Dan NICULA

ELECTRONICĂ DIGITALĂ
Carte de învăţătură 2.0

Editura Universităţii *TRANSILVANIA* din Braşov
ISBN 978-606-19-0563-8

2015

Lecţia 17

Registre şi numărătoare

17.1 Noţiuni teoretice

Extensia paralel a unui bistabil poartă numele de **registru**. Simbolul bloc şi structura unui registru de 4 biţi implementat cu bistabile D sunt prezentate în figura 17.1.

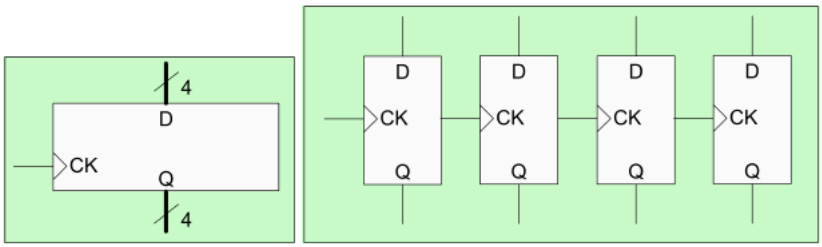

Figura 17.1 Registru D: simbol bloc şi structură.

În funcţie de modalitatea de încărcare şi accesare a datelor, registrele se clasifică astfel:

- **Registru paralel-paralel**. Datele sunt încărcate în registru şi accesate din exterior în paralel. Faţă de registrul "clasic", registrul paralel-paralel poate dispune de o intrare suplimentară care validează încărcarea doar când semnalul de validare este activ (în acest mod, registrul poate "memora" datele în tot intervalul de timp dintre două pulsuri de încărcare).

  Tabelul de funcţionare este:

LD	Q^+	Acţiune
1	D	Încarcă starea de pe intrarea de date
0	Q	Păstrează starea curentă

  Structura internă (figura 17.2) prezintă o buclă între ieşirea şi intrarea registrului intern peste un multiplexor prin care se selectează starea viitoare: se încarcă starea din exterior ($LD = 1$) sau se menţine starea curentă ($LD = 0$).

  Registrul este utilizat pentru stocarea datelor reprezentate pe mai mulţi biţi între două activări succesive ale semnalului de încărcare (încărcarea nu trebuie să se facă la fiecare front activ al ceasului, ca în cazul registrului clasic).

- **Registru paralel-serie (de serializare)**. Datele sunt încărcate în registru în paralel dar sunt accesate din exterior în mod serial. Sunt necesare două intrări de control pentru determinarea încărcării datelor şi a deplasării acestora în registrul intern (pentru deplasarea serială a datelor pe poziţia accesibilă din exterior).

  Tabelul de funcţionare este:

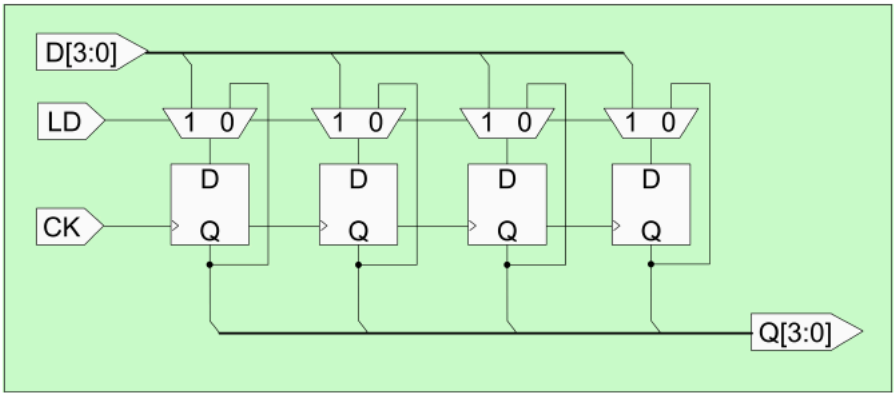

Figura 17.2 Registru paralel-paralel cu validare a încărcării.

LD	SHR	Q^+	Acţiune
1	X	D	Încarcă în paralel starea de pe intrarea de date
0	1	$\{S_I, Q >> 1\}$	Deplasează conţinutul registrului cu un bit
0	0	Q	Păstrează starea curentă

Structura internă (figura 17.3) prezintă o buclă între ieşirea şi intrarea registrului intern peste un multiplexor prin care se selectează starea viitoare: se încarcă starea din exterior ($LD = 1$), se deplasează datele din interior ($LD = 0, SHR = 1$) sau se menţine starea curentă ($LD = 0, SHR = 0$). Ieşirea serială de 1 bit provine de la cel mai puţin seminificativ bit al registrului.

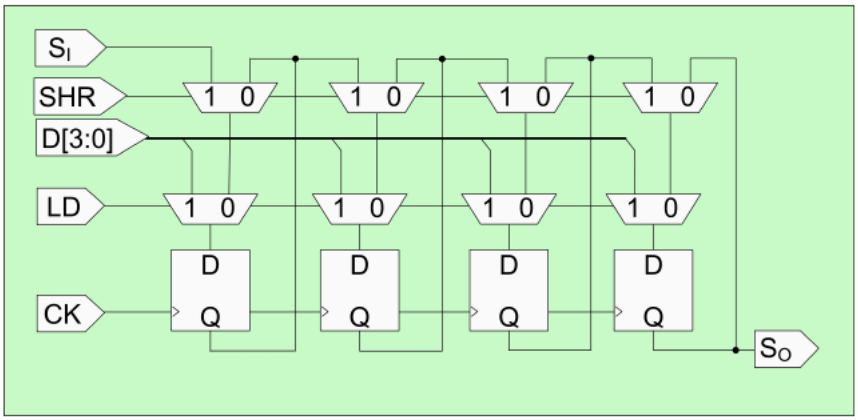

Figura 17.3 Registru paralel-serie (de serializare).

Aplicaţia clasică a registrului paralel-serie constă în serializarea datelor reprezentate pe mai mulţi biţi în vederea transmiterii acestora pe un singur fir. În acest caz, un impuls de încărcare $LD = 1$ este urmat de un număr de impulsuri de deplasare $SHR = 1$ egal cu numărul de biţi ai registrului intern.

- **Registru serie-paralel (de deserializare)**. Datele sunt încărcate în registru în mod serial, dar sunt accesate din exterior în mod paralel. Este necesară o intrare de control pentru determinarea deplasării datelor în registrul intern (pentru avansarea serială a datelor acceptate în vederea completării cuvântului de date accesibile din exterior în paralel). Registrul are o intrare de date de un singur bit. În cazul activării încărcării datelor, în poziţia cea mai semnificativă se încarcă datele de intrare S_I şi deplasează datele interne cu 1 bit. Cel mai puţin semnificativ bit se pierde.

  Tabelul de funcţionare este:

SHR	Q^+	Acţiune
1	$\{S_I, Q >> 1\}$	Încarcă datele de intrare (1 bit) şi deplasează conţinutul
0	Q	Păstrează starea curentă

Structura internă (figura 17.4) prezintă o buclă între ieşirea şi intrarea registrului intern peste un multiplexor prin care se selectează starea viitoare: se încarcă bitul din exterior şi se deplasează datele ($SHR = 1$), sau se menţine starea curentă ($SHR = 0$). Ieşirea paralelă este accesibilă din exterior în orice moment.

