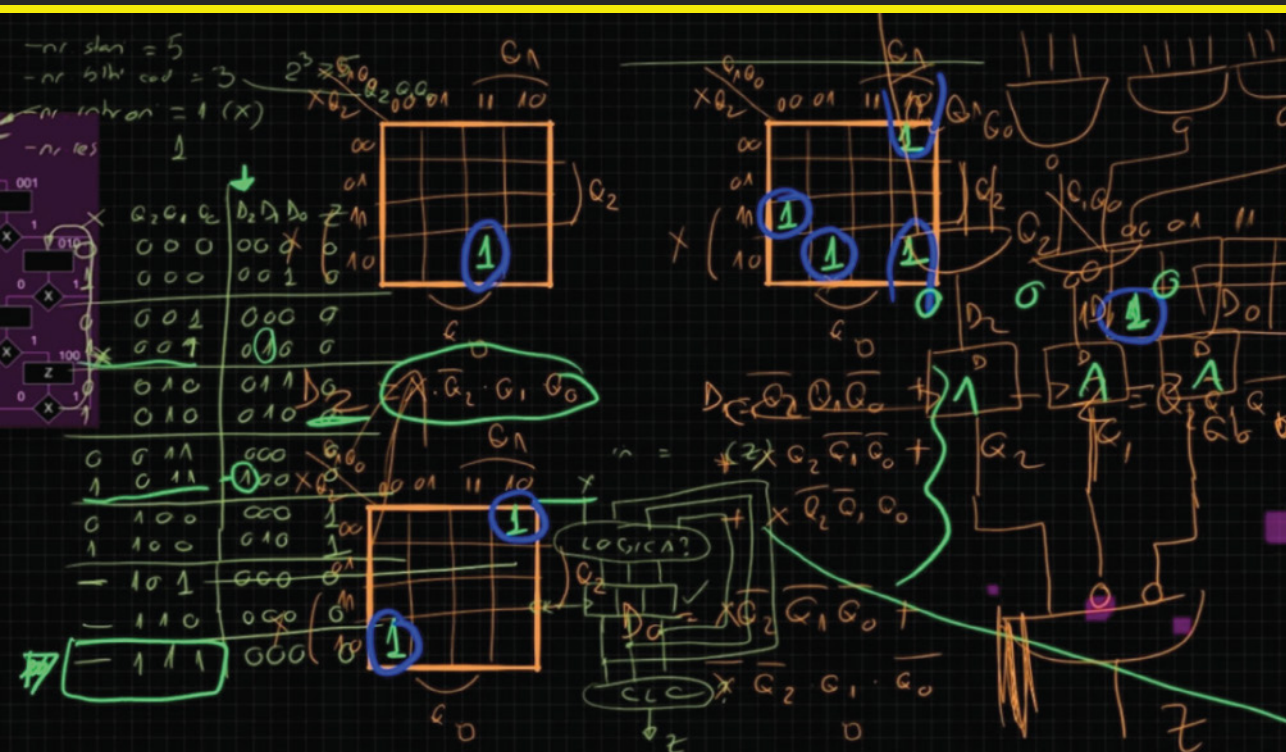


TOMUL PENTRU CHIBZUIT ȘI BUCHISIT

DAN NICULA

ELECTRONICĂ DIGITALĂ

CARTE DE ÎNVĂȚĂTURĂ



Dan NICULA

ELECTRONICĂ DIGITALĂ
Carte de învățatură



EDITURA
UNIVERSITĂȚII
TRANSILVANIA
DIN BRAȘOV

2025

EDITURA UNIVERSITĂȚII TRANSILVANIA DIN BRAȘOV

Adresa: Str. Iuliu Maniu nr. 41A

500091 Brașov

Tel.: 0268 476 050

Fax: 0268 476 051

E-mail: editura@unitbv.ro

Editură acreditată de CNCIS, cod 81

ISBN 978-606-19-1794-5 (ebook)

Copyright ©2025 Autorul

Lucrarea a fost avizată de

Consiliul Departamentului de Electronică și Calculatoare din cadrul

Facultății de Inginerie Electrică și Știința Calculatoarelor,

Universitatea Transilvania din Brașov.

Cuprins

Electronică digitală	I
Cuprins	i
Cuvântul cel dintâi către „cetitor”	xi
Tomul pentru zăbovit și gândit	1
Începutul învățaturii „Tomului pentru zăbovit și gândit”	3
1 De la matematică la inginerie	5
1.1 Lumea digitală. Între 0 și 1	7
1.1.1 De ce este nevoie de „Electronică digitală”?	8
1.1.2 Informația. Semnale	12
1.2 Matematică binară	16
1.2.1 Reprezentarea numerelor	17
1.2.2 Conversia numerelor între bazele de numerație 2, 10, 16	22
1.2.3 Reprezentarea numerelor întregi în complement față de 2	28
1.2.4 Adunarea și scăderea numerelor întregi reprezentate în complement față de 2	31
1.2.5 Codul ASCII	33
1.2.6 Codul BCD	35
1.2.7 Algebra Booleană	36
1.3 Inginerie binară	38
1.3.1 Porți logice	38
1.3.2 Justificarea noțiunii de poartă logică	40
1.3.3 Set complet de operatori logici	45
1.3.4 Parametrii porților și semnalelor logice	45
1.3.5 Timp de propagare	51
1.4 Modelarea sistemelor digitale în HDL	54
1.4.1 De ce este nevoie de HDL?	55
1.4.2 Metodologia de proiectare a sistemelor digitale	57

1.4.3	Mediul de simulare HDL	59
1.4.4	Modele HDL	60
1.4.5	Setul de valori	63
1.4.6	Tipuri de date	63
1.4.7	Codați cu stil!	65
1.4.8	Hard-ul nu-i ca soft-ul!	67
2	Circuite logice combinaționale	69
2.1	Reprezentarea circuitelor logice combinaționale	72
2.1.1	Tabele de adevăr	72
2.1.2	Forme standard	74
2.1.3	Diagrame Veitch-Karnaugh	76
2.2	Analiza circuitelor logice combinaționale	79
2.3	Sinteza circuitelor logice combinaționale	84
2.3.1	Minimizarea funcțiilor logice, diagrame V-K	84
2.3.2	Minimizarea funcțiilor logice incomplet definite	86
2.3.3	Minimizarea funcțiilor cu variabile reziduu	87
2.3.4	Minimizarea funcțiilor logice cu ieșiri multiple	88
2.4	CLC particulare	90
2.4.1	Decodificator	91
2.4.2	Codificator	99
2.4.3	Multiplexor	101
2.4.4	Demultiplexor	110
2.4.5	Sumator	117
2.4.6	Comparator	121
2.4.7	Unitate logică și aritmetică	123
2.4.8	Look-Up-Table și memorii ROM	126
2.5	Modelarea circuitelor logice combinaționale în HDL	129
2.5.1	Operatori	129
2.5.2	Operatori aritmetici	131
2.5.3	Operatori relaționali	138
2.5.4	Operatori de egalitate	138
2.5.5	Operatori logici	139
2.5.6	Operatori logici pe vectori de biți	140
2.5.7	Operatori de reducere	141
2.5.8	Operatori de deplasare	143
2.5.9	Operatorul condițional	145
2.5.10	Operatori de concatenare și replicare a biților	145
2.5.11	Precedența operatorilor	146
2.5.12	Modelarea circuitelor logice combinaționale	149
3	Circuite logice secvențiale	157
3.1	Circuite de memorare a stării	159
3.1.1	Latch	163
3.1.2	Bistabile	167

3.1.3	Inițializarea bistabilelor	181
3.2	Automate secvențiale	183
3.3	Reprezentarea automatelor	186
3.3.1	Tabele de tranziții	187
3.3.2	Organigrame	190
3.3.3	Grafuri	191
3.3.4	Analiza circuitelor logice secvențiale	193
3.3.5	Sinteza automatelor	197
3.3.6	Modelarea automatelor în Verilog	200
3.4	Registre și numărătoare	206
3.4.1	Registre cu accesare serie sau paralel	207
3.4.2	Numărătoare	211
3.4.3	Linear-Feedback-Shift-Register	211
3.4.4	Modelarea HDL a registrelor și a numărătoarelor	213
3.5	Memorii	225
3.5.1	Memoria ROM sincronă	225
3.5.2	Memoria RAM	228
3.5.3	Structuri de memorii	237
4	Considerente ingineresti ale sistemelor digitale	245
4.1	Timpul în sistemele digitale	248
4.1.1	Căi de propagare, frecvența semnalului de ceas, slack	248
4.1.2	Parametrii semnalului de ceas: skew, jitter, drift	251
4.1.3	Metastabilitatea bistabilelor	255
4.1.4	Domenii de ceas, circuite de sincronizare	256
4.1.5	Structura pipeline	259
4.2	Protocoale de comunicare sincrone între module digitale	262
4.2.1	Protocolul „request-acknowledge”	262
4.2.2	Protocolul „valid-ready”	266
4.3	Tehnologia circuitelor integrate digitale	270
4.3.1	Metodologia de proiectare a circuitelor integrate digitale	271
4.3.2	Fabricarea circuitelor integrate	274
4.3.3	Tehnologia CMOS	275
4.4	Noțiuni avansate de Verilog	279
4.4.1	Modelarea comportamentală RTL	281
4.4.2	Medii de testare	288
	Tomul pentru zăbovit și gândit - Final	300
	Tomul pentru chibzuit și buchisit	301
	Începutul învățaturii „Tomului pentru chibzuit și buchisit”	303
1	Reprezentarea datelor în sistemele digitale	305
1.1	Pentru seminar	305

1.2	Pentru cei ce vor doar să promoveze examenul	306
1.3	Pentru cei ce vor să învețe	307
1.4	Pentru cei ce vor să devină profesioniști	316
2	Algebră Booleană, porți logice, funcții logice	319
2.1	Pentru seminar	319
2.2	Pentru cei ce vor doar să promoveze examenul	323
2.3	Pentru cei ce vor să învețe	327
2.4	Pentru cei ce vor să devină profesioniști	344
3	Minimizare cu diagrame Veitch-Karnaugh	351
3.1	Pentru seminar	351
3.2	Pentru cei ce vor doar să promoveze examenul	351
3.3	Pentru cei ce vor să învețe	352
3.4	Pentru cei ce vor să devină profesioniști	356
4	Implementare CLC cu porți NAND	363
4.1	Pentru seminar	363
4.2	Pentru cei ce vor doar să promoveze examenul	365
4.3	Pentru cei ce vor să învețe	366
4.4	Pentru cei ce vor să devină profesioniști	371
5	Multiplexoare, demultiplexoare și codificatoare	377
5.1	Pentru seminar	377
5.2	Pentru cei ce vor doar să promoveze examenul	378
5.3	Pentru cei ce vor să învețe	379
5.4	Pentru cei ce vor să devină profesioniști	380
6	Modelarea circuitelor logice combinaționale	393
6.1	Pentru seminar	393
6.2	Pentru cei ce vor doar să promoveze examenul	397
6.3	Pentru cei ce vor să învețe	398
6.4	Pentru cei ce vor să devină profesioniști	399
6.5	Concluzie: Inginer vs. ChatGPT	409
7	Circuite cu bistabile	415
7.1	Pentru seminar	415
7.2	Pentru cei ce vor doar să promoveze examenul	418
7.3	Pentru cei ce vor să învețe	419
7.4	Pentru cei ce vor să devină profesioniști	422
8	Analiza circuitelor logice secvențiale	425
8.1	Pentru seminar	425
8.2	Pentru cei ce vor doar să promoveze examenul	426
8.3	Pentru cei ce vor să învețe	426
8.4	Pentru cei ce vor să devină profesioniști	432

9 Sinteza circuitelor logice secvențiale	439
9.1 Pentru seminar	439
9.2 Pentru cei ce vor doar să promoveze examenul	441
9.3 Pentru cei ce vor să învețe	441
9.4 Pentru cei ce vor să devină profesioniști	449
10 Modelarea automatelor în Verilog	453
10.1 Pentru seminar	455
10.2 Pentru cei ce vor doar să promoveze examenul	457
10.3 Pentru cei ce vor să învețe	457
10.4 Pentru cei ce vor să devină profesioniști	459
11 Registre și numărătoare	469
11.1 Pentru seminar	469
11.2 Pentru cei ce vor doar să promoveze examenul	471
11.3 Pentru cei ce vor să învețe	471
11.4 Pentru cei ce vor să devină profesioniști	474
12 Memorii	485
12.1 Pentru seminar	485
12.2 Pentru cei ce vor doar să promoveze examenul	486
12.3 Pentru cei ce vor să învețe	486
12.4 Pentru cei ce vor să devină profesioniști	488
13 Proiectarea sistemelor digitale	495
13.1 Pentru seminar	495
13.2 Pentru cei ce vor doar să promoveze examenul	496
13.3 Pentru cei ce vor să învețe	496
13.4 Pentru cei ce vor să devină profesioniști	507
14 Probleme de examen	511
14.1 Subiectul 1	511
14.1.1A. Noțiuni generale	511
14.1.2B. Circuite logice combinaționale	516
14.1.3C. Circuite logice secvențiale	518
14.1.4D. Proiectare sistem digital	521
14.2 Subiectul 2	523
14.2.1A. Noțiuni generale	523
14.2.2B. Circuite logice combinaționale	528
14.2.3C. Circuite logice secvențiale	531
14.2.4D. Proiectare sistem digital	532
14.3 Subiectul 3	534
14.3.1A. Noțiuni generale	534
14.3.2B. Circuite logice combinaționale	539
14.3.3C. Circuite logice secvențiale	540
14.3.4D. Proiectare sistem digital	541

14.4	Subiectul 4	542
14.4.1A.	Noțiuni generale	543
14.4.2B.	Circuite logice combinaționale	548
14.4.3C.	Circuite logice secvențiale	550
14.4.4D.	Proiectare sistem digital	552
14.5	Subiectul 5	553
14.5.1A.	Noțiuni generale	553
14.5.2B.	Circuite logice combinaționale	558
14.5.3C.	Circuite logice secvențiale	559
14.5.4D.	Proiectare sistem digital	562
15	Subiecte pentru evaluat ingineri	565
15.1	Ingineri începători	565
15.1.1	Cunoștințe generale	565
15.1.2	Definirea noțiunilor	566
15.1.3	Circuite cu bistabile	575
15.1.4	Proiectare circuite secvențiale pe baza specificațiilor	579
15.1.5	Proiectare circuite secvențiale după model	588
15.2	Ingineri avansați	589
15.2.1	Analiza circuitelor digitale	589
15.2.2	Proiectare sisteme digitale de control	596
	Tomul pentru chibzuit și buchisit - Final	610
	Tomul pentru meșterit și muncit	611
	Începutul învățaturii „Tomului pentru meșterit și muncit”	613
1	Modelarea circuitelor cu porți logice	615
1.1	Desfășurarea lucrării	615
1.2	Temă de casă	618
2	Modelarea circuitelor logice	623
2.1	Desfășurarea lucrării	624
2.2	Temă de casă	626
3	Desenare diagrame de timp, circuite și scheme	629
3.1	Desfășurarea lucrării	631
3.2	Temă de casă	633
4	Modelarea circuitelor digitale în Verilog	635
4.1	Noțiuni introductive	635
4.1.1	Definirea termenilor	635
4.1.2	Modele asociate unui proiect	637
4.1.3	Organizarea fișierelor	638

