PREFAŢĂ

Dispozitivele numerice (digitale) sunt componentele de bază ale calculatoarelor electronice şi ale altor sisteme şi aparate destinate procesării informaţiei. Aceste structuri numerice, pe care se bazează o foarte mare parte a tehnologiilor moderne din domeniul electronicii, asigură precizia, fiabilitatea, viteza şi complexitatea necesare prelucrării datelor. Începînd cu structurile logice cele mai simple şi pînă la circuitele secvenţiale complexe sau logica programată se utilizează pe scară largă diferite metode şi tehnici de proiectare (sinteză) şi analiză a acestora. Pentru a utiliza cu succes aceste metode şi tehnici proiectantul dispozitivelor numerice trebuie să aibă pregătirea teoretică respectivă şi să dispună de deprinderile practice necesare.

Lucrarea de faţă conţine materialul referitor la mijloacele şi principalele aspecte ale proiectării digitale a diferitor structuri numerice. În capitolul 1 este prezentată descrierea, posibilităţile şi principiile de utilizare a sistemului de proiectare **Logic Works.** Capitolul 2 este destinat abordării principalelor aspecte ale sintezei şi analizei circuitelor logice combinaţionale şi descrierii diferitor implementări ale acestora. Pentru studierea lor practică sunt descrise aplicaţii pentru cinci lucrări de laborator. În capitolul 3 este efectuată prezentarea teoretică şi sunt descrise aspectele de sinteză ale principalelor structuri ale circuitelor logice secvenţiale. Tot în acest capitol este prezentat materialul necesar îndeplinirii a două lucrări de laborator.

Îndeplinirea fiecărei lucrări de laborator presupune:

- asamblarea schemelor circuitelor numerice indicate de către profesor, sinteza cărora a fost efectuată anterior de către student în conformitate cu varianta sa;
- efectuarea analizei circuitelor asamblate şi colectarea datelor pentru darea de seamă a lucrării.

În lucrare sunt descrise două modalităţi de îndeplinire a lucrărilor de laborator. Prima din ele prevede simularea şi

analiza circuitelor cu ajutorul sistemului **Logic Works,** iar a doua - asamblarea circuitelor şi analiza lor cu ajutorul standului de laborator.

Pentru a fi admis la îndeplinirea lucrării de laborator fiecare student trebuie să îndeplinească următoarele condiţii:

- să aibă îndeplinită tema pentru acasă, care constă din sinteza uneia sau mai multor scheme ale circuitelor numerice;
- să răspundă la întrebările de control puse de către profesor.

Lucrarea de laborator se consideră îndeplinită doar după ce studenţii demonstrează profesorului funcţionarea corectă a circuitelor asamblate.

Pentru fiecare lucrare, fiecare student pregăteşte o dare de seamă pe care o susţine în faţa profesorului. Darea de seamă include: foaia de titlu, tema, scopul lucrării, lucrul îndeplinit acasă, schemele tuturor circuitelor asamblate, tabelele obţinute în rezultatul îndeplinirii lucrării respective.

Descrierea standului de laborator

Standul de laborator este destinat studierii practice a structurii şi sintezei elementelor funcţionale şi unităţilor calculatoarelor numerice.

Standul de laborator este realizat în baza circuitelor integrate din familia K155 şi este compus din: elemente ŞI-NU cu două, trei, patru şi opt intrări (respectiv circuitele K155LA3, K155LA4, K155LA1 şi K155LA2); elemente ŞI cu două intrări (circuitul K155LI1); elemente SAU cu două intrări (circuitul K155LL1); elemente NU (circuitul K155LN1); elemente SAU-NU cu două intrări (circuitul K155LE1); elemente 2-2ŞI-2SAU-NU şi 4-4ŞI-2SAU-NU cu posibilitatea de extindere prin elementul SAU) respectiv circuitele K155LR1, K155LR4); bistabilele JK (circuitul K155TV1); numărătorul binar-zecimal (circuitul K155IE5); numărătorul binar-reversibil

(circuitul K155IE7); registrul de deplasare (circuitul K155IR1); sumatoare binare de patru ranguri (circuitul K155IM3).

Intrările şi ieşirile circuitelor integrate sunt scoase în partea din faţă a standului. Comunicaţiile necesare între intrările şi ieşirile circuitelor integrate se realizează cu ajutorul firelor de contact.

Standul este dotat cu un generator de impulsuri şi poate funcţiona şi în regim "pas cu pas".

1. SISTEMUL DE PROIECTARE DIGITALĂ LOGIC WORKS (versiunea 4)

Sistemul **Logic Works** este destinat proiectării şi simulării circuitelor digitale. Asamblarea circuitelor se efectuează cu ajutorul elementelor logice şi circuitelor combinaţionale şi secvenţiale care se conţin în bibliotecile sistemului. Circuitul asamblat se analizează cu ajutorul diagramelor de timp şi a elementelor de vizualizare a valorilor logice generate de circuit.

1.1 Interfaţa

Fereastra de lucru – conţine circuitele proiectate. Circuitele din **Fereastra de lucru** funcţiunează în acord cu informaţia de timp din **Fereastra de timp**.

Biblioteca – conţine toate elementele necesare pentru proiectare.

Fereastra de timp – în stare activă afişează diagramele de timp ale circuitului proiectat.

Bara de instrucţiuni pentru proiectare – este folosită pentru controlul cursorului şi pentru modificarea statutului **Ferestrei de lucru**. Funcţiile din această bară permit deschiderea, salvarea şi tipărirea documentului, folosirea cursorului pentru inserarea elementelor şi trasarea liniilor, verificarea valorilor logice, marcarea elementelor, ştergerea liniilor şi elementelor ş.a.

Bara de instrucţiuni pentru simulare – este folosită pentru a controla **Fereastra de timp** şi anume frecvenţa de generare a impulsurilor. Instrucţiunile de simulare pot fi accesate şi selectând meniul **Simulation** din **Bara de meniuri**.

1.2 Proiectarea de bază

Amplasarea elementelor. Pentru amplasarea elementelor în **Fereastra de lucru** se execută dublu clic pe numele elementului din bibliotecă. Versiunea elementului va apărea în dreptul cursorului. Elementul se amplasează în locaţia dorită şi se execută clic. Elementul selectat va rămâne activ pentru o nouă amplasare până nu va fi dezactivat. Selectarea altui element se execută prin efectuarea dublului clic pe numele elementului corespunzător din bibliotecă şi procedura se repetă. Pentru a anula selectarea elementului se tastează **space bar** sau butonul din dreapta al mouseului. Pentru a schimba orientarea elementului se selectează instrucţiunea **Orientation** din meniul **Schematic**. Se poate alege una dintre cele opt opţiuni din meniul grafic care se va deschide. Toate elementele inserate vor avea aceeaşi orientare.

Trasarea liniilor. Pentru a uni elementele logice între ele se apasă şi se menţine butonul din stînga al mousului la capătul pinului de intrare sau ieşire al unuia din elemente sau la nodul de conexiune dintre firele circuitului. Cursorul se deplasează spre punctul dorit, care poate fi un alt pin, alt fir sau o locaţie liberă. *Fiţi atenţi deoarece acest mod de trasare poate duce la apariţia unor noduri nedorite în punctele de intersecţie a liniilor !*. O altă metodă de trasare a liniilor este de a folosi butoanele din bara de instrumente pentru proiectare.

Editarea. Pentru a deplasa o linie sau un element se execută clic pe el. Butonul din stânga al mousului se ţine apăsat şi se deplasează spre locaţia dorită. Elementele selectate vor fi evidenţiate cu negru, liniile cu galben. Pentru a deplasa mai multe elemente se apasă şi se menţine butonul din stânga al mousului pe o locaţie neocupată iar apoi se deplasează mousul până cand toate elementele dorite vor fi cuprinse în fereastra creată. Pentru a şterge un element sau o linie se execută clic pe el şi se apasă tasta **Del**. Pentru a şterge un număr mai mare de elemente toate elementele se selectează şi se apasă tasta **Del**.

1.3 Bara de instrucţiuni pentru proiectare

Funcţiile din această bară sunt folosite pentru schimbarea poziţiei elementelor şi tipului de linii, executarea corectărilor necesare în circuite, inserarea textului, schimbarea aspectului **Ferestrei de lucru** şi efectuarea probelor logice ale circuitului.

Bara de instrucţiuni pentru proiectare constă din cinci părţi: **Gestionarea fişierelor**, **Deplasarea şi modificarea elementelor** , **Modificarea cursorului**, **Modificarea Ferestrei de lucru** şi **Tipul elementului**.

Gestionarea fişierelor. Primele patru butoane sunt folosite pentru gestionarea fişierelor. Începând cu butonul din stânga aceste patru butoane sunt folosite pentru **crearea** fişierelor noi, **deschiderea** documentului existent, **salvarea** documentului activ şi **tipărirea** documentului activ. Fiecare din aceste instrucţiuni, la fel ca şi celelalte instrucţiuni de gestiune pot fi găsite în meniul **File.**

Modificarea elementelor. Următoarele cinci butoane sunt folosite pentru deplasarea şi modificarea elementelor.

Cut – ştergerea elementelor selectate în **Fereatra de lucru**.

Copy – copierea elementelor selectate în Cliboard,

Insert – inserarea conţinutului din Clipboard în **Fereastra de lucru**,

Duplicate – dublarea elementelor selectate,

Get Info – obţinerea şi setarea informaţiei despre elementul selectat. În acest caz doar un singur element poate fi selectat.

Modificarea cursorului. Următoarele şapte butoane servesc pentru modificarea cursorului.

Lupa. Prezintă imaginea centrată şi mărită a locaţiei selectate. Mărirea poate fi de asemenea efectuată şi utilizând comanda **Magnify** din meniul **View.** Revenirea la dimensiunile normale poate fi executată utilizând comanda **Reduce to**

Fit din meniul **View** sau butoanele de dirijare a dimensiunilor **Ferestrei de lucru**, examinate ulterior.

Probe. Butonul **Probe** are două funcţii. În primul rând el permite de a afla valoarea logică a pinului. Pentru aceasta cursorul se poziţionează pe pinul dorit şi se ţine apăsat butonul din stânga al mousului. În al doilea rând se poate ataşa pinului valoarea logică dorită dacă se tastează această valoare în timpul procedurii precedente. *Atenţie! În acest caz puteţi provoca conflicte logice care pot duce la deteriorarea circuitului*.

Pointer. Aceasta este starea implicită a cursorului. Când nu este activă, cursorul poate funcţiona în toate modurile descrise anterior.

Zap. În unele cazuri este nevoie de a şterge doar o porţiune de linie sau un element situat într-o regiune foarte aglomerată. Funcţia **Zap** permite de a face aceasta cu o mai mare flexibilitate. Pentru aceasta se execută clic pe porţiunea de linie sai pe elementul dorit.

Text. Utilizând funcţia **Text** se poate adăuga în schemă titluri, descrieri ale circuitelor şi elementelor. La activarea butonului cursorul va apărea în formă de pix. În locaţia selectată se va deschide o casetă de text, unde se poate introduce textul dorit. Pentru redactarea textului se execută clic pe textul dorit.

Această funcţie mai permite de a denumi un pin, semnal logic sau element. Numele pinului va avea culoarea albastră, numele elementului şi al semnalului va avea culoarea roşie.

Pentru a denumi semnalul logic e necesar de selectat linia respectivă.

Trasarea liniilor.

Draw signal. Permite trasarea liniilor de conexiune dintre elementele circuitului. Nu este necesar de a începe trasarea de la capătul pinului sau nodului spre deosebire de trasarea liniilor cu cursorul obişnuit al mousului.

Draw bus. Permite trasarea magistralelor.

Modificarea Ferestrei de lucru.

Zoom Out. Mărirea imaginii în **Fereastra de lucru**.

Zoom In. Micşorarea imaginii în **Fereastra de lucru**.

Fit to Window. Vizualizarea întregului circuit proiectat în **Fereastra de lucru**.

Normal Size. Revenirea la dimensiunile normale.

Tipul elementelor.

PROM/RAM/PLA Wizard. Această funcţie permite programarea şi cercetare circuitelor de memorie permanentă şi cu acces aleatoriu precum şi a matricelor logice programabile.

Parts Palette. Activează şi dezactivează **Fereastra de vizualizare a bibliotecilor.**

1.4 Bibliotecile

Pentru îndeplinirea lucrărilor de laborator se vor folosi următoarele biblioteci:

Simulation Gates.clf. Această bibliotecă conţine

următoarele porţi logice: **AND** (ŞI), **OR** (SAU), **NOT** (NU), **NAND** (ŞI-NU), **NOR** (SAU-NU), **XOR** (SAU-EXSCLUSIV), **XNOR** (SAU-NU-EXSCLUSIV). Sunt accesibile porţi logice cu numere diferite de intrări (**AND-2, AND-4**). Unele porţi logice au intrări inversate (**AND-4(2-Inv)**).

Simulation IO.clf. Această bibliotecă conţine mai multe dispozitive de intrare-ieşire care sunt folosite pentru introducerea şi vizualizarea informaţiei.

Seven-Segment Displays –display pe şapte segmente. Sunt disponibile mai multe variante color:

7-SegDisp-Color. Funcţionează în felul următor: aplicând 1 logic la una dintre cele şapte intrări informaţionale, segmentul corespunzător începe să lumineze.

7-SegDispInv-Color. La aplicarea valorii 0 logice la una dintre cele şapte intrări informaţionale, segmentul corespunzător începe să lumineze.

Poate fi folosit pentru vizualizarea cifrelor zecimale utilizând un decodificator de şapte segmente din biblioteca **7400devs.clf** (74-46, 74-47, 74-48, 74-49).

LEDs (light –emitting diodes) – diode color luminiscente. Pentru a conecta corect dioda este necesar de a uni cu pământul pinul inversat şi celălalt pin cu dispozitivul care urmează să dirijeze dioda (1 logic – dioda luminează).

Comutatoarele.

- **SPDT pushbutton** (Single-Pole Double-Throw) Întrerupător care deţine o anumită valoare şi trece la altă

valoare logică în momentul efectuării contactului cu ajutorul butonului din stânga al mousului.

- **SPDT switch.** Comutator care transmite una dintre cele două valori pe care le deţine în dependenţă de legătura de la intrarea comutatorului.
- **SPST switch** (Single-Pole Single-Throw) Comutator cu o singură intrare şi o singură ieşire care poate fi sau închis sau deschis.

Binary Switch. Reprezintă o varietate a **SPDT switch** ale cărui intrări sunt deja conectate la 1 şi 0 logic.

Binary Probe. Conectat la orice linie a circuitului proiectat afişează valoarea binară în acest punct, fiind de ajutor la detectarea erorilor sau la testarea circuitului. În afară de valorile binare 0 şi 1 pot apărea şi altele:

- **(X)** Valoarea nu este cunoscută / valoare indiferentă;
- **(Z)** Ieşirea nu este conectată la intrare;
- **(C)** Condiţie de conflict.

Hex Keyboard . Tastatură hexazecimală cu ajutorul căreia pot fi introduse valorile hexazecimale de la 0 la F, prezentate în formă de cod binar pe patru biţi.

Hex Display. Diplay hexazecimal folosit pentru vizualizarea informaţiei în formă hexazecimală.

Clock (semnalul de tact sau ceasul) este folosit la cercetarea circuitelor logice secvenţiale.

Simulation Logic.clf. Această bibliotecă conţine mai multe circuite combinaţionale şi secvenţiale specializate care pot

fi utilizate la proiectarea circuitelor digitale complexe: sumatoare (Adder-4, Adder-8, Adder-4 wo/Carry), numărătoare (Counter-4, Counter-4 Up wo/En), decodificatoare (Decoder-4, Decoder-8), multiplexoare (Mux-4, Mux-4 wo/En), bistabile (JK Flip Flop, D Flip Flop), registre (Reg-4, Reg-4 inv CLR) ş.a.

1.5 Simularea şi sincronizarea

Bara de instrumente pentru simulare reprezintă interfaţa cu **Simulatorul** şi cu **Fereastra De timp**. Instrucţiunile din această bară permit controlul asupra vitezei de simulare, statutului semnalelor din circuitul proiectat. Bara constă din trei părţi de bază: **Statutul Semnalului**, **Statutul Ferestrei Temporale** şi **Statutul Simulatorului.**

Statutul semnalului

Show/Hide Timing. Activează sau dezactivează **Fereastra de timp**.

Add Signals to Timing. Adaugă semnalul selectat în **Fereastra de timp** în cazul când opţiunea **Add Automatically** din meniul **Simulation** nu este activată.

Triggers. Deschide meniul opţional de declanşare al semnalului.

Simulation Parameters. Susţine aceleaşi funcţii ca şi comanda **Simulation Parameters** din meniul **Simulation**.

Stick/unstick Signals. Fixează semnalul la un anumit

nivel logic. Semnalul fixat rămâne la acest nivel logic indiferent de schimbările care au loc în circuit.

Reset Simulator. Resetează **Simulatorul**.

Clear Unknowns. Elimină toate semnalele cu valori neindetificate.

Statutul Ferestrei Temporale

Următoarele trei butoane controlează scara **Ferestrei Temporale**.

Zoom In. **Fereastra De timp** conţine mai puţine gradaţii.

Zoom Out. **Fereastra De timp** conţine mai multe gradaţii.

Normal Size. Revenirea la gradarea iniţială.

Statutul Simulatorului

Controlează viteza **Simulatorului** şi constă din trei butoane, bara de viteză şi un numărător.

Single Step. Afişează în **Ferastra De timp** schimbările care au loc în circuit la un singur pas de simulare. Un pas de simulare reprezintă perioada minimă de timp în care în circuit se produce o oarecare schimbare. Deci pasul de simulare poate fi de orice lungime şi de obicei reflectă timpul necesar pentru schimbarea valorii celei mai rapide variabile.

