2 + \bar{Q}_1$	23
10	$\bar{Q}_0$	$Q_1 \oplus Q_0$	47
11	$\bar{Q}_0$	$Q_2 \oplus Q_0$	79
12	$\overline{Q_1 \oplus Q_0}$	$\overline{Q_1 + Q_0}$	189
13	Q_0	$\overline{Q_2 \oplus Q_0 + (Q_1 Q_0)}$	335
14	$\bar{Q}_0 + \bar{Q}_2 \bar{Q}_1$	$\bar{Q}_0 + (Q_2 Q_1)$	1335
15	$\bar{Q}_0$	$\overline{\bar{Q}_0 + (Q_2 Q_1)}$	1335
16	$\bar{Q}_0$	$\overline{Q_0 \oplus Q_1 Q_2 + Q_0 Q_2}$	2895

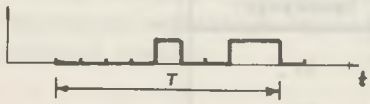
5.3. Generowanie przebiegów czasowych. Generatory słów

Za pomocą rejestru liczącego można generować przebiegi periodyczne o charakterze fali prostokątnej, przy czym kształt przebiegu w okresie jest dowolny, z zastrzeżeniem, że zmiany poziomu następują jedynie w określonych momentach, będących wielokrotnością pewnego odcinka czasu. Układami takimi są np. generatory słów zwane pamięciami stałymi na rejestrach.

Synteza struktury rejestru liczącego, realizującego zadany przebieg w czasie, polega na zapisie tego przebiegu w postaci sekwencji binarnej i określeniu metodą "dopisywania" (por. punkt 4.1) ciągu kodowego, który odpowiada tej sekwencji. Dalszy etap projektowania przebiegu zgodnie z procedurą określoną w punkcie 5.1.

Przykład 5.2

Zaprojektować rejestr liczący, realizujący przebieg czasowy przedstawiony na rys. 5.3.

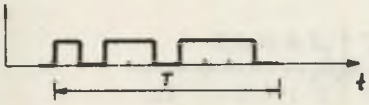


Rys. 5.3. Przebieg czasowy sygnału do przykładu 5.2

Przebieg ten odpowiada sekwencji binarnej 000010011, która jest generatorem 19 sekwencji 9-stanowej. Układ, który generuje tę sekwencję, został zaprojektowany w przykładzie 5.1. Wobec tego rejestr liczący, którego schemat jest pokazany na rys. 5.2, stanowi rozwiązanie zadania, generując żądany przebieg czasowy z rys. 5.3.

Przykład 5.3

Zaprojektować rejestr liczący, realizujący przebieg czasowy o okresie T przedstawiony na rys. 5.4.



Rys. 5.4. Przebieg czasowy sygnału do przykładu 5.3

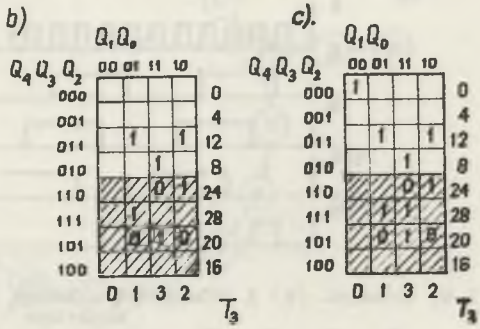
Przedstawionemu na rysunku 5.4 przebiegowi odpowiada sekwencja binarna 101101110. Stosując metodę dopisywania otrzymuje się następujący ciąg kodowy:

1	0	1	1	1	23
0	1	0	1	1	11
1	0	1	0	1	21
1	1	0	1	0	26
0	1	1	0	1	13
1	0	1	1	0	22
1	1	0	1	1	27
1	1	1	0	1	29
0	1	1	1	0	14

Ciągowi temu odpowiada tablica zależności i siatki Karnaugha przedstawione na rys. 5.5.

a).