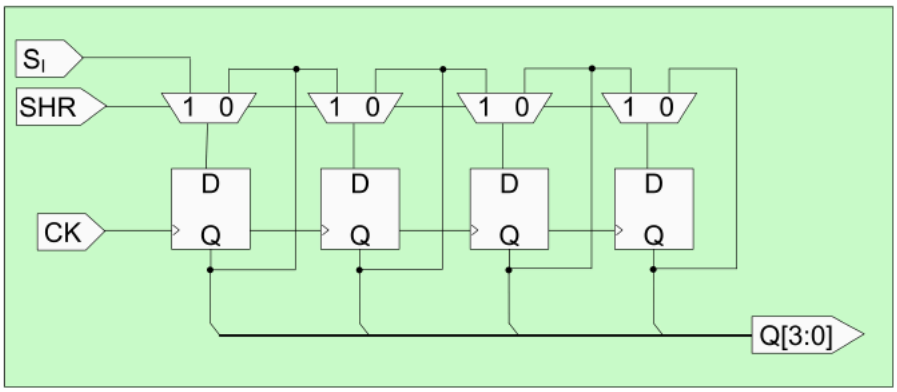

Figura 17.4 Registru serie-paralel (de deserializare).

Aplicaţia clasică a registrului serie-paralel constă în deserializarea datelor reprezentate pe mai mulţi biţi dar recepţionate serial. În acest caz, modulul de control extern trebuie să asigure preluarea datelor în paralel după un număr de impuls de încărcare $SHR = 1$ egal cu numărul de biţi.

- **Registru serie-serie (de deplasare)**. Deşi în interior există un registru pe mai mulţi biţi, datele sunt încărcate şi sunt accesate serial, câte un bit pe tact. Registrul realizează o deplasare internă a datelor. Este necesară o intrare de control pentru determinarea încărcării datelor şi a deplasării acestora în registrul intern (pentru deplasarea serială a datelor acceptate). Registrul are o intrare de date de un singur bit. În cazul activării încărcării datelor, în poziţia cea mai semnificativă se încarcă datele de intrare S_I şi deplasează datele interne cu 1 bit. Cel mai puţin semnificativ bit se pierde. Ieşirea serială de 1 bit provine de la cel mai puţin semnificativ bit al registrului.

  Registrul se comportă ca registrul serie-paralel cu deosebirea că la registrul serie-serie doar un bit (cel mai puţin semnificativ) este accesibil din exterior ($Q_{[0]}$).

  Tabelul de funcţionare este:

SHR	Q^+	Acţiune
1	$\{S_I, Q >> 1\}$	Încarcă datele de intrare (1 bit) şi deplasează conţinutul
0	Q	Păstrează starea curentă

Structura internă (figura 17.5) prezintă o buclă între ieşirea şi intrarea registrului intern peste un multiplexor prin care se selectează starea viitoare: se încarcă bitul din exterior şi se deplasează datele ($SHR = 1$), sau se menţine starea curentă ($SHR = 0$). Ieşirea serială (cel mai puţin semnificativ bit) este accesibilă din exterior în orice moment.

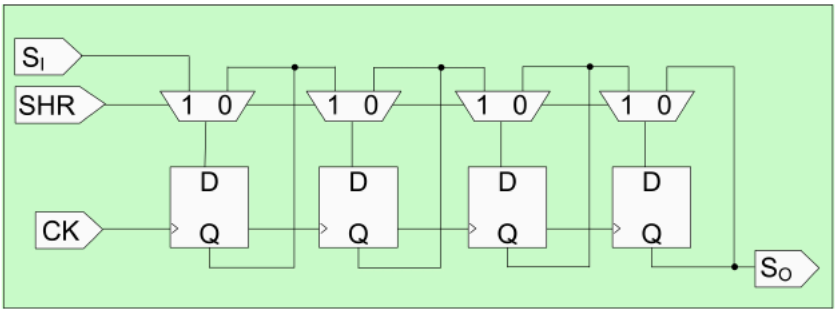

Figura 17.5 Registru serie-serie (de deplasare).

Aplicaţia clasică a registrului serie-serie constă în stocarea datelor sub forma unei "cozi de aşteptare" care avansează doar la comandă ($SHR = 1$).

© 2015 Dan NICULA - dan.nicula@unitbv.ro
www.DanNicula.ro/ed_ci

Un registru poate avea intrări suplimentare de **set** sau **reset asincrone** (care acţionează prioritar şi independent de semnalul de ceas).

Numărătoarele sunt circuite logice secvenţiale (automate) care îşi determină starea următoarea exclusiv pe baza stării prezente printr-o funcţie de numărare. Structura "clasică" a unui numărător sincron (figura 17.6) prezintă un sumator conectat în bucla între ieşirea şi intrarea unui registru de stare. Numărătoarele pot avea diverse intrări care controlează funcţia de numărare (direcţia de numărare, condiţia de numărare, intervalul de numărare, codul de numărare).

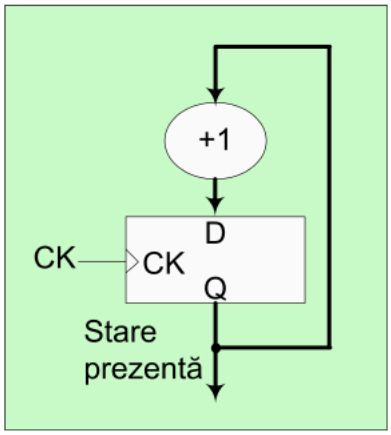

Figura 17.6 Schema bloc a numărătoarelor sincrone.

Un tip particular de registre îl constituie circuitele **LFSR** (Engl. *"Linear Feedback Shift Register, Registru de deplasare cu reacţie liniară"*).

Registrul de deplasare cu reacţie liniară reprezintă o modalitate de obţinere a unei secvenţe de numărare lungi cu un număr mic de bistabile şi care să funcţioneze la frecvenţă mare. În structura unui LSFR de 4 biţi, 3 din cele 4 bistabile primesc pe intrare semnalul de la ieşirea bistabilului de ordin superior. Un singur bistabil, cel mai semnificativ, primeşte la intrare o funcţie logică XOR determinată pe baza unor biţi mai puţini semnificativi. Determinarea biţilor care intră în calculul valorii viitoare a bitului cel mai semnificativ se face pe baza unui polinom. Polinomul are gradul egal cu numărul de biţi ai LFSR şi prezintă indecşii biţilor care concură la determinarea valorii viitoare a celui mai semnificativ bit. De exemplu, dacă polinomul este $X^4 + X^3 + 1$, figura 17.7, valoarea viitoare a bitului cel mai semnificativ este determinată prin funcţia XOR între bitul 0 şi bitul 3. Se remarcă următoarele:
- Funcţia XOR realizează funcţia de adunare aritmetică a biţilor.
- Pentru N bistabile se obţine o secvenţă de numărarea de lungime maximă $2^N - 1$. Starea 000...0 este o stare de blocare.
- Nu orice polinom produce secvenţe de numărare de lungime maximă. În funcţie de polinomul ales, se pot obţine mai multe secvenţe de numărare de lungimi mai mici decât lungimea maximă.
- Un circuit LFSR poate fi considerat un **generator de numere pseudo-aleatorii**, dacă are o secvenţă de numărare foarte lungă (număr mare de biţi) şi se utilizează un număr mic de eşantioane succesive.