4.1.4 Metodologia de proiectare a circuitelor integrate digitale	640
4.1.5 Metodologia de simulare HDL cu <i>ModelSim</i>	642
4.2 Desfășurarea lucrării	650
4.3 Temă de casă	651
5 Multiplexoare, demultiplexoare, codificatoare	653
5.1 Desfășurarea lucrării	654
5.2 Temă de casă	656
6 Modelarea circuitelor logice combinaționale în Verilog	659
6.1 Desfășurarea lucrării	660
6.2 Teme de casă	663
7 Modelarea circuitelor cu bistabile	665
7.1 Desfășurarea lucrării	667
7.1.1 Mediu de testare pentru circuite cu bistabile	669
7.2 Teme de casă	672
8 Modelarea circuitelor secvențiale cu bistabile	677
8.1 Desfășurarea lucrării	677
8.2 Temă de casă	680
9 Modelarea FSM	683
9.1 Desfășurarea lucrării	684
9.2 Temă de casă	686
10 Modelarea registrelor și a numărătoarelor	689
10.1 Desfășurarea lucrării	689
10.1.1 Registru cu validare	689
10.1.2 Numărător	690
10.1.3 LFSR	691
10.2 Temă de casă	692
11 Modelarea memoriilor	695
11.1 Noțiuni teoretice	695
11.1.1 Memorii	696
11.1.2 Inițializarea memoriei cu o constantă	698
11.1.3 Inițializarea memoriei	700
11.1.4 Funcții pentru accesul la fișiere de intrare/ieșire	700
11.1.5 Inițializarea memoriei din fișier	701
11.1.6 Salvarea conținutului memoriei în fișier	702
11.2 Desfășurarea lucrării	703
11.2.1 Testarea memoriei single-port	703
11.2.2 Operațiuni cu fișiere	704
11.2.3 Modificări asupra mediului de testare	705
11.3 Temă de casă	706

11.4 Fișiere cu modelul mediului de test	708
11.4.1 Modulul generatorului de stimuli	708
11.4.2 Modulul mediului de testare	709
12 Modelarea sistemelor digitale complexe	713
12.1 Multiplicator secvențial	714
12.2 Circuit watchdog	719
12.3 Circuit pipeline pentru determinarea maximului dintre 3 date	724
13 Subiecte pentru colocviul de laborator	735
Tomul pentru meșterit și muncit - Final	742
Tomul pentru grabă mare	743
Începutul învățăturii „Tomului pentru grabă mare”	745
1 Aritmetică binară	749
1.1 Întrebări	749
1.2 Răspunsuri	758
2 Porți logice	761
2.1 Întrebări	761
2.2 Răspunsuri	772
3 Funcții logice	775
3.1 Întrebări	775
3.2 Răspunsuri	784
4 Circuite logice combinaționale	785
4.1 Întrebări	785
4.2 Răspunsuri	791
5 Circuite logice combinaționale particulare	793
5.1 Întrebări	793
5.2 Răspunsuri	807
6 Elemente de memorare	809
6.1 Întrebări	809
6.2 Răspunsuri	821
7 Automate cu număr finit de stări	823
7.1 Întrebări	823
7.2 Răspunsuri	828
8 Circuite logice secvențiale particulare	829

8.1 Întrebări	829
8.2 Răspunsuri	851
9 Probleme pentru testat inteligența naturală/artificială	853
9.1 Probleme de proiectare	853
Tomul pentru grabă mare - Final	862
Grăirea de încheiere	863
Electronică digitală - Final	866

Cuvântul cel dintâi către „cetitor”

Omul a simțit dintotdeauna dorința să își transmită învățătura către generațiile următoare.

Prima ediție **„Electronică digitală - Carte de învățatură”** a apărut în anul 2012. Subtitlul arhaic își are originea în sentimentul meu că „*învățătura*”¹ devine învechită, fiind înlocuită treptat de „*descurcăreala*”².

După 3 ani, în 2015, am publicat **„Electronică digitală - Carte de învățatură 2.0”**, o ediție revizuită, în care „*m-am adaptat pieteii*”. Adică, am mai îndulcit tonul, am mai coborât ștacheta și am sortat problemele pentru a realiza o secțiune minimală, dedicată „*celor ce vor doar să promoveze examenul*”.

În 2019, pentru „*cei ce vor să învețe și să devină profesioniști*” am scris volumul **„Verilog - Carte de învățatură”**.

În anul 2020, am simțit nevoia să public un set de chestionare de electronică digitală, cu subtitlul **„Electronică digitală - Carte de învățatură în regim de urgență”**. Problemele au fost adaptate examinării online, prin transformarea problemelor de proiectare și gândire în chestionare potrivite pentru „*metoda ghici*”. Am vrut să marchez astfel perioada nefastă la nivel mondial și să îmi exprim convingerea că nu se pot pregăti profesioniști de electronică în *regim de urgență*.

După 2022, observând schimbarea evidentă a electronicii în ultimii 30 de ani, am actualizat conținutul cursului de Electronică digitală, prin aducerea limbajului de descriere hardware în strânsă conexiune cu circuitele electronice digitale. Am sesizat că studenții sunt mult mai pragmatici, dornici să rezolve repede problema, și mai puțin dornici să înțeleagă justificarea. După încă 2 ani, feed-back-ul primit din partea studenților a fost că „*nu există o bibliografie adaptată cerințelor mele*”.

¹**învățatură** = Procesul de a dobândi cunoștințe. dannicula.ro

²**descurcăreală** = Arta de a rezolva o problemă, fără a o înțelege pe deplin, eventual „*asistat de inteligență artificială*”. dannicula.ro

Înțelegând corectitudinea acestei opinii, m-am decis să rescriu cartea de învățătură în integralitatea ei, sub forma a patru „tomuri”³:

- **Tomul pentru zăbovit și gândit**⁴ suport de curs, noțiuni teoretice alăturate modelării Verilog.
- **Tomul pentru chibzuit și buchisit**^{5 6} suport pentru seminar și probleme de proiectare.
- **Tomul pentru meșterit și muncit**⁷ suport pentru laborator și mici proiecte pentru practică.
- **Tomul pentru grabă mare**⁸ chestionare pentru auto-evaluare și evidențierea noțiunilor ce mai trebuie studiate.

Subtitlurile arhaice sunt intenționate, pentru că tot mai cred că „*a învăța dintr-o carte cu creionul în mână*” este ceva învechit. Mă aștept ca studenții să citească din această carte pe un monitor mare, în care pot vedea două pagini alăturate (similar cu o carte pe hârtie, deschisă), având în față un caiet (sau o tabletă de scris) și un creion (clasic sau „pen”). Din acest motiv, în carte am preferat să pun link-uri de acces rapid în versiunea electronică și să evit să dau detalii despre anumite subiecte cu o relevanță scăzută dar ușor de descoperit pe Internet.

Versiunea actuală, 2025, aduce o structurare și o corelare a întregului subiect al electronicii digitale, oferind suport pentru curs, seminar, laborator și studiu individual pentru studenții de la specializările:

- Electronică aplicată
- Telecomunicații
- Calculatoare
- Automatică
- Tehnologia Informației

³**tom** = Fiecare dintre părțile unei lucrări, ale unei opere de proporții mai întinse, formând adesea o unitate independentă și purtând, de cele mai multe ori, un număr de ordine. Volum, carte, lucrare, operă. dexonline.ro

⁴**a zăbovi** = A lucra prea încet, a nu se grăbi. dexonline.ro

⁵**a chibzui** = A reflecta asupra unei situații cumpănind toate eventualitățile. A se gândi. dexonline.ro

⁶**a buchisi** = A citi sau a învăța ceva cu efort și cu mîgală. dexonline.ro

⁷**a meșteri** = A lucra, a munci (cu pricepere) pentru a realiza ceva. dexonline.ro

⁸**grabă mare** = Termen generat de ChatGPT ca un arhaism pentru „stare de urgență”.

Deși subtitlurile sunt arhaice, conținutul este redactat într-o formă ce s-a dovedit acceptată de generația actuală de studenți, obișnuită (și nevoită) să citească mai mult de pe monitor decât de pe hârtie: paragrafe și cuvinte **cheie** evidențiate cu galben, **avertizări** marcate cu roșu, link-uri la pagini web, recomandări ale  ChatGPT și multe încurajări... 😊. ⁹

Asemănările cu aplicațiile de socializare actuale sunt intenționate. 😞

Conceptul cărții este de a permite studenților, la finalul anului al doilea de studii, accesarea stagiilor de pregătire oferite de firme în domeniul hardware.

Fișierele necesare pentru efectuarea lucrărilor se află pe site-ul autorului, dannicula.ro.



Brașov, 2025.

Dan NICULA

»> CONTINUT

⁹Da, m-am gândit și la audio-book. Și la benzi desenate... 😊

Începutul învățaturii „Tomului pentru chibzuit și buchisit”

Tomul pentru chibzuit și buchisit^{1 2} a fost conceput pentru a include probleme practice pentru învățarea electronicii digitale, în vederea proiectării circuitelor integrate digitale cu o metodologie actuală: **modelare HDL urmată de sinteză de hardware**. Accentul este pus pe modelarea comportamentului sistemelor digitale în Verilog și mai puțin pe determinarea precisă a structurilor de porți logice și bistabile care implementează o anumită funcție. Nu se tratează nici proiectarea cu circuite SSI/MSI.³

Cartea este structurată în 13 lecții, asociate celor 13 săptămâni ale unui semestru (sigur se pierde câte una... 😊). Fiecare lecție are patru secțiuni pentru uzul diverselor categorii de cititori:

- **„Pentru seminar”** este secțiunea minimală, ce conține chestiuni abordate la seminar, sub supervizarea profesorului. Chestiunile nu sunt rezolvate, pentru că se presupune că vor fi rezolvate în sala de seminar.
- **„Pentru cei ce vor doar să promoveze examenul”** este secțiunea dedicată studenților ce se consideră „*auto-didacti*”, interesați doar de a promova examenul, cu efort minim.
- **„Pentru cei ce vor să învețe”** este secțiunea dedicată majorității studenților, care sunt determinați să se pregătească pentru profesiunea de inginer și depun un efort pentru aceasta. Este cea mai întinsă secțiune și conține atât probleme rezolvate, cât și nerezolvate.
- **„Pentru cei ce vor să devină profesioniști”** este secțiunea dedicată celor pasionați,⁴ care vor să susțină interviuri de angajare în domeniul electronicii digitale.

¹ **a chibzui** = A reflecta asupra unei situații cumpănind toate eventualitățile. A se gândi. dexonline.ro

² **a buchisi** = A citi sau a învăța ceva cu efort și cu mîgală. dexonline.ro

³ **SSI/MSI** = „*Small/Medium Scale Integration*”, engl.

⁴ **pasiune** = Încălnație vie, însoțită de plăcere pentru obiectul studiat sau pentru profesia exercitată. dexonline.ro

La final, am inclus două anexe:

- Secțiunea „**Probleme de examen**” cuprinde subiectele utilizate în anul 2024, majoritatea rezolvate. Ar trebui ca acestea să nu fie considerate dificile dacă au fost parcurse primele două volume ale „Cărții de învățatură”:

Tomul pentru zăbovit și gândit și

Tomul pentru chibzuit și buchisit

- Secțiunea „**Probleme de evaluat ingineri**” cuprinde subiecte utilizate de-a lungul timpului la interviuri de angajare, atât pentru poziții de ingineri începători cât și de ingineri avansați. Această secțiune poate fi o sursă de inspirație pentru testarea studenților în cadrul stagiilor de practică oferite de firme din domeniul proiectării și verificării de sisteme digitale integrate.

Cartea conține multe probleme rezolvate sau care prezintă codul Verilog ce modelează sistemul digital propus spre proiectare.

Așteptarea mea este ca un student să parcurgă **i-n-c-e-t** această carte, „cu creionul în mână”, să *chibzuiască* asupra soluțiilor și să *buchisească* acest volum (chiar citit sub formă electronică).

Desigur, cineva poate utiliza  ChatGPT pentru a găsi soluții la probleme. Încurajez aceasta. Însă, în perioada studenției, este momentul de a alocă un timp pentru a testa limitele inteligenței artificiale. Până la urmă, „GPT”⁵ înglobează experiențele anterioare ale umanității. Este bine de a prelua și învăța din experiențele înaintașilor. Desigur, n-ar fi rău să evităm greșelile acestora...

Este însă esențială înțelegerea conceptelor, pentru a putea crea dincolo de cunoștințele actuale.

Atenție! ChatGPT nu generează întotdeauna adevăruri.⁶ Trebuie să verificați răspunsurile acestei aplicații.

Cartea aceasta este pentru a vă „calibra etalonul” de cunoștințe pentru a putea face diferența dintre o soluție corectă și una greșită.

Brașov, 2025.

Dan NICULA

⁵GPT = Generative Pre-trained Transformers, engl. Model de limbaj care generează text, învățat dinainte, bazat pe o arhitectură numită Transformer. (definiție dată de ChatGPT la întrebarea „Ce înseamnă GPT?”)

⁶În partea de jos a ferestrei ChatGPT este scris „ChatGPT can make mistakes. Check important info.”

Lecția 1

Reprezentarea datelor în sistemele digitale

Această lecție introductivă utilizează cunoștințele dobândite la cursurile anterioare de **matematică** și scoate în evidență noțiunile esențiale pentru electronica digitală:

- Reprezentarea și conversia numerelor între bazele de numerație 2, 10 și 16.
- Reprezentarea numerelor întregi în complement față de 2 și operații aritmetice cu acestea.

1.1 Pentru seminar

1. Care este cel mai mare număr binar, exprimat pe 16 biți? Care este echivalentul său în baza 10? Dar în baza 16?

Soluție

Cel mai mare număr binar, exprimat pe 16 biți este:

$$1111_1111_1111_1111|_2 = FFFF|_{16} = 2^{16} - 1 = 65.535|_{10}$$

2. Care este cel mai mare număr pozitiv, exprimat în baza 10 care se poate reprezenta pe 8 biți? Dar pe 4 bytes?

Soluție

Cel mai mare număr pozitiv, exprimat în baza 10, care se poate reprezenta pe 8 biți este $2^8 - 1 = 255$, adică $1111_1111|_2$.

4 bytes = 4×8 biți = 32 biți.

Deci, cel mai mare număr pozitiv, exprimat în baza 10, care poate fi reprezentat pe 4 bytes este $2^{32} - 1 = 4.294.967.295$. În binar acest număr are toți cei 32 de biți egali cu 1.

3. Valoarea $100|_{16}$ este egală cu valoarea exprimată în baza 10:
a) 100 **b)** 256 **c)** 2^{10} **d)** 400

Soluție

$$100|_{16} = 1 \cdot 16^2 + 0 \cdot 16^1 + 0 \cdot 16^0 = 256|_{10}. \text{ Răspuns corect: } \mathbf{b}).$$

4. Realizați următoarele conversii:

- a)** $10001000|_2 = ?|_{10}$
b) $DEAD_BEEF|_{16} = ?|_2 = ?|_{10}$
c) $369|_{10} = ?|_2 = ?|_{16}$
d) $1101_0110|_2 = ?|_{10} = ?|_{16}$
e) $372|_{16} = ?|_2 = ?|_{10}$
f) $DAC|_{16} = ?|_2 = ?|_{10}$

Soluție

- a)** $1000_1000|_2 = 1 \cdot 2^7 + 1 \cdot 2^3 = 136|_{10}$
c) $369|_{10} = 256 + 64 + 32 + 16 + 1 = 1 \cdot 2^8 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^0 = 1_0111_0001|_2 = 171|_{16}$
d) $1101_0110|_2 = D6|_{16} = 13 \cdot 16^1 + 6 \cdot 16^0 = 214|_{10}$

5. Realizați următoarele operații cu numere reprezentate în complement față de 2, pe 8 biți:

$$\begin{array}{cccccc} 7 + (-3) & 97 + 28 & 100 + (-7) & 38 + (-23) & 40 + (-40) & 58 + (-42) \\ 7 - (-3) & 97 - 28 & 100 - (-7) & -38 - (-23) & 40 - (-40) & -58 - (-42). \end{array}$$

1.2 Pentru cei ce vor doar să promoveze examenul

1. Realizați următoarele **conversii**:

$$\begin{array}{l} 2025|_{10} = ?|_2 = ?|_{16} \\ 0101_1100|_2 = ?|_{10} = ?|_{16} \\ DA7C|_{16} = ?|_{10} = ?|_2 \end{array}$$

2. Reprezentați următoarele numere întregi, în **complement față de 2**, pe 16 biți:

$$66, -73, 51, -84, 200, -200.$$

3. Realizați următoarele **adunări** cu numere întregi exprimate în complement față de 2, pe 16 biți:

$$\begin{array}{ll} 70 + 17 = ? & 88 + (-15) = ? \\ 120 + (-122) = ? & 73 + (-73) = ? \\ (-85) + (-15) = ? & (-14) + 14 = ? \\ (-68) + 15 = ? & (-66) + 90 = ? \end{array}$$