	Q_4	T_3
23	1	1
11	0	1
21	1	0
26	1	1
13	0	1
22	1	0
27	1	0
29	1	1
14	0	1



Rys. 5.5. Tablica zależności (a) i siatki Karnaugha (b,c) do przykładu 5.3

Na podstawie siatek Karnaugha (b) otrzymuje się następujące funkcje przełączające:

$$J = 1$$

$$K = Q_3 \oplus Q_1Q_0$$

Rejestr liczący z tego typu sprzężeniem zwrotnym nie ma właściwości samokorekcji. Uzupełnienie siatki Karnaugha "1" dla stanów 00000 i 11111 (rys. 5.5c) pozwala na wypisanie następujących funkcji przełączających:

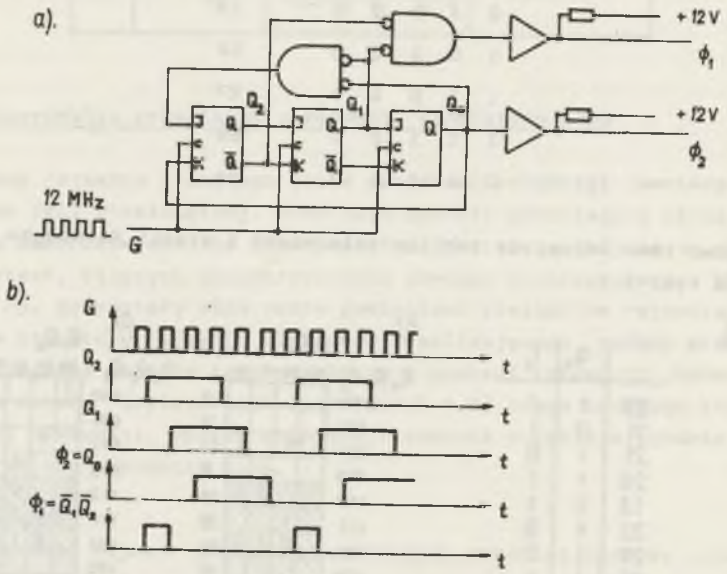
$$J = 1$$

$$K = Q_3 \oplus Q_1Q_0 + Q_3Q_2,$$

które określają sprzężenia zwrotne rejestru liczącego samokorekcyjnego, realizującego na swych wyjściach przebieg czasowy z rys. 5.4.

Przykład 5.4

Na rys. 5.6 przedstawiono układ generujący sygnały zegarowe dla mikroprocesora INTEL 8080, zrealizowany przez autora. Istotnym fragmentem układu jest rejestr liczący 6-stanowy (tzw. pseudo pierścieniowy) z samokorekcją. Szczegóły syntezy pominięto.



Rys. 5.6. Schemat (a) i przebiegi czasowe (b) układu generującego sygnały zegarowe

Przykład 5.5

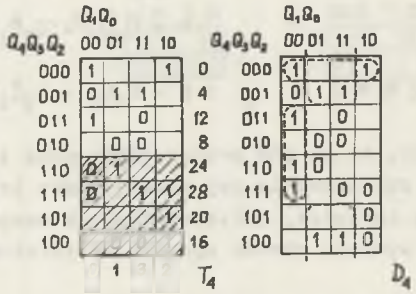
Zaprojektować generator następujących cyklicznie słów liczb binarnych 6, 7, 8 i 9 [1].

Etapy syntezy tego układu przedstawia rys. 5.7. Mniej bramek logicznych wymaga realizacja sygnału D; schemat układu pominięto. Otrzymany generator jest układem samokorekcyjnym.

a).

	Q_4	Q_3	Q_2	Q_1	Q_0	T_4	D_4
22	1	0	1	1	0	1	0
11	0	1	0	1	1	0	0
5	0	0	1	0	1	1	1
18	1	0	0	1	0	1	0
9	0	1	0	0	1	0	0
4	0	0	1	0	0	0	0
2	0	0	0	1	0	1	1
17	1	0	0	0	1	0	1
24	1	1	0	0	0	0	1
28	1	1	1	0	0	0	1
30	1	1	1	1	0	1	0
15	0	1	1	1	1	0	0
7	0	0	1	1	1	1	1
19	1	0	0	1	1	0	1
25	1	1	0	0	1	1	0
12	0	1	1	0	0	1	1

b).