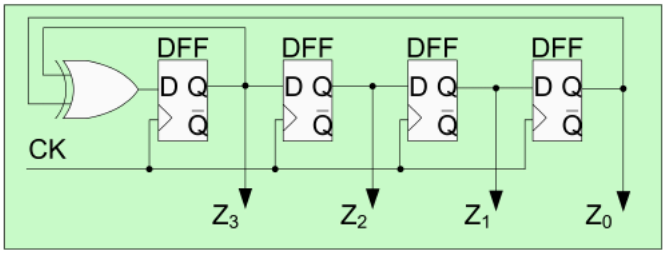

Figura 17.7 Circuit LFSR pe baza polinomului $X^4 + X^3 + 1$.

© 2015 Dan NICULA - dan.nicula@unitbv.ro
www.DanNicula.ro/ed_ci

17.2 Pentru cei ce vor doar să promoveze examenul

1. Un registru de 4 biţi având iniţial conţinutul 1011 este deplasat spre stânga secvenţial, primind pe intrarea de date serială secvenţa 1010111. Precizaţi conţinutul registrului după fiecare front activ de ceas.

   Soluţie

   Starea viitoare a biţilor registrului este:
   $Q_3^+ = Q_2$
   $Q_2^+ = Q_1$
   $Q_1^+ = Q_0$
   $Q_0^+ = In$

   Conţinutul registrului după fiecare front activ de ceas, în funcţie de intrarea de date, este redat în tabelul:

Tact	Intrare	Stare registru			
	In	Q_3	Q_2	Q_1	Q_0
1	1	1	0	1	1
2	0	0	1	1	1
3	1	1	1	1	0
4	0	1	1	0	1
5	1	1	0	1	0
6	1	0	1	0	1
7	1	1	0	1	1
8	-	0	1	1	1

2. Un registru serial de 4 biţi se află iniţial în starea 0001 şi primeşte pe intrarea serială secvenţa 101110011 (cel mai din stânga bit primul). Care este starea registrului după fiecare front de ceas, în cazul unei deplasări spre dreapta?

3. Proiectaţi cu porţi şi bistabile un numărător sincron de 4 biţi.

17.3 Pentru cei ce vor să înveţe

1. Determinaţi secvenţa de numărare a circuitului din figura 17.8, indiferent de starea iniţială.

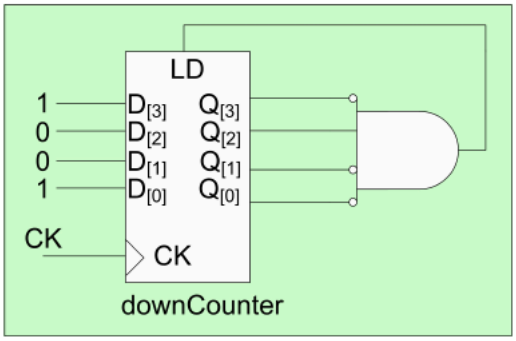

Figura 17.8 Numărător, problema 1.

   Soluţie

   Circuitul este un numărător în sens descrescător cu facilitate de presetare. Condiţia de preset este activă în starea $Q_{[3:0]} = 0100|_2 = 4|_{10}$ (conform legăturilor porţii AND). Starea de presetare este $D_{[3:0]} = 1001|_2 = 9|_{10}$ (conform valorilor logice aplicate pe intrările D).
   Rezultă că circuitul va cicla între stările $9, 8, 7, 6, 5, 4$. Dacă numărătorul se va afla în celelalte stări, condiţia de presetare va fi inactivă, deci numărătorul va număra în sens descrescător.
   Rezultă secvenţa: $3, 2, 1, 0, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 9, 8, ...$ Se remarcă faptul că stările care nu aparţin ciclului de numărare ar putea să apară doar după iniţializare (figura 17.9).

© 2015 Dan NICULA - dan.nicula@unitbv.ro
www.DanNicula.ro/ed_ci

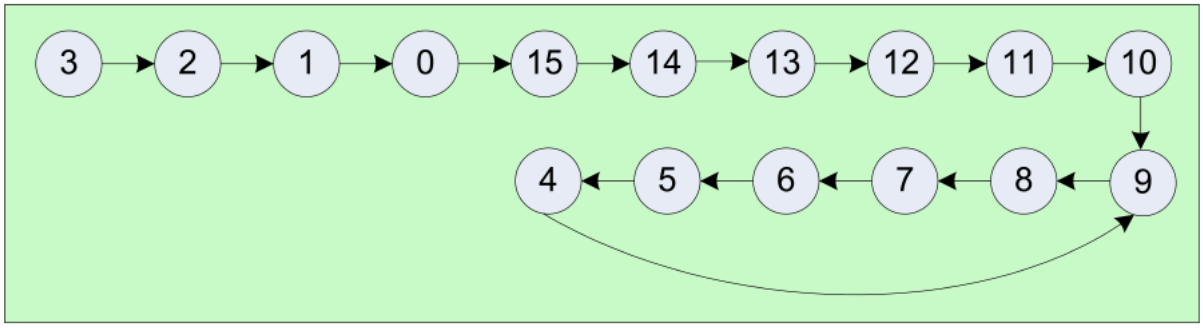

Figura 17.9 Graf de tranziţii al numărătorului, problema 1.

2. Completaţi diagramele temporale asociate circuitului prezentat în figura 17.10-a.

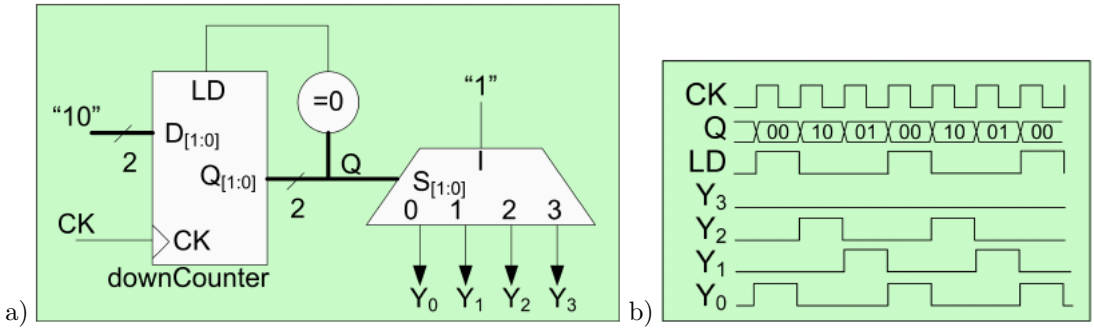

Figura 17.10 a) Circuit propus la problema 2. **b)** Forme de undă ale circuitului.

Soluţie

Se observă că numărătorul în sens descrescător se încărcă cu valoarea binară $10|_2$ de fiecare dată când ajunge în starea $00|_2$. Deci, secvenţa de numărare este: $00, 10, 01, 00, 10, 01, ...$ Ieşirea numărătorului este conectată la selecţia demultiplexorului, ceea ce înseamnă că vor fi activate ciclic ieşirile Y_2, Y_1 şi Y_0. Formele de undă ale semnalelor sunt prezentate în figura 17.10-b.

3. Să se determine secvenţa de numărare a circuitelor prezentate în figura 17.11. Presupuneţi starea iniţială 0. Studiaţi cum evoluează circuitele pornind din toate stările.

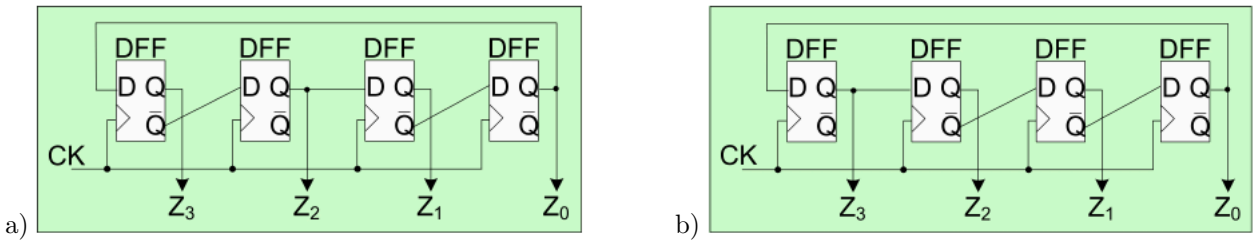

Figura 17.11 Circuitele de numărare propuse la problema 3.