Stop Simulator. Opreşte **Simulatorul,** fără a-l reseta. Simularea poate fi restartată activând butonul **Run Simulator** sau selectând viteza de simulare de pe **Bara de viteză**. La fel simularea poate fi continuată în regimul **Single Step**.

Simulation Speed. Reprezintă bara de viteză cu ajutorul căreia poate fi schimbată viteza de simulare a circuitului.

Run Simulator. Activarea acestui buton duce la rularea simulatorului la viteză maximă.

Simulation Counter. Această fereastră afişează pasul de simulare curent care coincide cu pasul din **Fereastra de timp**.

Fereastra de timp

Fereastra de timp afişează toate schimbările care se produc cu semnalele denumite. Aceste schimbări pot fi statice(manuale) şi dinamice(automate).

Simularea manuală. Pentru simularea manuală se folosesc comutatoarele şi verificatoarele binare (**Binary Probe**). Deoarece Logic Works este un simulator bazat pe evenimente discrete de timp, orice schimbare a valorii variabelei de intrare produce o reînnoire a valorii variabilei respective. Această schimbare este reflectată în verificatorul binar sau în **Fereastra De timp**.

Simularea automată. Într-un circuit complex utilizarea manuală a comutatoarelor devine neavantajoasă şi poate duce la erori. În aceste cazuri procesul de simulare poate fi automatizat utilizând **fişiere temporale**. Un fişier temporal reprezintă o metodă compactă şi simplă de a indica valorile semnalelor de intrare în momente discrete de timp.

Un fişier temporal necesită cel puţin trei antete şi cel puţin un rând de date, cu un element din fiecare rând corespunzător fiecărui antet. Aceste antete sunt:

- **$T -** indică momentele discrete de timp. În Logic Works măsurarea timpului începe de la 0 şi fiecare unitate de timp este echivalentă cu o nanosecundă;
- **$D -** indică diferenţa dintre valoarea timpului din rândul curent şi cel următor.
- **$O** – antetul este folosit pentru toate semnalele de intrare şi este urmat de numele acestor semnale. În cazul când fişierul temporal este exportat toate semnalele, atât cele de intrare cât şi cele de ieşire sunt indicate cu ajutorul antetului **$O**, urmat de numele fiecărui semnal.

În continuare este prezentat un exemplu a unui fişier temporal, creat pentru simularea unui circuit cu trei semnale de intrare: A, B şi C

$T	$D	$O A	$O B	$O C
0	10	0	0	0
10	10	0	0	1
20	10	0	1	0

Importarea şi exportarea fişierelor temporale

Fişierul temporal cu extensia **.TIM** poate fi creat în orice redactor de texte, de exemplu **Notepad** sau **Wordpad**. Pentru a importa fişierul temporal în **Simulator** selectaţi comanda **Import Timing** din meniul **Simulation**. După selectarea numelui fişierului temporal în **Fereastra De timp** va apărea o versiune întretăiată a semnalelor de intrare. La pornirea

Simulatorului programul **Logic Works** va aplica valorile acestor semnale în locaţiile necesare. Orice semnal de ieşire afişat în **Fereastra De timp** va fi reînnoit. În cazul când doriţi să obţineţi rezultatele simulării în formă de tabel puteţi să exportaţi fişierul temporal. Pentru aceasta selectaţi comanda **Export Timing** din meniul **Simulation**. Acest fişier va conţine cât semnalele de intrare atât şi cele de ieşire.

Timpul de reţinere a semnalelor

În **Logic Works** toate porţile logice la fel ca şi circuitele logice mai complexe de tipul sumatoarelor, multiplexoarelor ş.a. au timpul de reţinere egal cu o nanosecundă, ceea ce este departe de realitate. De exemplu o poartă logică ŞI-NU cu două intrări are, în cel mai rău caz, o întârziere de 22 nanosecunde. În cazul când apare necesitatea de a studia comportamentul circuitelor în condiţii mai reale puteţi să schimbaţi timpul de reţinere.

Pentru a schimba timpul de reţinere la una sau mai multe porţi logice selectaţi-le pe toate, ţinând apăsată tasta **CTRL**, apoi activaţi comanda **Simulation Parameters** din meniul **Simulation**. În fereastra ce va apărea introduceţi valoarea necesară. Toate porţile selectate vor avea acelaşi timp de reţinere.

Pentru a schimba timpul de reţinere la circuitele din biblioteca **7400devs.clf** este necesar la început de deblocat porţile logice componente. Pentru aceasta apăsaţi butonul din dreapta al mousului pe circuit şi selectaţi **Device Info**. În fereastra care va apărea desactivaţi caseta de validare **Lock Opening**

Subcircuit. Acum puteţi schimba parametrii porţilor logice componente ca şi în cazul unor porţi logice separate. Pentru aceasta executaţi dublu clic pe circuitul pe care doriţi să-l modificaţi. Va apărea o nouă **Fereastră de lucru** cu toate porţile logice componente ale circuitului deblocat unde puteţi executa modificările necesare.

2. CIRCUITELE LOGICE COMBINAŢIONALE

2.1 Prezentare teoretică

Orice circuit logic se caracterizează prin natura semnalelor de intrare, a celor de ieşire, prin clasele de funcţii intrare-ieşire şi prin natura prelucrărilor de date ce au loc în structura sa internă.

Circuitele logice se împart în două clase: combinaţionale şi secvenţiale. Un circuit logic combinaţional (CLC) se caracterizează prin aceea că starea ieşirilor sale la un moment dat depinde numai de starea intrărilor sale în acest moment. Legătura între starea intrărilor şi starea ieşirilor circuitului este dată de funcţiile de transfer ale acestuia, denumite în acest caz funcţii de comutare, care sunt funcţii booleene (logice).

CLC este circuitul, care are ***n*** intrări (x_1, x_2, x_3, ..., x_n) şi ***m*** ieşiri (y_1, y_2, y_3, ..., y_m), la care ieşirile pot fi exprimate numai în dependenţă de variabilele de intrare:

$$y_1=f_1(x_1, x_2, x_3, ..., x_n);$$
$$y_2=f_2(x_1, x_2, x_3, ..., x_n);$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$y_m=f_m(x_1, x_2, x_3, ..., x_n).$$

Pentru că în acest model matematic nu intervin ca variabile independente timpul şi nici mărimile de ieşire, rezultă,

că în structura sa un CLC nu prezintă circuite de memorie şi nici legături de reacţie (variabilele de ieşire nu sunt aplicate la intrare).

Sinteza unui CLC se efectuează în următoarele etape:

- descrierea necesităţilor ce trebuie să le rezolve circuitul combinaţional respectiv (prin text, desen, diagrame etc.);
- reprezentarea acestei descrieri sub forma unui tabel de adevăr;
- deducerea funcţiilor logice şi minimizarea acestora;
- implementarea acestor funcţii minimizate sub forma unor reţele de comutare prin intermediul circuitelor integrate;

Tabelul de adevăr conţine $\boldsymbol{n+m}$ coloane şi $\mathbf{2^n}$ rînduri. Fiecare rînd al tabelului reprezintă una din combinaţiile posibile ale valorilor variabilelor şi valorile funcţiilor pentru combinaţia respectivă.

Implementarea funcţiilor logice minimizate sub forma reţelelor de comutare poate fi realizată sau în forma canonică disjunctivă (ŞI/SAU), sau în forma canonică conjunctivă (SAU/ŞI), sau în orice altă formă normală, adică ŞI-NU/ŞI-NU, SAU/ŞI-NU, SAU-NU/SAU, ŞI/SAU-NU, ŞI-NU/ŞI, SAU-NU/SAU-NU.

Trecerea de la o formă normală la alta se efectuează prin utilizarea succesivă a formulelor lui De Morgan, avînd iniţial forma canonică disjunctivă normală (ŞI/SAU) şi forma canonică conjunctivă normală (SAU/ŞI) a funcţiei.

De exemplu:
din forma disjunctivă normală:

(forma ŞI/SAU):

$$y = x_2\bar{x}_4 \vee \bar{x}_1\bar{x}_3x_4 \vee \bar{x}_1\bar{x}_2x_4 \vee x_1\bar{x}_3\bar{x}_4 =$$

(forma ŞI-NU/ŞI-NU):

$$= \overline{(x_2\bar{x}_4)(\bar{x}_1\bar{x}_3x_4)(\bar{x}_1\bar{x}_2x_4)(x_1\bar{x}_3\bar{x}_4)} =$$

(forma SAU/ŞI-NU):

$$= \overline{(\bar{x}_2 \vee x_4)(x_1 \vee x_3 \vee \bar{x}_4)(x_1 \vee x_2 \vee \bar{x}_4)(\bar{x}_1 \vee x_3 \vee x_4)} =$$

(forma SAU-NU/SAU):

$$= \overline{(\bar{x}_2 \vee x_4)} \vee \overline{(x_1 \vee x_3 \vee \bar{x}_4)} \vee \overline{(x_1 \vee x_2 \vee \bar{x}_4)} \vee \overline{(\bar{x}_1 \vee x_3 \vee x_4)}$$

din forma conjunctivă normală:

(forma SAU/ŞI):

$$y = (\bar{x}_1 \vee \bar{x}_4)(x_2 \vee x_3 \vee x_4)(\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4)(x_1 \vee x_2 \vee x_4) =$$

(forma ŞI-NU/ŞI):

$$= \overline{(x_1x_4)}\,\overline{(\bar{x}_2\bar{x}_3\bar{x}_4)}\,\overline{(x_2x_3x_4)}\,\overline{(\bar{x}_1\bar{x}_2\bar{x}_4)} =$$

(forma ŞI/SAU-NU):

$$= \overline{(x_1x_4) \vee (\bar{x}_2\bar{x}_3\bar{x}_4) \vee (x_2x_3x_4) \vee (\bar{x}_1\bar{x}_2\bar{x}_4)} =$$

(formaSAU-NU/SAU-NU):

$$= \overline{\overline{(\bar{x}_1 \vee \bar{x}_4)} \vee \overline{(x_2 \vee x_3 \vee x_4)} \vee \overline{(\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4)} \vee \overline{(x_1 \vee x_2 \vee x_4)}}$$

2.2 Lucrarea de laborator nr. 1

Tema: Sinteza circuitelor logice combinaţionale

Scopul lucrării: studierea practică şi cercetarea procesului de sinteză a circuitelor logice combinaţionale.

Tema pentru acasă

1. Se efectuează minimizarea funcţiilor logice y_1 şi y_2 conform variantei din tabelul 2.1. Pentru ambele funcţii se

efectuează sinteza circuitul logic în setul de elemente ŞI-NU.

2. Funcţia y_1 se reprezintă în forma disjunctivă normală perfectă şi forma conjunctivă normală perfectă. Pentru forma disjunctivă normală perfectă se efectuează sinteza circuitul logic în setul de elemente ŞI-NU.

3. Funcţia y_2 se reprezintă în toate cele 8 forme normale.

Tabelul 2.1

Nr. Var.	Funcţiile logice
1	2
1	$y_1=\vee(0,1,2,4,5,7,9,10,11,14,15)$ $y_2=\vee(2,3,4,5,8,9,12,13)$
2	$y_1=\vee(1,3,4,7,8,10,12,13,14)$ $y_2=\vee(3,4,5,7,9,11,13,14,15)$
3	$y_1=\vee(0,2,4,5,8,10,12,14)$ $y_2=\vee(1,2,3,4,7,8,9,12,13,14)$
4	$y_1=\vee(0,2,3,5,6,7,9,11,12,13,14)$ $y_2=\vee(1,2,4,5,6,8,9,11,14,15)$
5	$y_1=\vee(2,4,5,7,8,9,12,14,15)$ $y_2=\vee(0,1,2,7,8,10,11,14)$
6	$y_1=\vee(1,2,4,5,6,8,10,14,15)$ $y_2=\vee(0,1,2,5,6,7,9,11,12,13)$
7	$y_1=\vee(0,1,5,6,7,8,10,12,14,15)$ $y_2=\vee(1,2,4,8,9,10,11,12)$
1	2
8	$y_1=\vee(0,1,2,4,6,8,11,12,15)$ $y_2=\vee(0,1,2,5,6,7,8,9,12,13)$

9	$y_1=\vee(0,2,4,5,7,8,10,12,15)$ $y_2=\vee(2,3,4,5,7,8,9,11,12,14)$
10	$y_1=\vee(0,3,4,5,6,8,10,12,13)$ $y_2=\vee(4,5,6,7,9,11,12,13,14)$
11	$y_1=\vee(1,2,4,5,8,9,10,12,13,14,)$ $y_2=\vee(3,4,5,7,8,9,11,12,13)$
12	$y_1=\vee(0,1,2,4,5,8,9,12,14,)$ $y_2=\vee(1,2,3,5,6,8,10,11,12)$
13	$y_1=\vee(0,2,4,5,6,7,9,12,13,15)$ $y_2=\vee(2,3,4,5,7,8,9,10,11,)$
14	$y_1=\vee(0,2,3,4,6,9,10,11,13,14,15)$ $y_2=\vee(3,4,5,7,8,10,11,14,15,)$
15	$y_1=\vee(0,1,4,5,7,8,10,11,12)$ $y_2=\vee(1,3,5,6,7,9,10,12,15)$
16	$y_1=\vee(0,2,3,4,6,7,9,11,12,13)$ $y_2=\vee(3,4,5,8,9,11,12,14)$
17	$y_1=\vee(0,3,4,5,7,8,12,13,14)$ $y_2=\vee(2,4,5,6,8,10,11,15)$
18	$y_1=\vee(1,2,3,4,6,7,8,9,10)$ $y_2=\vee(2,3,5,6,7,10,12,15)$
19	$y_1=\vee(0,1,2,5,6,7,14,15)$ $y_2=\vee(2,3,4,7,8,9,10,12,13,14,15)$
20	$y_1=\vee(3,4,5,6,7,8,10,12,13)$ $y_2=\vee(0,1,2,5,6,8,9,11,12,14)$
21	$y_1=\vee(0,1,5,6,7,8,9,10,12,13)$ $y_2=\vee(1,2,3,4,5,6,8,9,11,14,15)$
22	$y_1=\vee(0,2,3,5,8,9,10,12,13)$ $y_2=\vee(1,3,4,6,7,8,9,10,11,14)$

Desfăşurarea lucrării

a) la standul de laborator:

1. Se verifică corectitudinea funcţionării circuitelor integrate ale standului de laborator.

2. Se asamblează şi se reglează circuitul logic combinaţional, care realizează două funcţii din tema pentru acasă în setul de elemente ŞI-NU (la indicaţia profesorului).

3. Pentru circuitele asamblate se determină costul şi timpul de reţinere.

b) în LogicWorks:

1. Din biblioteca de elemente **Simulation Gates.clf** se selectează elementele **NAND** cu numărul corespunzător de intrări. Din biblioteca **Simulation IO.clf** se selectează dispozitivele de intrare-ieşire **Binary Probe** şi **Hex Keyboard.**

2. Se asamblează circuitul logic combinaţional în **Fereastra de lucru** şi se verifică corectitudinea lui. Se studiază diagrama de timp. Un exemplu al circuitului asamblat este prezentat în fig. 2.1.

3. Pentru circuitele asamblate se determină costul şi timpul de reţinere.

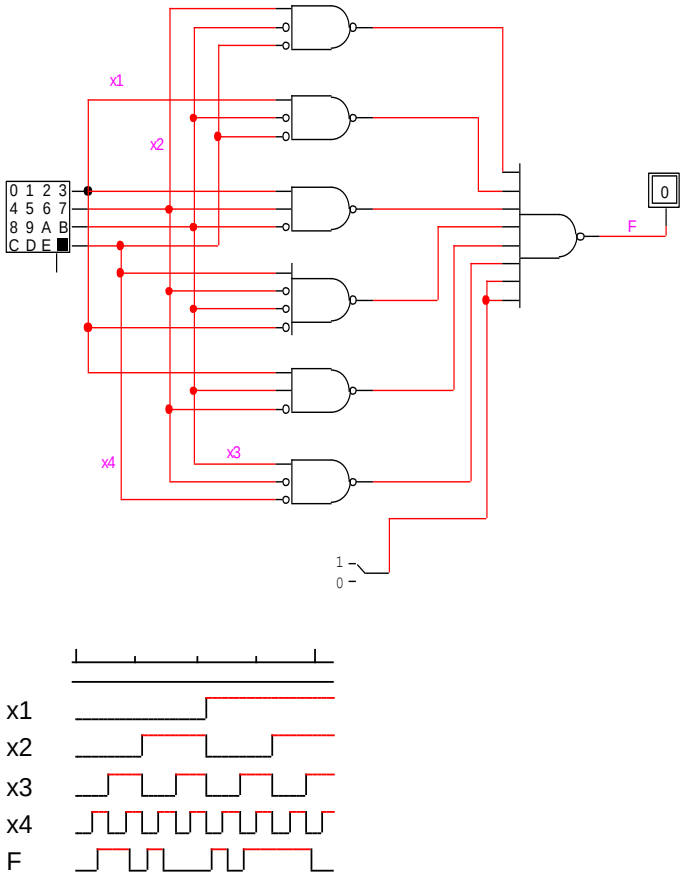


Fig. 2.1. Un circuit logic combinaţional asamblat în LogicWorks şi diagrama lui de timp.

Întrebări

1. Care sunt particularităţile care

caracterizează circuitele logice combinaţionale?

2. Care sunt etapele de sinteză ale circuitelor logice combinaţionale ?

3. Cum se calculează timpul de funcţionare a unui circuit logic combinaţional ?

2.3. Convertoarele de cod

Prezentare teoretică

Convertoarele de cod sunt elementele funcţionale destinate transformării unui cod binar în altul. De obicei, aceste elemente funcţionale reprezintă circuite logice combinaţionale cu ***n*** intrări şi ***m*** ieşiri. Aria tipurilor de convertoare de cod este foarte largă, iar valorile ***n*** şi ***m*** ale lor pot coincide, dar pot fi şi diferite, în dependenţă de tipul de coduri de la intrarea şi, respectiv, de la ieşirea convertorului de cod.