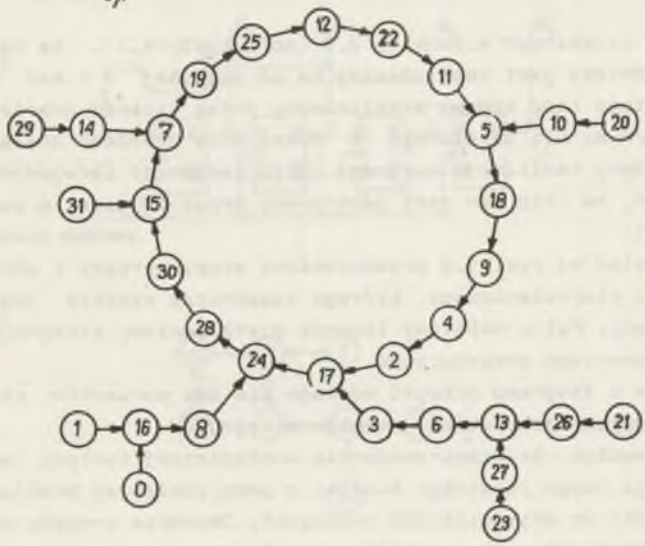


$$J_4 = \bar{Q}_3\bar{Q}_2 + (Q_3 \oplus Q_0)$$

$$K_4 = Q_1\bar{Q}_0 + Q_3Q_0$$

$$D_4 = \bar{Q}_3Q_0 + Q_3\bar{Q}_1\bar{Q}_0 + \bar{Q}_4\bar{Q}_3\bar{Q}_2$$

c).



Rys. 5.7. Tablica zależności (a), siatki Karnaugha (b) i diagram przejść (c) do przykładu 5.5

Rozwiązanie tego generatora w [1] jest pozbawione właściwości samokorekcji - generator nie wychodzi ze stanu samych zer - proponuje się tam zdekodowanie tego stanu i wprowadzenie uzyskanego sygnału do obwodu sprzężenia zwrotnego.

Jeżeli porównać oba rozwiązania

w [1]: $D = \bar{Q}_3 Q_0 + Q_3 \bar{Q}_1 \bar{Q}_0 + \bar{Q}_4 Q_1 \bar{Q}_0$

na rys. 5.7: $D = \bar{Q}_3 Q_0 + Q_3 \bar{Q}_1 \bar{Q}_0 + \bar{Q}_4 \bar{Q}_3 \bar{Q}_2$,

to widać, że są one prawie takie same i ich realizacja np. na bramkach NOR wymaga zastosowania tej samej liczby bramek. Przykład ten może być ilustracją korzyści, jakie wynikają z uwzględnienia problemu samokorekcji na etapie syntezy obwodu sprzężenia zwrotnego.

5.4. Uwagi dotyczące konstrukcji licznika pierścieniowego

Powszechnie znana zasada tworzenia tego licznika polega na doprowadzeniu sygnału wyjściowego Q_0 z ostatniego stopnia rejestru przesuwającego na wejście D względnie na doprowadzeniu sygnałów Q_0 i $\bar{Q}_0$ odpowiednio na wejścia J i K, jeśli stopniem wejściowym rejestru jest przerzutnik J - K. Otrzymany w ten sposób licznik nie ma właściwości samokorekcji.

Z wyników uzyskanych w punkcie 4.3 (por. tabl. 4.1), że każdy s-bitowy ciąg zasadniczy jest realizowalny na co najwyżej $n = s-1$ przerzutnikach. Wobec tego ciąg kodowy zrealizowany przez licznik pierścieniowy jest ciągiem pozornym. Dla ustalonego s można więc próbować znaleźć ciąg zasadniczy, będący realizacją żądanego ciągu sekwencji zerojedynkowej. Łatwo się przekonać, że ciąg ten jest generowany przez rejestr przesuwający o długość $s - 1$.

Jako przykład na rys. 5.8 przedstawiono etapy syntezy i schemat 4-bitowego licznika pierścieniowego, którego zasadniczą częścią jest 3-bitowy rejestr liczący. Pełny 4-bitowy licznik pierścieniowy otrzymuje się przez dołączenie czwartego przerzutnika.

Jak wynika z diagramu przejść ważnego dla obu wariantów, otrzymany rejestr przesuwający jest układem samokorekcyjnym.

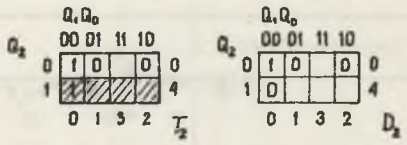
Można sprawdzić, że przeprowadzenie analogicznej syntezy przy założeniu realizacji ciągu pozornego 8-4-2-1 z uwzględnieniem problemu samokorekcji prowadzi do analogicznych rozwiązań. Jednakże syntezę wykonuje się z uwzględnieniem czterech sygnałów wyjściowych przerzutników (dwukrotnie większe siatki Karnaugha).