4. Proiectaţi cu porţi şi bistabile un registru de 4 biţi cu facilităţi de preîncărcare (dacă $LD = 1$) şi deplasare stânga (dacă $LD = 0$ şi $SH = 1$). Dacă intrările de control sunt inactive ($LD = SH = 0$) circuitul îşi păstrează starea.

Soluţie

Circuitul este format din 4 bistabile. La intrarea acestora se află circuite multiplexor care selectează datele încărcate pe baza intrărilor de selecţie. Intrarea LD fiind mai prioritară, se aplică pe multiplexorul mai apropiat bistabilului. Circuitul rezultat este prezentat în figura 17.12.

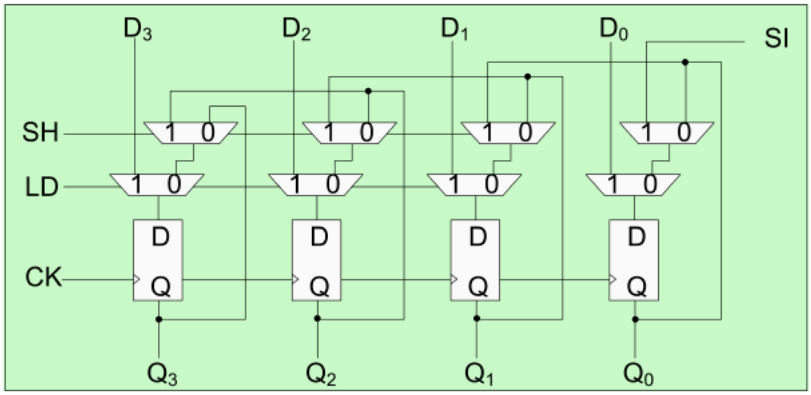

Figura 17.12 Circuit propus la problema 4.

5. Proiectaţi cu multiplexoare şi bistabile un registru de 4 biţi cu facilităţile prezentate în tabelul următor:

S_1	S_0	Operaţia
0	0	păstrează valoarea curentă
0	1	deplasare stânga cu încărcare serială
1	0	deplasare dreapta cu încărcare serială
1	1	încărcare paralelă

6. Proiectaţi un circuit de generare a unui grup de semnale cu 4 faze, perioadă de 50 ns şi factor de umplere 1/2 pe baza unui semnal cu frecvenţă de 80 MHz.

   Soluţie

   Perioada ceasului de frecvenţă 80 MHz este 1/80 MHz=12, 5 ns. Perioada semnalului generat este de 4 ori mai mare (50 ns /12, 5 ns = 4). Deci, o perioadă de semnal generat este egală cu 4 perioade de ceas.

   Soluţia de implementare conţine un numărător de 2 biţi, modulo 4, care repetă periodic 4 faze. Cele 4 semnale se pot genera cu un circuit combinaţional pe baza stării numărătorului conform tabelului de adevăr:

Fază	Q_1Q_0	O_0	O_{90}	O_{180}	O_{270}
0	00	0	1	1	0
1	01	0	0	1	1
2	10	1	0	0	1
3	11	1	1	0	0

   $O_0 = Q_1$
   $O_{90} = \overline{Q_1 \oplus Q_0}$
   $O_{180} = \overline{Q_1}$
   $O_{270} = Q_1 \oplus Q_0$

   Implementarea circuitului este prezentată în figura 17.13-a. Acest circuit are dezavantajul că ieşirile sunt generate de un circuit combinaţional. Soluţia de a genera ieşirile direct din bistabile se poate implementa fără porţi logice suplimentare obsevând că $O_0 = Q_1$ (bitul cel mai semnificativ al numărătorului) iar celelate 3 faze se obţin din întârzierea acestuia printr-o serie de 3 bistabile D (figura 17.13-b). Se remarcă faptul că, pentru a se genera secvenţele corecte imediat după semnalul de reset, bistabilele trebuie iniţializate la valorile menţionate în figură: numărătorul la 00, iar bistabilele la 110.

7. Proiectaţi un registru de 4 biţi cu facilităţi de resetare sincronă utilizând 4 bistabile D şi 4 circuite MUX 2:1.

8. Proiectaţi un numărător de 16 biţi cu încărcare paralelă utilizând 4 numărătoare de 4 biţi cu interfaţa descrisă în continuare:

© 2015 Dan NICULA - dan.nicula@unitbv.ro
www.DanNicula.ro/ed_ci

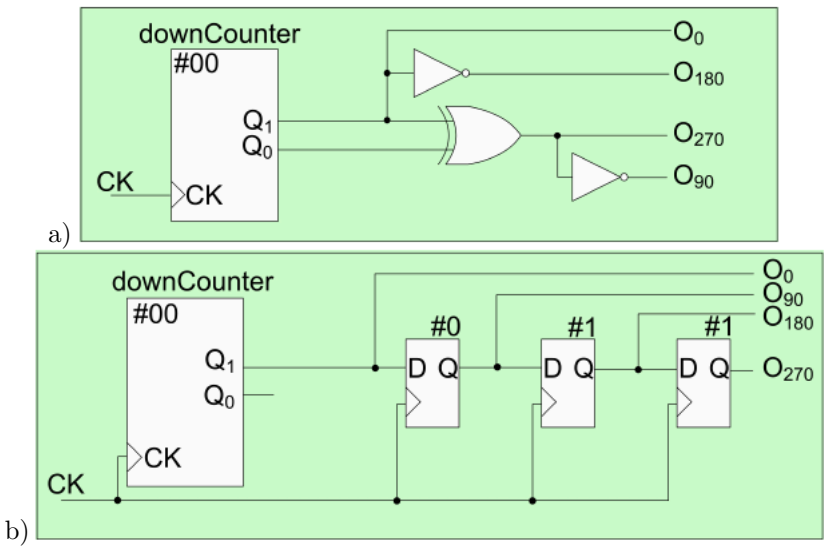

Figura 17.13 Generator de 4 faze: **a)** implementare cu ieşiri din circuit combinaţional, **b)** implementare cu întârziere prin bistabile.

Port	Direcţie	Dimensiune	Semnificaţie
clk	in	1	intrare de ceas
reset	in	1	intrare de reset asincron
countUp	in	1	intrare de control, activă în 1, numărare în sens crescător
load	in	1	intrare de control, încărcare, activă în 1, prioritară faţă de *countUp*
din	in	4	intrare de date
dout	out	4	ieşire de date
co	out	1	transport de ieşire, activ în 1 în cazul numărării din 15 în 0

9. Proiectaţi un circuit secvenţial care generează şase semnale periodice cu faze diferite, cu formele de undă prezentate în figura 17.14. Proiectaţi circuitul:
a) doar cu bistabile D,
b) cu numărător şi decodificator.
Comparaţi cele două soluţii de implementare.