1.3 Pentru cei ce vor să învețe

1. Realizați următoarele conversii:

- | | |
|---|---|
| a) $13 _{10} = ? _2 = ? _{16}$ | g) $1999 _{10} = ? _2 = ? _{16}$ |
| b) $110_1101 _2 = ? _{16} = ? _{10}$ | h) $2012 _{10} = ? _2 = ? _{16}$ |
| c) $11101010101 _2 = ? _{10}$ | i) $BEEF _{16} = ? _2 = ? _{10}$ |
| d) $100 _{16} = ? _{10}$ | j) $0101_1101 _2 = ? _{16} = ? _{10}$ |
| e) $1000_1101 _2 = ? _{16} = ? _{10}$ | k) $201 _{10} = ? _2 = ? _{16}$ |
| f) $11010011 _2 = ? _{10}$ | l) $ABCD _{16} = ? _2 = ? _{10}$ |

Soluție

- a)** $13|_{10} = 8 + 4 + 1 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1101|_2 = D|_{16}$.
- b)** $1101101|_2 = 110_1101|_2 = 6D|_{16} = 6 \cdot 16 + 13 = 109|_{10}$
- c)** $111_0101_0101|_2 = 1 \cdot 2^{10} + 1 \cdot 2^9 + 1 \cdot 2^8 + 1 \cdot 2^6 + 1 \cdot 2^4 + 1 \cdot 2^2 + 1 \cdot 2^0 = 1024 + 512 + 256 + 64 + 16 + 4 + 1 = 1877|_{10}$
- e)** $1000_1101|_2 = 8D|_{16}$
 $8D|_{16} = 8 \cdot 16^1 + 13 \cdot 16^0 = 128 + 13 = 141|_{10}$
- f)** $1101_0011|_2 = 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^4 + 1 \cdot 2^1 + 1 \cdot 2^0 = 128 + 64 + 16 + 2 + 1 = 211|_{10}$
- g)** $1999|_{10} = 1024 + 512 + 256 + 128 + 64 + 8 + 4 + 2 + 1 = 1 \cdot 2^{10} + 1 \cdot 2^9 + 1 \cdot 2^8 + 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 111_1100_1111|_2$
- h)** $2012|_{10} = 1024 + 512 + 256 + 128 + 64 + 16 + 8 + 4 = 1 \cdot 2^{10} + 1 \cdot 2^9 + 1 \cdot 2^8 + 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 = 111_1101_1100|_2 = 7DC|_{16}$
- i)** $BEEF|_{16} = 1011_1110_1110_1111|_2$
 $BEEF|_{16} = 11 \cdot 16^3 + 14 \cdot 16^2 + 14 \cdot 16^1 + 15 \cdot 16^0 = 48.879|_{10}$
- j)** $0101_1101|_2 = 5D|_{16}$
 $5D|_{16} = 5 \cdot 16^1 + 13 \cdot 16^0 = 93|_{10}$
- k)** $201|_{10} = 128 + 64 + 8 + 1 = 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^3 + 1 \cdot 2^0 = 1100_1001|_2 = C9|_{16}$
- l)** $ABCD|_{16} = 1010_1011_1100_1101|_2$
 $ABCD|_{16} = 10 \cdot 16^3 + 11 \cdot 16^2 + 12 \cdot 16^1 + 13 \cdot 16^0 = 43.981|_{10}$

2. Rezolvați următoarele operații, după conversia tuturor operanzilor în baza rezultatului:

- a)** $100|_{10} + 100|_2 + 100|_{16} = ?|_{10}$
- b)** $100|_{10} + 100|_2 + 100|_{16} = ?|_{16}$
- c)** $100|_{10} + 100|_2 + 100|_{16} = ?|_2$
- d)** $11|_{10} + 11|_{16} = ?|_{16}$
- e)** $2012|_{16} + ABCD|_{16} = ?|_{16}$

f) $2012|_{16} + BEEF|_{16} = ?|_{16}$

g) $2012|_{10} + 2025|_{16} = ?|_{16}$

Soluție

a) $100|_2 = 1 \cdot 2^2 = 4|_{10}$

$100|_{16} = 1 \cdot 16^2 = 256|_{10}$

În baza 10, calculul devine: $100 + 4 + 256 = 360|_{10}$

b) $100|_{10} = 6 \cdot 16^1 + 4 \cdot 16^0 = 64|_{16}$

$100|_2 = 4|_{16}$

În baza 16, calculul devine $64|_{16} + 4|_{16} + 100|_{16} = 168|_{16}$.

c) $100|_{10} = (64 + 32 + 4)|_{10} = (1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^2)|_{10} = 0110_0100|_2$

$100|_{16} = 1_0000_0000|_2$

În baza 2, calculul devine:

$$\begin{array}{r} 0_0110_0100+ \\ 0_0000_0100 \\ 1_0000_0000+ \\ \hline 1_0110_1000 \end{array}$$

d) $11|_{10} = B|_{16}$

În baza 16, calculul devine: $B|_{16} + 11|_{16} = 1C|_{16}$.

e), f) Numerele se pot aduna direct în baza 16 prin dispunerea lor unul sub altul și adunarea de la dreapta la stânga a câte unei cifre, cu considerarea transportului.

000	<- transport	011	<- transport
2012	+	2012	+
ABCD		BEEF	
----		----	
CBDF		DF01	

g) $2012|_{10} = 7 \cdot 16^1 + 13 \cdot 16^0 + 12 = 7DC|_{16}$;

În baza 16, calculul devine:

$$\begin{array}{r} 011 \quad <- \text{transport} \\ 7DC + \\ 2025 \\ \hline 2801 \end{array}$$

3. Convertiți următoarele numere din baza 2 în baza 10.

$1101_0111|_2 = ?|_{10}$

$1010_1010|_2 = ?|_{10}$

$110_0100_1001|_2 = ?|_{10}$

$$\begin{aligned}
1_1000_0110_1001_1111|_2 &= ?|_{10} \\
111_1101_1100|_2 &= ?|_{10} \\
0011_1000_0111_0000_0111|_2 &= ?|_{10}
\end{aligned}$$

Soluție

O soluție rapidă constă în conversia numerelor din baza 2 în baza 16, prin gruparea a câte 4 biți, și apoi din baza 16 în baza 10.

$$\begin{aligned}
1101_0111|_2 &= D7|_{16} = 13 \cdot 16^1 + 7 \cdot 16^0 = 215|_{10} \\
1010_1010|_2 &= AA|_{16} = 10 \cdot 16^1 + 10 \cdot 16^0 = 170|_{10} \\
110_0100_1001|_2 &= 649|_{16} = 6 \cdot 16^2 + 4 \cdot 16^1 + 9 \cdot 16^0 = 1609|_{10}
\end{aligned}$$

4. Converteți următoarele numere din baza 10 în baza 2.

104, 1024, 1907, 1921, 1989, 2012, 2020, 4095, 4097, 7777, 8192.

Soluție

Conversiile în binar din cadrul acestui exercițiu sunt realizate prin descompunerea numărului reprezentat în baza 10, ca sume ponderate ale puterilor lui 2.

$$\begin{aligned}
104|_{10} &= 64 + 32 + 8 = 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^3 = 110_1000|_2 \\
1024|_{10} &= 1 \cdot 2^{10}|_{10} = 100_0000_0000|_2 \\
1907|_{10} &= 1024 + 512 + 256 + 64 + 32 + 16 + 2 + 1 = 1 \cdot 2^{10} + 1 \cdot 2^9 + 1 \cdot 2^8 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^1 + 1 \cdot 2^0 = 111_0111_0011|_2 \\
1989|_{10} &= 1024 + 512 + 256 + 128 + 64 + 4 + 1 = 1 \cdot 2^{10} + 1 \cdot 2^9 + 1 \cdot 2^8 + 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^2 + 1 \cdot 2^0 = 111_1100_0101|_2 \\
2012|_{10} &= 1024 + 512 + 256 + 128 + 64 + 16 + 8 + 4 = 1 \cdot 2^{10} + 1 \cdot 2^9 + 1 \cdot 2^8 + 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 = 111_1101_1100|_2 = 0000_0111_1101_1100|_2.
\end{aligned}$$

În cazul conversiei intermediare în baza 16, se reprezintă numărul în baza 16, iar apoi se reprezintă fiecare cifră în binar pe 4 biți. Conversia din baza 10 în baza 16 se poate face prin împărțiri succesive la 16 și considerarea câturilor intermediare și a restului final.

$$\begin{aligned}
104|_{10} &= 6 \cdot 16^1 + 8 \cdot 16^0 = 68|_{16} = 110_1000|_2. \\
1024 &= 4 \cdot 16^2 + 0 \cdot 16^1 + 0 \cdot 16^0 = 400|_{16} = 100_0000_0000|_2. \\
1907|_{10} &= 7 \cdot 16^2 + 7 \cdot 16^1 + 3 \cdot 16^0 = 773|_{16} = 111_0111_0011|_2. \\
1989|_{16} &= 7 \cdot 16^2 + 12 \cdot 16^1 + 5 \cdot 16^0 = 7C5|_{16} = 111_1100_0101|_2.
\end{aligned}$$

5. Converteți numerele din baza 16 în baza 2.

6ABC|₁₆, ABCD|₁₆, BEBE|₁₆, 2012|₁₆, DEAD|₁₆, BEEF|₁₆, 2025|₁₆, DEAF|₁₆, DAD|₁₆.

Soluție

Fiecare cifră hexazecimală va genera un grup de 4 cifre binare.

$$\begin{aligned}
6ABC|_{16} &= 0110_1010_1011_1100|_2 \\
ABCD|_{16} &= 1010_1011_1100_1101|_2 \\
BEBE|_{16} &= 1011_1110_1011_1110|_2
\end{aligned}$$

6. Convertiți următoarele numere din binar în hexazecimal grupând câte 4 biți pentru a forma o cifră în baza 16:

$$010101010011|_2 = ?|_{16}$$

$$111111010010|_2 = ?|_{16}$$

$$100101101|_2 = ?|_{16}$$

$$11100011100|_2 = ?|_{16}$$

$$1000001|_2 = ?|_{16}$$

$$100|_2 = ?|_{16}$$

$$10111101|_2 = ?|_{16}$$

$$10100111|_2 = ?|_{16}$$

$$11011110101011011011111011101111|_2 = ?|_{16}$$

Soluție

$$010101010011|_2 = 0101_0101_0011|_2 = 553|_{16}$$

$$111111010010|_2 = 1111_1101_0010|_2 = FD2|_{16}$$

$$100101101|_2 = 1_0010_1101|_2 = 12D|_{16}$$

$$11100011100|_2 = 111_0001_1100|_2 = 71C|_{16}$$

$$1000001|_2 = 100_0001|_2 = 41|_{16}$$

$$100|_2 = 4|_{16}$$

$$10111101|_2 = 1011_1101|_2 = BD|_{16}$$

$$10100111|_2 = 1010_0111|_2 = A7|_{16}$$

$$11011110101011011011111011101111|_2 =$$

$$= 1101_1110_1010_1101_1011_1110_1110_1111|_2 = DEAD_BEEF|_{16}$$

7. Convertiți următoarele numere din hexazecimal în zecimal:

EA2, ABC, 777, BABA, ABC5, ABACE, E, 14, 234, FF01.

Convertiți aceleași numere în binar prin asocierea a 4 cifre binare fiecărei cifre hexazecimale.

Soluție

Conversia numerelor din hexazecimal direct în zecimal:

$$EA2|_{16} = 14 \cdot 16^2 + 10 \cdot 16^1 + 2 \cdot 16^0 = 3584 + 160 + 2 = 3746|_{10};$$

$$ABC|_{16} = 10 \cdot 16^2 + 11 \cdot 16^1 + 12 \cdot 16^0 = 2560 + 176 + 12 = 2748|_{10};$$

$$777|_{16} = 7 \cdot 16^2 + 7 \cdot 16^1 + 7 \cdot 16^0 = 1792 + 112 + 7 = 1911|_{10};$$

$$BABA|_{16} = 11 \cdot 16^3 + 10 \cdot 16^2 + 11 \cdot 16^1 + 10 \cdot 16^0 = 45056 + 2560 + 176 + 10 = 47802|_{10}.$$

Conversia numerelor din hexazecimal în zecimal, cu conversie intermediară în binar prin gruparea a câte 4 biți:

$$EA2|_{16} = 1110_1010_0010|_2 = 3746|_{10};$$

$$ABC|_{16} = 1010_1011_1100|_2 = 2748|_{10};$$

$$777|_{16} = 0111_0111_0111|_2 = 1911|_{10};$$

$$BABA|_{16} = 1011_1010_1011_1010|_2 = 47802|_{10}.$$

8. Scrieți următoarele numere reprezentate pe 8 biți, în complement față de 2:

33, -27, 14, -17, 100, -98, 125, -123.

Determinați numerele negare, în complement față de 2:

-33, 27, -14, 17, -100, 98, -125, 123.

Soluție

- $33|_{10} = 10_0001|_2 = 0_0100001|_{C2}$
- $27|_{10} = 1_1011|_2 = 0001_1011|_2$. -27 fiind un număr negativ are bitul de semn egal cu 1.
 $-27|_{10} = 1_1100101|_{C2}$
- $14|_{10} = 1110|_2 = 0_000_1110|_{C2}$.
- $17|_{10} = 1_0001|_2 = 0001_0001|_2$.
 $-17|_{10} = 1_110_1111|_{C2}$
- $100|_{10} = 110_0100|_2 = 0_110_0100|_{C2}$.

9. Determinați numerele întregi reprezentate în complement față de 2 după cum urmează (bitul cel mai semnificativ este bitul de semn):
 01101110, 1101111010101101, 01011010011, 1100000001,
 001111110, 110001010.

Soluție

- $0_11_0111|_{C2} = +37|_{16} = +55|_{10}$
- $1_101_1110_1010_1101|_{C2}$ are bitul de semn egal cu 1, deci este negativ.
 $\overline{1_101_1110_1010_1101}|_{C2} + 1 = 0_010_0001_0101_0011|_2 = 2153|_{16} = 2 \cdot 16^3 + 1 \cdot 16^2 + 5 \cdot 16^1 + 3 \cdot 16^0 = 8531|_{10}$.
 Deci, numărul în baza 10 este $-8531|_{10}$
- $0_10_1101_0011|_{C2} = +2D3|_{16} = +723|_{10}$

S-a notat cu bară deasupra negatul unui număr binar: $\bar{1} = 0, \bar{0} = 1$.

10. Precizați domeniul numerelor ce pot fi reprezentate astfel:

- numere pozitive, pe 7 biți
- numere pozitive, pe 8 biți
- numere pozitive, pe 4 bytes
- numere întregi, codificate în complement față de 2, pe 7 biți
- numere întregi, codificate în complement față de 2, pe 8 biți
- numere întregi, codificate în complement față de 2, pe 4 bytes

Soluție

Pe N biți, pot fi reprezentate 2^N numere pozitive, între 0 și $2^N - 1$.

Pe N biți, pot fi reprezentate $2^N - 1$ numere întregi, între $-(2^{N-1} - 1)$ și $+(2^{N-1} - 1)$. Combinația $10\dots 0$ ar putea reprezenta cel mai mic număr negativ, -2^{N-1} , însă nu va exista un corespondent ca număr pozitiv, $+2^{N-1}$.

a) Pentru $N = 7$ biți, pot fi reprezentate numerele pozitive între 0 și $2^7 - 1$, adică între 0 (000_0000₂) și 127 (111_1111₂).

b) Pentru $N = 8$ biți, pot fi reprezentate numerele pozitive între 0 și $2^8 - 1$, adică între 0 (0000_0000₂) și 255 (1111_1111₂).

c) Pentru $N = 32$ biți, pot fi reprezentate numerele pozitive între 0 și $2^{32} - 1$.

d) Pentru $N = 7$ biți, domeniul de reprezentare a numerelor în complement față de 2 este între -63 și $+63$.

Numărul zero are o singură reprezentare 000_0000_{C2}.

Numărul pozitiv maxim este 63, reprezentat ca 011_1111_{C2}.

Numărul negativ -63 este reprezentat ca 100_0001_{C2}.

e) Pentru $N = 8$ biți, domeniul de reprezentare a numerelor în complement față de 2 este între -127 și $+127$.

Numărul zero are o singură reprezentare 0000_0000_{C2}.

Numărul pozitiv maxim este 127, reprezentat ca 0111_1111_{C2}.

Numărul negativ -127 este reprezentat ca 1000_0001_{C2}.