În calitate de convertoare de cod, la care numărul de intrări coincide cu numărul de ieşiri pot servi cele care transformă codul direct al unui număr binar în codul lui invers, sau în cel complementar. Tot în această categorie intră convertoarele unui cod binar- zecimal în altul etc.

Convertoare de cod cu număr diferit de intrări şi ieşiri sunt acelea care efectuează conversia numerelor dintr-un sistem de numeraţie în altul, convertoarele care transformă codul binar-zecimal de patru biţi în codul pentru indicatoarele numerice de şapte segmente (şapte biţi) ş.a.

Sinteza convertoarelor de cod, indiferent de tipul lor, are loc în felul următor:

1. Elaborarea tabelului de adevăr cu următorii parametri: numărul de variabile este egal cu numărul biţilor codului, care se aplică la intrările convertorului, iar numărul funcţiilor cu numărul de biţi ai codului, care trebuie obţinut la ieşirile convertorului de cod. Funcţiile logice pot fi parţial determinate, dacă

numărul combinaţiilor codului de intrare este mai mic decît $\mathbf{2^n}$.

2. Minimizarea tuturor funcţiilor din tabelul de adevăr.

3. Depistarea conjuncţiilor comune ale formelor minimale ale tuturor funcţiilor, pentru a evita dublarea elementelor logice, care realizează părţi comune ale mai multor funcţii.

4. Implementarea funcţiilor minimizate prin circuite integrale digitale.

Vom ilustra cele descrise mai sus printr-un exemplu de sinteză a unui convertor de cod binar-zecimal. Tabelul de adevăr care descrie structura convertorului de cod 8 7 (-2) (-7) $\rightarrow$ 4 2 2 1 este prezentat în tabelul 2.2

Tabelul 2.2

Nr.	87(-2)(-4)				4221			
	x_4	x_3	x_2	x_1	f_4	f_3	f_2	f_1
0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	1
2	1	0	1	1	0	0	1	0
3	0	1	0	1	0	0	1	1
4	1	0	0	1	0	1	1	0
5	0	1	1	0	1	0	0	1
6	1	0	1	0	1	1	0	0
7	0	1	0	0	1	1	0	1
8	1	0	0	0	1	1	1	0
9	1	1	1	1	1	1	1	1
10	0	0	0	1	*	*	*	*
11	0	0	1	0	*	*	*	*

12	0	0	1	1	*	*	*	*
13	1	1	0	0	*	*	*	*
14	1	1	0	1	*	*	*	*
15	1	1	1	0	*	*	*	*

În fig. 2.2 sunt prezentate diagramele Karnaugh pentru minimizarea funcţiilor f_4, f_3, f_2, f_1.

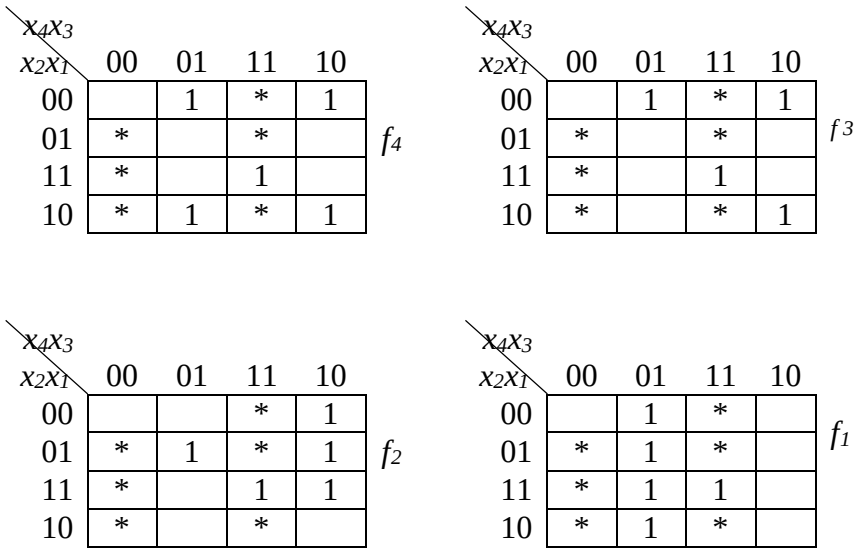


Fig. 2.2 Diagramele Karnaugh pentru minimizarea funcţiilor f_4, f_3, f_2, f_1.

În rezultatul minimizării au fost obţinute următoarele funcţii logice:

$$
\begin{aligned}
f_4 &= x_3\bar{x}_1 \vee x_4\bar{x}_1 \vee x_4x_3;\\
f_3 &= x_3\bar{x}_2\bar{x}_1 \vee x_4\bar{x}_2 \vee x_4\bar{x}_1 \vee x_4x_3;\\
f_2 &= x_4\bar{x}_2 \vee \bar{x}_2x_1 \vee x_4x_1;\\
f_1 &= x_3.
\end{aligned}
\tag{3.1}
$$

Luînd în consideraţie conjuncţiile comune, funcţiile f_4, f_3, f_2, f_1 pot fi scrise în felul următor:

$$
\begin{aligned}
f_4 &= x_3\bar{x}_1 \vee z_1;\\
f_3 &= x_3\bar{x}_2\bar{x}_1 \vee z_1 \vee z_2;\\
f_2 &= \bar{x}_2x_1 \vee x_4x_1 \vee z_2;\\
f_1 &= x_3.
\end{aligned}
\tag{3.2}
$$

unde :

$$
\begin{aligned}
z_1 &= x_4\bar{x}_1 \vee x_4x_3;\\
z_2 &= x_4\bar{x}_2.
\end{aligned}
\tag{3.3}
$$

Schema convertorului de cod 8 7 (-2) (-7) → 4 2 2 1 este prezentată în fig. 2.3.

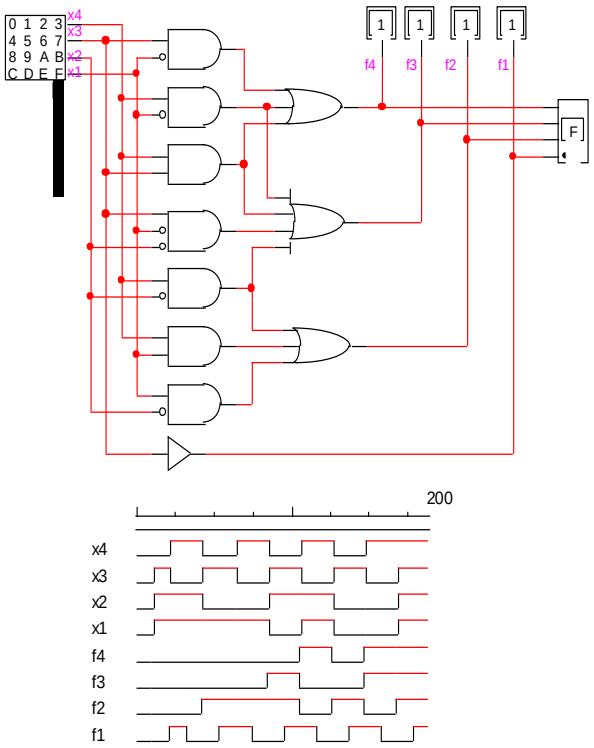


Fig. 2.3 Circuitul convertorului de cod 87(-2)(-7)→4221 şi diagrama lui de timp.

2.4 Lucrarea de laborator nr. 2

Tema: Sinteza convertoarelor de cod

Scopul lucrării: studierea practică a metodelor de sinteză a convertoarelor de cod.

Tema pentru acasă

1. Să se efectueze sinteza unui convertor de cod binar-zecimal în altul conform variantei din tabelul 2.3 (la indicaţia profesorului).

2. Funcţiile să se reprezinte în forma disjunctivă normală perfectă şi forma disjunctivă minimală. Pentru forma minimală să se prezinte schema în setul de elemente ŞI-NU.

Desfăşurarea lucrării

a) la standul de laborator:

1. Se verifică corectitudinea funcţionării circuitelor integrate ale standului de laborator.

2. Se asamblează şi se reglează schema convertorului de cod binar-zecimal din tema pentru acasă în setul de elemente ŞI-NU.

3. Pentru circuitele asamblate se determină costul şi timpul de reţinere.

b) în LogicWorks:

1. Din biblioteca de elemente **Simulation Gates.clf** se selectează elementele **NAND** cu numărul corespunzător de intrări. Din biblioteca **Simulation IO.clf** se selectează dispozitivele de intrare-ieşire **Binary Probe** şi **Hex Keyboard.**

2. Se asamblează schema convertorului de cod binar-zecimal din tema pentru acasă în setul de elemente ŞI-NU în **Fereastra de lucru** şi se verifică corectitudinea lui. Se studiază diagrama de timp.

3. Pentru circuitul asamblat se determină costul şi timpul de reţinere.

Tabelul 2.3

Nr. var.	Codul binar-zecimal intrare	Codul binar-zecimal ieşire	Nr. var.	Codul binar-zecimal intrare	Codul binar-zecimal ieşire
1.	8 4 2 (-1)	4 4 2 1	16.	4 4 2 1	8 4 2 (-1)
2.	8 4 2 (-3)	5 2 1 1	17.	5 2 1 1	8 4 2 (-3)
3.	8 4 1 (-2)	5 2 2 (-1)	18.	5 2 2 (-1)	8 4 1 (-2)
4.	8 3 2 (-4)	5 3 2 (-1)	19.	5 3 2 (-1)	8 3 2 (-4)
5.	8 4 2 (-5)	5 2 2 1	20.	5 2 2 1	8 4 2 (-5)
6.	8 4 1 (-6)	5 3 1 (-1)	21.	5 3 1 (-1)	8 4 1 (-6)
7.	8 4 2 (-3)	5 3 2 (-1)	22.	5 3 2 (-1)	8 4 2 (-3)
8.	8 7 (-2)(-4)	3 3 2 1	23.	3 3 2 1	8 7 (-2)(-4)
9.	8 6 (-1)(-4)	4 2 2 1	24.	4 2 2 1	8 6 (-1)(-4)
10.	8 5 (-2)(-4)	4 3 1 1	25.	4 3 1 1	8 5 (-2)(-4)
11.	8 4 3 (-6)	4 3 2 (-1)	26.	4 3 2 (-1)	8 4 3 (-6)
12.	8 6 1 (-4)	4 3 2 1	27.	4 3 2 1	8 6 1 (-4)
13.	8 5 2 (-4)	4 4 1 (-2)	28.	4 4 1 (-2)	8 5 2 (-4)
14.	8 4 3 (-2)	4 4 2 (-1)	29.	4 4 2 (-1)	8 4 3 (-2)
15.	8 4 2 1	4 4 3 (-2)	30.	4 4 3 (-2)	8 4 2 1

Întrebări

1. Care este raportul dintre numărul de intrări şi numărul de ieşiri ale convertoarelor de cod ?

2. Enumeraţi etapele şi specificul procesului de sinteză a convertoarelor de cod.

2.5 Decodificatoarele şi codificatoarele

Prezentare teoretică

Decodificatorul este un element funcţional, care reprezintă un circuit logic combinaţional şi este destinat decodificării cuvintelor binare aplicate la intrările lui. Dacă notăm numărul de intrări ale decodificatorului prin ***n*** şi numărul de ieşiri prin ***m,*** atunci relaţia dintre aceste numere pentru un decodificator complet este de $m=2^n$. Fiecărei combinaţii de variabile de intrare, care se mai numesc şi variabile de selecţie îi corespunde o singură ieşire, care este activă cînd combinaţia respectivă se aplică la intrare, celelalte ieşiri fiind inactive.

Tabelul de adevăr la sinteza unui decodificator complet are dimensiunile de $n+m$ coloane şi 2^n rînduri. În primele ***n*** coloane sunt reprezentate toate 2^n combinaţii posibile ale variabilelor, care pot fi aplicate la intrările decodificatorului, iar în celelalte ***m*** sunt reprezentate valorile funcţiilor logice care descriu ieşirile decodificatorului. Specificul acestui tabel constă în faptul că fiecare funcţie poate avea valoarea egală cu unu doar pentru o singură combinaţie a variabilelor de intrare, iar pentru celelalte valorile ei sunt egale cu zero. De aceea, este inutilă minimizarea acestor funcţii şi, în consecinţă, fiecare din ele va fi egală cu o conjuncţie a variabilelor de intrare, iar schema unui decodificator complet va include ***m*** elemente logice ŞI cu ***n*** intrări fiecare.

Relaţia dintre numărul de intrări şi ieşiri poate fi şi $m<2^n$. În acest caz decodificatorul se numeşte incomplet şi cheltuielile de aparataj pot fi micşorate dacă la sinteza decodificatorului se iau în consideraţie combinaţiile neutilizate. În acest caz sinteza schemei decodificatorului practic se reduce la minimizarea a ***m*** funcţii logice parţial determinate. Particularităţile acestor funcţii sunt următoarele: numai pentru o singură combinaţie funcţia este egală cu unu, pentru $m-1$ combinaţii valoarea ei este

egală cu zero, iar pentru $\mathbf{2^n}$-$\boldsymbol{m}$ combinaţii funcţia nu este determinată.

Vom ilustra cele descrise mai sus printr-un exemplu de sinteză a unui decodificator binar-zecimal. În tabelul 2.4 este prezentată codificarea cifrelor zecimale cu ajutorul codului 842(-3). Tot aici este şi tabelul de adevăr pentru cele 10 funcţii, care descriu structura decodificatorului.

Tabelul 2.4

Cifra zeci-	Codul				Funcţiile									
	8	4	2	-3										
mală	x_4	x_3	x_2	x_1	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	1	0	0	0	0	0	0	0
3	0	1	1	1	0	0	0	1	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	1	0	0	0	0	0
5	1	0	0	1	0	0	0	0	0	1	0	0	0	0
6	0	1	1	0	0	0	0	0	0	0	1	0	0	0
7	1	0	1	1	0	0	0	0	0	0	0	1	0	0
8	1	0	0	0	0	0	0	0	0	0	0	0	1	0
9	1	1	0	1	0	0	0	0	0	0	0	0	0	1

Combinaţiile pentru care funcţiile nu sunt determinate: 0001, 0011, 1010, 1100, 1110, 1111.

În fig. 2.4 sunt prezentate digramele Karnaugh pentru minimizarea funcţiilor y_0 -y_9.

x_2x_1 \ x_4x_3	00	01	11	10
00	1		*	
01	*			
11	*		*	
10			*	*

y_0

x_2x_1 \ x_4x_3	00	01	11	10
00			*	
01	*	1	*	
11	*			
10			*	*

y_1

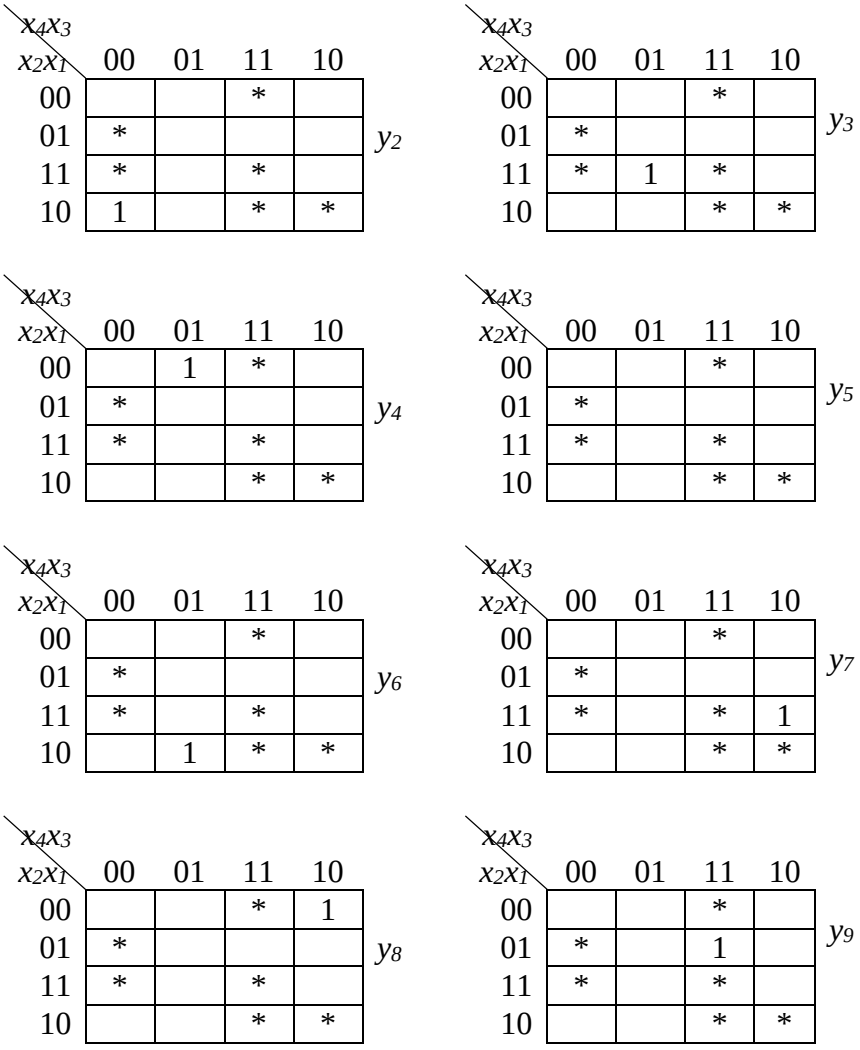


Fig. 2. 4 Diagramele Vetch-Karnaugh pentru minimizarea funcţiilor y_0-y_9.