Przedstawione w tym punkcie podejście do syntezy licznika pierścieniowego umożliwia zmniejszenie zmiennych wejściowych o jeden, co ułatwia syntezę w szczególności dla dłuższych rejestrów przesuwających.

a).

	Q_2 Q_1 Q_0	T_2 D_2
4	1 0 0	1 0
2	0 1 0	0 0
1	0 0 1	0 0
0	0 0 0	1 1

b).

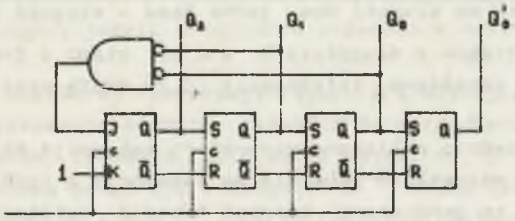


$J_2 = \overline{Q}_1 \overline{Q}_0$

$K_2 = 1$

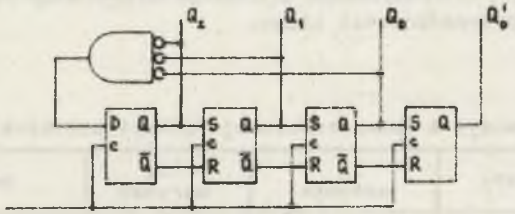
$D_2 = \overline{Q}_2 \overline{Q}_1 \overline{Q}_0$

c).



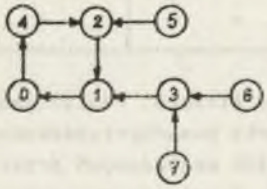
impulsy zegarowe

d).



impulsy zegarowe

e).



Rys. 5.8. Tablica zależności (a), siatki Karnaugha (b), schemat dla rejestru z wejściem J - K (c), schemat dla rejestru z wejściem D (d) i diagram przejść (e) 4-bitowego rejestru przesuwającego z przykładu 5.6

5.5. Generatory sekwencji pseudoprzypadkowych

Zazwyczaj [3] w technice cyfrowej jako pseudoprzypadkowe uważa się ciągi zerojedynkowe, spełniające następujące warunki:

I. Funkcja autokorelacji ciągu jest 2-wartościowa, tzn:

$$c(\tau) = \frac{1}{s}\sum_{n=1}^{s} a_n a_{n+\tau} = \begin{cases} 1 \text{ dla } \tau = 0 \\ -\frac{1}{s} \text{ dla } 0 < \tau < s \end{cases} \qquad (5.2)$$

przy czym przy obliczaniu $c(\tau)$ zamiast wartości elementu równych 0 podstawia się wartość (- 1).

II. Dla każdego okresu liczba "1" i "0" może się różnić co najwyżej o jeden.

III. W każdym okresie połowa ciągów identycznych zdarzeń ma długość jeden, jedna czwarta ma długość dwa, jedna ósma - długość trzy t.i.d.

Obliczono, że dla ciągów o długościach $s \leqslant 16$ ciągi o 2-wartościowej funkcji autokorelacji określonej zależnością (5.2) muszą mieć długość s = = 3, 7, 11 i 15.

Wykorzystując poprzednio obliczone generatory sekwencji obliczono funkcje autokorelacji dla wszystkich generatorów sekwencji o tych długościach. W tablicy 5.4 zebrano te generatory, których funkcja autokorelacji jest dwuwartościowa, zaznaczając równocześnie, które z nich spełniają pozostałe dwa warunki pseudoprzypadkowości ciągu.

Tablica 5.4

Generatory sekwencji o dwuwartościowej funkcji autokorelacji

s	Generatory	II warunek	III warunek	Uwagi
3	1	+	+	+ oznacza, że
7	11, 13	+	+	warunek jest
11	151, 183	+	-	spełniony
15	1335, 1893	+	+	

Jeżeli zaprojektować rejestry liczące, realizujące sekwencje spełniające poprzednio sformułowane warunki pseudoprzypadkowości, to uzyskuje się funkcje przełączające dla sygnałów wejściowych przerzutnika wejściowego rejestru, zebrane w tablicy 5.5. Z tablicy tej widać, że poza powszechnie znanymi rejestrami liniowymi, realizującymi sekwencje pseudoprzypadkowe, istnieją rozwiązania bazujące na innym typie sprzężenia zwrotnego. Interesujące jest to, że te inne rozwiązania są układowo tańsze. Dotyczy to sekwencji o s = 7 i s = 15 ze sprzężeniami j = K = Q_0 (bez samokorekcji).