11. Aflați reprezentarea binară a numărului 0010100100010111_{BCD}.

Soluție

Se grupează numărul reprezentat în format BCD în grupuri de câte 4 biți, de la dreapta la stânga, obținându-se:

0010_1001_0001_0111.

Pentru fiecare grup de 4 biți ce reprezintă o cifră în format BCD, se află corespondentul în baza 10:

0010 = 2, 1001 = 9, 0001 = 1, 0111 = 7.

Rezultă că 0010_1001_0001_0111_{BCD} = 2917₁₀.

Prin transformarea numărului din baza 10 în baza 2, rezultă:

2917₁₀ = 1011_0110_0101₂.

În concluzie:

0010_1001_0001_0111_{BCD} = 1011_0110_0101₂.

12. Reprezentați numerele 123 și 789 în format BCD și apoi însumați-le. Repetați operația cu numerele (777 și 456), (839 și 352), (784 și 609).

Soluție

Se plasează cele două numere unul sub altul, ca la o adunare în baza 10:

123+	0001_0010_0011+
789	0111_1000_1001
---	-----
912	1001_0001_0010

Adunarea cifrelor BCD nu trebuie făcută ca a numerelor în binar deoarece, în cazul numerelor BCD, transportul este generat la valoarea zecimală 10.

Astfel:

$0011 + 1001 = 0010$, $transport = 1$, adică $3 + 9 = 2$, $transport = 1$.

La fel:

$0010 + 1000 + 1 = 0001$, $transport = 1$ ($2 + 8 + 1 = 1$, $transport = 1$)

și similar

$0001 + 0111 + 1 = 1001$, $transport = 0$ ($1 + 7 + 1 = 9$).

839+	1000_0011_1001+	784+	0111_1000_0100+
352	0011_0101_0010	609	0110_0000_1001
----	-----	----	-----
1191	0001_0001_1001_0001	1393	0001_0011_1001_0011

13. Cu ce operație aritmetică se pot determina codurile ASCII ale caracterelor ce reprezintă cifre, pe baza cifrei respective?

Soluție

În codul ASCII, caracterele asociate cifrelor sunt codificate cu primele 4 cifre binare $msb = 0011$. Ultimele 4 cifre binare reprezintă cifra scrisă în binar.

Astfel, codul ASCII al caracterului „0” este $0011_0000|_2 = 30|_{16}$. Se poate considera că pentru a obține codul ASCII al unei cifre, trebuie adunată acea cifră la codul ASCII asociat caracterului „0”.

Astfel: „1”=„0”+1, „7”=„0”+7, „9”=„0”+9.

Această observație este utilizată pentru conversia numerelor reprezentate în binar în codul ASCII corespunzător cifrei, în vederea afișării acestuia în sisteme de afișaj.

14. Ce bit trebuie complementat în codul ASCII pentru a converti un șir de litere minuscule în corespondentele lor majuscule?

Soluție

Între codurile ASCII asociate unei litere majuscule și aceeași literă minusculă este o diferență de $20|_{16} = 32|_{10} = 2^5$. Cu alte cuvinte, minusculele și majuscule au același cod ASCII cu excepția bitului 5 care este 0 la litera majusculă și 1 la litera minusculă. Pentru conversia unei litere din minusculă în majusculă trebuie complementat bitul 5.

15. Ce valoare în baza 10 reprezintă șirul binar 1000_1001_0111 dacă valoarea este codificată în:
- a) cod BCD
 - b) cod complement față de 2
 - c) număr pozitiv.

Soluție

- a) $1000_1001_0111|_{BCD} = 897|_{10}$
 b) $1000_1001_0111|_{C2} = -(0111_0110_1000|_2 + 1) = -(0111_0110_1001|_2) = -(1024 + 512 + 256 + 64 + 32 + 8 + 1) = -1897|_{10}$
 c) $1000_1001_0111|_2 = 2^{11} + 2^7 + 2^4 + 2^2 + 2^1 + 2^0 = 2048 + 128 + 16 + 4 + 2 + 1 = 2199|_{10}$

16. Convertiți următoarele numere reprezentate în BCD în baza 10 și prezentați modul în care ați obținut rezultatul:

10010111, 10000010, 11001001001, 11000011010011000, 11101011000, 00100011000100010111.

Convertiți în binar numerele prezentate.

Soluție

Pentru a converti un număr reprezentat în format BCD în format zecimal, se grupează câte 4 biți, de la dreapta la stânga. Fiecare grup de 4 cifre binare va genera o cifră a numărului exprimat în baza 10. Conversia în binar se va face din reprezentarea în baza 10.

$1001_0111|_{BCD} = 97|_{10} = 110_0001|_2,$
 $1000_0010|_{BCD} = 82|_{10} = 101_0010|_2,$
 $110_0100_1001|_{BCD} = 649|_{10} = 10_1000_1001|_2,$
 $1_1000_0110_1001_1000|_{BCD} = 18698|_{10} = 100_1001_0000_1010|_2,$
 $111_0101_1000|_{BCD} = 758|_{10} = 10_1111_0110|_2,$
 $0010_0011_0001_0001_0111|_{BCD} = 23117|_{10} = 5A4D|_{16} = 101_1010_0100_1101|_2.$

17. Realizați următoarele scăderi ale numerelor pozitive precizate prin determinarea numărului negativ asociat scăzătorului și realizarea unei operații de adunare, conform formulei $A - B = A + (-B)$. Extindeți bitul de semn pentru a putea reprezenta numerele negative.

- a) $11011 - 11001$
- b) $110110 - 10101$
- c) $1100 - 110111$
- d) $1001010 - 1001011$

Soluție

Extinderea bitului de semn are ca scop obținerea unei forme de reprezentare echivalente pe un număr mai mare de biți, necesari pentru efectuarea unor operații aritmetice care generează un rezultat ce nu poate fi reprezentat pe numărul inițial de biți. Extinderea bitului de

semn constă în adăugarea unor biți egali ca valoare cu bitul de semn, în partea stângă, ca biți mai semnificativi.

a) Numerele pozitive au 5 biți. Pentru a reprezenta numerele negative asociate este necesară extinderea operanzilor cu un bit suplimentar $101011 - 11001 = 0_11011 - 0_11001$.

Operația binară corespunde operației cu numere zecimale:

$$27 - 25 = 27 + (-25) = 2$$

$$-25 = -(0_11001|_{C_2}) = 1_00110 + 1 = 1_00111$$

```
011011+
100111
=====
000010
```

b) Cel mai mare număr pozitiv are 6 biți. Pentru a reprezenta numerele negative asociate este necesară extinderea operanzilor la 7 biți: $110110 - 10101 = 0_110110 - 0_010101$

Operația binară corespunde operației cu numere zecimale:

$$54 - 21 = 54 + (-21) = 33$$

$$-21 = -(0_010101|_{C_2}) = 1_101010 + 1 = 1_101011$$

```
0110110+
1101011
=====
0100001
```

c) Cel mai mare număr pozitiv are 6 biți. Pentru a reprezenta numerele negative asociate este necesară extinderea operanzilor la 7 biți: $1100 - 110111 = 0_001100 - 0_110111$

Operația binară corespunde operației cu numere zecimale:

$$12 - 55 = 12 + (-55) = -43$$

$$-43 = -(0_110111|_{C_2}) = 1_001000 + 1 = 1_001001$$

$$-(1_010101|_{C_2}) = 0_101011|_2 = 43|_{10}$$

```
0001100+
1001001
=====
1010101
```

d) Operația binară corespunde operației cu numere zecimale:

$$74 - 75 = 74 + (-75) = -1.$$

```

01001010+
10110101
=====
11111111

```

18. Reprezentați următoarele operații cu numere codificate în complement față de 2 pe 16 biți. Verificați rezultatele prin conversia numerelor în baza 10.

- a)** $(+87) + (+22)$
- b)** $(+87) + (-22)$
- c)** $(-87) + (+22)$
- d)** $(-87) + (-22)$
- e)** $(+8765) + (+502)$
- f)** $(-8765) + (+502)$
- g)** $(+8765) + (-502)$
- h)** $(-8765) + (-502)$

Soluție

- a)** $0000_0000_0101_0111|_2 + 0000_0000_0001_0110|_2 = 0000_0000_0110_1101|_2 = 109|_{10};$
- b)** $0000_0000_0101_0111|_2 + 1111_1111_1110_1010|_2 = 0000_0000_0100_0001|_2 = 65|_{10};$
- c)** $1111_1111_1010_1001|_2 + 0000_0000_0001_0110|_2 = 1111_1111_1011_1111|_2 = -65|_{10};$

1.4 Pentru cei ce vor să devină profesioniști

1. Câte cifre sunt egale cu 1 în reprezentarea binară a următoarelor numere zecimale: 4, 16, 33, 63, 126, 257, 1022?

Găsiți o modalitate mai rapidă de determinare, fără a face conversia numerelor în baza 2.

Soluție

- 4 este o putere a numărului 2, $4 = 2^2$. Deci, în reprezentarea binară, are un singur 1 ($100|_2$).
- 16 este o putere a numărului 2, $16 = 2^4$. Deci, în reprezentarea binară, are un singur 1 ($1_0000|_2$).
- $33 = 32 + 1 = 1 \cdot 2^5 + 1 \cdot 2^0$. Deci, are două cifre egale cu 1 în reprezentarea binară ($10_0001|_2$).
- $257 = 256 + 1 = 1 \cdot 2^8 + 1 \cdot 2^0$. Deci, două cifre egale cu 1 ($1_0000_0001|_2$).

- 63 poate fi scris ca $64 - 1$. $64 = 2^6$, deci are reprezentarea $100_0000|_2$ (un 1 urmat de 6 cifre 0). Dacă se scade 1, rezultă un număr binar cu 6 cifre de 1 ($011_1111|_2$).
- $1022 = 1023 - 1$, $1023 = 2^{10} - 1$ are 10 cifre 1 ($011_1111_1111|_2$), rezultă că 1022 are 9 cifre egale cu 1, $lsb = 0$ ($011_1111_1110|_2$).

Lecția 2

Algebră Booleană, porți logice, funcții logice

Această lecție face trecerea de la operatorii logici prezentați la matematică spre **poarta logică** (*circuit electronic care implementează o funcție logică*).

Sunt reamintite **axiomele și teoremele algebrei Booleene**, dar sunt introduse și primele modele **Verilog**.

2.1 Pentru seminar

1. Desenați diagramele temporale asociate circuitelor cu porți logice prezentate în figura 2.1.

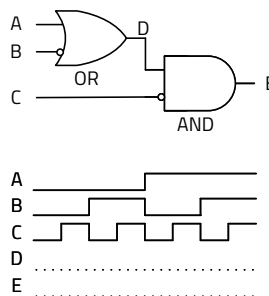


Figura 2.1 Circuite cu porți logice și diagrame temporale asociate (problema 1).

Soluție

Circuitul are trei intrări: A, B și C.

Stimulii de intrare vor avea $2^3 = 8$ valori diferite. Pentru fiecare combinație de intrare se determină valoarea logică a ieșirii porților, în funcție de tipul acesteia.

$$D = A + \overline{B}$$

$$E = D \cdot \overline{C}$$

Formele de undă pentru D și E se determină prin aplicarea funcțiilor menționate pentru toate combinațiile intrărilor. O alternativă constă în utilizarea unui tabel de adevăr pentru centralizarea datelor.

A	B	C	$\overline{B}$	$\overline{C}$	$D = A + \overline{B}$	$E = D \cdot \overline{C}$
0	0	0	1	1	1	1
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	1	0	0	0	0
1	0	0	1	1	1	1
1	0	1	1	0	1	0
1	1	0	0	1	1	1
1	1	1	0	0	1	0

2. Operațiile logice cu șiruri de biți (bus-uri de date) se efectuează considerând fiecare bit separat (operații „bit cu bit”). Se consideră următoarele bus-uri de 8 biți: $A = 10101101$ și $B = 10001110$.

Determinați rezultatul operațiilor:

$$A \cdot B, \overline{A \cdot B}, A + B, \overline{A + B}, A \oplus B, \overline{A \oplus B}, \overline{A}, \overline{B}.$$

Soluție

$$A \cdot B = 1010_1101 \cdot 1000_1110 = 1000_1100;$$

$$\overline{A \cdot B} = \overline{1010_1101 \cdot 1000_1110} = 0111_0011;$$

$$A + B = 1010_1101 + 1000_1110 = 1010_1111;$$

$$\overline{A + B} = \overline{1010_1101 + 1000_1110} = 0101_0000;$$

$$A \oplus B = 1010_1101 \oplus 1000_1110 = 0010_0011$$

$$\overline{A \oplus B} = \overline{1010_1101 \oplus 1000_1110} = 1101_1100$$

$$\overline{A} = \overline{1010_1101} = 0101_0010$$

$$\overline{B} = \overline{1000_1110} = 0111_0001.$$

Puteti verifica soluția cu codul Verilog prezentat în continuare:

```
module op_logice;
  reg [8-1:0] A;
  reg [8-1:0] B;

  initial begin
    A <= 8'b1010_1101;
```

```

B <= 8'b1000_1110;
#10;
$display("A      =%b", A      ); // A
$display("B      =%b", B      ); // B
$display("A.B     =%b", A&B   ); // A and B
$display("~(A.B)  =%b", ~(A&B)); // not (A and B)
$display("A+B     =%b", A|B   ); // A or B
$display("~(A+B)  =%b", ~(A|B)); // not (A or B)
$display("A^B     =%b", A^B   ); // A xor B
$display("~(A^B)  =%b", ~(A^B)); // not (A xor B)
$display("~A      =%b", ~A    ); // not A
$display("~B      =%b", ~B    ); // not B
$stop;
end

endmodule // op_logice

```

3. Se consideră funcția logică:

$$F(W, X, Y, Z) = X \cdot \bar{Y} \cdot Z + \bar{X} \cdot \bar{Y} \cdot Z + \bar{W} \cdot X \cdot Y + W \cdot \bar{X} \cdot Y + W \cdot X \cdot Y$$

- Obțineți tabelul de adevăr al funcției.
- Determinați structura de porți logice care implementează funcția, conform expresiei originale.
- Simplificați expresia funcției, utilizând alegebra Booleană.
- Obțineți tabelul de adevăr al funcției din expresia simplificată și demonstrați că este identic cu cel obținut anterior.
- Determinați structura de porți logice care implementează funcția, conform expresiei minimizate.
- Comparați costurile celor două implementări ca număr de porți logice și ca număr de intrări în porți logice.

Soluție

a) Tabelul de adevăr al funcției este prezentat în continuare. Completarea tabelului se face pe baza observației că funcția $F(W, X, Y, Z)$ este compusă dintr-o expresie OR cu 5 expresii AND cu câte 3 variabile de intrare. Fiecare expresie AND de câte 3 variabile de intrare va genera două valori 1 în coloana funcției, în tabelul de adevăr.

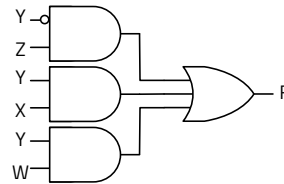
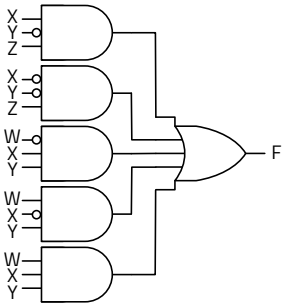
Astfel:

$$\begin{aligned}
 X \cdot \bar{Y} \cdot Z &= \sum(5, 13), \text{ rândurile 5 și 13,} \\
 \bar{X} \cdot \bar{Y} \cdot Z &= \sum(1, 9), \text{ rândurile 1 și 9,} \\
 \bar{W} \cdot X \cdot Y &= \sum(6, 7), \text{ rândurile 6 și 7,} \\
 W \cdot \bar{X} \cdot Y &= \sum(10, 11), \text{ rândurile 10 și 11,} \\
 W \cdot X \cdot Y &= \sum(14, 15), \text{ rândurile 14 și 15.}
 \end{aligned}$$

$$\text{Deci, } F(W, X, Y, Z) = \sum(1, 5, 6, 7, 9, 10, 11, 13, 14, 15)$$

Nr	W	X	Y	Z	F
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

b) Structura de porți inițială este prezentată în figura 2.2-a.



a)

b)

Figura 2.2 Structurile de porți logice: a) inițială, b) simplificată (problema 3).

$$\begin{aligned}
 \text{c) } F &= X \cdot \bar{Y} \cdot Z + \bar{X} \cdot \bar{Y} \cdot Z + \bar{W} \cdot X \cdot Y + W \cdot \bar{X} \cdot Y + W \cdot X \cdot Y = \\
 &= \bar{Y} \cdot Z \cdot (X + \bar{X}) + X \cdot Y \cdot (W + \bar{W}) + W \cdot \bar{X} \cdot Y = \\
 &= \bar{Y} \cdot Z + X \cdot Y + W \cdot \bar{X} \cdot Y = \\
 &= \bar{Y} \cdot Z + Y \cdot (X + \bar{X} \cdot W) = \\
 &= \bar{Y} \cdot Z + Y \cdot (X + W) = \\
 &= \bar{Y} \cdot Z + Y \cdot X + Y \cdot W
 \end{aligned}$$

d) Expresia simplificată a funcției este:

$$F(W, X, Y, Z) = \overline{Y} \cdot Z + Y \cdot X + Y \cdot W$$

Fiecare expresie AND de câte 2 variabile de intrare va genera patru valori 1 în coloana funcției, în tabelul de adevăr.