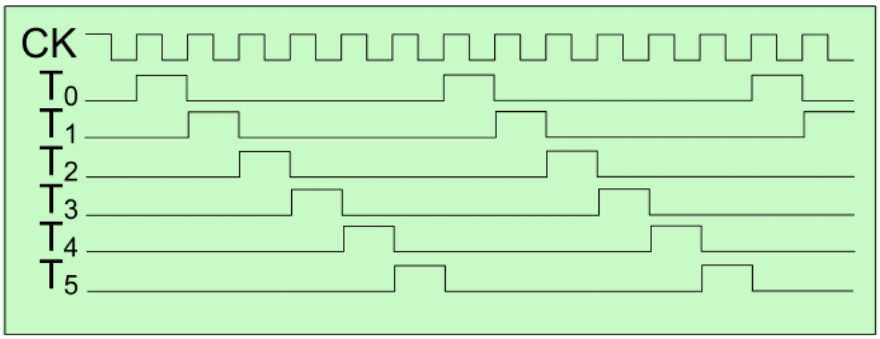

Figura 17.14 Semnale posibil de utilizat ca semnale de ceas multi-fază, problema 9.

Soluţie

Din formele de undă prezentate în figura 17.14 se observă că cele 6 semnale au aceeaşi formă de undă şi diferă doar prin fază (T_i este identic cu T_{i-1}, dar este întârziat cu o perioadă de ceas).

a) Implementarea cu bistabile D se bazează pe o structură de registru de deplasare stânga în inel, cu 6 bistabile, iniţializate cu valoarea 000001. Circuitul este prezentat în figura 17.15-a.

Dacă bistabilele au doar facilitate de reset (se pot aduce iniţial, asincron, doar în starea 0), atunci se poate adăuga un circuit combinaţional care din starea 000000 va conduce circuitul într-o stare dorită. Circuitul din

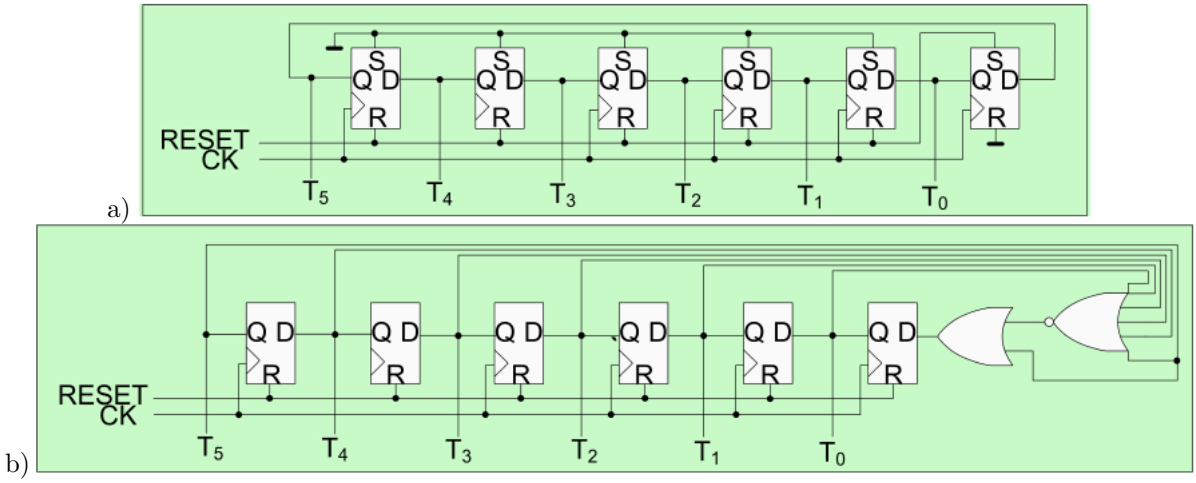

Figura 17.15 Registru de deplasare în inel **a)** iniţializat cu 000001, **b)** cu iniţializare în starea 000000 şi comutare în starea 000001.

figura 17.15-b este un registru de deplasare stânga în inel cu iniţializare în starea 000000 şi comutare în starea 000001.

b) Aceleaşi forme de undă pot fi generate de către un numărător modulo 6 şi un decodificator 3:8. Numărătorul va număra ciclic de la 0 la 5. Ieşirea numărătorului este conectată la intrarea decodificatorului. La ieşirile decodificatorului se obţin semnalele cerute. Circuitul este prezentat în figura 17.16.

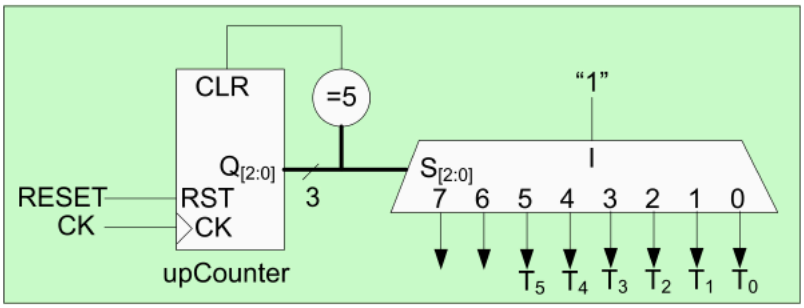

Figura 17.16 Generator de faze implementat cu numărător şi decodificator.

Prima abordare necesită 6 bistabile şi nu necesită porţi logice suplimentare (în cazul bistabilelor cu intrări asincrone de set şi reset). Ieşirile circuitului sunt direct din bistabile (recomandat).
A doua schemă utilizează un numărător de 3 biţi, deci s-ar putea spune că sunt necesare doar 3 bistabile. Însă, în acest caz este necesară logica pentru numărător (circuit de incrementare de 3 biţi, circuit de detecţie a stării 000 şi decodificatorul). În plus, în acest caz, ieşirile sunt din circuit combinaţional (nerecomandat).

Concluzia este că, deşi a două schemă pare mai directă ca rezolvare a problemei, prima soluţie este atât mai ieftină ca resurse hardware cât şi mai rapidă (ca frecvenţă maximă de operare).

10. Un numărător în inel (Engl. *"ring counter"*) utilizează N bistabile pentru a obţine un ciclu de numărare de N stări. Structura acestui numărător, figura 17.17, prezintă N bistabile D, fiecare având ieşirea conectată la intrarea următorului bistabil. De obicei, numărătorul este iniţializat cu o valoare având un singur bit egal cu 1.

    a) Listaţi secvenţa de stări a numărătorului în inel pornind de la starea iniţială 0001.
    b) Listaţi celelalte stări ale numărătorului în inel şi precizaţi evoluţia acestuia în ipoteza apariţiei acestora.
    c) Modificaţi circuitul astfel încât, în ipoteza unei stări din afara ciclului dorit, numărătorul să revină în ciclul proiectat.

11. Proiectaţi structurile LFSR pe baza polinoamelor: $X^4 + X^3 + 1$, $X^2 + X + 1$, $X^5 + X^3 + 1$, $X^6 + X^5 + 1$, $X^9 + X^5 + 1$. Studiaţi comportamentul circuitului la apariţia stării de blocare (toţi biţii egali cu 0). Propuneţi un circuit care să iniţializeze circuitul într-o stare validă. Listaţi secvenţa de stări prezentată de circuite. Pentru

© 2015 Dan NICULA - dan.nicula@unitbv.ro
www.DanNicula.ro/ed_ci

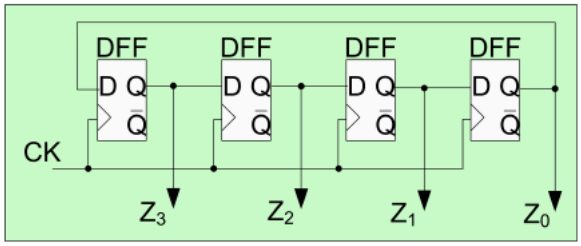

Figura 17.17 Numărător în inel de 4 biţi, problema 10.

ultimul circuit, folosiţi calculatorul pentru determinarea secvenţei.