În rezultatul minimizării au fost obţinute următoarele funcţii logice:

$y_0 = \bar{x}_4\bar{x}_3\bar{x}_2$ $y_1 = \bar{x}_4\bar{x}_2x_1$ $y_2 = \bar{x}_4\bar{x}_3x_2$ $y_3 = x_3x_2x_1$

$y_4 = x_3\bar{x}_2\bar{x}_1$ $y_5 = \bar{x}_3\bar{x}_2x_1$ $y_6 = x_3x_2\bar{x}_1$ $y_7 = x_4x_2$

$y_8 = x_4\bar{x}_1$ $y_9 = x_4x_3$

Schema decodificatorului binar-zecimal 842(-3) şi diagrama lui de timp sunt prezentate în figura 2.5.

Costul decodificatorului elaborat în baza acestor funcţii logice va fi de 27 unităţi Quine, spre deosebire de cazul clasic cînd costul ar fi fost de 40 unităţi Quine.

Codificatorul este un element funcţional, care reprezintă un circuit logic combinaţional şi este destinat codificării prin ***m*** biţi a uneia din ***n*** intrări active dintr-un număr maxim de $\mathbf{2^m}$ intrări. La sinteza codificatoarelor trebuie de ţinut cont de faptul, că concomitent nu pot fi active două sau mai multe intrări, de aceea la sinteza codificatoarelor în tabelul de adevăr fiecare combinaţie a variabilelor de intrare poate avea valoarea egală cu unu doar pentru o singură variabilă şi zero pentru toate celelalte. În acest caz sinteza codificatorului se reduce la reprezentarea fiecărei ieşiri prin disjuncţia variabilelor de intrare, care determină egalitatea cu unu a funcţiei respective.

Cele descrise mai sus sunt ilustrate mai jos printr-un exemplu de sinteză a unui codificator pentru codul binar-zecimal 8 5 –2 –4. Tabelul 4.2 reprezintă tabelul de adevăr pentru sinteza acestui codificator.

Cele descrise mai sus sînt ilustrate mai jos printr-un exemplu de sinteză a unui codificator pentru codul binar-zecimal 8 5–2 –4. Tabelul 2.5 reprezintă tabelul de adevăr pentru sinteza acestui codificator.

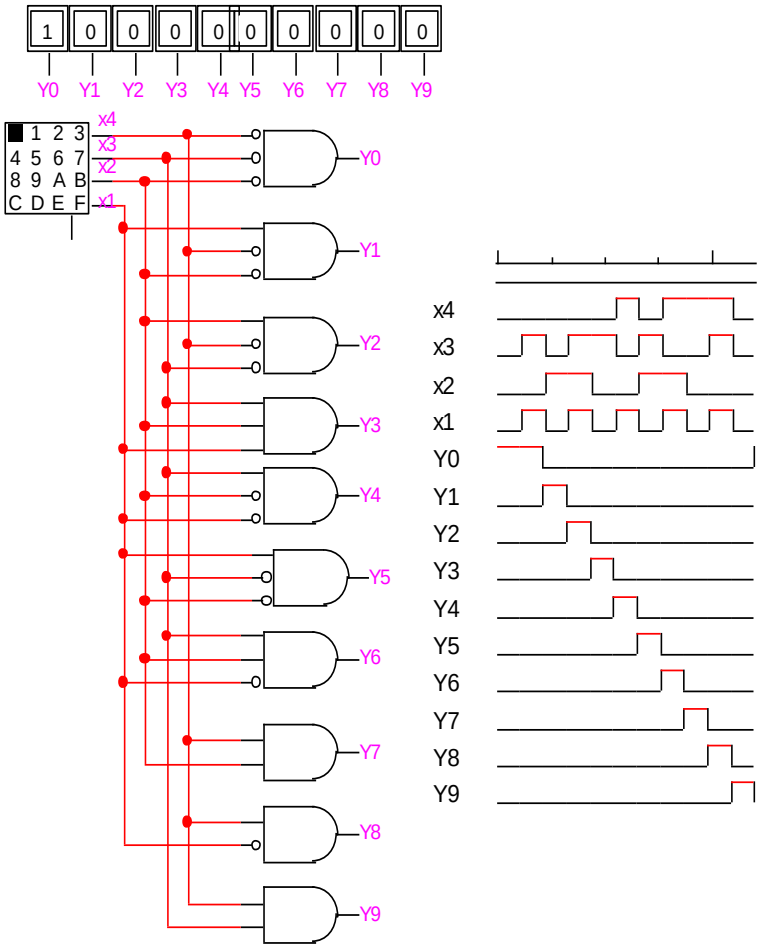


Fig. 2.5 Schema decodificatorului binar-zecimal 842(-3) şi diagrama lui de timp.

Tabelul 2.5

Cifra zecimală	Intrările										Ieşirile			
											8	5	-2	-4
	x_0	x_1	X_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	f_4	f_3	f_2	F_1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	1	0	1
2	0	0	1	0	0	0	0	0	0	0	1	0	1	1
3	0	0	0	1	0	0	0	0	0	0	0	1	1	0
4	0	0	0	0	1	0	0	0	0	0	1	0	0	1
5	0	0	0	0	0	1	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0	1	0	0	0	1	0	1	0
7	0	0	0	0	0	0	0	1	0	0	1	1	1	1
8	0	0	0	0	0	0	0	0	1	0	1	0	0	0
9	0	0	0	0	0	0	0	0	0	1	1	1	0	1

Setul de funcţii care realizează codificatorul din tabelul de mai sus este următorul:

$$\begin{aligned} f_4 &= x_2 \vee x_4 \vee x_6 \vee x_7 \vee x_8 \vee x_9; \\ f_3 &= x_1 \vee x_3 \vee x_5 \vee x_7 \vee x_9; \\ f_2 &= x_2 \vee x_3 \vee x_6 \vee x_7; \\ f_1 &= x_1 \vee x_2 \vee x_4 \vee x_7 \vee x_9. \end{aligned} \qquad (2.1)$$

Folosind legile lui De Morgan, transformăm relaţiile 2.1 pentru setul de elemente ŞI-NU:

$$f_4 = \overline{\bar{x}_2 * \bar{x}_4 * \bar{x}_6 * \bar{x}_7 * \bar{x}_8 * \bar{x}_9};$$
$$f_3 = \overline{\bar{x}_1 * \bar{x}_3 * \bar{x}_5 * \bar{x}_7 * \bar{x}_9};$$
$$f_2 = \overline{\bar{x}_2 * \bar{x}_3 * \bar{x}_6 * \bar{x}_7};$$
$$f_1 = \overline{\bar{x}_1 * \bar{x}_2 * \bar{x}_4 * \bar{x}_7 * \bar{x}_9}. \quad (2.2)$$

Schema codificatorului binar-zecimal 85(-2)(-4) şi diagrama lui de timp sunt prezentate în figura 2.6

2.6 Lucrarea de laborator nr. 3

Tema: Sinteza decodificatoarelor şi codificatoarelor

Scopul lucrării: studierea practică a structurii şi a metodelor de sinteză a decodificatoarelor şi codificatoarelor.

Tema pentru acasă

1. Efectuaţi sinteza unui decodificator complet cu trei variabile de intrare.
2. Efectuaţi sinteza unui decodificator binar-zecimal conform variantei din tabelul 4.3 (la indicaţia profesorului).
3. Efectuaţi sinteza unui codificator binar-zecimal conform variantei din tabelul 4.3 (la indicaţia profesorului).

Desfăşurarea lucrării
a) la standul de laborator:

1. Se verifică corectitudinea funcţionării circuitelor integrate ale standului de laborator.
2. Se asamblează şi se reglează schema unui decodificator

binar-zecimal din tema pentru acasă în setul de elemente ŞI-NU.

3. Se asamblează şi se reglează schema unui codificator binar-zecimal din tema pentru acasă în setul de elemente ŞI-NU.

4. Pentru circuitele asamblate se determină costul şi timpul de reţinere.

b) în LogicWorks:

1. Din biblioteca de elemente **Simulation Gates.clf** se selectează elementele **NAND** cu numărul corespunzător de intrări. Din biblioteca **Simulation IO.clf** se selectează dispozitivele de intrare-ieşire **Binary Probe**, **Hex Keyboard** şi **Binary Switch.**

2. Se asamblează schema unui decodificator binar-zecimal din tema pentru acasă în setul de elemente ŞI-NU în **Fereastra de lucru** şi se verifică corectitudinea lui. Se studiază diagrama de timp.

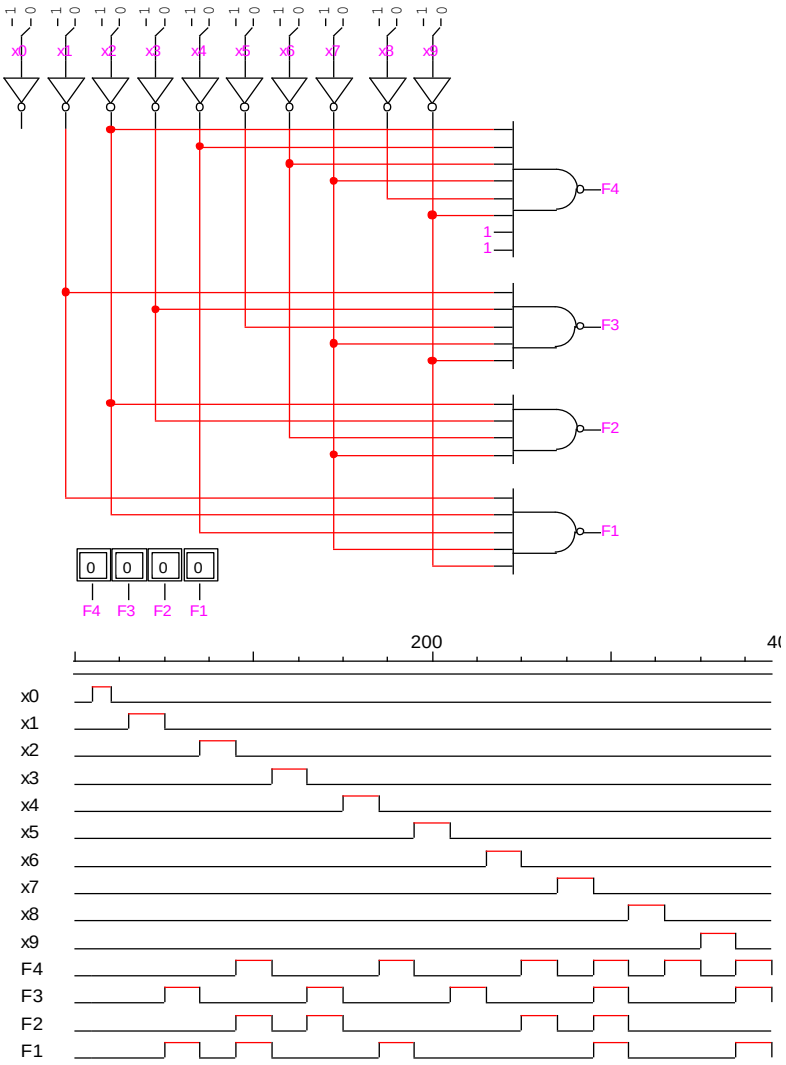


Fig. 2.6 Schema codificatorului binar-zecimal 85(-2)(-4) şi diagrama lui de timp.

Tabelul 2.6

Nr. crt.	Codul binar-zecimal		Nr. crt.	Codul binar-zecimal	
	Decodificator	Codificator		Decodificator	Codificator
1.	8 7 (-2)(-4)	5 3 2 (-1)	16.	3 3 2 1	8 6 1 (-4)
2.	8 6 (-1)(-4)	5 3 1 (-1)	17.	4 2 2 1	8 6 (-1)(-4)
3.	8 5 (-2)(-4)	3 3 2 1	18.	4 3 1 1	8 7 (-2)(-4)
4.	8 4 3 (-6)	4 2 2 1	19.	4 3 2 (-1)	8 5 (-2)(-4)
5.	8 6 1 (-4)	4 3 1 1	20.	4 3 2 1	8 6 1 (-4)
6.	8 5 2 (-4)	4 3 2 (-1)	21.	4 4 1 (-2)	8 5 2 (-4)
7.	8 4 3 (-2)	4 3 2 1	22.	4 4 2 (-1)	8 4 3 (-2)
8.	8 4 2 1	4 4 1 (-2)	23.	4 4 3 (-2)	8 4 2 1
9.	8 4 2 (-1)	4 4 2 (-1)	24.	4 4 2 1	8 4 2 (-1)
10.	8 4 2 (-3)	4 4 3 (-2)	25.	5 2 1 1	8 4 2 (-3)
11.	8 4 1 (-2)	4 4 2 1	26.	5 2 2 (-1)	8 4 1 (-2)
12.	8 3 2 (-4)	5 2 1 1	27.	5 3 2 (-1)	8 3 2 (-4)
13.	8 4 2 (-5)	5 2 2 (-1)	28.	5 2 2 1	8 4 2 (-5)
14.	8 4 1 (-6)	5 3 2 (-1)	29.	5 2 2 (-1)	8 4 1 (-6)
15.	8 4 1 (-2)	5 2 2 1	30.	5 3 2 (-1)	8 4 1 (-2)

3. Se asamblează schema unui codificator binar-zecimal din tema pentru acasă în setul de elemente ŞI-NU în **Fereastra de lucru** şi se verifică corectitudinea lui. Se studiază diagrama de timp.

4. Pentru circuitele asamblate se determină costul şi timpul de reţinere.

Întrebări

1. De ce depinde numărul de funcţii logice, care trebuie minimizate la sinteza decodificatoarelor incomplete şi care sunt particularităţile lor?

2. Care este raportul dintre numărul de intrări şi numărul de ieşiri ale decodificatoarelor complete şi incomplete.

2.7. Sumatoarele binare

Prezentare teoretică

Sumatoarele se includ în clasa de circuite logice combinaţionale, în care semnalele prelucrate sunt asociate unor numere. Sumatoarele execută operaţia de adunare a două numere şi operaţia de scădere, care constă în sumarea descăzutului cu codul complementar al scăzătorului.

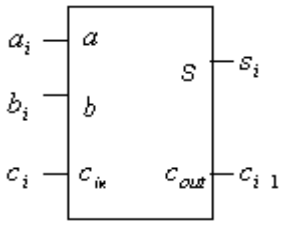
Realizarea structurii tuturor sumatoarelor porneşte de la două scheme logice de sumare pe un bit, cunoscute în literatura de specialitate ca schema semisumatorului şi respectiv schema sumatorului complet. Un rang al unui sumator binar complet are două intrări de date a cifrelor operanzilor din rangul respectiv şi o intrare de transport din rangul vecin mai puţin semnificativ şi produce la ieşire rezultatul sumei în rangul respectiv şi bitul de transport în rangul următor mai semnificativ. Tabelul de adevăr care descrie funcţionarea unui rang al sumatorului complet şi simbolul de reprezentare al acestuia sunt prezentaţi în fig. 2.7.

În rangul i sunt sumaţi doi biţi a_i şi b_i (valorile binare din rangul i ale ambilor operanzi) prezenţi la cele două intrări de date, precum şi bitul de transport c_i de la rangul vecin mai puţin semnificativ. Se generează două ieşiri s_i şi c_{i+1}, care sunt rezultatul sumei în rangul i şi respectiv cifra de transport în rangul următor mai semnificativ. Din tabelul de adevăr putem deduce, că sistemul de funcţii logice, cu ajutorul căruia se descrie structura unui rang a sumatorului binar complet este următoarea:

$$s_i = \bar{a}_i b_i \bar{c}_i \vee a_i \bar{b}_i \bar{c}_i \vee \bar{a}_i \bar{b}_i c_i \vee a_i b_i c_i = a_i \oplus b_i \oplus c_i;$$
$$c_{i+1} = a_i b_i \vee a_i c_i \vee b_i c_i$$

a_i	b_I	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

a)



b)

Fig. 2.7. Sumatorul complet: a) tabelul de adevăr; b) simbolul de reprezentare.

Pe baza sumatorului complet se poate realiza simplu structura unui sumator de ***n*** biţi. Pentru ***n***=4 structura este cea din fig. 2.8. Se observă propagarea succesivă a semnalului pe traseul bitului de transport. În consecinţă, rezultatul va fi disponibil la

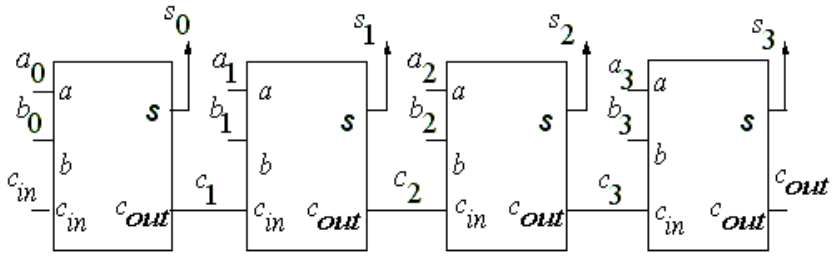


Figura 2.8 Structura sumatorului cu transport succesiv.

ieşire doar după ce semnalul corespunzător bitului de transport va parcurge întreg traseul. La sumatoarele cu propagarea succesivă a transportului, lanţul de propagare va introduce o întîrziere maximă în cazul cînd transportul produs în rangul cel mai puţin semnificativ al sumatorului se propagă prin toate rangurile lui pînă ajunge la cel mai semnificativ. Este evident că timpul maximal de funcţionare a sumatorului cu transport succesiv creşte în dependenţă liniară de numărul său de ranguri. Intervalul de timp mare de sumare este preţul plătit de sumatoarele cu transport succesiv pentru simplitatea structurii.

Sumatorul este una din cele mai importante componente a oricărui calculator, de aceea viteza lui de lucru reprezintă un criteriu de calitate primordial. Pentru creşterea vitezei de lucru a sumatoarelor au fost propuse şi implementate diverse măsuri de natură tehnologică şi arhitecturală. Măsurile luate în plan tehnologic pot fi de exemplu:

a) circuitele logice pe care le parcurge semnalul de transport trebuie proiectate în aşa fel, încît să fie redus timpul de propagare;

b) se operează cu semnal de transport inversat între intrare şi ieşire;

Aceste măsuri însă nu sunt suficiente atunci cînd trebuie realizate sumatoare pentru cuvinte cu un număr mare de biţi. Soluţia în acest caz este arhitecturală şi presupune renunţarea la transportul succesiv în favoarea celui anticipat (accelerat). Aceasta de fapt înseamnă redefinirea funcţiei logice a acelei părţi a sumatorului, care formează semnalul de transport.