Astfel:

$$\overline{Y} \cdot Z = \sum(1, 5, 9, 13), \text{ rândurile } 1, 5, 9 \text{ și } 13,$$

$$Y \cdot X = \sum(6, 7, 14, 15), \text{ rândurile } 6, 7, 14 \text{ și } 15,$$

$$Y \cdot W = \sum(10, 11, 14, 15), \text{ rândurile } 10, 11, 14 \text{ și } 15.$$

$$\text{Deci, } F(W, X, Y, Z) = \sum(1, 5, 6, 7, 9, 10, 11, 13, 14, 15)$$

Structura de porți logice ce implementează expresia simplificată este prezentată în figura 2.2-b.

f) Costurile implementării, calculate ca număr de porți logice și ca număr de intrări în porți logice, sunt:

Implementare inițială

3 porți NOT $\times$ 1 intrare;

5 porți AND $\times$ 3 intrări;

1 poartă OR $\times$ 5 intrări.

Total: 9 porți $\times$ 23 intrări.

Implementare după simplificare

1 poartă NOT $\times$ 1 intrare;

3 porți AND $\times$ 2 intrări;

1 poartă OR $\times$ 3 intrări.

Total: 5 porți $\times$ 10 intrări.

2.2 Pentru cei ce vor doar să promoveze examenul

Soluțiile problemelor următoare fac referire la axiomele și teoremele algebrei Booleene (tabelul 2.1).

1. Determinați și justificați valoarea de adevăr a fiecărei afirmații:

a) $A \cdot \overline{B} + \overline{A} \cdot B + B \cdot C = A \cdot \overline{B} + \overline{A} \cdot B + A \cdot C$

b) $A + B \cdot C = A + \overline{A} \cdot B \cdot C$

2. Să se aplice teorema lui De Morgan următoarelor expresii:

a) $\overline{A \cdot B \cdot (C + D)}$

b) $\overline{A \cdot B \cdot C \cdot \overline{D}}$

c) $\overline{A + B + \overline{C}}$

d) $\overline{A \cdot B \cdot (C \cdot D + E)}$

e) $\overline{\overline{A} \cdot B + A \cdot \overline{B}}$

f) $\overline{\overline{A} \cdot B + A \cdot \overline{B} + C}$

Axioma	Forma cu operatorul	
	AND	OR
Axioma 1. Mulțimea $B=\{0, 1\}$ este închisă în raport cu operatorii AND și OR	$\forall X, Y \in B \Rightarrow X \cdot Y \in B$	$\forall X, Y \in B \Rightarrow X + Y \in B$
Axioma 2. Asociativitatea	$X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$	$X + (Y + Z) = (X + Y) + Z$
Axioma 3. Comutativitatea	$X \cdot Y = Y \cdot X$	$X + Y = Y + X$
Axioma 4. Distributivitatea	$X \cdot (Y + Z) = X \cdot Y + X \cdot Z$	$X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$
Axioma 5. Existența elementului neutru	$X \cdot 1 = 1 \cdot X = X$	$X + 0 = 0 + X = X$
Axioma 6. Existența complementului	$X \cdot \bar{X} = \bar{X} \cdot X = 0$	$X + \bar{X} = \bar{X} + X = 1$

Teorema	Forma cu operatorul	
	AND	OR
Teorema 1. Idempotența (tautologia)	$X \cdot X = X$	$X + X = X$
Teorema 2. Legea lui 0 și a lui 1	$X \cdot 0 = 0 \cdot X = 0$	$X + 1 = 1 + X = 1$
Teorema 3. Dubla negație (involuția)	$\bar{\bar{X}} = X$	$\bar{\bar{X}} = X$
Teorema 4. Absorbția directă Absorbția inversă	$X \cdot (X + Y) = X$ $X \cdot (\bar{X} + Y) = X \cdot Y$	$X + X \cdot Y = X$ $X + \bar{X} \cdot Y = X + Y$
Teorema 5. Teorema lui De Morgan	$\overline{X \cdot Y} = \bar{X} + \bar{Y}$	$\overline{X + Y} = \bar{X} \cdot \bar{Y}$

Tabelul 2.1

Axiomele și teoremele algebrei Booleene.

Soluție

Aplicarea teoremei lui De Morgan:

- negata unei funcții AND este o funcție OR în care toți termenii se neagă individual.
- negata unei funcții OR este o funcție AND în care toți termenii se neagă individual.

$$\text{b)} \quad \overline{A \cdot B \cdot C \cdot \overline{D}} = \overline{A} + \overline{B} + \overline{C} + D$$

$$\text{c)} \quad \overline{A + B + \overline{C}} = \overline{A} \cdot \overline{B} \cdot C$$

3. Folosind algebra Booleană, simplificați expresiile și aduceți-le la o formă echivalentă, exprimată cu un număr minim de litere.

- $X \cdot Y + X \cdot \overline{Y}$
- $(X + Y) \cdot (X + \overline{Y})$
- $Y \cdot Z + Y \cdot \overline{Z} + X \cdot \overline{Y} \cdot \overline{Z}$
- $X \cdot Y \cdot Z + \overline{X} \cdot Y + X \cdot Y \cdot \overline{Z}$
- $\overline{(X + Y)} \cdot \overline{(X + \overline{Y})}$
- $(Y + Z) \cdot (\overline{Y} + Z) \cdot (X + \overline{Y} + Z)$

Soluție

- Utilizând Axioma 4 (A4) și ulterior A6 și A5, expresia devine:

$$X \cdot Y + X \cdot \overline{Y} = X \cdot (Y + \overline{Y}) = X$$
- Aplicând A4 și Teorema 1 (T1) expresia devine:

$$(X + Y) \cdot (X + \overline{Y}) = X \cdot X + X \cdot \overline{Y} + Y \cdot X + Y \cdot \overline{Y} =$$

$$X + X \cdot (Y + \overline{Y}) + 0 = X + X \cdot 1 + 0 = X + X + 0 = X$$
- Conform A4, expresia devine:

$$X \cdot Y \cdot Z + \overline{X} \cdot Y + X \cdot Y \cdot \overline{Z} = X \cdot Y \cdot (Z + \overline{Z}) + \overline{X} \cdot Y$$
Prin aplicarea A2 și ulterior A4 și încă o dată A2:

$$X \cdot Y + \overline{X} \cdot Y = Y \cdot (X + \overline{X}) = Y$$
- Prin aplicarea T5 și ulterior A6, expresia devine:

$$\overline{(X + Y)} \cdot \overline{(X + \overline{Y})} = \overline{X} \cdot \overline{Y} \cdot X \cdot Y = 0$$

4. Aflați expresiile complementare (negate) următoarelor expresii:

- $X \cdot Y + \overline{X} \cdot \overline{Y}$
- $X \cdot \overline{Y} + Z(W + Q)$
- $(X + \overline{Y} + \overline{Z}) \cdot X$
- $A \cdot B + \overline{B} \cdot C + \overline{A} \cdot \overline{B} \cdot \overline{C}$

5. Să se determine porțile logice echivalente porților:

- a) $A \cdot \overline{B}$
- b) $\overline{A + \overline{B}}$
- c) $\overline{A \cdot \overline{B} \cdot C}$
- d) $\overline{A + B + \overline{C}}$

Soluție

O poartă logică echivalentă se obține prin negarea atât a intrărilor cât și a ieșirilor și interschimbarea operatorilor logici AND $\Leftrightarrow$ OR. Transformarea se poate justifica analitic prin aplicarea teoremei lui De Morgan (T5). Eventual, anterior aplicării teoremei lui De Morgan, se aplică întregii expresii o dublă negare (T3).

- a) $A \cdot \overline{B} = \overline{\overline{A \cdot \overline{B}}} = \overline{\overline{A} + \overline{\overline{B}}} = \overline{\overline{A} + B}$
- c) $\overline{A \cdot \overline{B} \cdot C} = \overline{A} + \overline{\overline{B}} + \overline{C} = \overline{A} + B + \overline{C}$

6. Să se determine structura de porți logice care realizează următoarea funcție logică:

$$Y = (A \cdot B + C + D) \cdot \overline{E \cdot F \cdot G}$$

Modelați în Verilog circuitul. Utilizați specificații `assign` și operatorii asociați porților logice prezente în circuit.

Soluție

Structura de porți reprezintă o implementare a operatorilor prezenți în expresia funcției, ținând cont de precedența acestora (figura 2.3).

Modelul Verilog asociat este următorul:

```
module cir_logic (
input      A, B, C, D, E, F, G,
output     Y
);

assign wire_and   = A & B;
assign wire_or    = wire_and | C | D;
assign wire_nand  = ~(E & F & G);
assign Y          = wire_or & wire_nand;

endmodule // cir_logic
```

7. Să se determine funcția logică a structurii de porți logice prezentate în figura 2.4. Modelați în Verilog circuitul. Utilizați specificații `assign` și operatorii asociați porților logice prezente în circuit.
8. Operațiile logice cu șiruri de biți (bus-uri de date) se efectuează considerând fiecare bit separat (operații „bit cu bit”). Se consideră următoarele bus-uri de 8 biți: $A = 1001_0111$ și $B = 0111_0011$.

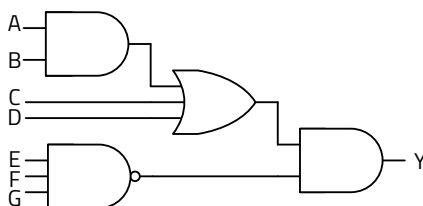


Figura 2.3 Structura de porți logice pentru implementarea funcției de la problema 6.

Determinați rezultatul operațiilor: $A \cdot B$, $\overline{A \cdot B}$, $A + B$, $\overline{A + B}$, $A \oplus B$, $\overline{A \oplus B}$, $\overline{A}$, $\overline{B}$.

Completați codul cu specificații `assign` potrivite și verificați răspunsul în simulare.

```
wire [8-1:0] A      ;
wire [8-1:0] B      ;

wire [8-1:0] and_AB  ;
wire [8-1:0] nand_AB ;
wire [8-1:0] or_AB   ;
wire [8-1:0] nor_AB  ;
wire [8-1:0] xor_AB  ;
wire [8-1:0] xnor_AB ;

assign and_AB  = ..... ;
assign nand_AB = ..... ;
assign or_AB   = ..... ;
assign nor_AB  = ..... ;
assign xor_AB  = ..... ;
assign xnor_AB = ..... ;
```

2.3 Pentru cei ce vor să învețe

1. Determinați și justificați valoarea de adevăr a fiecărei afirmații:

- a) $W \cdot X + \overline{Y} + Z = (W + \overline{Y} + Z) \cdot (X + \overline{Y} + \overline{W} \cdot Z + W \cdot Z)$
- b) $X \cdot Y + X \cdot \overline{Y} = X$

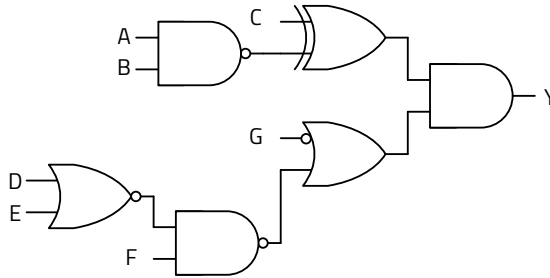


Figura 2.4 Structura de porți logice (problema 7).

- c) $X + \overline{X} \cdot Y = X + Y$
d) $X \cdot (\overline{X} + Y) = X \cdot Y$

Soluție

a) Expresia din partea dreaptă devine, după aplicarea A4:

$(W + \overline{Y} + Z) \cdot (X + \overline{Y} + \overline{W} \cdot Z + W \cdot Z) = (W + \overline{Y} + Z) \cdot (X + \overline{Y} + Z \cdot (W + \overline{W})) =$
utilizând A6 și după aplicarea A4:

$= (W + \overline{Y} + Z) \cdot (X + \overline{Y} + Z) = W \cdot X + W \cdot \overline{Y} + W \cdot Z + \overline{Y} \cdot X + \overline{Y} \cdot \overline{Y} +$
 $\overline{Y} \cdot Z + Z \cdot X + Z \cdot \overline{Y} + Z \cdot Z = W \cdot X + \overline{Y} + Z \cdot \overline{Y} + Z + W \cdot Z + X \cdot Z =$

Utilizând T4, expresia devine egală cu expresia din partea stângă:

$= W \cdot X + \overline{Y} + Z$

b) Prin aplicarea A4 și utilizând A6, expresia din partea stângă devine:

$X \cdot Y + X \cdot \overline{Y} = X \cdot (Y + \overline{Y}) = X$

c) Prin aplicarea T4 este justificată valoarea de adevăr a următoarei afirmații:

$X + \overline{X} \cdot Y = X + Y$

d) Prin aplicarea A4 și ulterior A6 expresiei din partea stângă, aceasta devine egală cu expresia din partea dreaptă:

$X \cdot (\overline{X} + Y) = X \cdot \overline{X} + X \cdot Y = X \cdot Y$

2. Utilizând axiomele și teoremele algebrei Booleene, să se demonstreze următoarele identități:

a) $B + \overline{A} \cdot C = (A + B + C) \cdot (\overline{A} + B + C) \cdot (\overline{A} + B + \overline{C})$

b) $\overline{A} \cdot D + \overline{C} \cdot D + A \cdot \overline{B} =$
 $= \overline{A} \cdot \overline{C} \cdot D + A \cdot \overline{C} \cdot D + A \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot C + \overline{A} \cdot C \cdot D$

$$\begin{aligned} \text{c)} \quad & D \cdot (\overline{A} + B + C + \overline{D}) \cdot (A + B + \overline{C} + \overline{D}) = \\ & = (D + A \cdot \overline{C} + \overline{A} \cdot C) \cdot (\overline{A} \cdot \overline{C} + B \cdot D + A \cdot C) \end{aligned}$$