Soluţie

Circuitele LFSR au o structură de registru de deplasare: intrarea unui bistabil asociat unui bit provine de la ieşirea bistabilului asociat bit-ului mai semnificativ. Intrarea celui mai semnificativ bit provine de la un circuit logic combinaţional implementat cu o poartă XOR cu intrările conectate la ieşirile bistabilelor, conform polinomului. Numărul de bistabile este egal cu puterea cea mai mare a variabilei în polinom. Circuitele LFSR asociate polinoamelor sunt prezentate în figura 17.18.

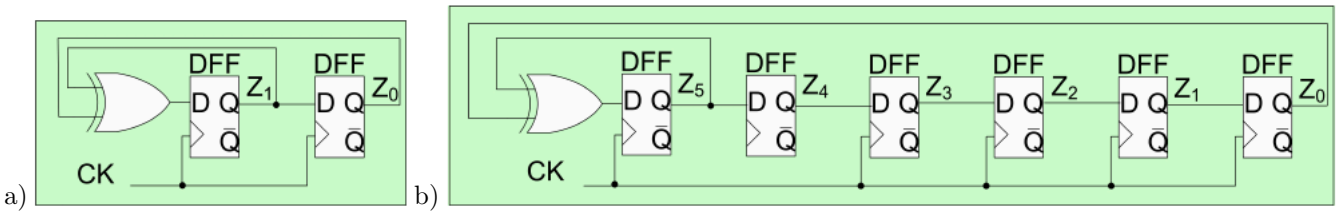

Figura 17.18 Circuit LFSR pe baza polinomelor: **a)** X^2+X+1, **b)** X^6+X^5+1 .

În cazul în care toate bistabile se află în starea 0, starea viitoare va fi aceeaşi. Circuitul nu va mai putea părăsi această stare. Pentru iniţializarea circuitului într-o stare validă, cel puţin un bistabil va trebui adus în starea 1 printr-un semnal extern de reset. Secvenţa generată de circuitul LFSR pe baza polinomului X^2+X+1 este:

nr.	$Z_{[1:0]}$
1	01
2	10
3	11

Secvenţa generată de circuitul LFSR pe baza polinomului X^5+X^3+1 este:

nr.	$Z_{[4:0]}$	nr.	$Z_{[4:0]}$	nr.	$Z_{[4:0]}$	nr.	$Z_{[4:0]}$
1	00001	9	01110	17	11100	25	10110
2	10000	10	10111	18	11110	26	01011
3	01000	11	11011	19	11111	27	00101
4	10100	12	01101	20	01111	28	10010
5	01010	13	00110	21	00111	29	01001
6	10101	14	00011	22	10011	30	00100
7	11010	15	10001	23	11001	31	00010
8	11101	16	11000	24	01100		

12. Studiaţi comportamentul circuitelor LFSR caracterizate de polinoamele: X^3+X^2+1, X^4+X^2+1, X^5+X^2+1.

13. Implementaţi cu bistabile D/T/JK un numărător modulo 6/10/16 cu intrare de validare.
Implementaţi acelaşi numărător utilizând un numărător de 4 biţi cu presetare.
Implementaţi un circuit capabil să numere modulo 6 sau 10 sau 16, numărul fiind stabilit pe baza unui cod de 2 biţi.

© 2015 Dan NICULA - dan.nicula@unitbv.ro
www.DanNicula.ro/ed_ci

14. Implementaţi automatele descrise de organigramele din figura 17.19 cu numărătoare presetabile.

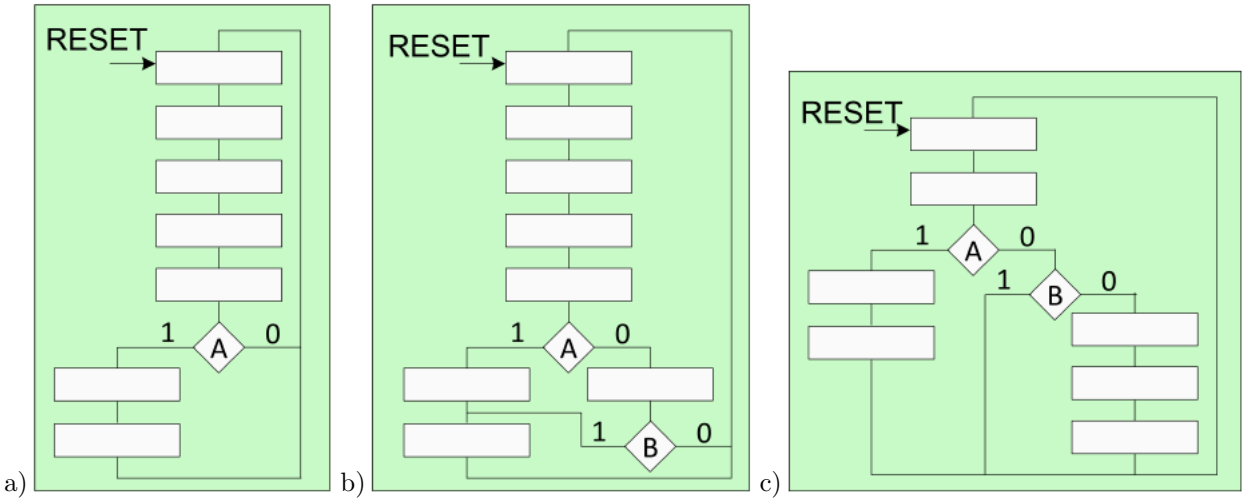

Figura 17.19 Organigrame ce descriu automate recomandate a fi implementate cu numărător.

Soluţie

Registrul de stare este implementat cu un numărător. Numărătorul are două facilităţi: să determine prin incrementare starea următoare sau să fie încărcat cu o anumită stare din exterior. Pe baza organigramei, se determină funcţia executată de numărător în fiecare stare: numărare sau încărcare. În cazul încărcării, trebuie asigurată existenţa pe intrările de date ale numărătorului a codului stării următoare. Structura automatului implementat cu numărător este prezentată în figura 17.20.

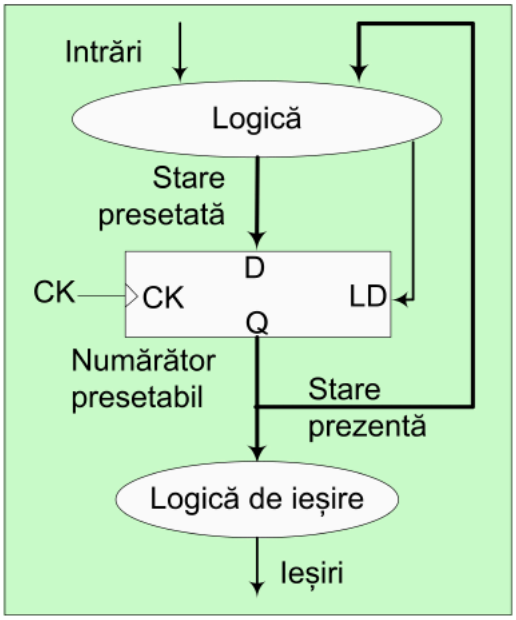

Figura 17.20 Structură de automat implementat cu numărător.

Beneficiile implementării cu numărător sunt vizibile în cazul organigramelor care au majoritatea tranziţiilor necondiţionate între stări cu coduri ordonate în sensul de numărare. În acest caz, circuitul care determină logica semnalului de încărcare şi a valorilor de presetare va fi de mici dimensiuni.