Relaţia de definire a transportului poate fi scrisă sub forma:

$$c_{i+1} = a_i b_i \vee a_i c_i \vee b_i c_i = a_i b_i \vee (a_i \vee b_i) c_i$$

Notînd $G_i = a_i b_i$ şi $P_i = a_i \vee b_i$, relaţia de definire a transportului devine:

$$c_{i+1} = G_i \vee P_i c_i$$

Funcţia de generare $G_i=1$ indică faptul că din rangul ***i*** al sumatorului se generează o cifră de transport egală cu 1 indiferent de valoarea cifrei de transport, care vine din rangul vecin mai puţin semnificativ. Funcţia de propagare $P_i=1$ indică faptul că prin rangul ***i*** al sumatorului se va propaga valoarea transportului c_i de la ieşirea din rangul vecin mai puţin semnificativ.

Structura sumatorului cu transport anticipat este compusă din două module: modulul de sumare propriu-zisă, alcătuit din mai multe sumatoare de un rang şi modulul (schema) de transport anticipat, care generează simultan semnalele de transport pentru toate rangurile sumatorului. Pentru cuvinte cu lungimea de patru biţi schema sumatorului cu transport anticipat este prezentată în fig. 2.9. Modulul de sumare este format din patru sumatoare complete de un rang. Generarea semnalelor de transport se bazează pe următoarele relaţii:

$$c_1 = G_0 \vee P_0 c_{IN}$$
$$c_2 = G_2 \vee P_1 G_0 \vee P_1 P_0 c_{IN}$$
$$c_3 = G_2 \vee P_2 G_1 \vee P_2 P_1 G_0 \vee P_2 P_1 P_0 c_{IN}$$

Timpul total de sumare al unui sumator cu transport anticipat este egal cu timpul de sumare al unui sumator de un rang plus întîrzierea introdusă de schema transportului anticipat, nedepinzînd de numărul de ranguri.

2.8 Lucrarea de laborator nr. 4

Tema:Sinteza sumatoarelor binare

Scopul lucrării: însuşirea deprinderilor practice de sinteză a

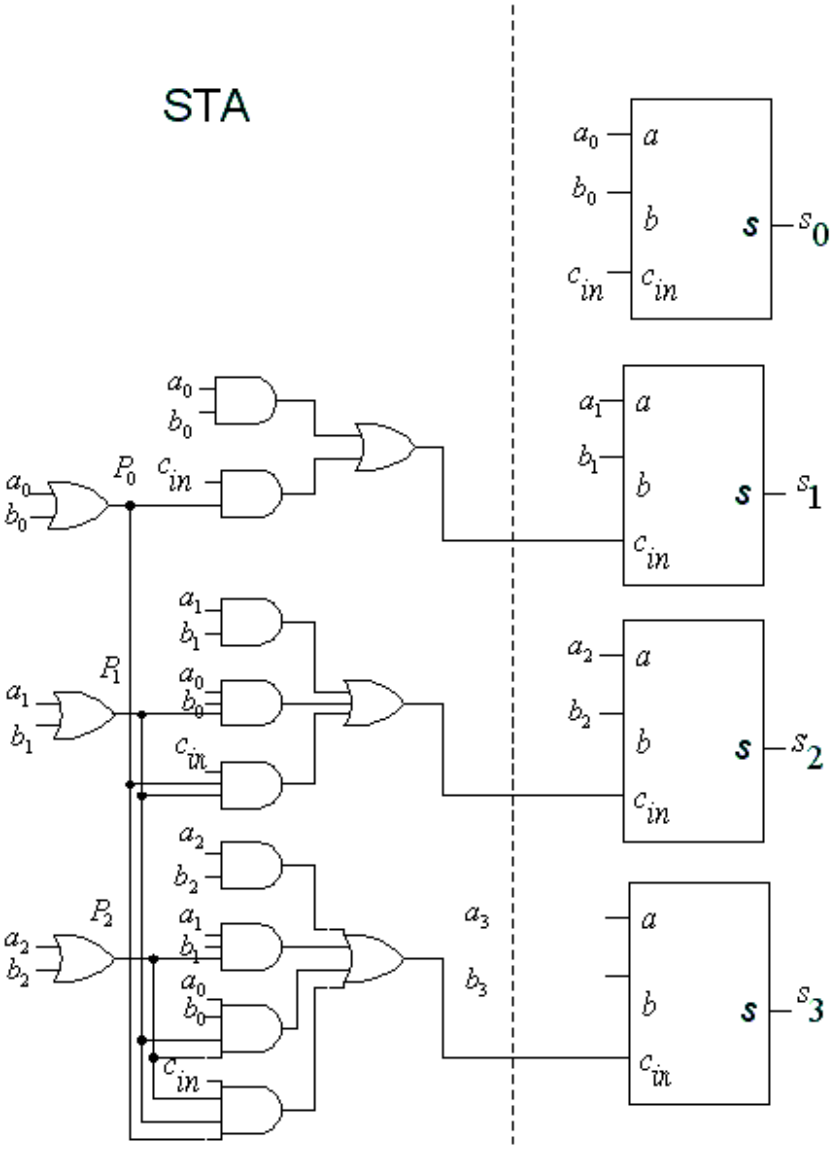


Figura 2.9. Structura sumatorului cu transport anticipat

sumatorului complet şi a sumatorului binar cu transport succesiv şi anticipat.

Tema pentru acasă

1. De efectuat sinteza sumatorului complet în setul de elemente ŞI-NU.
2. De efectuat sinteza sumatorului binar pe 4 biţi cu transport anticipat în setul de elemente ŞI-NU.

Desfăşurarea lucrării

a) la standul de laborator:

1. Se verifică corectitudinea funcţionării circuitelor integrate ale standului de laborator.
2. Se asamblează şi se reglează schema sumatorului binar complet.
3. Se asamblează şi se reglează schema sumatorului binar pe 4 biţi cu transport anticipat. Se realizează prin intermediul acestei scheme un exemplu de adunare şi unul de scădere (la indicaţia profesorului).
4. Pentru circuitele asamblate se determină costul şi timpul de reţinere.

b) în LogicWorks:

1. Din biblioteca de elemente **Simulation Gates.clf** se selectează elementele **NAND şi XOR** cu numărul corespunzător de intrări. Din biblioteca **Simulation IO.clf** se selectează dispozitivele de intrare-ieşire **Binary Probe**, **Hex Keyboard** şi **Binary Switch.**
2. Se asamblează două scheme ale sumatorului binar complet
a) în setul de elemente ŞI-NU;
b) utilizând elemente XOR şi SI-NU.

Se verifică corectitudinea lor. Se studiază diagramele de timp.

3. Se asamblează schema sumatorului binar pe 4 biţi cu transport anticipat utilizând elemente XOR pentru fiecare rang de sumare şi elemente ŞI-NU pentru transportul anticipat. Se verifică corectitudinea schemei. Se studiază diagrama de timp.

4. Pentru circuitele asamblate se determină costul şi timpul de reţinere.

Întrebări

1. Determinaţi timpul maximal de sumare pentru structurile din figurile 2.8 şi 2.9 considerînd că reţinerea semnalului într-un element logic este egală cu τ.

2. Care sunt restricţiile care intervin în sinteza schemei transportului anticipat odată cu creşterea lungimii operanzilor?

2.9 COMPARATOARELE

Prezentare teoretică

Comparatoarele sunt incluse în clasa circuitelor logice combinaţionale care asigură compararea cuvintelor binare. Comparatorul elementar reprezintă un circuit combinaţional simplu capabil să detecteze egalitatea a două cuvinte binare aplicate la intrările de date, furnizînd la ieşire valoarea respectivă a unui bit special de egalitate. Evident, că cuvintele de la intrările de date ar putea să nu fie egale. În cazul cînd cuvintele comparate sunt numere, atunci prin comparare se va putea stabili nu doar dacă sunt egale sau nu cuvintele respective, ci şi care din ele este mai mare sau mai mic. În acest caz comparatorul va fi mai complex şi va produce trei semnale de ieşire: bitul de superioritate, bitul de inferioritate şi bitul de egalitate.

Cele mai simple sunt comparatoarele, la care unul

dintre cuvintele, ce se compară este cunoscut dinainte, adică este o constantă. De aceea ele se numesc comparatoare cu constante şi pot fi folosite atît la detectarea egalităţii, cît şi a inegalităţii cuvintelor comparate. Structura acestor comparatoare este foarte simplă şi sinteza lor se efectuează în felul următor. Pentru detectarea egalităţii se foloseşte un element logic ŞI, numărul de intrări al căruia este egal cu numărul de biţi al cuvintelor comparate. Cuvîntul ce se compară cu constanta se aplică la intrările elementului logic în felul următor. Dacă valoarea constantei în rangul ***i*** este egală cu 1, atunci rangul respectiv se aplică direct, iar dacă această valoare este 0, atunci rangul respectiv se aplică printr-un inversor. În consecinţă la ieşirea elementului ŞI respectiv vom avea 1 logic doar atunci cînd cuvîntul aplicat va coincide cu constanta respectivă.

Pentru a efectua sinteza unui comparator de detectare a inegalităţii unui cuvînt cu o constantă se foloseşte un element logic SAU, numărul de intrări al căruia este egal cu numărul de biţi al cuvintelor comparate. Aplicarea la intrările elementului logic a cuvîntului ce se compară cu constanta se efectuează în felul următor. Dacă valoarea constantei în rangul ***i*** este egală cu 0, atunci rangul respectiv se aplică direct, iar dacă această valoare este 1, atunci rangul respectiv se aplică printr-un inversor. La ieşirea elementului SAU în acest caz vom avea 1 logic atunci cînd cuvîntul aplicat nu va coincide cu constanta respectivă şi 0 doar în cazul cînd cuvintele comparate vor coincide.

Sinteza unui comparator, care este destinat de a stabili dacă un număr oarecare face sau nu parte dintr-un interval, se efectuează prin metoda clasică, adică tabelul de adevăr al funcţiei care exprimă ieşirea comparatorului, se completează cu 1 pentru combinaţiile ce exprimă valorile numerelor din intervalul respectiv şi cu 0 pentru celelalte combinaţii. După aceea funcţia respectivă se minimizează.

Sinteza comparatoarelor, destinate stabilirii care din două numere este mai mare, este practic imposibilă dacă se

folosesc metodele clasice. Să presupunem, că este necesară sinteza unui comparator, care ar stabili egalitatea sau care din două numere de opt biţi este mai mare sau mai mic. Metoda clasică de sinteză ar necesita construirea unui tabel de adevăr cu $2^{(8+8)}$=65536 rînduri şi ulterioara minimizare a funcţiei (funcţiilor) respective. În asemenea cazuri soluţia este utilizarea metodei de decompoziţie a problemei, care presupune soluţionarea prin fragmentare. Datorită fragmentării rezolvarea şi soluţionarea unei probleme complexe se reduce la formularea şi soluţionarea unor probleme mai simple. Să exemplificăm cele expuse, prezentînd în continuare sinteza comparatorului cu trei ieşiri.

Fie că avem de comparat două cuvinte binare $A=a_3a_2a_1a_0$ şi $B=b_3b_2b_1b_0$. Sinteza directă ar necesita scrierea formelor canonice pentru trei funcţii $F_{A=B}$ – de egalitate, $F_{A>B}$ – de superioritate, $F_{A<B}$ – de inferioritate dintr-un tabel de adevăr cu $2^{(4+4)}$ = 256 rînduri. Practic sinteza se va realiza prin compararea separată a cifrelor de rang 3, 2, 1, 0. Pentru aceasta este necesară sinteza unui element de comparare, care compară două cuvinte de un bit, producînd trei ieşiri. Pentru aceste ieşiri vom obţine funcţiile f_e – de egalitate, f_s - de superioritate şi f_i - de inferioritate. Apoi cu ajutorul elementului proiectat se va construi un comparator de patru biţi.

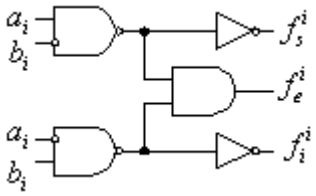
În figura 2.10 a) este prezentat tabelul de adevăr pentru funcţiile f_e, f_s şi f_i. Aceste trei funcţii se reprezintă cu ajutorul următoarelor expresii logice:

$$\begin{aligned} f_e &= \bar{a}_i\bar{b}_i \vee a_i b_i = \overline{a_i\bar{b}_i} \wedge \overline{\bar{a}_i b_i} \\ f_s &= a_i\bar{b}_i \\ f_i &= \bar{a}_i b_i \end{aligned} \qquad (2.3)$$

Circuitul care realizează funcţiile logice de mai sus este prezentat în fig. 2.10 b).

a_i	b_i	f_e	f_s	f_i
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

a) Tabelul de adevăr



b) Schema logică

Figura 2.10. Elementul de comparare a doi biţi.

Folosind relaţiile (2.3) putem scrie cele trei funcţii logice în conformitate cu care funcţionează comparatorul de patru biţi:

- relaţia de egalitate $A=B$ se exprimă prin funcţia logică:

$$F_{A=B} = f_{e3} f_{e2} f_{e1} f_{e0}, \qquad (2.4)$$

deoarece relaţia $A=B$ presupune că $a_3=b_3$, $a_2=b_2$, $a_1=b_1$ şi $a_0=b_0$;

- relaţia de superioritate $A>B$ presupune că $a_3>b_3$; sau $a_3=b_3$ şi $a_2>b_2$; sau $a_3=b_3$ şi $a_2=b_2$ şi $a_1>b_1$; sau $a_3=b_3$ şi $a_2=b_2$ şi $a_1=b_1$ şi $a_0>b_0$ ceea ce conduce la funcţia logică:

$$F_{A>B} = f_{s3} \vee f_{e3} f_{s2} \vee f_{e3} f_{e2} f_{s1} \vee f_{e3} f_{e2} f_{e1} f_{s0} \qquad (2.5)$$

relaţia de inferioritate $A<B$ presupune că $a_3<b_3$; sau $a_3=b_3$ şi $a_2<b_2$; sau $a_3=b_3$ şi $a_2=b_2$ şi $a_1<b_1$; sau $a_3=b_3$ şi $a_2=b_2$ şi $a_1=b_1$ şi $a_0<b_0$ de unde rezultă funcţia logică:

$$F_{A<B} = f_{i3} \vee f_{e3} f_{i2} \vee f_{e3} f_{e2} f_{i1} \vee f_{e3} f_{e2} f_{e1} f_{i0} \qquad (2.6)$$

Schema comparatorului paralel care realizează funcţiile (2.4), (2.5) şi (2.6) este prezentată în fig. 2.11 (este notat prin C_i elementul pentru compararea cifrelor de rangul i).

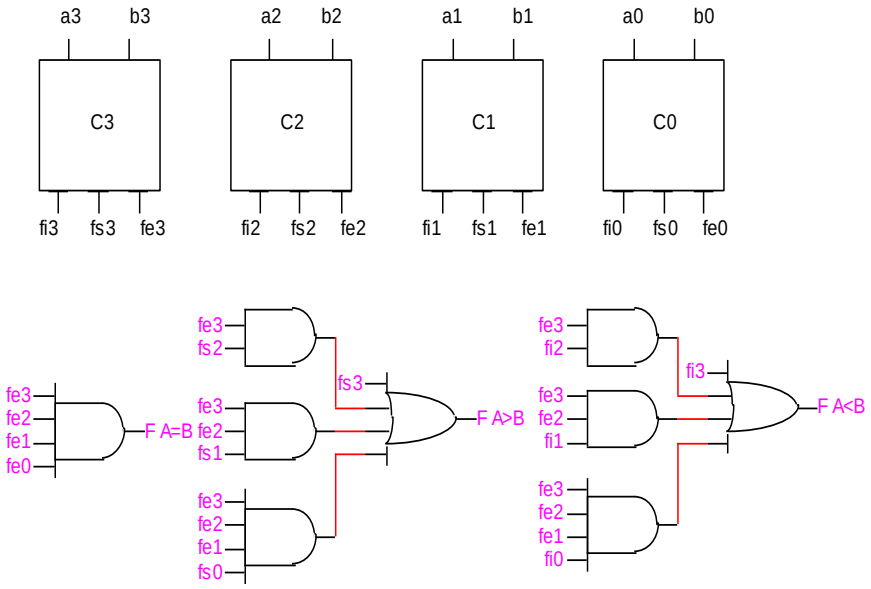


Fig. 2.11 Modulul comparator paralel de patru biţi

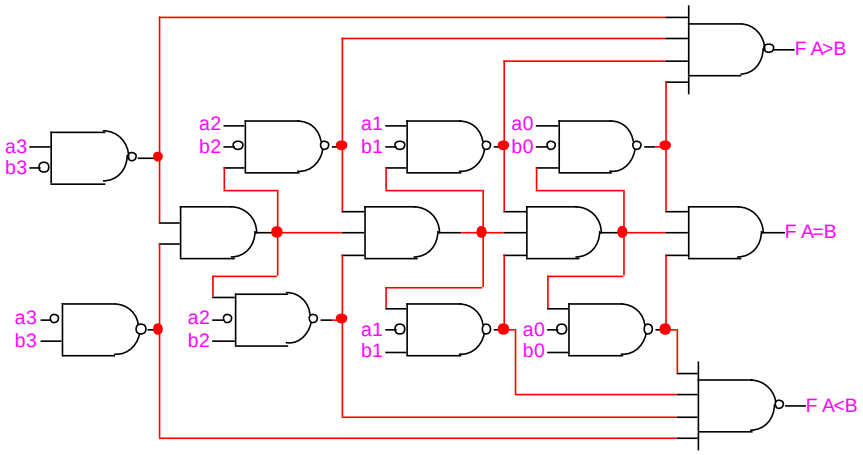


Figura 2.12 Modulul comparator succesiv de patru biţi

Acelaşi circuit este realizat în varianta succesivă în figura 2.12.