Soluție

a) Expresia din partea dreaptă devine, după aplicarea A4:

$$\begin{aligned} & (A + B + C) \cdot (\overline{A} + B + C) \cdot (\overline{A} + B + \overline{C}) = \\ & = (A \cdot \overline{A} + A \cdot B + A \cdot C + \overline{A} \cdot B + B \cdot B + B \cdot C + \overline{A} \cdot C + B \cdot C + C \cdot C) \cdot (\overline{A} + B + \overline{C}) = \\ & \text{utilizând A6, T1 și ulterior A4 (invers, pentru B și C):} \\ & = (A \cdot B + A \cdot C + \overline{A} \cdot B + B + B \cdot C + \overline{A} \cdot C + C) \cdot (\overline{A} + B + \overline{C}) = \\ & = [C \cdot (A + B + \overline{A} + 1) + B \cdot (A + \overline{A} + 1)] \cdot (\overline{A} + B + \overline{C}) = \\ & \text{utilizând T2, A5 și aplicarea A4 (invers, pentru B):} \\ & = (C + B) \cdot (\overline{A} + B + \overline{C}) = \overline{A} \cdot C + B \cdot C + C \cdot \overline{C} + \overline{A} \cdot B + B \cdot B + B \cdot \overline{C} = \\ & \text{aplicând A5, rezultă:} \\ & = \overline{A} \cdot C + B \cdot (C + \overline{A} + B + \overline{C}) = \overline{A} \cdot C + B = B + \overline{A} \cdot C, \\ & \text{expresie egală cu expresia din partea stângă.} \end{aligned}$$

b) Prin gruparea termenilor din partea dreaptă și aplicarea A4:

$$\begin{aligned} & (\overline{A} \cdot \overline{C} \cdot D + A \cdot \overline{C} \cdot D) + (A \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot C) + \overline{A} \cdot C \cdot D = \\ & = \overline{C} \cdot D \cdot (\overline{A} + A) + A \cdot \overline{B} \cdot (\overline{C} + C) + \overline{A} \cdot C \cdot D = \\ & \text{aplicarea A6 și gruparea primului termen cu ultimul:} \\ & = \overline{C} \cdot D + A \cdot \overline{B} + \overline{A} \cdot C \cdot D = (\overline{C} \cdot D + \overline{A} \cdot C \cdot D) + A \cdot \overline{B} = D \cdot (\overline{C} + \overline{A} \cdot C) + A \cdot \overline{B} = \\ & \text{la paranteză se aplică T4 și apoi A4:} \\ & = D \cdot (\overline{C} + \overline{A}) + A \cdot \overline{B} = \\ & \text{după aplicarea A4 și A3:} \\ & = D \cdot \overline{C} + D \cdot \overline{A} + A \cdot \overline{B} = \overline{A} \cdot D + \overline{C} \cdot D + A \cdot \overline{B}, \\ & \text{care este expresia din partea stângă.} \end{aligned}$$

c) Prin aplicarea A4 expresiilor din parantezele din partea stângă:

$$\begin{aligned} & D \cdot (\overline{A} + B + C + \overline{D}) \cdot (A + B + \overline{C} + \overline{D}) = \\ & = D \cdot (\overline{A} \cdot A + \overline{A} \cdot B + \overline{A} \cdot C + \overline{A} \cdot \overline{D} + A \cdot B + B \cdot B + B \cdot \overline{C} + B \cdot \overline{D} + A \cdot C + \\ & \quad B \cdot C + C \cdot \overline{C} + C \cdot \overline{D} + A \cdot \overline{D} + B \cdot \overline{D} + \overline{C} \cdot \overline{D} + \overline{D} \cdot \overline{D}) = \\ & \text{și restrângerea termenilor utilizând A6 și T1:} \\ & = D \cdot (0 + \overline{A} \cdot B + \overline{A} \cdot C + \overline{A} \cdot \overline{D} + A \cdot B + B + B \cdot \overline{C} + B \cdot \overline{D} + A \cdot C + B \cdot C + \\ & \quad 0 + C \cdot \overline{D} + A \cdot \overline{D} + B \cdot \overline{D} + \overline{C} \cdot \overline{D} + \overline{D}) = \\ & \text{pe baza A3, se grupează B și } \overline{D}: \\ & = D \cdot [B \cdot (\overline{A} + A + 1 + \overline{C} + \overline{D} + C) + \overline{A} \cdot \overline{C} + A \cdot C + \overline{D} \cdot (\overline{A} + A + \overline{C} + 1)] = \\ & \text{și se observă că parantezele rotunde sunt egale cu 1, conform T2:} \\ & = D \cdot (B + \overline{A} \cdot \overline{C} + A \cdot C + \overline{D}) = B \cdot D + \overline{A} \cdot \overline{C} \cdot D + A \cdot C \cdot D \end{aligned}$$

Expresia din partea dreaptă se procesează conform A4, A6 și T1:

$$\begin{aligned} & (D + A \cdot \overline{C} + \overline{A} \cdot C) \cdot (\overline{A} \cdot \overline{C} + B \cdot D + A \cdot C) = \\ & = \overline{A} \cdot \overline{C} \cdot D + B \cdot D \cdot D + A \cdot C \cdot D + A \cdot \overline{C} \cdot \overline{A} \cdot \overline{C} + \\ & \quad A \cdot \overline{C} \cdot B \cdot D + A \cdot \overline{C} \cdot A \cdot C + \overline{A} \cdot C \cdot \overline{A} \cdot \overline{C} + \overline{A} \cdot C \cdot B \cdot D + \overline{A} \cdot C \cdot A \cdot C = \\ & = \overline{A} \cdot \overline{C} \cdot D + B \cdot D + A \cdot C \cdot D + A \cdot B \cdot \overline{C} \cdot D + \overline{A} \cdot B \cdot C \cdot D = \end{aligned}$$

Prin aplicarea A3 se grupează favorabil termenii pentru a da în factor $B \cdot D$:

$$= (B \cdot D + A \cdot B \cdot \overline{C} \cdot D + \overline{A} \cdot B \cdot C \cdot D) + \overline{A} \cdot \overline{C} \cdot D + A \cdot C \cdot D =$$

$$= B \cdot D \cdot (1 + A \cdot \overline{C} + \overline{A} \cdot C) + \overline{A} \cdot \overline{C} \cdot D + A \cdot C \cdot D =$$

Prin aplicarea T2 se ajunge la aceeași expresie ca după procesarea părții stângi:

$$= B \cdot D + \overline{A} \cdot \overline{C} \cdot D + A \cdot C \cdot D$$

3. Utilizând axiomele și teoremele algebrei Booleene, să se demonstreze următoarele identități:

a) $A \cdot B \cdot C + \overline{A} \cdot \overline{B} + \overline{A \cdot B \cdot C} \cdot D = A \cdot B \cdot C + \overline{A} \cdot \overline{B} + D$

b) $\overline{A \cdot B \cdot C} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot C + A \cdot B \cdot C =$
 $= B \cdot C + A \cdot \overline{B} + \overline{B} \cdot \overline{C}$

c) $A \cdot \overline{B} \cdot C \cdot (B \cdot D + C \cdot D \cdot E) + A \cdot \overline{C} = A \cdot (\overline{C} + \overline{B} \cdot D \cdot E)$

d) $X \cdot Y + X \cdot \overline{Y} = X$

e) $X + \overline{X} \cdot Y = X + Y$

4. Utilizând axiomele și teoremele algebrei Booleene, să se determine forma minimă a funcțiilor complementare următoarelor:

a) $F = \overline{[(\overline{A \cdot B}) \cdot A] \cdot [(\overline{A \cdot B}) \cdot B]}$
 (funcția $A \oplus B$ exprimată prin operatorul NAND)

b) $F = \overline{(A + B + \overline{C}) \cdot (\overline{A \cdot B} + \overline{C \cdot D}) + B \cdot C \cdot D}$

c) $F = \overline{(A \cdot B \cdot C + B \cdot \overline{C} \cdot D) + (\overline{A \cdot C \cdot D} + \overline{B \cdot \overline{C} \cdot D} + B \cdot C \cdot \overline{D})}$

d) $F = X \cdot (X + X \cdot \overline{Y})$

e) $F = \overline{\overline{X}} \cdot (\overline{X} + 1)$

Soluție

a) $\overline{F} = \overline{[(\overline{A \cdot B}) \cdot A] \cdot [(\overline{A \cdot B}) \cdot B]} =$
 $= (\overline{A \cdot B} + \overline{A}) \cdot (\overline{A \cdot B} + \overline{B}) =$
 $= A \cdot B \cdot A \cdot B + \overline{A} \cdot A \cdot B + A \cdot B \cdot \overline{B} + \overline{A} \cdot \overline{B} =$
 $= A \cdot B + 0 + 0 + \overline{A} \cdot \overline{B} = A \cdot B + \overline{A} \cdot \overline{B} = A \oplus B$

b) $\overline{F} = \overline{(A + B + \overline{C}) \cdot (\overline{A \cdot B} + \overline{C \cdot D}) + B \cdot C \cdot D} =$
 $= \overline{(A + B + \overline{C}) \cdot (\overline{A \cdot B} + \overline{C \cdot D})} + \overline{B \cdot C \cdot D} =$
 $= (\overline{A \cdot B \cdot \overline{C}}) \cdot (\overline{\overline{A \cdot B} \cdot \overline{C \cdot D}}) + \overline{B \cdot C \cdot D} =$
 $= (\overline{A \cdot B \cdot C}) \cdot (A \cdot B \cdot C \cdot D) + \overline{B \cdot C \cdot D} =$
 $= \overline{B \cdot C \cdot D}$

c) $\overline{F} = \overline{(A \cdot B \cdot C + B \cdot \overline{C} \cdot D) + (\overline{A \cdot C \cdot D} + \overline{B \cdot \overline{C} \cdot D} + B \cdot C \cdot \overline{D})} =$
 $= (A \cdot B \cdot C + B \cdot \overline{C} \cdot D) \cdot (\overline{A \cdot C \cdot D} + \overline{B \cdot \overline{C} \cdot D} + B \cdot C \cdot \overline{D}) =$
 $= (A \cdot B \cdot C + B \cdot \overline{C} \cdot D) \cdot (\overline{A} + \overline{C} + \overline{D} + \overline{B \cdot \overline{C} \cdot D} + B \cdot C \cdot \overline{D}) =$
 $= (A \cdot B \cdot C + B \cdot \overline{C} \cdot D) \cdot (\overline{A} + \overline{D} \cdot (1 + B \cdot C) + \overline{C} \cdot (1 + \overline{B \cdot D})) =$
 $= (A \cdot B \cdot C + B \cdot \overline{C} \cdot D) \cdot (\overline{A} + \overline{D} + \overline{C}) =$
 $= A \cdot B \cdot C \cdot \overline{A} + A \cdot B \cdot C \cdot \overline{D} + A \cdot B \cdot C \cdot \overline{C} +$
 $B \cdot \overline{C} \cdot D \cdot \overline{A} + B \cdot \overline{C} \cdot D \cdot \overline{D} + B \cdot \overline{C} \cdot D \cdot \overline{C} =$

$$\begin{aligned}
 &= 0 + A \cdot B \cdot C \cdot \overline{D} + 0 + B \cdot \overline{C} \cdot D \cdot \overline{A} + 0 + B \cdot \overline{C} \cdot D = \\
 &= A \cdot B \cdot C \cdot \overline{D} + (\overline{A} \cdot B \cdot \overline{C} \cdot D + B \cdot \overline{C} \cdot D) = A \cdot B \cdot C \cdot \overline{D} + B \cdot \overline{C} \cdot D
 \end{aligned}$$

5. Să se aplice teorema lui De Morgan următoarelor expresii:

- a) $\overline{A \cdot \overline{B} \cdot (C + \overline{D})}$
- b) $\overline{(A + \overline{B} + C + \overline{D}) + A \cdot B \cdot C \cdot \overline{D}}$
- c) $\overline{A \cdot \overline{B} \cdot (C \cdot D + \overline{E} \cdot F) \cdot (\overline{A \cdot B} + \overline{C \cdot D})}$
- d) $\overline{(A + \overline{B \cdot \overline{C}} + C \cdot D) + \overline{B \cdot \overline{C}}}$
- e) $\overline{A \cdot B \cdot (C \cdot D + E \cdot F)}$
- f) $\overline{(\overline{A + B + C + D}) \cdot (\overline{A \cdot \overline{B} \cdot \overline{C} \cdot D})}$
- g) $\overline{(\overline{A \cdot B \cdot C}) \cdot (\overline{E \cdot F \cdot G}) + (\overline{H \cdot I \cdot J}) \cdot (\overline{K \cdot L \cdot M})}$
- h) $\overline{(\overline{A + B}) \cdot (\overline{C + D}) \cdot (\overline{E + F}) \cdot (\overline{G + H})}$

Soluție

- a) $\overline{A \cdot \overline{B} \cdot (C + \overline{D})} = \overline{A} + \overline{\overline{B}} + \overline{(C + \overline{D})} =$
 $= \overline{A} + B + \overline{C \cdot \overline{D}} = \overline{A} + B + \overline{C} \cdot D$
- b) $\overline{(A + \overline{B} + C + \overline{D}) + A \cdot B \cdot C \cdot \overline{D}} =$
 $= \overline{A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}} + \overline{(A + \overline{B} + \overline{C} + \overline{D})} =$
 $= \overline{A \cdot B \cdot \overline{C} \cdot D} + \overline{(A + \overline{B} + \overline{C} + D)}$
- e) $\overline{A \cdot B \cdot (C \cdot D + E \cdot F)} =$
 $= \overline{A} + \overline{B} + \overline{(C \cdot D + E \cdot F)} =$
 $= \overline{A} + \overline{B} + \overline{C \cdot D \cdot E \cdot F} =$
 $= \overline{A} + \overline{B} + (\overline{C} + \overline{D}) \cdot (\overline{E} + \overline{F})$
- f) $\overline{(\overline{A + B + C + D}) \cdot (\overline{A \cdot \overline{B} \cdot \overline{C} \cdot D})} =$
 $= \overline{(\overline{A + B + C + D})} + \overline{(A \cdot \overline{B} \cdot \overline{C} \cdot D)} =$
 $= (\overline{A} + \overline{B} + \overline{C} + \overline{D}) + A \cdot \overline{B} \cdot \overline{C} \cdot D$

6. Demonstrați identitățile analitic, folosind axiomele și teoremele algebrei Booleene:

- a) $Y + \overline{X} \cdot Z + X \cdot \overline{Y} = X + Y + Z$
- b) $\overline{X \cdot Y \cdot Z} = \overline{X} + \overline{Y} + \overline{Z}$
- c) $X + Y \cdot Z = (X + Y) \cdot (X + Z)$
- d) $\overline{X \cdot Y} + \overline{Y \cdot Z} + X \cdot \overline{Z} = X \cdot \overline{Y} + Y \cdot \overline{Z} + \overline{X} \cdot Z$
- e) $\overline{X \cdot \overline{Y}} + \overline{X \cdot Y} + X \cdot Y = \overline{X} + Y$
- f) $\overline{X \cdot Y} + \overline{Y \cdot Z} + X \cdot Y + \overline{Y \cdot Z} = 1$
- g) $\overline{X \cdot \overline{Y}} + \overline{Y \cdot Z} + X \cdot Z + X \cdot Y + Y \cdot \overline{Z} = \overline{X \cdot \overline{Y}} + X \cdot Z + Y \cdot \overline{Z}$
- h) $A \cdot \overline{B} + \overline{A \cdot \overline{C} \cdot \overline{D}} + \overline{A \cdot \overline{B} \cdot D} + \overline{A \cdot \overline{B} \cdot C \cdot \overline{D}} = \overline{B} + \overline{A \cdot \overline{C} \cdot \overline{D}}$
- i) $X \cdot Z + W \cdot \overline{Y \cdot \overline{Z}} + \overline{W \cdot Y \cdot \overline{Z}} + W \cdot \overline{X \cdot \overline{Z}} =$

$$= X \cdot Z + W \cdot \bar{Y} \cdot \bar{Z} + W \cdot X \cdot \bar{Y} + \bar{W} \cdot X \cdot Y + \bar{X} \cdot Y \cdot \bar{Z}$$

Soluție

a) Expresia din partea stângă se procesează conform T4:

$$Y + \bar{X} \cdot Z + X \cdot \bar{Y} = Y + X + \bar{X} \cdot Z = Y + X + Z = X + Y + Z$$

Prin aplicarea A3 se ajunge la o expresie egală cu expresia din partea dreaptă.

b) Expresia din partea stângă devine, prin aplicarea T5, egală cu expresia din partea dreaptă:

$$\bar{X} \cdot Y \cdot \bar{Z} = \bar{X} + \bar{Y} + \bar{Z}$$

c) Utilizând A4 și ulterior T4 expresia din partea dreaptă devine egală cu cea din partea stângă:

$$(X + Y) \cdot (X + Z) = X \cdot X + X \cdot Z + Y \cdot X + Y \cdot Z = X + Y \cdot Z$$

d) $\bar{X} \cdot Y + \bar{Y} \cdot Z + X \cdot \bar{Z} =$

$$= \bar{X} \cdot Y \cdot Z + \bar{X} \cdot Y \cdot \bar{Z} + X \cdot \bar{Y} \cdot Z + \bar{X} \cdot \bar{Y} \cdot Z + X \cdot Y \cdot \bar{Z} + X \cdot \bar{Y} \cdot \bar{Z}$$

$$X \cdot \bar{Y} + Y \cdot \bar{Z} + \bar{X} \cdot Z = X \cdot \bar{Y} \cdot Z + X \cdot \bar{Y} \cdot \bar{Z} + X \cdot Y \cdot \bar{Z} + \bar{X} \cdot Y \cdot \bar{Z} + \bar{X} \cdot Y \cdot Z + \bar{X} \cdot \bar{Y} \cdot Z$$

e) Prin aplicarea A4 și ulterior T1 expresia din partea dreaptă devine:

$$\bar{X} \cdot \bar{Y} + \bar{X} \cdot Y + X \cdot Y = \bar{X} \cdot (\bar{Y} + Y) + X \cdot Y = \bar{X} + X \cdot Y = \bar{X} + Y$$

Aplicând T4 expresia devine egală cu expresia din partea stângă.