15. Implementaţi cu bistabile D şi multiplexoare un registru care să funcţioneze conform tabelului:

© 2015 Dan NICULA - dan.nicula@unitbv.ro
www.DanNicula.ro/ed_ci

SH	LD	Operaţie
1	-	deplasare stânga
0	1	încărcare paralelă
0	0	păstrează starea

16. Implementaţi cu bistabile D şi multiplexoare un registru care să funcţioneze conform tabelului:

S_1	S_0	Operaţie
0	0	păstrează starea
0	1	resetare sincronă
1	0	deplasare dreapta
1	1	încărcare paralelă

17. Implementaţi cu bistabile D şi multiplexoare un registru care să funcţioneze conform tabelului:

EN	LD	SH	CNT	SENS	Operaţie
0	-	-	-	-	păstrează starea
1	1	-	-	-	încărcare paralelă
1	0	1	-	1	deplasare dreapta
1	0	1	-	0	deplasare stânga
1	0	0	1	1	decrementare
1	0	0	1	0	incrementare
1	0	0	0	-	păstrează starea

18. Proiectaţi un circuit LFSR pe baza unui polinom parametrizabil.

    Soluţie

    Circuitul este format dintr-un registru de deplasare. Intrarea celui mai semnificativ bit se determină pe baza stării curente şi a polinomului caracteristic. Coeficienţii polinomului sunt intrări în circuit. Fiecare bit de stare intră în conjuncţie cu coeficientul cu indice corespunzător.

19. Utilizaţi un numărător presetabil de 4 biţi pentru a implementa un numărător BCD (între 0 şi 9).

20. Justificaţi valoarea de adevăr a fiecărei afirmaţii:
    a) Un numărător de 4 biţi are 8 stări valide.
    b) Un circuit LFSR de 4 biţi are întotdeauna un ciclu de lungime 15 şi o stare de blocare.

    Soluţie

    a) Un numărător de 4 biţi are $2^4 = 16$ stări valide, deci afirmaţia este falsă.

21. Care este numărul minim de bistabile necesar pentru implementarea unui numărător modulo 10^4?
    a) 10
    b) 12
    c) 14
    d) 15
    e) 16

    Soluţie

    $10^4 = 10000 < 16384 = 2^{14}$, rezultă ca sunt necesari 14 biţi pentru a codifica cele 10^4 stări. Deci, răspunsul corect este **c)**.

22. Care este deosebirea de implementare a unui reset asincron faţă de un reset sincron?

23. Care este numărul maxim de stări într-un ciclu pentru un circuit LFSR de 10 biţi?

24. Câte bistabile trebuie să comute la schimbarea stării unui numărător dacă starea curentă este:
    a) 01101111, **b)** 01101101, **c)** 01011111, **d)** 01111110.

© 2015 Dan NICULA - dan.nicula@unitbv.ro
www.DanNicula.ro/ed_ci

17.4 Pentru cei ce vor să devină profesionişti

Registru paralel (cod Verilog)

```
input[7:0] Dff;
reg[7:0] Qff;
always @(posedge ck)
Qff <= Dff;
```

Registru paralel-paralel cu intrare de încărcare (cod Verilog)

```
input[7:0] Dff;
reg[7:0] Qff;
always @(posedge ck)
if (load)  Qff <= Dff;
```

Registru paralel cu reset asincron (cod Verilog)

```
input[7:0] Dff;
reg[7:0] Qff;
always @(posedge ck or posedge resetAsync)
if (resetAsync) Qff <= 'b0; else
Qff <= Dff;
```

Registru paralel cu reset sincron (cod Verilog)

```
input[7:0] Dff;
reg[7:0] Qff;
always @(posedge ck)
if (resetSync) Qff <= 'b0; else
Qff <= Dff;
```

Registru paralel cu reset sincron şi asincron (cod Verilog)

```
input[7:0] Dff;
reg[7:0] Qff;
always @(posedge ck or posedge resetAsync)
if (resetAsync) Qff <= 'b0; else
if (resetSync) Qff <= 'b0; else
Qff <= Dff;
```

Registre paralel-serie (cod Verilog)

```
input[7:0] Dff;
reg[7:0] Qff;
always @(posedge ck or posedge resetAsync)
if (resetAsync) Qff <= 'b0; else
if (load) Qff <= Dff; else
if (shr) Qff <= (Qff >> 1);
assign serialOut = Qff[0];
```

Registre serie-paralel (cod Verilog)

```
input[7:0] Dff;
reg[7:0] Qff;
always @(posedge ck or posedge resetAsync)
if (resetAsync) Qff <= 'b0; else
if (load) Qff <= {Qff[6:0], serialIn};
```


© 2015 Dan NICULA - dan.nicula@unitbv.ro
www.DanNicula.ro/ed_ci

Registre serie-serie (cod Verilog)

```
input[7:0] Dff;
reg[7:0] Qff;
always @(posedge ck or posedge resetAsync)
if (resetAsync) Qff <= 'b0; else
if (load) Qff <= {Qff[6:0], serialIn};
assign serialOut = Qff[7];
```

1. Un numărător Johnson (Engl. *"Johnson counter"*) utilizează N bistabile pentru a obţine un ciclu de numărare de $2 \times N$ secvenţe. Structura acestui numărător, figura 17.21, este similară cu a numărătorului în inel, prezintă N bistabile D, fiecare având ieşirea conectată la intrarea următorului bistabil, cu excepţia unui bistabil care utilizează ieşirea inversată. De obicei, numărătorul este iniţializat cu o valoare având toţi biţii egali cu 0. La fiecare comutare, bitul mai puţin semnificativ este negat şi reintrodus la intrarea lanţului de bistabile.

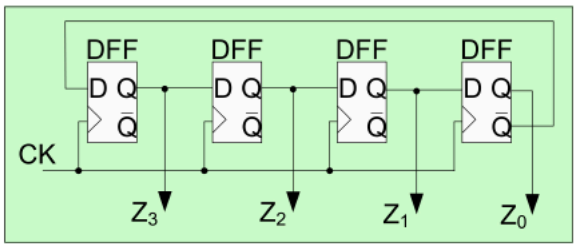

Figura 17.21 Numărător Johnson de 4 biţi, problema 1.

a) Listaţi secvenţa de stări a numărătorului în inel pornind de la starea iniţială 0000.
b) Listaţi celelalte stări ale numărătorului în inel şi precizaţi evoluţia acestuia în ipoteza apariţiei acestora.
c) Modificaţi circuitul astfel încât, în ipoteza unei stări din afara ciclului dorit, numărătorul să revină în ciclul proiectat (în următoarea stare sau într-un număr finit de stări). Evaluaţi soluţiile din punct de vedere al costului implementării.