Altă modalitate de sinteză a comparatorului presupune realizarea operaţiei ***(A-B)*** pe un sumator, după care urmează analiza rezultatului obţinut. După cum se ştie, sumatoarele execută operaţia de scădere prin sumarea descăzutului la codul complementar al scăzătorului. De aceea cuvîntul ***A*** se va aplica direct la una dintre cele două intrări de date ale sumatorului, iar pentru a obţine codul complementar al lui ***B*** acesta trebuie aplicat la a doua intrare de date, fiind în prealabil inversat, iar la intrarea de transport a celui mai puţin semnificativ rang al sumatorului trebuie aplicat unu logic.

Pentru a stabili cum se va determina relaţia dintre cuvintele comparate cu ajutorul sumatorului vom lua ca exemplu două numere pozitive (bitul semnului lipseşte) ***A*** şi ***B*** cu lungimea de patru biţi. Efectuăm operaţia de scădere şi analizăm rezultatele obţinute pentru toate cele trei cazuri posibile: ***A>B, A=B, A<B.***

$A>B$	$A=B$	$A<B$
$\boldsymbol{A}=11_{(10)}=1011_{(2)}$ $\boldsymbol{B}=10_{(10)}=1010_{(2)}$	$\boldsymbol{A}=11_{(10)}=1011_{(2)}$ $\boldsymbol{B}=11_{(10)}=1011_{(2)}$	$\boldsymbol{A}=9_{(10)}=1001_{(2)}$ $\boldsymbol{B}=12_{(10)}=1100_{(2)}$
(A-B)$\Rightarrow$ 1011 + 0101 1 $\underbrace{01}_{C_{out}}\ \underbrace{0001}_{S}$	***(A-B)***$\Rightarrow$ 1011 + 0100 1 $\underbrace{01}_{C_{out}}\ \underbrace{0000}_{S}$	***(A-B)***$\Rightarrow$ 1001 + 0011 1 $\underbrace{00}_{C_{out}}\ \underbrace{1101}_{S}$

Din exemplul prezentat rezultă că:

- relaţia de superioritate ***A>B*** are loc cînd cifra

transportului următor C_{out}=1 şi suma S≠0;

- relaţia de egalitate A=B are loc cînd suma S=0.
- relaţia de inferioritate A<B este adevărată cînd cifra transportului următor C_{out}=0.

Schema comparatorului obţinută în urma analizei efectuate mai sus pentru două cuvinte de patru biţi este prezentată în fig. 2.13

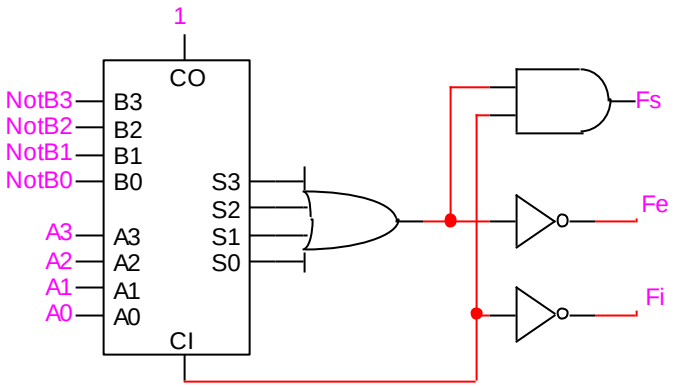


Fig. 2.13 Comparatorul de patru biţi în baza sumatorului

2.10 Lucrarea de laborator nr. 5

Tema: Sinteza comparatoarelor

Scopul lucrării: însuşirea deprinderilor practice de sinteză a diferitor tipuri de comparatoare.

Tema pentru acasă

1. Efectuaţi sinteza unui comparator cu cinci intrări şi trei

ieşiri conform variantei proprii din tabelul 2.7, avînd în vedere că:

I ieşire – compararea la egalitate cu o constantă

II ieşire – compararea la inegalitate cu o constantă

III ieşire – depistarea intervalului

2. Efectuaţi sinteza modulului comparator de patru biţi, utilizînd setul de elemente SI-NU.

3. Efectuaţi sinteza comparatorului de opt biţi în baza sumatorului

Tabelul 2.7

Nr. crt.	I ieşire	II ieşire	III ieşire	Nr. crt.	I ieşire	II ieşire	III ieşire
1.	1.	30	10-19	16.	16.	15	7-15
2.	2.	29	11-19	17.	17.	14	7-16
3.	3.	28	12-18	18.	18.	13	7-17
4.	4.	27	10-20	19.	19.	12	8-19
5.	5.	26	1-11	20.	20.	11	8-20
6.	6.	25	2-10	21.	21.	10	8-18
7.	7.	24	3-12	22.	22.	9	9-19
8.	8.	23	4-12	23.	23.	8	9-24
9.	9.	22	14-19	24.	24.	7	9-15
10.	10.	21	1-14	25.	25.	6	9-26
11.	11.	20	13-20	26.	26.	5	10-21
12.	12.	19	16-29	27.	27.	4	10-22
13.	13.	18	1-26	28.	28.	3	10-23
14.	14.	17	1-27	29.	29.	2	15-24
15.	15.	16	14-28	30.	30.	1	16-25

Desfăşurarea lucrării

a) la standul de laborator:

1. Se verifică corectitudinea funcţionării circuitelor integrate ale standului de laborator.

2. Se asamblează şi se reglează schema comparatorului cu cinci intrări şi trei ieşiri.

3. Se asamblează şi se reglează schema modulului comparator de patru biţi. Se efectuează compararea a două cuvinte binare de patru biţi (la indicaţia profesorului).

4. Se asamblează şi se reglează schema comparatorului de opt biţi în baza sumatorului. Se efectuează compararea a două cuvinte binare de opt biţi (la indicaţia profesorului).

5. Pentru circuitele asamblate se determină costul şi timpul de reţinere.

b) în LogicWorks:

1. Din biblioteca de elemente **Simulation Gates.clf** se selectează elementele **NAND** cu numărul corespunzător de intrări. Din biblioteca **Simulation IO.clf** se selectează dispozitivele de intrare-ieşire **Binary Probe**, **Hex Keyboard** şi **Binary Switch.**

2. Se asamblează schema comparatorului cu cinci intrări şi trei ieşiri în **Fereastra de lucru** şi se verifică corectitudinea ei. Se studiază diagrama de timp.

3. Se asamblează schema modulului comparator de patru biţi. Se efectuează compararea a două cuvinte binare de patru biţi (la indicaţia profesorului). Se studiază diagrama de timp.

4. Se asamblează schema comparatorului de opt biţi în baza sumatorului. Se efectuează compararea a două cuvinte binare de opt biţi (la indicaţia profesorului). Se studiază diagrama de timp.

5. Pentru circuitul asamblat se determină costul şi timpul de reţinere.

Întrebări

1. Demonstraţi că funcţia de egalitate f_e din 2.3 poate fi descrisă cu relaţia $f_e = \overline{a_i \overline{b_i} \vee \overline{a_i} b_i}$. Cum poate fi

modificată, în acest caz implementarea funcţiei f_e din figura 2.10, b?

2. Care este viteza de lucru a comparatorului din fig. 2.11? Depinde ea de numărul de ranguri?

3. Efectuaţi o analiză comparativă a schemelor din figurile 2.11 şi 2.12.

4. E posibilă oare sinteza comparatorului numai cu funcţiile de superioritate f_s şi inferioritate f_i, excluzînd funcţia de egalitate? Dacă da, atunci scrieţi relaţiile pentru funcţiile logice $F_{A=B}$, $F_{A>B}$, $F_{A<B}$ şi caracterizaţi eventualul comparator sub aspectul costului de implementare şi al vitezei de lucru. Comparaţi-l cu cel din figura 2.11.

5. Modificaţi modulul comparator din figura 2.11 astfel încît să devină posibilă expandarea acestuia pentru compararea cuvintelor de 4 biţi.

3. SINTEZA CIRCUITELOR LOGICE SECVENŢIALE

3.1 Particularităţile procesului de sinteză a circuitele logice secvenţiale

Circuitele logice secvenţiale (CLS) se caracterizează prin faptul că în orice moment de timp vectorul de ieşire al circuitului depinde nu numai de semnalele de la intrare din acel moment (ignorînd timpul de propagare) ci şi de semnalele de la intrare aplicate în momentele de timp anterioare. Această dependenţă se datorează prezenţei în CLS a unor elemente de memorie, care reprezintă mai multe stări logice stabile, comandate prin intrări şi participă la formarea semnalelor de ieşire. Rezultă că un CLS conţine o schemă combinaţională completată cu o structură de memorare.

Sinteza unui CLS se efectuează în următoarele etape:

- descrierea necesităţilor ce trebuie să le rezolve circuitul respectiv (prin text, desen, diagrame etc.);

- reprezentarea acestei descrieri sub forma unui tabel de tranziţie;
- deducerea funcţiilor logice şi minimizarea acestora;
- implementarea acestor funcţii minimizate sub forma unor reţele de comutare prin intermediul circuitelor integrate;

Tabelul de tranziţie se deosebeşte de tabelul de adevăr prin faptul că aici trebuie luat în consideraţie şi factorul timpului. Acest tabel include următoarele părţi componente:

- valoarea variabilelor de intrare la orice moment de timp ***t***;
- valoarea funcţiilor (vectorilor) de ieşire la momentul de timp ***t***;
- valoarea funcţiilor(vectorilor) de ieşire, care vor rezulta la momentul de timp ***t+1*** în urma aplicării semnalelor de intrare în momentul de timp precedent ***t.***

În calitate de valori a funcţiilor logice, care trebuie deduse şi minimizate, se iau valorile funcţiilor de ieşire la momentul ***t+1***, iar în calitate de variabile a acestor funcţii se iau valorile variabilelor de intrare la momentul ***t*** şi valorile funcţiilor de ieşire la momentul ***t***, adică toate datele care asigură tranziţia circuitului respectiv de la starea care o are în momentul ***t*** la starea care trebuie s-o aibă în momentul ***t+1***.

3.2 Circuitele basculante bistabile

Circuitele basculante bistabile **(**CBB) sunt circuite elementare secvenţiale şi se caracterizează prin faptul că tot timpul prezintă una din două posibile stări stabile. Aceste două stări codificabile prin cele două valori binare (0 sau 1) se utilizează de regulă în calculatoarele numerice ca suport pentru memorarea unui bit.

Există mai multe tipuri de circuite basculante bistabile sau prescurtat bistabile. Primul ca ordine a apariţiei şi cel mai simplu este bistabilul de tip RS. Funcţionarea acestuia este descrisă de tabelul de tranziţie 3.1. În acest tabel *S* (Set) este intrarea de

scriere, R (Reset) este intrarea de ştergere, Q_t este starea bistabilului în momentul ***t***, iar Q_{t+1} este starea bistabilului în momentul ***t+1***, care rezultă în urma aplicării valorilor semnalelor respective la intrare şi stării bistabilului în momentul ***t***. Combinaţia $S = R = 1$ este interzisă deoarece în urma aplicării ei starea bistabilului va fi incertă din cauza asimetriei reale a elementelor logice folosite la implementarea bistabilului.

Tabelul 3.1

S	R	Q_t	Q_{t+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	*
1	1	1	*

Tabelul de tranziţie al bistabilului RS

Din tabelul 3.1 uşor se poate deduce că:

$$Q_{t+1} = S \vee \overline{R} Q_t, \tag{3.1}$$

cu condiţia

$$RS = 0, \tag{3.2}$$

Cu ajutorul formulelor de Morgan expresia (3.1) poate fi uşor transformată în una din următoarele două forme:

$$Q_{t+1} = S \vee \overline{R \vee \overline{Q}_t} \tag{3.3}$$

$$Q_{t+1} = \overline{\overline{S} \wedge \overline{\overline{R} Q_t}} \tag{3.4}$$

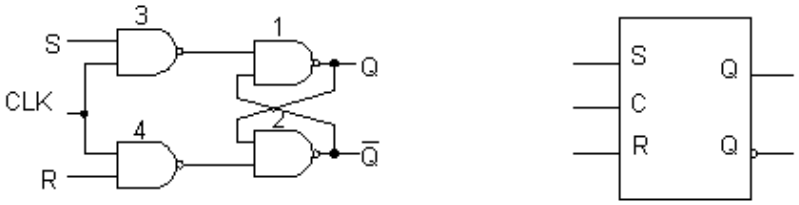
Expresiile (3.3) şi (3.4) pot fi implementate cu ajutorul a două elemente logice SAU-NU (figura 3.1, a) ori cu ajutorul a

două elemente logice ŞI-NU (figura 3.1, b) şi în aşa fel se obţine schema bistabilelor RS asincrone.

a) b)

Figura 3.1. circuitul bistabil RS asincron: a – în setul ŞI-NU; b – în setul SAU-NU.

În foarte multe aplicaţii este utilizat bistabilul *RS* sincron (figura 3.2). Elementele logice 1 şi 2 formează bistabilul propriu-zis, iar elementele 3 şi 4 realizează sincronizarea cu un semnal de ceas, *CLK*. Acest semnal activează intrările *S* şi *R* numai cînd se află în starea logică 1, inhibîndu-le pe durata cît se află în stare logică 0.



a) b)

Figura 7.2. Circuitul bistabil RS sincron: a)–schema logică; b)-simbolul.

Pornind de la bistabilul *RS*, se obţine bistabilul de tip *D* în

cazul cînd se stabileşte condiţia $S = \overline{R}$. În acest caz semnalul de la ieşire va fi identic cu cel de la intrare. Funcţionarea bistabilului de tip *D* este descrisă de relaţia

$$Q_{t+1} = D$$

în care *D* – intrarea de date. Această ultimă relaţie reflectă capacitatea bistabilului de tip *D* de a păstra direct o informaţie, aplicaţia principală a circuitului fiind memorarea cuvintelor binare. Cea mai simplă schemă a circuitului bistabil *D* sincron este prezentată în figura 3.3. Faptul că informaţia de la intrare este transmisă la ieşire şi apoi intrarea este inactivată a condus la denumirea de latch (lacăt) atribuită bistabilelor de tip *D*.

a) b)

Figura 7.3. Circuitul bistabil D sincron: a - schema;

b – simbolul.

Existenţa restricţiei *S*=*R*=1 constituie o dificultate pentru proiectanţi, de aceea foarte des se recurge la utilizarea circuitului bistabil de tip *JK*. Bistabilul *JK* îşi păstrează funcţionalitatea şi în cazul cînd *S*=*R*= 1. Această combinaţie a semnalelor de la intrare se foloseşte pentru a inversa starea bistabilului. Funcţionarea bistabilului *JK* este prezentată în tabelul de tranziţie 3.2, iar funcţia logică ce rezultă din acest tabel este reflectată de expresia logică (3.5).

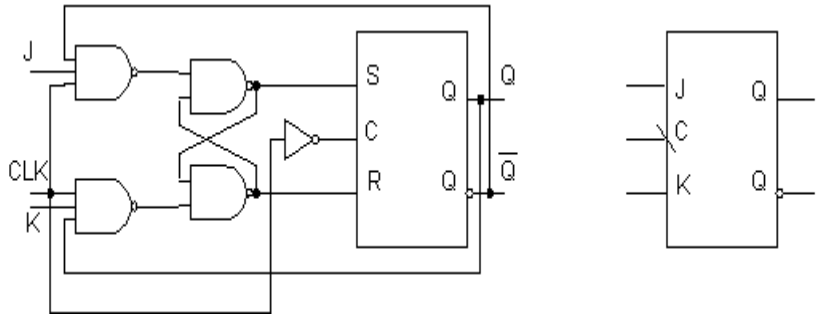
$$Q_{t+1} = J\overline{Q}_t \vee \overline{K}Q_t, \qquad (3.5)$$

Tabelul 3.2

Tabelul de tranziţie al bistabilului *JK*

J	K	Q_t	Q_{t+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Din tabelul 3.2 se observă că intrările *J* şi *K* pot fi tratate ca intrări *S* şi respectiv *R*. Însă în cazul, cînd *J*=*K*=1, bistabilul îşi schimbă starea anterioară (0→1, respectiv 1→0). Bistabilul *JK* se realizează de obicei în baza bistabilului *RS* într-o formă cunoscută sub numele de master-slave (stăpîn-sclav). Circuitul *JK* de tip master-slave (figura 3.4) conţine două bistabile *RS*. Analizînd modul de funcţionare a circuitului se poate constata că informaţia de la intrare este transferată la ieşire pe frontul negativ al semnalului de ceas. Asemenea circuite sunt denumite în literatura de specialitate ca bistabile flip-flop.



a) b)

Figura 3.4. Circuitul bistabil JK sincron: a) – schema; b) – simbolul.

Bistabilul *JK* este universal, avînd aplicaţii multiple. După cum s-a menţionat mai sus, el poate funcţiona în regim de bistabil *RS*. Dacă intrările *J* şi *K* sunt conectate între ele printr-un inversor, circuitul devine bistabil *D*, cu intrarea *D* pe linia *J*. Dacă *J*=*K*=1, atunci bistabilul *JK* devine bistabil de tip *T*, care basculează la fiecare impuls de ceas.