Tablica 5.5

Sprzężenia zwrotne dla rejestrów liczących generujących sekwencje binarne o pseudoprzypadkowym rozkładzie 0 i 1

s	n	G	J	K	U w a g i
3	2	1	$\bar{Q}_0$	1	samokorekcyjny
7	3	11	$\bar{Q}_0$	$\bar{Q}_0 + (Q_1)$	*
		13	$\overline{Q_1 \oplus Q_0}$	$Q_1 \oplus Q_0$	samokorekcyjny
15	4	1335	$\bar{Q}_0$	$\bar{Q}_0 + (Q_1 Q_2)$	*
		1893	$\overline{Q_1 \oplus Q_0}$	$Q_1 \oplus Q_0 + (Q_2 Q_0)$	*

* samokorekcyjny, jeżeli uwzględnić składnik w nawiasie.

Ta prosta możliwość realizacji rejestru liczącego, generującego pseudoprzypadkową sekwencję binarną, skłoniła do sprawdzenia, jak długo utrzymuje się ta reguła. Istnieje tylko kilka rejestrów, w których uzyskuje się pseudoprzypadkową sekwencję o długości 2^{n-1} przez wykonanie tego typu sprzężenia zwrotnego ($J = K = \bar{Q}_0$). Są to rejestry liczące o długościach s = 3, 4, 6 i 7, generujące sekwencje odpowiednio o długościach 7, 15, 63 i 127.

6. PODSUMOWANIE

Pierwsza część pracy poświęcona jest wprowadzeniu sposobów opisu działania rejestru liczącego. Opisy te w postaci tablicy stanów (2.1) względnie układu równań (2.6) zostały wykorzystane do udowodnienia względnie obliczenia pewnych właściwości sekwencji generowanych przez rejestr liczący. Możliwości stosowania tych opisów do prowadzenia konkretnych obliczeń są ograniczone ze względu na wydłużające się czasy obliczeń. Autor prowadził obliczenia na kalkulatorze programowanym HP 9820, co umożliwiło mu analizę układów o liczbie stanów nie przekraczających w zasadzie 16, przy przyjęciu rozsądnych czasów obliczeń nie większych od kilku godzin. Przejście na dużą maszynę cyfrową zwiększyłoby możliwości obliczeń mniej więcej dwukrotnie. Rezultatem pracy jest dogłębna analiza rejestrów liczących, generujących do 16 stanów, realizowanych na rejestrach liczących o długościach nie przekraczających 15 stopni. Pewne twierdzenie czy wnioski mają charakter ogólny, niezależny od długości sekwencji, czy długości rejestru.

Ujęcie problemu oraz uzyskane wyniki są oryginalne, stanowiąc rozwiązanie przedstawionego w rozdziale 1 zadania. Praca wyjaśnia szereg problemów związanych z generowaniem sekwencji przez rejestry liczące, a w zakresie rejestrów krótkich odpowiada w zasadzie na wszystkie pytania interesujące projektanta urządzeń cyfrowych.

Wprowadzone w rozdziale 2 sposoby opisu działania rejestru liczącego stanowią nie tylko poprawny zapis funkcjonowania rejestru liczącego, lecz stanowią również podstawę do metod obliczenia ciągów kodowych wprowadzonych w dalszej części pracy. W pracy wprowadzono kilka pojęć, takich jak: generator sekwencji, grupa generująca, ciąg zasadniczy, ciąg pozorny czy ciąg pojedynczy, które umożliwiły właściwe zdefiniowanie i uporządkowanie sekwencji binarnych względnie ciągów kodowych generowanych przez rejestr liczący. W efekcie umożliwiło to nie tylko wprowadzenie oszacowań typu wzoru (3.4), lecz również podanie algorytmów obliczeń, których wynikami są dane zebrane w kilku tabelach, a które dają pełny obraz sekwencji uzyskiwanych na rejestrach o niezbyt dużej liczbie stanów.

Ważne rezultaty pracy związane są z generowaniem założonych sekwencji zerojedynkowych, których charakterystycznym zastosowaniem są tzw. generatory słów, zwane również pamięciami stałymi na rejestrach przesuwających. Przedstawiony w pracy sposób "dopisywania" pozwala na realizację tych generatorów za pomocą najkrótszych rejestrów, przy czym metoda jest bardzo prosta i łatwa do realizacji zarówno ręcznej jak i maszynowej.