7. Folosind algebra Booleană, simplificați expresiile și aduceți-le la o formă echivalentă, exprimată cu un număr minim de litere.

a) $\overline{(X + Y)} \cdot (\bar{X} + \bar{Y})$

b) $\bar{X} \cdot Y \cdot Z + X \cdot Z$

c) $\bar{X} \cdot \bar{Z} + X \cdot Y \cdot Z + X \cdot \bar{Z}$

d) $X \cdot Y \cdot Z + \bar{X} \cdot Y + X \cdot Y \cdot \bar{Z}$

e) $(\bar{X} + \bar{Z}) \cdot (\bar{X} + \bar{Z}) \cdot (X + Y + \bar{Z} \cdot W)$

f) $\overline{(A + B)} \cdot (\bar{A} + \bar{B})$

g) $B \cdot C + B \cdot (A \cdot D + A \cdot \bar{D})$

h) $\bar{X} \cdot \bar{Y} + X \cdot Y \cdot Z + \bar{X} \cdot Y$

i) $\bar{W} \cdot X \cdot (\bar{Z} + \bar{Y} \cdot Z) + X \cdot (W + \bar{W} \cdot Y \cdot Z)$

j) $X \cdot (X + X \cdot \bar{Y})$

k) $(Y \cdot Z + \bar{X} \cdot W) \cdot (X \cdot \bar{Y} + Z \cdot \bar{W})$

l) $X \cdot Y + X \cdot (W \cdot Z + W \cdot \bar{Z})$

m) $\overline{(\bar{X} \cdot \bar{Y} + Z)} + Z + X \cdot Y + W \cdot Z$

n) $\bar{X} \cdot Y \cdot (\bar{W} + \bar{Z} \cdot W) + Y \cdot (X + X \cdot \bar{Z} \cdot W)$

o) $A \cdot B \cdot C + A \cdot B \cdot \bar{C} + \bar{A} \cdot B$

p) $\bar{A} \cdot B \cdot C + A \cdot C$

q) $(A + \bar{B} + A \cdot \bar{B}) \cdot (A \cdot B + \bar{A} \cdot C + B \cdot C)$

r) $X + Y \cdot (Z + \bar{X} + \bar{Z})$

s) $(A \cdot B + \bar{A} \cdot \bar{B}) \cdot (\bar{C} \cdot \bar{D} + C \cdot D) + \bar{A} \cdot \bar{C}$

t) $\overline{\overline{X}} \cdot (\overline{X} + 1)$

Soluție

a) Utilizând T6 și ulterior T4, expresia devine:

$$\overline{(X + Y)} \cdot (\overline{X} + \overline{Y}) = \overline{X} \cdot \overline{Y} \cdot (\overline{X} + \overline{Y}) = \overline{X} \cdot \overline{Y}$$

k) După aplicarea A4, expresia devine:

$$\begin{aligned} (Y \cdot Z + \overline{X} \cdot W) \cdot (X \cdot \overline{Y} + Z \cdot \overline{W}) &= \\ &= Y \cdot Z \cdot (X \cdot \overline{Y} + Z \cdot \overline{W}) + \overline{X} \cdot W \cdot (X \cdot \overline{Y} + Z \cdot \overline{W}) = \\ &= Y \cdot Z \cdot X \cdot \overline{Y} + Y \cdot Z \cdot Z \cdot \overline{W} + \overline{X} \cdot W \cdot X \cdot \overline{Y} + \overline{X} \cdot W \cdot Z \cdot \overline{W} \end{aligned}$$

După restrângerea termenilor utilizând A6:

$$0 + Y \cdot Z \cdot \overline{W} + 0 + 0 = Y \cdot Z \cdot \overline{W}$$

8. Aflați complementul expresiei $F = X + Y \cdot Z$. Arătați că $F \cdot \overline{F} = 0$ și $F + \overline{F} = 1$.

Soluție

$$\overline{F} = \overline{X + Y \cdot Z} = \overline{X} \cdot \overline{Y \cdot Z} = \overline{X} \cdot (\overline{Y} + \overline{Z}) = \overline{X} \cdot \overline{Y} + \overline{X} \cdot \overline{Z}$$

$$\begin{aligned} F \cdot \overline{F} &= (X + Y \cdot Z) \cdot (\overline{X} \cdot \overline{Y} + \overline{X} \cdot \overline{Z}) = X \cdot \overline{X} \cdot \overline{Y} + X \cdot \overline{X} \cdot \overline{Z} + Y \cdot Z \cdot \overline{X} \cdot \overline{Y} + Y \cdot Z \cdot \overline{X} \cdot \overline{Z} = \\ &= 0 + 0 + 0 + 0 = 0 \end{aligned}$$

$$\begin{aligned} F + \overline{F} &= (X + Y \cdot Z) + (\overline{X} \cdot \overline{Y} + \overline{X} \cdot \overline{Z}) = (X + \overline{X} \cdot \overline{Y}) + Y \cdot Z + \overline{X} \cdot \overline{Z} = \\ &= X + (\overline{Y} + Y \cdot Z) + \overline{X} \cdot \overline{Z} = X + \overline{Y} + (Z + \overline{X} \cdot \overline{Z}) = \\ &= X + \overline{Y} + Z + \overline{X} = (X + \overline{X}) + \overline{Y} + Z = 1 + \overline{Y} + Z = 1 \end{aligned}$$

9. Aflați expresiile complementare următoarelor expresii:

a) $X \cdot \overline{Y} + \overline{X} \cdot Y$

b) $X \cdot \overline{Y} + Z \cdot W + Q$

c) $\overline{V} \cdot W + X \cdot Y + \overline{Z}$

d) $(A + \overline{B} + C) \cdot (\overline{A} \cdot \overline{B} + C) \cdot (A + \overline{B} \cdot \overline{C})$

e) $(X + \overline{Y} + \overline{Z}) \cdot (\overline{X} + \overline{Z}) \cdot (X + Y)$

f) $A \cdot \overline{B} + \overline{A} \cdot B$

g) $W \cdot X \cdot (\overline{Y} \cdot Z + Y \cdot \overline{Z}) + \overline{W} \cdot \overline{X} \cdot (\overline{Y} + Z) \cdot (Y + \overline{Z})$

Soluție

Formele complementare ale expresiilor se obțin prin negarea acestora:

a) $\overline{X \cdot \overline{Y} + \overline{X} \cdot Y} = \overline{X \cdot \overline{Y}} \cdot \overline{\overline{X} \cdot Y} = (\overline{X} + Y) \cdot (X + \overline{Y})$

b) $\overline{X \cdot \overline{Y} + Z \cdot W + Q} = \overline{(X \cdot \overline{Y}) \cdot (Z \cdot W) \cdot Q} = (\overline{X} + Y) \cdot (\overline{Z} + \overline{W}) \cdot \overline{Q}$

e) $\overline{(X + \overline{Y} + \overline{Z}) \cdot (\overline{X} + \overline{Z}) \cdot (X + Y)} = \overline{(X + \overline{Y} + \overline{Z})} \cdot \overline{(\overline{X} + \overline{Z})} \cdot \overline{(X + Y)} = \overline{X} \cdot Y \cdot Z + X \cdot Z + \overline{X} \cdot \overline{Y}$

f) $\overline{A \cdot \overline{B} + \overline{A} \cdot B} = \overline{A \cdot \overline{B}} \cdot \overline{\overline{A} \cdot B} = \overline{A \cdot \overline{B}} \cdot \overline{\overline{A} \cdot B} = (\overline{A} + B) \cdot (A + \overline{B})$

10. Să se determine porțile logice echivalente porților:

- a) $A \cdot \overline{B}$ b) $A \cdot \overline{B} \cdot C$ c) $\overline{A \cdot B}$ d) $A + B + \overline{C}$

Soluție

O poartă logică echivalentă se obține prin negarea atât a intrărilor cât și a ieșirilor și interschimbarea operatorilor logici AND $\Leftrightarrow$ OR. Figura 2.5 prezintă porți logice echivalente.

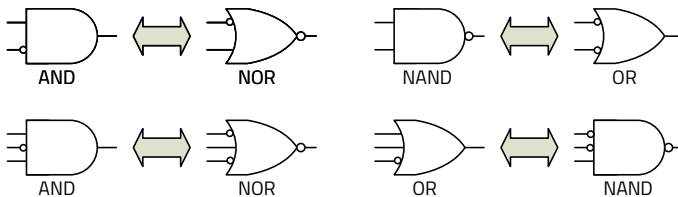


Figura 2.5 Porți logice echivalente (problema 10).

11. Să se determine structura de porți logice care realizează următoarea funcție logică:

$$Y = (A \cdot B \cdot C + D) \cdot \overline{E \cdot F} + G \cdot H \cdot (\overline{I + J} + K)$$

Modelați în Verilog circuitul. Utilizați specificații `assign` și operatorii asociați porților logice prezente în circuit.

Soluție

Structura de porți este prezentată în figura 2.6.

12. Să se determine structurile de porți logice care realizează următoarele funcții logice. Să se simplifice expresiile utilizând prelucrări analitice și să se determine structurile de porți logice simplificate.

- a) $Y_1 = (\overline{A \cdot B} + C) \cdot [\overline{(D + E)} \cdot F + \overline{G}]$
b) $Y_2 = (\overline{A \cdot B} + C) \cdot \overline{A \cdot B} + \overline{B} \cdot C$
c) $Y_3 = \overline{A} \cdot B \cdot (\overline{C \cdot D} + \overline{C} \cdot D) + A \cdot B \cdot (\overline{C} \cdot \overline{D} + \overline{C} \cdot D) + A \cdot \overline{B} \cdot \overline{C} \cdot D$
d) $Y_4 = (\overline{A \cdot B} + A \cdot \overline{B}) \cdot (C \cdot D + C \cdot \overline{D})$
e) $Y_5 = A \cdot B \cdot (C + D \cdot E \cdot F) + C \cdot E \cdot (A + B + F)$

Soluție

Structurile de porți sunt prezentate în figurile 2.7, 2.8 și 2.9.

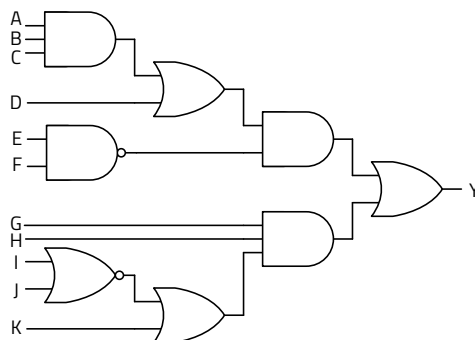


Figura 2.6 Structura de porți logice (problema 11).

13. Să se determine funcția logică a structurii de porți logice prezentate în figura 2.10.

Modelați în Verilog circuitul. Utilizați specificații `assign` și operatorii asociați porților logice prezente în circuit.

Soluție

Pornind de la intrarea rețelei spre ieșire, după fiecare poartă se determină expresia logică a ieșirii respective care este apoi propagată la intrarea următoarelor porți logice. Pentru rețeaua din figura 2.10, la ieșirile porților se deduc:

$$G_1 = \overline{A}$$

$$G_2 = \overline{D}$$

$$G_3 = G_1 \cdot B = \overline{A} \cdot B$$

$$G_4 = C \cdot D$$

$$G_5 = B \cdot C$$

$$G_6 = A \cdot C \cdot G_2 = A \cdot C \cdot \overline{D}$$

$$G_7 = G_4 + G_5 = C \cdot D + B \cdot C$$

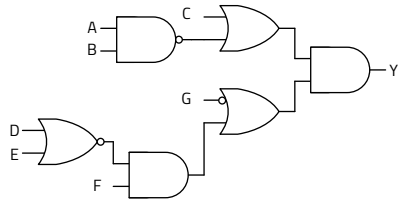
$$G_8 = G_3 \cdot G_7 = (\overline{A} \cdot B) \cdot (C \cdot D + B \cdot C)$$

$$G_9 = F = G_6 + G_8 = A \cdot C \cdot \overline{D} + \overline{A} \cdot B \cdot (C \cdot D + B \cdot C) = A \cdot C \cdot \overline{D} + \overline{A} \cdot B \cdot C \cdot D + A \cdot B \cdot C.$$

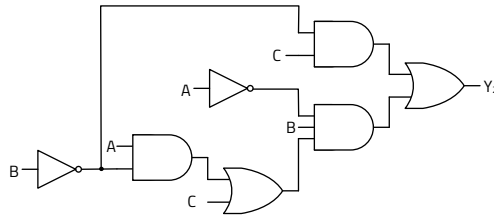
14. Determinați funcțiile logice implementate de structurile de porți logice din figura 2.11.

Soluție

a) $Y = \overline{A \cdot B \cdot C} + (E \cdot F + \overline{G}) \cdot B$



a)



b)

Figura 2.7 Structuri de porți logice (problema 12, a, b).

$$\text{b)} \quad Y = B \cdot (C \cdot \overline{D} \cdot E + F \cdot G \cdot \overline{E}) \cdot (\overline{A \cdot B} + C)$$

15. Desenați diagramele temporale asociate circuitelor cu porți logice prezentate în figura 2.12-a,b.

Soluție

Circuitele au trei intrări: A, B și C.

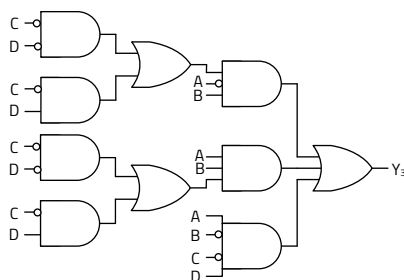
Stimulii de intrare vor avea $2^3 = 8$ valori diferite. Pentru fiecare combinație de intrare se determină valoarea logică a ieșirii porților, în funcție de tipul acestora.

$$\text{a)} \quad D = A \cdot \overline{B}, \quad E = D + \overline{C}$$

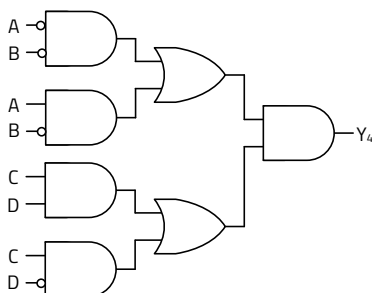
$$\text{b)} \quad D = A \cdot \overline{B}, \quad F = B \cdot C, \quad G = D + F$$

Diagramele temporale sunt prezentate în aceeași figură 2.12-a,b.

16. Determinați tipul porților logice caracterizate de diagramele temporale prezentate în figura 2.13. A, B și C sunt intrări iar D și E sunt ieșiri.



c)



d)

Figura 2.8 Structuri de porți logice (problema 12, c, d).

Soluție

Diagramele temporale prezintă formele de undă ale intrărilor A, B, C și ale ieșirilor D și E. Se observă că diagramele prezintă toate combinațiile intrărilor, $2^3 = 8$.

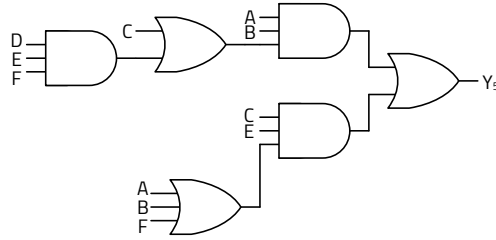
În primul caz, se remarcă faptul că ieșirea D este egală cu 1 dacă un număr impar de intrări sunt egale cu 1. Aceasta implică relația:

$$D = A \oplus B \oplus C.$$

În al doilea caz, funcția este:

$$E = \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot C.$$

Implementarea directă a funcției este prezentată în figura 2.14. Alternativa este să se realizeze tabelul de adevăr și să se completeze coloanele ieșirilor pe baza formelor de undă. Ulterior, expresia funcției se poate deduce din tabelul de adevăr.



e)

Figura 2.9 Structuri de porți logice (problema 12, e).