3.3 Registrele

Registrele se includ în categoria elementelor funcţionale secvenţiale şi sunt destinate memorării şi procesării cuvintelor binare. Componenta de bază a oricărui registru sunt bistabilele. Structura generală a unui registru este prezentată în figura 3.5. şi este constituită din ***n*** bistabile (în cazul de faţă de tip *D*), avînd un semnal de ceas CLK comun pentru toate bistabilele (B). Intrarea de ştergere $\overline{CLR}$, activă pe zero logic, este prezentă la majoritatea registrelor şi permite resetarea celor ***n*** bistabile. Intrările S_1 şi S_0 comandă cele ***n*** comutatoare logice (CM), asigurînd astfel selecţia regimului de lucru al registrului. Comutatoarele, în dependenţă de codul de selecţie, por asigura conectarea intrărilor bistabilelor în trei moduri: la ieşirea B_i din

stînga, din dreapta sau la intrarea de date *D*. În dependenţă de conectarea intrărilor bistabilelor, registrul poate încărca un cuvînt binar în cod paralel sau succesiv.

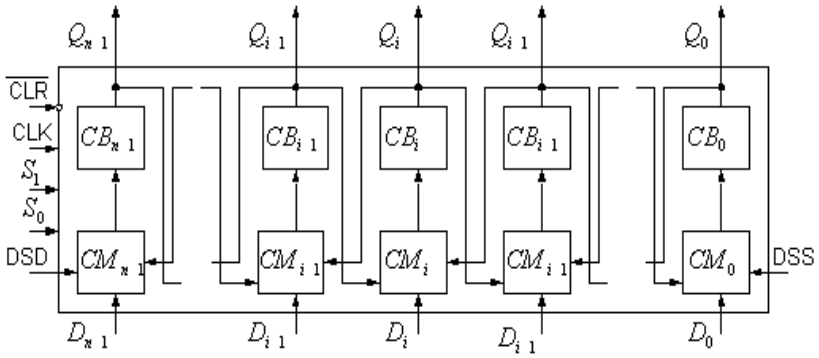


Figura 3.5 Structura generală a unui registru

În regimul de încărcare paralel, cuvîntul pentru înscriere se aplică la intrările de date D_{n-1}, …, D_0 – portul de intrare. Cuvîntul înscris este accesibil la ieşirile Q_{n-1}, …, Q_0 – portul de ieşire. Încărcarea datelor în registru se realizează la aplicarea semnalului de ceas.

În regimul succesiv de încărcare a datelor cuvîntul binar poate fi deplasat spre dreapta sau spre stînga. Pentru înscrierea succesivă se folosesc două intrări de date: spre dreapta *DSD* şi respectiv spre stînga *DSS*. Registrul capabil să deplaseze datele atît la stînga, cît şi la dreapta se numeşte registru cu deplasare bidirecţională. Acest registru poate fi utilizat nu numai pentru memorarea unui cuvînt binar, ci şi pentru procesarea lui, deoarece deplasarea spre dreapta cu ***i*** poziţii este echivalentă cu operaţia de împărţire a cuvîntului la $\boldsymbol{2^i}$ iar deplasarea spre stînga – cu operaţia de inmulţire cu $\boldsymbol{2^i}$.

Cuvîntul de ***n*** biţi înscris pe intervalul de ***n*** tacte prin intrarea DSD, respectiv DSS este pierdut secvenţial bit cu bit

la ieşirea Q_0, respectiv Q_{n-1} pe următorul interval de ***n*** tacte. Dacă însă se conectează ieşirea Q_0 la DSD, respectiv Q_{n-1} la *DSS* se obţine structura de registru în inel sau registru cu deplasare ciclică. Într-un asemenea registru cuvîntul înscris iniţial este recirculat în interiorul registrului.

În continuare vom exemplifica sinteza unui registru de patru biţi cu încărcare paralelă şi deplasare bidirecţională. Pentru o generalizare completă se va asigura, de asemenea, păstrarea cuvîntului încărcat în registru. Pentru a realiza păstrarea datelor, comutatoarele trebuie să aibă cîte o intrare suplimentară conectată la ieşirea bistabilului din acelaşi rang. Funcţionarea unui asemenea registru este descrisă în tabelul 3.3.

Tabelul 3.3

Regim de lucru	Intrări					Ieşiri				
	S_1	s_0	D_i	DSD	*DSS*	Q_3	Q_2	Q_1	Q_0	t
Păstrare	0	0	X	X	X	Q_3	Q_2	Q_1	Q_0	$t+1$
Deplasare stînga	0	1	X	X		Q_2	Q_1	Q_0	*DSS*	
Deplasare dreapta	1	0	X		X	*DSD*	Q_3	Q_2	Q_1	
Încărcare paralelă	1	1	D_i	X	X	D_3	D_2	D_1	D_0	

Notînd prin J_i, K_i intrările de setare, respectiv de resetare ale bistabilelor cu ieşirile Q_i şi folosind tabelul 3.1, se pot obţine relaţiile care descriu funcţionarea registrului în cele patru moduri de operare:

$$J_3 = \overline{K}_3 = \bar{s}_1\bar{s}_0 Q_3 \vee \bar{s}_1 s_0 Q_2 \vee s_1\bar{s}_0 DSD \vee s_1 s_0 D_3,$$
$$J_2 = \overline{K}_2 = \bar{s}_1\bar{s}_0 Q_2 \vee \bar{s}_1 s_0 Q_1 \vee s_1\bar{s}_0 Q_3 \vee s_1 s_0 D_2,$$
$$J_1 = \overline{K}_1 = \bar{s}_1\bar{s}_0 Q_1 \vee \bar{s}_1 s_0 Q_0 \vee s_1\bar{s}_0 Q_2 \vee s_1 s_0 D_1, \qquad 3.6$$
$$J_0 = \overline{K}_0 = \bar{s}_1\bar{s}_0 Q_0 \vee \bar{s}_1 s_0 DSS \vee s_1\bar{s}_0 Q_1 \vee s_1 s_0 D_0.$$

Schema registrului sintetizat este dată în figura 3.6.

3.4 Lucrarea de laborator nr. 6

Tema: SINTEZA REGISTRELOR

Scopul lucrării: Studierea registrelor şi metodelor lor de proiectare.

Tema pentru acasă

Conform variantei indicate de profesor efectuaţi sinteza unui registru pe 4 biţi, setul ŞI-NU, care realizează operaţiile definite în tabelul 3.4.

Tabelul 3.4.

Nr. var.	Bista-bilul	Modurile de operare	Nr. var.	Bista-bilul
1	JK	Încărcare paralelă Deplasare aritmetică stînga	13	D
2	JK	Încărcare paralelă Deplasare aritmetică dreapta	14	D
3	JK	Menţinere Deplasare aritmetică stînga	15	D
4	JK	Menţinere Deplasare aritmetică dreapta	16	D

5	JK	Încărcare paralelă Deplasare ciclică stînga	17	D
6	JK	Încărcare paralelă Deplasare ciclică stînga	18	D
7	JK	Menţinere Deplasare ciclică stînga	19	D
8	JK	Menţinere Deplasare ciclică dreapta	20	D
9	JK	Încărcare paralelă Deplasare aritmetică stînga	21	D
10	JK	Încărcare paralelă Deplasare aritmetică dreapta	22	D
11	JK	Menţinere Deplasare aritmetică stînga	23	D
12	JK	Menţinere Deplasare aritmetică dreapta	24	D

Desfăşurarea lucrării

a) la standul de laborator:

1. Se verifică corectitudinea funcţionării circuitelor integrate ale standului de laborator.

4. Se asamblează şi se reglează schema registrului (conform variantei). Se verifică pe un exemplu concret corectitudinea funcţionării lui.

5. Pentru circuitul asamblat se determină costul şi timpul de reţinere.

b) în LogicWorks:

1. Din biblioteca de elemente **Simulation Gates.clf** se selectează elementele necesare.. Din biblioteca **Simulation IO.clf** se selectează dispozitivele de intrare-ieşire **Binary Probe**, **Hex Keyboard, Binary Switch** şi **Clock.**

2. Se asamblează schema registrului (conform variantei). în **Fereastra de lucru** şi se verifică corectitudinea ei. Se studiază diagrama de timp.

3. Pentru circuitul asamblat se determină costul şi timpul de reţinere.

Întrebări

1. Demonstraţi că circuitul bistabil *JK* (fig. 3.4,a) se comportă ca un bistabil de tip *T* dacă *J=K=1*.

2. Realizaţi conversia bistabilelor $SR \rightarrow T$ şi $D \rightarrow T$. Prezentaţi schemele posibile.

3. Avem un registru de patru biţi realizat cu bistabile de tip *D* în care este înscris un cuvînt binar. Realizaţi conexiunile necesare astfel încît să se asigure într-un singur tact deplasarea la dreapta cu două ranguri.

4. Propuneţi schema unui registru cu bistabile de tip *D* care să asigure realizarea operaţiei de înmulţire cu 2 a cuvîntului binar înscris în registru.

5. Modificaţi schema registrului din fig. 3.6, utilizînd circuite multiplexoare pentru realizarea relaţiilor 3.6.

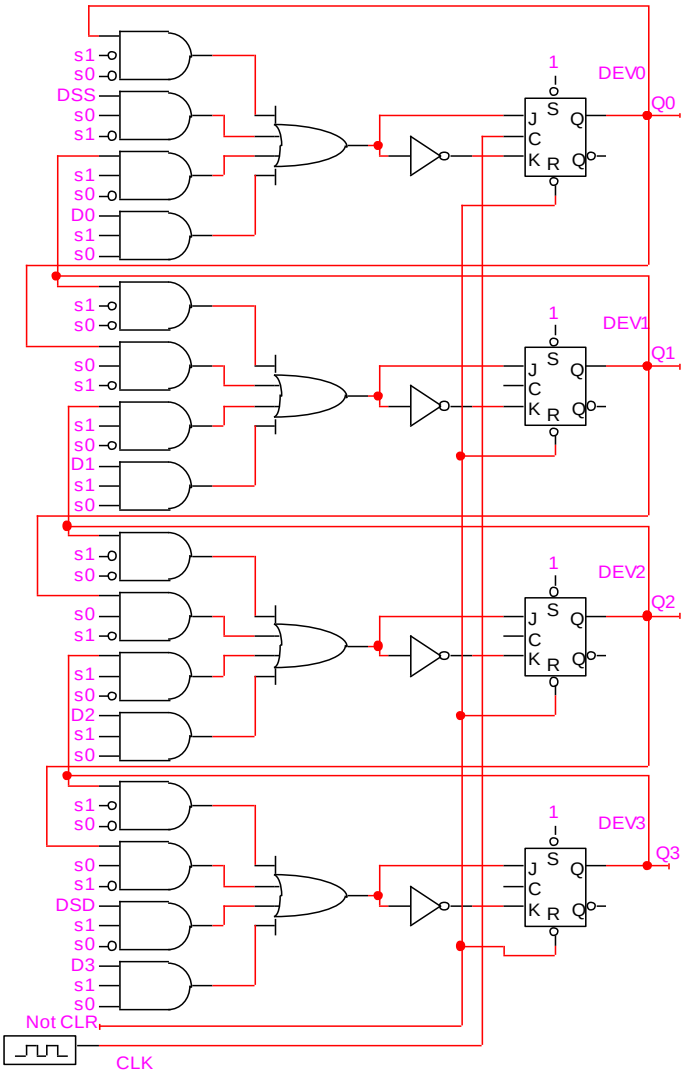


Figura 3.6. Registru cu deplasare bidirecţională**.**

3.5 Numărătoarele

Prezentare teoretică

O clasă importantă de circuite logice secvenţiale o reprezintă numărătoarele (contoarele). Destinaţia numărătorului este înregistrarea numărului de impulsuri aplicate la intrarea lui şi divizarea frecvenţei acestor impulsuri. Componenta de bază a numărătoarelor sunt bistabilele. Un numărător are ***M*** stări distincte. Tranziţia între stările succesive se produce în urma impulsurilor aplicate la intrarea numărătorului. După aplicarea unui număr de ***M*** impulsuri, numărătorul revine în starea iniţială (de exemplu în starea zero). Un astfel de circuit reprezintă un numărător modulo ***M***. Relaţia dintre modulo ***M*** şi numărul de bistabile ***n,*** ce intră în componenţa numărătorului este următoarea: $n = [\log_2 M]$. Dacă numărul ***n*** nu este întreg, atunci el se rotunjeşte pînă la cel mai apropiat număr întreg mai mare decît cel obţinut.

Numărătoarele se clasifică după două criterii: în dependenţă de ordinea numărării şi în dependenţă de faptul cum îşi schimbă starea bistabilele, Dacă numărătorul realizează o succesiune de stări codificate în ordine crescătoare, atunci el se numeşte numărător direct, iar dacă succesiunea stărilor este în ordine descrescătoare, atunci el se numeşte numărător invers. Numărătorul care realizează atît numărarea directă, cît şi cea inversă se numeşte reversibil. Numărătoarele se divizează în sincrone sau asincrone în dependenţă de faptul cum îşi schimbă starea bistabilele ce intră în componenţa lor. Dacă bistabilele îşi schimbă starea concomitent, atunci numărătoarele se numesc sincrone, iar dacă starea lor se schimbă succesiv numărătoarele se numesc asincrone.

Pentru un numărător modulo 8 vor fi necesare trei bistabile, succesiunea de numărare în cele două sensuri este prezentată în tabelul 3.5. Analiza succesiunii cifrelor de

rang 0 conduce la concluzia că succesiunea respectivă poate fi realizată în baza bistabilelor de tip *T*, care basculează la fiecare impuls de ceas. Cu alte cuvinte bistabilul *T* este un numărător modulo 2. Extinzînd analiza tabelului 3.5 pentru rangurile 1, 2, …, se poate observa că un numărător modulo $M=2^n$ se obţine prin înserierea a ***n*** bistabile de tip *T*. Într-o asemenea înseriere fiecare din bistabilele de tip *T* cînd revine în starea iniţială (zero) trebuie să transmită un semnal de comandă la intrarea de ceas a bistabilului aflat în rangul vecin mai semnificativ. Impulsurile de numărare trebuie aplicate la intrarea de ceas a bistabilului de rang 0.

Tabelul 3.5

Nr. de impul-suri	Numărare directă			Numărare inversă		
	rang			Rang		
	2	1	0	2	1	0
0	0	0	0	1	1	1
1	0	0	1	1	1	0
2	0	1	0	1	0	1
3	0	1	1	1	0	0
4	1	0	0	0	1	1
5	1	0	1	0	1	0
6	1	1	0	0	0	1
7	1	1	1	0	0	0
8	0	0	0	1	1	1

Schema numărătorului asincron, direct modulo 8 obţinut prin înserierea a trei bistabile JK, ajustate la bistabile T prin conectarea de unu logic la intrările J şi K, este prezentată în figura 3.7.

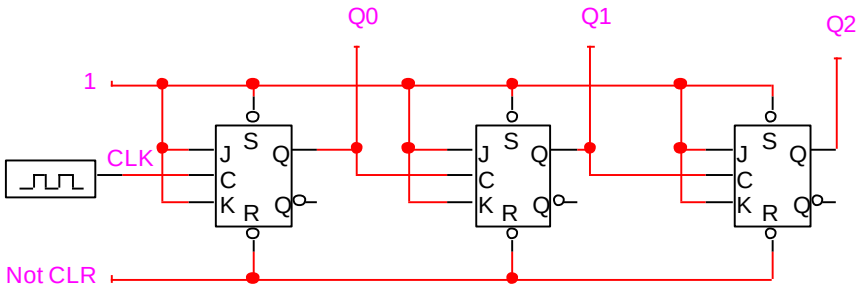


Figura 3.7. Numărător asincron modulo 8

Pe frontul negativ al primului impuls de numărare bistabilul Q_0 se va comuta din 0 în 1, iar Q_1 şi Q_2 vor rămîne în starea de zero logic. Pe frontul negativ al celui de-al doilea impuls de numărare Q_0 se va comuta din 1 în 0. Deoarece ieşirea Q_0 comandă intrarea de ceas a bistabilului Q_1, atunci frontul negativ la ieşirea Q_0 va duce la comutarea lui Q_1 din 0 în 1, Q_2 rămînînd în starea zero. Deci se observă că pe fiecare front negativ al impulsului de numărare basculează Q_0, Q_1 se inversează pe frontul negativ al ieşirii lui Q_0, iar Q_2 se inversează pe frontul negativ al lui Q_1. Pentru a realiza numărarea în sens invers conectarea bistabilelor din figura 3.7 se face astfel încît intrarea de ceas a fiecărui bistabil să fie comandată de ieşirea inversă a bistabilului din rangul vecin mai puţin semnificativ. Numărătorul din figura 3.7 este asincron, deoarece, după cum s-a menţionat mai sus, comutarea bistabilelor este succesivă în timp. Un numărător asincron are structura simplă dar şi rapiditatea lui este relativ mică, fiind determinată de comutarea succesivă a bistabilelor.

La numărătoarele sincrone impulsul de numărare se aplică simultan la intrările de ceas ale tuturor bistabilelor şi acestea se vor comuta sincron. Oricare bistabil al unui numărător sincron se comută simultan cu celelalte la aplicarea impulsului de numărare în starea, dependentă de valoarea semnalelor la

intrările lui. Se poate observa din analiza succesiunii directe (tabelul 3.5) că oricare bistabil trebuie să-şi inverseze starea numai atunci cînd toate bistabilele de rang inferior sunt în starea de unu logic. Pentru un numărător sincron modulo 16 condiţiile de basculare se stabilesc conform relaţiilor:

$$T_0=1,$$
$$T_1=Q_0,$$
$$T_2=Q_0Q_1, \qquad (3.7)$$
$$T_3=Q_0Q_1Q_2.$$

Schema numărătorului este prezentată în figura 3.8. În acest numărător este implementată propagarea paralelă a transportului.