Wskazano, w jaki sposób rozwiązywać problem samokorekcji na etapie tworzenia funkcji obwodu sprzężenia zwrotnego. Sposób ten wydaje się być lepszy od uzupełnienia gotowego układu obwodami dekodującymi stany,z których nie może on wyjść; jak wynika z wielu przeliczonych przykładów prowadzi on do tańszych rozwiązań.

Korzyści wynikające z przedstawionego w pracy rozwiązywania problemu samokorekcji są oczywiste, a szczególnie sugestywną ilustracją tego są wyniki przykładu 5.5, w którym uzyskano układ samokorekcyjny przy tych samych kosztach realizacji co dla układu pozbawionego właściwości samokorekcji.

Podkreślono zalety wykorzystywania rejestrów liczących jako liczników, podając gotowe rozwiązania wszystkich liczników liczących modulo 3-16, skonstruowanych za pomocą 4-bitowego rejestru przesuwającego.

Ilustracją korzyści, jakie wynikają z przedstawionego w pracy podejścia do rejestrów liczących, może być punkt 5.4, w którym omówiono nieco inną realizację rejestru przesuwającego.

Na zakończenie przeanalizowano również możliwości realizacji przebiegów pseudoprzypadkowych na rejestrach o długościach $n = 2, 3, 4$.

Autor ma nadzieję, że wyniki teoretyczne stanowią pewien wkład do teorii automatów, a wyniki możliwie wszechstronnej analizy krótkich rejestrów liczących mogą znaleźć bezpośrednie zastosowanie przy projektowaniu układów cyfrowych.

LITERATURA

[1] Anderson W.D. i inni: Projektowanie układów z TTL obwodami scalonymi, Warszawa, Ośrodek Informacji o Energii Jądrowej, 1973.

[2] Derjatkov V.V.: Metody realizacji konecznych avtomatov na sdvigovych registrach, Izd. Energija, 1974.

[3] Golomb S.W.: Shift register sequences. San Francisco, Holden Day Inc., 1967.

[4] Hurd W.J.: Efficient generation of statistically good pseudonoise by linearly interconnected shift registers. IEEE Trans. Comput., Febr. 1974, 146-152.

[5] Lantschoot E.J.: Double adjacencies between cycles of a circulating shift register. IEEE Trans. Comput., Oct. 1973, 944-955.

[6] Leavy L.S., Freeman M.: Every finite-state machine can be simulated (realised) by a synchronous (asynchronous) binary shift-register machine. IEEE Trans. Comput., Febr. 1974, 174-178.

[7] Misiurewicz P., Grzybek M.: Półprzewodnikowe układy logiczne, WNT, Warszawa 1975.

[8] Mykkeltveit J.: Generating and counting the double adjacencies in a pure circulating shift register. IEEE Trans. Comput., March 1975, 299-304.

[9] Neumann F., Martin C.F.: The autocorrelation structure of Tausworthe pseudorandom number generators. IEEE Trans. Comput. May 1976, 460-464.

[10] Neuman F., Merrick R.B.: Autocorrelation peaks in congruential pseudorandom number generators. IEEE Trans. Comput., May 1976, 457-464.

[11] Schmidt K.J.: Schieberegister in Rauschgeneratoren. Elektronik 5, 1975, 79-81.

[12] Smith A.R.: General shift-register sequences of arbitrary cycle length. IEEE Trans. Comput., April 1971, 456-459.

[13] Wagner F.: Liczniki elektroniczne w przemysłowych układach sterowania, WNT, Warszawa 1971.

[14] Wagner F.: Kody rejestrów liczących. Archiwum Automatyki i Telemechaniki, Zeszyt 2, 1976.

[15] Wagner F.: Projektowanie urządzeń cyfrowych, WNT, Warszawa, (przyjęte do druku).

[16] Wagner F.: A Wiring Tester. Proceedings of the second Symposium on CAMAC in Computer Applications, Brussels, March 1976.

PROJEKTOWANIE KRÓTKICH REJESTRÓW LICZĄCYCH

S t r e s z c z e n i e

Praca poświęcona jest rejestrom przesuwającym ze sprzężeniem zwrotnym, zwanym rejestrami liczącymi. Przeanalizowano w niej sekwencje binarne generowane przez rejestry liczące, pokazano jak realizować rejestry liczące, generujące żądane sekwencje oraz podano efektywną metodę projektowania rejestrów liczących z uwzględnieniem problemu samokorekcji sekwencji generowanych przez te rejestry.