Soluţie

a) Secvenţa de stări pe care numărătorul le parcurge pornind din starea 0000 este următoarea:
$0000, 1000, 1100, 1111, 1110, 0111, 0011, 0001, 0000, ...$
b) Din totalul de 16 stări posibile rămân 8 stări care nu fac parte din ciclul dorit al numărătorului:
$0010, 0100, 0101, 0110, 1001, 1010, 1011, 1101, 0010, ...$
Din starea 0010 evoluţia este următoarea:
$0010, 1001, 0100, 1010, 1101, 0110, 1011, 0101$

Se observă că, în cazul apariţiei oricărei stări din afara ciclului dorit, numărătorul va trece succesiv prin toate cele 8 stări şi nu va reveni la funcţionarea dorită.
c) Circuitul are 8 stări în ciclul dorit şi 8 stări într-un alt ciclu. Pentru a reveni la ciclul proiectat, în cazul apariţiei unei stări nedorite, circuitul trebuie extins cu logică adiţională. Dacă se doreşte ca numărătorul să revină la ciclul normal într-un singur tact trebuie să se detecteze toate stările ilegale şi să se forţeze ca următoarea stare să fie una din ciclul valid (de exemplu 0000). În tabelul de adevăr s-a notat $Clear1$ semnalul care detectează apariţia unei stări din afara ciclului. Dacă $Clear1 = 1$ starea următoare este 0000. Acest lucru este implementat cu circuitul din figura 17.22-a.
$Clear1 = Z_3 \cdot \overline{Z_1} \cdot Z_0 + Z_3 \cdot \overline{Z_2} \cdot Z_1 + \overline{Z_3} \cdot Z_2 \cdot \overline{Z_1} + \overline{Z_3} \cdot Z_1 \cdot \overline{Z_0}$

Dacă se doreşte ca numărătorul să revină la ciclul normal, dar nu neapărat într-un singur tact, va trebui să se detecteze o singură stare ilegală (de exemplu, 0010) şi să se forţeze ca următoarea stare să fie una din ciclul valid (de exemplu 0000). Revenirea se va obţine în maximum 8 tacte, presupunând că numărătorul este iniţial în starea 1001 şi parcurge tot ciclul alternativ. În tabelul de adevăr s-a notat $Clear$ semmnalul care detectează apariţia stări 1001 (din afara ciclului). Dacă $Clear = 1$ starea următoare este 0000. Acest lucru este implementat cu circuitul din figura 17.22-b.
$Clear = \overline{Z_3} \cdot \overline{Z_2} \cdot Z_1 \cdot \overline{Z_0}$

© 2015 Dan NICULA - dan.nicula@unitbv.ro www.DanNicula.ro/ed_ci

Z_3	Z_2	Z_1	Z_0	$Clear1$	$Clear$
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	1	1
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	1	0	0

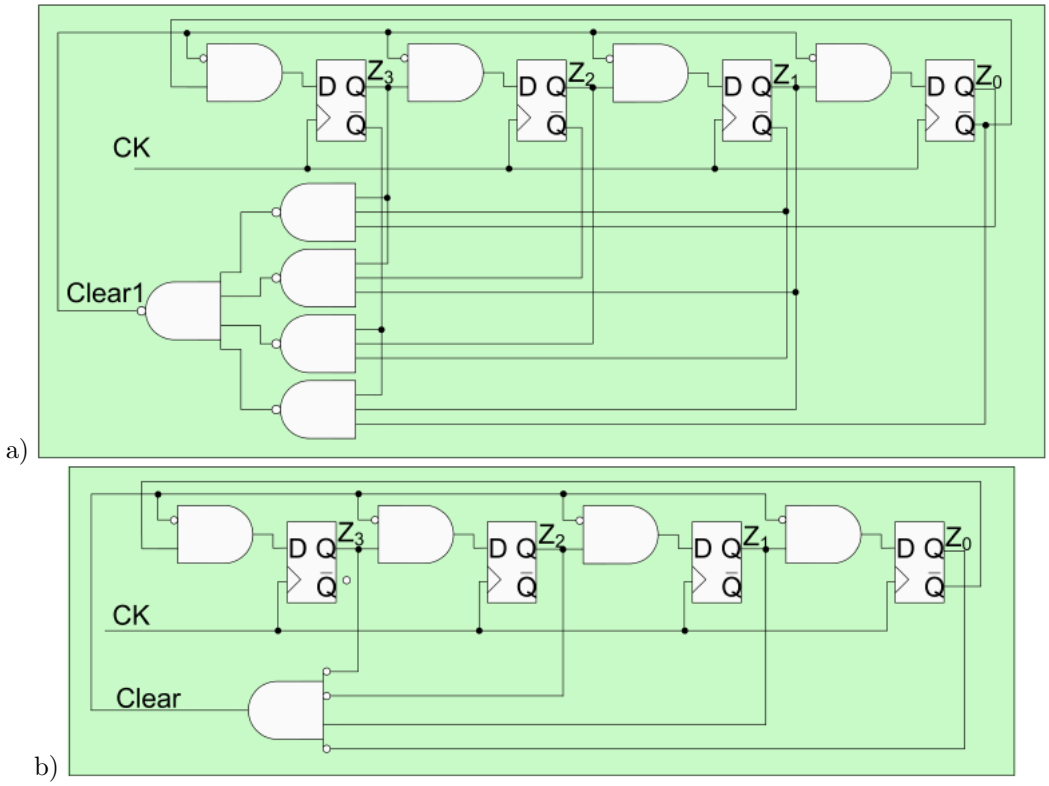

Figura 17.22 Numărător Johnson de 4 biţi cu revenire în cadrul ciclului **a)** într-un tact, **b)** în maximum 8 tacte (problema 1).

A doua soluţie necesită un circuit combinaţional mai simplu, însă determină revenirea în ciclul necesar în maxim 8 perioade de ceas. Prima soluţie permite revenirea în ciclul necesar într-o singură perioadă de ceas, însă utilizând o logică mai complexă (posibil cu o frecvenţă maximă mai joasă).

2. Utilizaţi un numărător presetabil de 4 biţi pentru a implementa un numărător între 3 şi 10.

3. Utilizaţi un numărător presetabil de 4 biţi pentru a implementa un numărător între două valori *min* şi *max* parametrizabile. Valorile *min* şi *max* sunt intrări de câte 4 biţi pentru circuitul proiectat.

4. Circuitul din figura 17.23 este un numărător de 4 biţi reversibil, presetabil a cărui funcţionare este descrisă de tabelul următor:

LD	EN	UP	Q^+	Acţiune
1	X	X	D	Încărcare
0	1	1	Q+1	Incrementare
0	1	0	Q-1	Decrementare
0	0	X	Q	Menţinere

C_O (Carry Output) este o ieşire activă în 1 generată combinaţional:
- când valoarea curentă este 15 şi numărătorul se incrementează şi
- când valoarea curentă este 0 şi numărătorul se decrementează.

Determinaţi secvenţa stărilor circuitului prezentat, presupunând că porneşte din starea 0001.

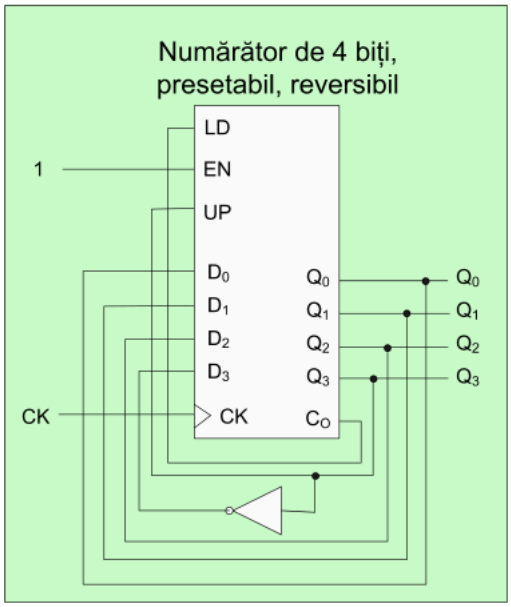

Figura 17.23 Circuitul cu numărător analizat la problema 4.

© 2015 Dan NICULA - dan.nicula@unitbv.ro
www.DanNicula.ro/ed_ci