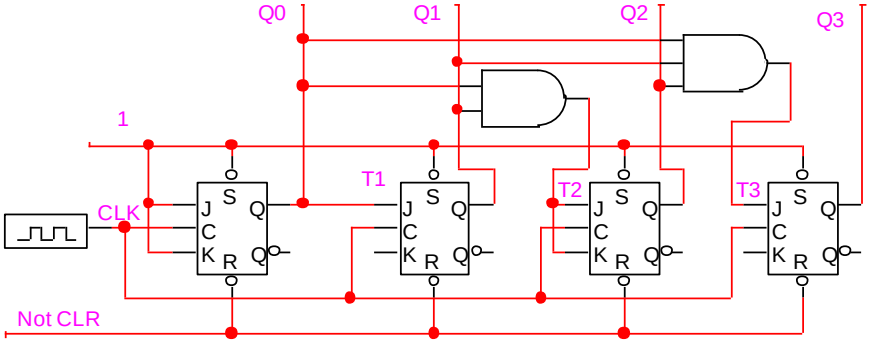


Figura 3.8 Numărător sincron cu transport paralel.

Condiţiile (3.7) pot fi rescrise astfel:

$$T_0=1,$$
$$T_1=T_0Q_0=Q_0,$$
$$T_2=T_1Q_1, \qquad (3.8)$$
$$T_3=T_2Q_2.$$

Implementarea relaţiilor (3.8) conduce la structura numărătorului sincron cu transport succesiv, care are avantajul utilizării de elemente logice ŞI numai cu două intrări, dar şi dezavantajul conectării succesive a elementelor ŞI.

Dacă se doreşte o funcţionare reversibilă,

numărătorul trebuie să conţină un semnal de comandă Up/Down. Acest semnal va determina sensul de numărare direct/invers prin selectarea fie a ieşirilor directe *Q*, fie a ieşirilor inverse *Q*. O structură reversibilă se obţine simplu dacă la ieşirea fiecărui bistabil este conectat un comutator logic care va selecta pentru numărarea directă ieşirea *Q*, iar pentru numărarea inversă ieşirea *Q*, după cum este arătat în figura 3.9.

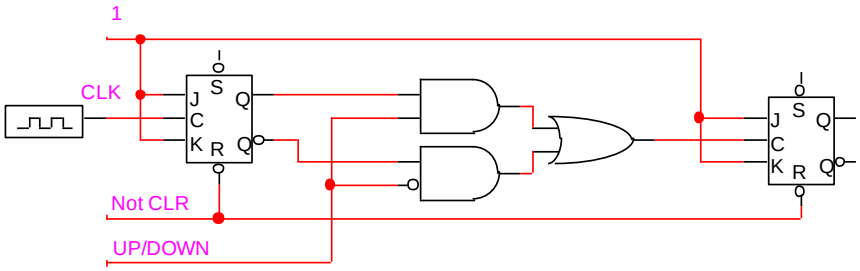


Figura 3.9 Numărător asincron reversibil.

Pînă acum au fost prezentate numărătoarele asincrone şi sincrone modulo $M=2^n$. În continuare vom prezenta sinteza numărătoarelor modulo *K*, $2^{n-1}<K<2^n$.

Structura unui numărător asincron modulo *K* se obţine dintr-un numărător asincron modulo 2^n la care se adaugă un decodificator. În momentul cînd starea binară a numărătorului are valoarea $K_{(10)}$ decodificatorul trebuie să activeze semnalul asincron de ştergere *CLR* pentru a reseta bistabilele numărătorului . În figura 3.10 este prezentat un numărător asincron modulo 10.

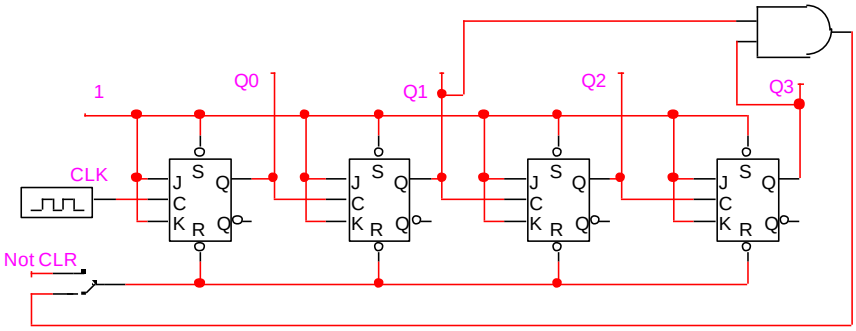


Figura 3.10 Numărător asincron modulo 10.

Un circuit numărător este în fond un automat de tip Moore, la care vectorul de ieşire este determinat de vectorul de stare. De aceea sinteza unui numărător sincron modulo ***M*** se efectuează în cîteva etape, care sunt următoarele:

- Se determină numărul de bistabile ale numărătorului conform relaţiei $n=[\log_2 M]$;
- Se elaborează tabelul de tranziţie al numărătorului, în care se completează coloanele pentru starea prezentă a numărătorului (momentul ***t***), starea următoare (momentul ***t+1***) şi valorile ce trebuie aplicate la intrările tuturor bistabilelor pentru a asigura tranziţia numărătorului din starea de la momentul ***t*** la starea de la momentul ***t+1***.
- Din tabelul de tranziţie al numărătorului se obţin funcţiile de instalare a bistabilelor, care se minimizează;
- În baza funcţiilor minimizate se elaborează circuitele de conexiune a bistabilelor între ele în urma implementării cărora se obţine schema numărătorului.

La sinteza unui numărător modulo ***M***, din cele 2^n stări posibile un număr de 2^n-***M*** sunt stări neutilizate (ilegale) şi se consideră ca nedeterminate în diagrama Veitch-Karnaugh. Se poate

întîmpla, însă, ca numărătorul la pornire sau sub influenţa unor semnale parazitare să nimerească în una din aceste stări. Dacă după cîteva tacturi numărătorul poate ajunge într-o stare legală, atunci funcţionarea lui de mai departe este corectă. Însă se poate întîmpla ca numărătorul să nu poată ieşi din stările ilegale, decît numai printr-o nouă pornire. Pentru a evita aceste cazuri, sinteza numărătorului trebuie efectuată în aşa fel ca în tabelul de tranziţie să se facă tranziţia spre starea iniţială din oricare din stările ilegale la următorul impuls de numărare.

Mai jos este prezentată sinteza unui numărător direct, sincron modulo 10 în baza bistabilelor *JK*. Numărul de bistabile necesar realizării acestui numărător este $n=[\log_2 10]=4$. În tabelul de tranziţie 3.6 Aceste bistabile sunt notate respectiv: Q_3,Q_2, Q_1, Q_0.

Tabelul 3.6

Ieşirile bistabilelor		Funcţiile de instalare a bistabilelor							
$(Q_3Q_2Q_1Q_0)$ t	$(Q_3Q_2Q_1Q_0)$ t+1	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0 0 0 0	0 0 0 1	0	*	0	*	0	*	1	*
0 0 0 1	0 0 1 0	0	*	0	*	1	*	*	1
0 0 1 0	0 0 1 1	0	*	0	*	*	0	1	*
0 0 1 1	0 1 0 0	0	*	1	*	*	1	*	1
0 1 0 0	0 1 0 1	0	*	*	0	0	*	1	*
0 1 0 1	0 1 1 0	0	*	*	0	1	*	*	1
0 1 1 0	0 1 1 1	0	*	*	0	*	0	1	*
0 1 1 1	1 0 0 0	1	*	*	1	*	1	*	1
1 0 0 0	1 0 0 1	*	0	0	*	0	*	1	*
1 0 0 1	0 0 0 0	*	1	0	*	0	*	*	1
1 0 1 0	0 0 0 0	*	1	0	*	*	1	0	*
1 0 1 1	0 0 0 0	*	1	0	*	*	1	*	1
1 1 0 0	0 0 0 0	*	1	*	1	0	*	0	*

1 1 0 1	0 0 0 0	*	1	*	1	0	*	*	1
1 1 1 0	0 0 0 0	*	1	*	1	*	1	0	*
1 1 1 1	0 0 0 0	*	1	*	1	*	1	*	1

Tabelul 3.7

Q_t	Q_{t+1}	J	K
0	0	0	*
0	1	1	*
1	0	*	1
1	1	*	0

Tabelul de tranziţie al numărătorului (Tabelul 3.6) este elaborat utilizînd tabelul de tranziţie al bistabilului *JK* modificat după cum este prezentat în tabelul 3.7. Funcţiile de instalare a bistabilelor sunt minimizate cu ajutorul diagramelor Veitch-Karnaugh după cum este arătat în figura 3.11, iar schema numărătorului este prezentată în figura 3.12.

Q_3Q_2 / Q_1Q_0	00	01	11	10
00			*	*
01			*	*
11		1	*	*
10			*	*

$J_3=Q_2Q_1Q_0$

Q_3Q_2 / Q_1Q_0	00	01	11	10
00	*	*	1	
01	*	*	1	1
11	*	*	1	1
10	*	*	1	1

$$K_3 = Q_2 \vee Q_1 \vee Q_0 = \overline{\overline{Q_2}\,\overline{Q_1}\,\overline{Q_0}}$$

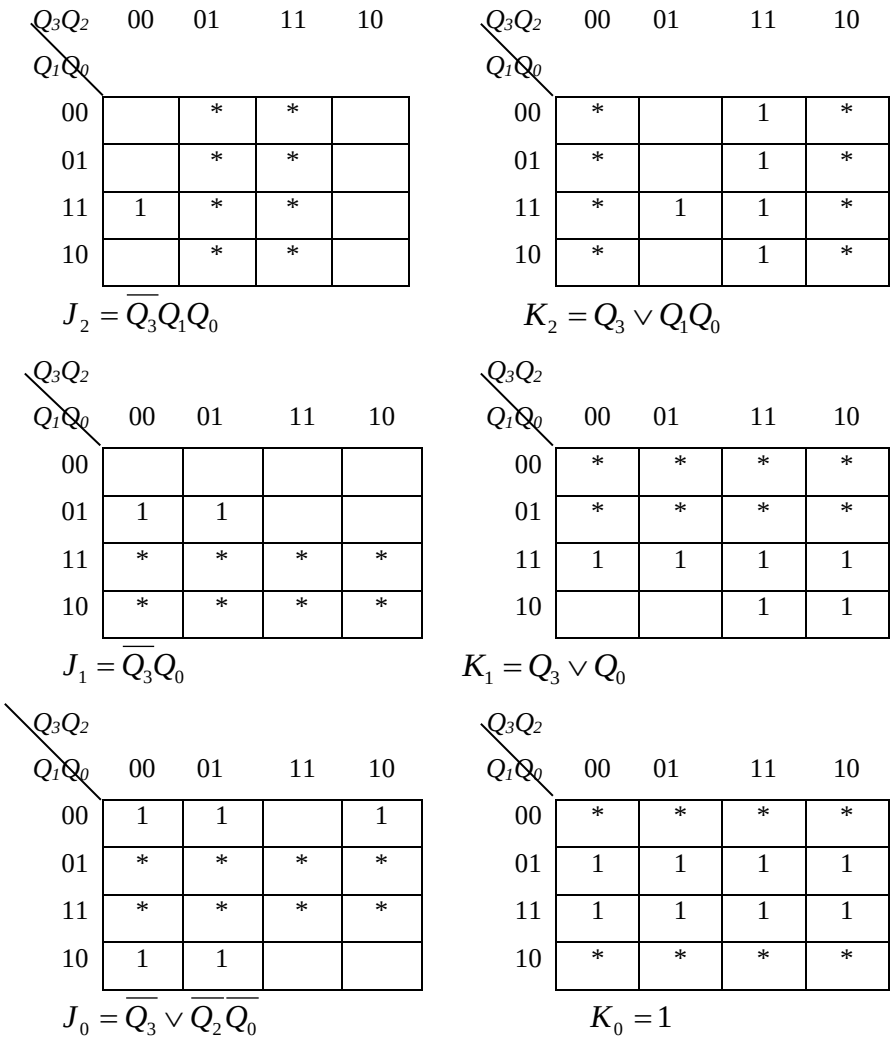


Figura 3.11. Diagramele Veitch-Karnaugh pentru minimizarea funcţiilor de instalare a bistabilelor.

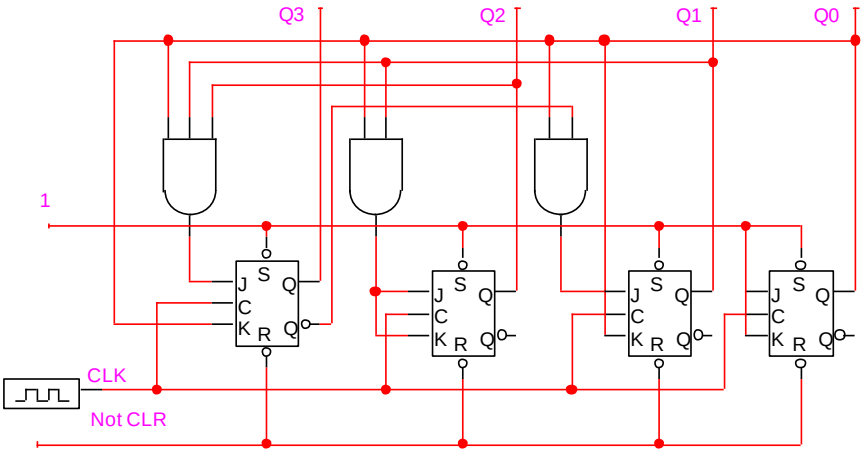


Figura 3.12 Numărător sincron modulo 10.

3.6 Lucrarea de laborator nr.7

Tema: Sinteza numărătoarelor

Scopul lucrării: studierea diferitor tipuri de numărătoare şi a metodelor lor de proiectare.

Tema pentru acasă

1. Executaţi sinteza numărătorului sincron incomplet în baza bistabilelor *JK* sau *D*, setul SI-NU, conform variantei indicate de profesor (tabelul 3.8).
2. Executaţi sinteza numărătorului asincron incomplet în baza bistabilelor *T*, conform variantei indicate de profesor (tabelul 3.9).
3. Executaţi sinteza numărătorului reversibil

asincron modulo 32 în baza bistabilelor *T*.

Tabelul 3.8

Nr. var.	Bistabilul	Tipul numărătorului	Nr. var.	Bistabilul
1.	JK	Direct, modulo 9	13	D
2.	JK	Direct, modulo 11	14	D
3.	JK	Direct, modulo 12	15	D
4.	JK	Direct, modulo 13	16	D
5.	JK	Direct, modulo 14	17	D
6.	JK	Direct, modulo 15	18	D
7.	JK	Invers, modulo 10	19	D
8.	JK	Invers, modulo 11	20	D
9.	JK	Invers, modulo 12	21	D
10.	JK	Invers, modulo 13	22	D
11.	JK	Invers, modulo 14	23	D
12.	JK	Invers, modulo 15	24	D

Tabelul 3.9

Nr. var.	Tipul numărătorului	Nr. var.	Tipul numărătorului
1.	Direct, modulo 17	15.	Invers, modulo 17
2.	Direct, modulo 18	16.	Invers, modulo 18
3.	Direct, modulo 19	17.	Invers, modulo 19
4.	Direct, modulo 20	18.	Invers, modulo 20
5.	Direct, modulo 21	19.	Invers, modulo 21
6.	Direct, modulo 22	20.	Invers, modulo 22
7.	Direct, modulo 23	21.	Invers, modulo 23
8.	Direct, modulo 24	22.	Invers, modulo 24
9.	Direct, modulo 25	23.	Invers, modulo 25
10.	Direct, modulo 26	24.	Invers, modulo 26

11.	Direct, modulo 27	25.	Invers, modulo 27
12	Direct, modulo 28	26.	Invers, modulo 28
13.	Direct, modulo 29	27.	Invers, modulo 29
14.	Direct, modulo 30	28.	Invers, modulo 30

Desfăşurarea lucrării

a) la standul de laborator:

1. Se verifică corectitudinea funcţionării circuitelor integrate ale standului de laborator.

2. Se asamblează şi se reglează schema numărătorului sincron incomplet în baza bistabilelor *JK* sau *D* (p. 1 din tema pentru acasă).

3. Se asamblează şi se reglează schema numărătorului asincron incomplet în baza bistabilelor *T* (p. 2 din tema pentru acasă).

4. Se asamblează şi se reglează schema numărătorului reversibil asincron modulo 32 în baza bistabilelor *T* (p. 3 din tema pentru acasă).

5. Pentru circuitele asamblate se determină costul şi timpul de reţinere.

b) în LogicWorks:

1. Din biblioteca de elemente **Simulation Gates.clf** se selectează elementele necesare.. Din biblioteca **Simulation IO.clf** se selectează dispozitivele de intrare-ieşire **Binary Probe**, **Hex Keyboard, Binary Switch** şi **Clock.**

2. Se asamblează schema numărătorului sincron incomplet în baza bistabilelor *JK* sau *D* în **Fereastra de lucru** şi se verifică corectitudinea ei. Se studiază diagrama de timp.

3. Se asamblează schema numărătorului asincron incomplet în baza bistabilelor *T* în **Fereastra de lucru** şi se verifică corectitudinea ei. Se studiază diagrama de timp.

4. Se asamblează schema numărătorului reversibil asincron modulo 32 în baza bistabilelor *T* în **Fereastra de lucru** şi se verifică corectitudinea ei. Se studiază diagrama de timp.

5. Pentru circuitele asamblate se determină costul şi timpul de reţinere.

Întrebări:

1. Prezentaţi schemele numărătoarelor asincrone modulo 8 cu numărare directă şi inversă în baza bistabilelor de tip *T* cu comutare pe front crescător.

2. Cînd apare timpul de propagare maxim întru-un numărător asincron modulo 2^n cu numărare directă?

3. Analizaţi succesiunea inversă din tabelul 3.5 şi scrieţi relaţiile de calculare a condiţiilor de basculare pentru un numărător sincron modulo 2^n.

4. Prezentaţi schema numărătorului sincron modulo 16 cu transport succesiv. Comparaţi sub aspectul problemelor de implementare şi al timpului de propagare maxim numărătoarele sincrone cu transport succesiv şi paralel.

5. Demonstraţi că numărătorul din figura 3.12 poate ieşi după cel mult două tacturi dintr-o posibilă stare ilegală.