Część pierwsza pracy poświęcona jest analizie sekwencji generowanych przez rejestry liczące. Wprowadzono w niej sposoby zapisu funkcjonowania rejestrów liczących oraz metody obliczania ciągów kodowych. Pozwoliło to na przeprowadzenie szeregu obliczeń, których rezultaty dają pełny obraz rejestrów liczących, generujących sekwencje nie dłuższe niż 16 stanów.

W części drugiej pracy przedstawiono metody projektowania rejestrów liczących oraz zebrano szereg wyników uzyskanych przez wykorzystanie wcześniej sformułowanej metodyki postępowania. Zasadniczą zaletą proponowanej metody jest właściwe potraktowanie problemu samokorekcji. Rezultaty obliczeń zebrane w tej części pracy mogą znaleźć bezpośrednie wykorzystanie w praktyce projektowania urządzeń cyfrowych. Jako takie można uważać optymalne struktury liczników oraz rezultaty dotyczące sprzężeń zwrotnych dla rejestrów liczących generujących sekwencje pseudoprzypadkowe

Р е з ю м е

Работа посвящена сдвиговым регистрам с обратной связью, называемым счётными регистрами. Сделано анализ двойных последовательностей генерированных счётными регистрами, указано, как реализовать счётные регистры, которые генерируют требуемые последовательности и представлено эффективный метод проектирования счётных регистров с учётом проблемы самокоррекции последовательностей, которая генерируется этими регистрами.

Первая часть работы посвящена анализу последовательностей генерированных счётными регистрами. Введено способы записи действия счетных регистров, а также методы вычисления кодовых последовательностей. Это позволило осуществить ряд вычислений, которых результаты дают полное изображение о счётных регистрах генерирующих последовательности, которых длина не больше 16 состоянии.

Во второй части работы представлено метод проектирования счётных регистров, а также собрано ряд результатов полученных благодаря использованию раньше сформулированных методов. Основное преимущество метода это соответственное решение проблемы самокоррекции. Результаты вычислений собранные в этой части работы могут найти применение в практике проектирования цифровых устройств. Такими можно считать оптимальные структуры счётчиков,а также результаты относящиеся к обратным связям для счётных регистров генерирующих псевдослучайные последовательности.

S u m m a r y

In the paper the shift registers with feedback (so called counter-registers) are discussed. The binary sequences generated by the counter-registers have been analyzed. It has been shown how to realise the counter register which generates a given binary sequence and how to design the counter-register with self-starting feature.

In the first part of the paper the sequences generated by the counter-registers are analyzed. The description methods of counter-register operation and the calculation methods of states sequences have been introduced. They allowed to make some calculations whose results give the complete picture of counter-registers generating sequences not longer then 16 states.

In the second part of the paper the designing method of counter-register has been shown. This part comprises several results derived from application of the designing method discussed earlier. The main merit of the method is the self-starting feature of the counter-registers. The results collected in that part may find direct applications in the design of digital circuits; the minimal structures of counters and the feedbacks for pseudonoise generators are the examples.

BIBLIOTEKA GŁÓWNA
Politechniki Śląskiej
P 3341/77/39

WYDAWNICTWA NAUKOWE I DYDAKTYCZNE POLITECHNIKI ŚLĄSKIEJ MOŻNA NABYĆ W NASTĘPUJĄCYCH PLACÓWKACH:

Gliwice — Księgarnia Techniczna, ul. M. Strzody 14b;
Gliwice — Punkt sprzedaży Studenckiej Spółdzielni Pracy, ul. Wrocławska 4/6;
Katowice — Księgarnia nr 004, ul. Warszawska 11;
Katowice — Księgarnia nr 015, ul. Żwirki i Wigury 33;
Chorzów — Księgarnia nr 063, ul. Wolności 22;
Dąbrowa Górnicza — Księgarnia nr 081, ul. ZBoWiD-u 2;
Rybnik — Księgarnia nr 126, Rynek 1;
Warszawa — Ośrodek Rozpowszechniania Wydawnictw PAN, Pałac Kultury i Nauki.

Wszystkie wydawnictwa naukowe i dydaktyczne zamawiać można poprzez Składnicę Księgarską w Warszawie, ul. Mazowiecka 9.