17. Să se implementeze funcția logică $F = A \oplus B$ cu porți:

- a) NAND cu două intrări;
- b) NOR cu două intrări.

Soluție

Pentru implementarea cu porți NAND, funcția F se rescrie:

$$F = A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B = \overline{A \cdot B} + \overline{\overline{A} \cdot \overline{B}} = \overline{(A \cdot B) \cdot (\overline{A} \cdot \overline{B})}$$

Pentru implementarea cu porți NOR, funcția F se rescrie:

$$F = A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B = \overline{\overline{A \cdot \overline{B}} + \overline{\overline{A} \cdot B}} = \overline{(\overline{A} + B) + (A + \overline{B})} = \overline{(\overline{A} + B) + (A + \overline{B})}$$

Ambale implementări sunt reprezentate în figura 2.15.

Se observă că în ambele implementări anterioare sunt necesare două porți pentru realizarea intrărilor negate. Se pot rescrie expresiile pentru a se pune în evidență termeni comuni astfel:

$$\overline{A \cdot B} = \overline{A \cdot B} + A \cdot \overline{A} = \overline{A \cdot (B + \overline{A})} = \overline{A \cdot (\overline{A} \cdot B)}$$

$$\overline{\overline{A} \cdot B} = \overline{\overline{A} \cdot B} + \overline{B} \cdot B = \overline{B \cdot (\overline{A} + \overline{B})} = \overline{B \cdot (\overline{A} \cdot B)}$$

$$\text{Deci: } F = A \oplus B = \overline{\overline{A \cdot B} \cdot \overline{\overline{A} \cdot B}} = \overline{\overline{A \cdot B} \cdot \overline{B \cdot (\overline{A} \cdot B)}}.$$

Similar:

$$\overline{A + B} = \overline{(\overline{A + B}) + B}$$

$$\overline{A + \overline{B}} = \overline{A + (\overline{A + B})}$$

$$\text{Deci: } F = A \oplus B = \overline{\overline{(\overline{A + B}) + B} + \overline{A + (\overline{A + B})}}.$$

Cele două implementări optimizate sunt prezentate în figura 2.16.

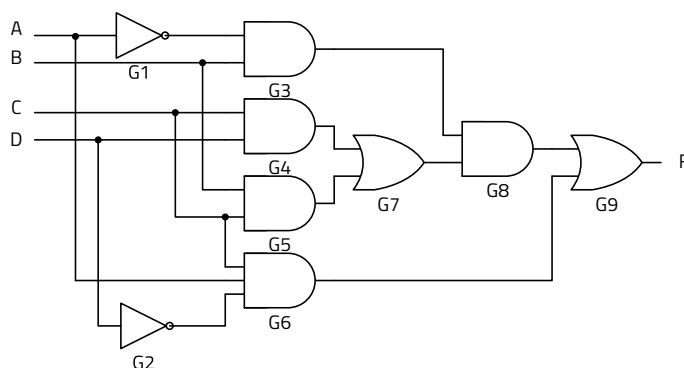


Figura 2.10 Structura de porți logice (problema 13).

18. Precizați porți logice echivalente (opțional cu intrări sau ieșiri negate) pentru circuitele prezentate în figura 2.17.
19. Precizați circuitele logice echivalente cu porți NAND pentru circuitele prezentate în figura 2.18.
20. Determinați structura de porți logice care implementează funcțiile, fără a le simplifica.

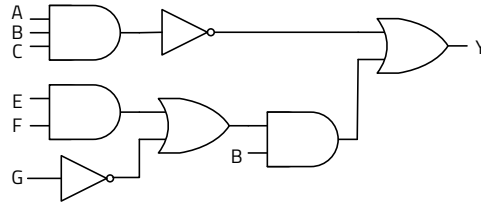
Simplificați funcțiile prin prelucrări algebrice și propuneți o structură de porți logice simplificată.

- a) $B \cdot \overline{C} + A \cdot B + A \cdot C \cdot D$
- b) $(A + B) \cdot (C + D) \cdot (\overline{A} + B + D)$
- c) $A \cdot (B \cdot \overline{C} + \overline{B} \cdot C) + C \cdot (B \cdot D + \overline{B} \cdot \overline{D})$
- d) $W \cdot \overline{Y} \cdot (X + Z) + \overline{X} \cdot Y \cdot (W + Z) + W \cdot \overline{Z} \cdot (X + Y)$
- e) $(A \cdot B + \overline{A} \cdot \overline{B}) \cdot (C \cdot \overline{D} + \overline{C} \cdot D)$
- f) $W \cdot \overline{X} \cdot \overline{Y} + \overline{W} \cdot Z + X \cdot Y$

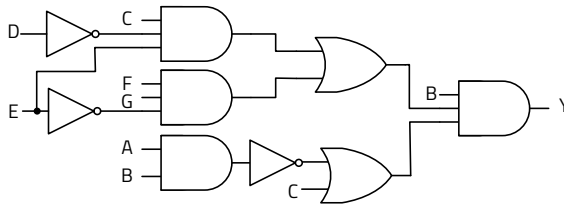
21. Un circuit logic are trei intrări și două ieșiri. O ieșire este egală cu 1 dacă toate intrările sunt egale cu 1. A doua ieșire este egală cu 0 dacă două sau mai puțin de două intrări sunt egale cu 0. Să se determine tabelul de adevăr al funcțiilor și expresiile analitice ale acestora.

Soluție

Definirea celei de-a doua funcții este similară cu: „ieșirea este egală cu 0 dacă există 0, 1 sau 2 intrări egale cu 0”, adică „ieșirea este egală



a)



b)

Figura 2.11 Structura de porți logice (problema 14).

cu 1 dacă există 3 intrări egale cu 0". Tabelul de adevăr al celor două funcții este:

A	B	C	O_1	O_2
0	0	0	0	1
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	0

Expresiile analitice ale ieșirilor sunt:

$$O_1 = A \cdot B \cdot C$$

$$O_2 = \overline{A} \cdot \overline{B} \cdot \overline{C} = \overline{A + B + C}$$

22. Determinați funcția logică cu trei intrări care este egală cu 1 doar în

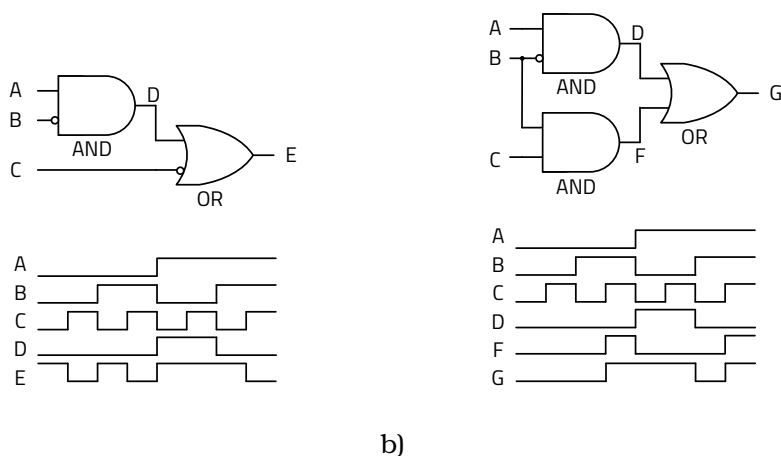


Figura 2.12 Circuite cu porți logice și diagrame temporale asociate (problema 15).



Figura 2.13 Diagrame temporale (problema 16).

cazul în care valoarea de la intrare (exprimată în binar) este mai mică decât 3. Să se determine tabelul de adevăr și expresia ieșirii.

Soluție

Valoare	A	B	C	O
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

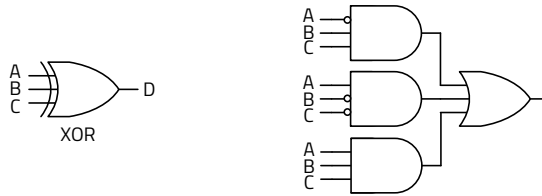


Figura 2.14 Circuite determinate pe baza diagramelor temporale prezentate în figura 2.13 (problema 16).

Expresia ieșirii este: $O = \overline{A} \cdot (\overline{B} + \overline{C})$.

23. Proiectați un circuit logic cu trei intrări și trei ieșiri. Dacă valoarea de la intrare (exprimată în binar) este 0, 1, 2 sau 3 atunci ieșirea este mai mare cu o unitate față de intrare. Dacă valoarea de la intrare este 4, 5, 6, sau 7 atunci ieșirea este mai mică cu o unitate față de intrare. Să se determine tabelul de adevăr și expresiile celor trei ieșiri.

Soluție

Valoare intrare	I_2	I_1	I_0	O_2	O_1	O_0	Valoare ieșire
0	0	0	0	0	0	1	1
1	0	0	1	0	1	0	2
2	0	1	0	0	1	1	3
3	0	1	1	1	0	0	4
4	1	0	0	0	1	1	3
5	1	0	1	1	0	0	4
6	1	1	0	1	0	1	5
7	1	1	1	1	1	0	6

Expresiile analitice ale ieșirilor sunt:

$$\begin{aligned}
 O_2 &= \overline{I_2} \cdot I_1 \cdot I_0 + I_2 \cdot \overline{I_1} \cdot I_0 + I_2 \cdot I_1 \cdot \overline{I_0} + I_2 \cdot I_1 \cdot I_0 = \\
 &= \overline{I_2} \cdot I_1 \cdot I_0 + I_2 \cdot (\overline{I_1} \cdot I_0 + I_1 \cdot \overline{I_0} + I_1 \cdot I_0) = \\
 &= \overline{I_2} \cdot I_1 \cdot I_0 + I_2 \cdot (\overline{I_1} \cdot I_0 + I_1 \cdot (\overline{I_0} + I_0)) = \\
 &= \overline{I_2} \cdot I_1 \cdot I_0 + I_2 \cdot (\overline{I_1} \cdot I_0 + I_1 \cdot 1) = \\
 &= \overline{I_2} \cdot I_1 \cdot I_0 + I_2 \cdot (\overline{I_1} \cdot I_0 + I_1) = \\
 &= \overline{I_2} \cdot I_1 \cdot I_0 + I_2 \cdot (I_0 + I_1)
 \end{aligned}$$

$O_1 = I_2 \oplus I_1 \oplus I_0$, (se observă că $O_1 = 1$ dacă un număr impar de intrări sunt egale cu 1).

$O_0 = \overline{I_0}$, (se observă coloana O_0 complementară coloanei I_0).

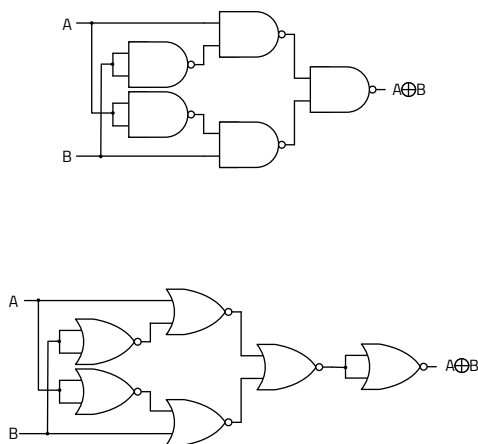


Figura 2.15 Implementarea funcției XOR cu porți NAND și NOR (problema 17).

24. Să se proiecteze un circuit logic care compară 2 biți, A și B , și determină 3 ieșiri:

$G = 1$ dacă $A > B$ (*greater*, engl.),

$E = 1$ dacă $A = B$ (*equal*, engl.),

$L = 1$ dacă $A < B$ (*less*, engl.).

Modelați în Verilog circuitul. Utilizați specificații `assign` și operatorii asociați porților logice prezente în circuit.

Soluție

Tabelul de adevăr al circuitului comparator este:

A	B	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Din tabelul de adevăr, rezultă ecuațiile circuitului comparator de 1 bit și structura de porți logice reprezentată în figura 2.19.

$$G = A \cdot \overline{B}$$

$$E = \overline{A \oplus B}$$

$$L = \overline{A} \cdot B$$

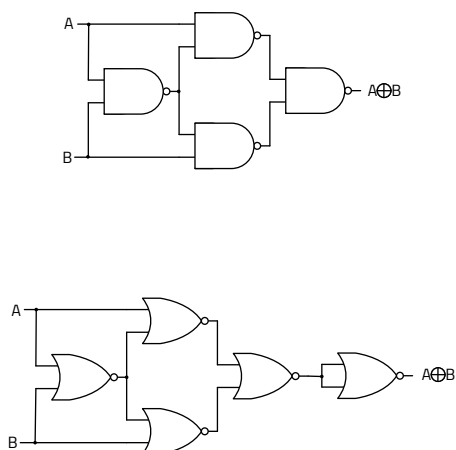


Figura 2.16 Implementarea funcției XOR cu porți NAND și NOR, optimizate (problema 17).

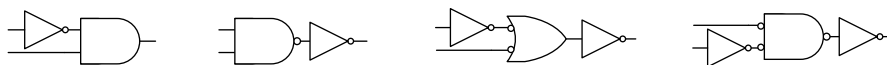


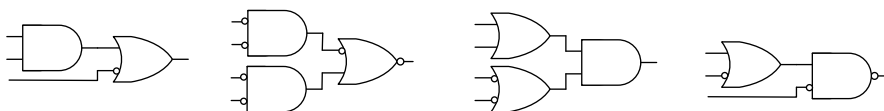
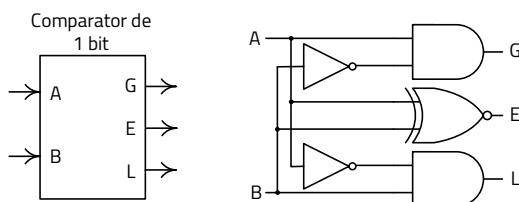
Figura 2.17 Circuite logice (problema 18).

2.4 Pentru cei ce vor să devină profesioniști

1. Determinați relația dintre numărul de variabile ale unei funcții și numărul total de funcții diferite existente. Calculați numărul total de funcții de 2, 3, 4, 5 și 6 variabile.

Soluție

O funcție cu N variabile de intrare are 2^N combinații diferite ale intrărilor. În fiecare din cele 2^N combinații, funcția poate avea două valori, 0 sau 1. Numărul de funcții distincte cu N intrări este 2^{2^N} .

**Figura 2.18** Circuite logice (problema 19).**Figura 2.19** Circuitul comparator de 1 bit: simbol bloc și structură de porți logice.

Număr de intrări	Număr de funcții distincte
1	$2^{2^1} = 2^2 = 4$
2	$2^{2^2} = 2^4 = 16$
3	$2^{2^3} = 2^8 = 256$
4	$2^{2^4} = 2^{16} = 65.536$
5	$2^{2^5} = 2^{32} = 4.294.967.296$
6	$2^{2^6} = 2^{64} = 18.446.744.073.709.551.616$

2. Determinați și justificați valoarea de adevăr a fiecărei afirmații:

- $(A \oplus B) \cdot C = (A \cdot B) \oplus (B \cdot C)$
- $Z \cdot X + \bar{Y} + W \cdot \bar{X} \cdot \bar{Y} \cdot \bar{Z} = (X + \bar{Y} + \bar{Z}) \cdot (\bar{X} + Y + Z) \cdot (Z + W + Y)$
- $\bar{X} \cdot Y + \bar{Y} \cdot Z + X \cdot \bar{Z} = X \cdot \bar{Y} + Y \cdot \bar{Z} + \bar{X} \cdot Z$
- $\bar{X} \cdot Z + Y = (X + Y + Z) \cdot (\bar{X} + Y + \bar{Z})$
- $A \cdot \bar{B} \cdot \bar{D} + A \cdot B \cdot \bar{C} + B \cdot \bar{C} \cdot \bar{D} + \bar{A} \cdot \bar{B} \cdot C =$
 $= A \cdot \bar{C} \cdot \bar{D} + B \cdot \bar{C} \cdot D + B \cdot C \cdot \bar{D} + \bar{A} \cdot \bar{B} \cdot C$

3. Utilizând axiomele și teoremele algebrei Booleene, să se demonstreze următoarele identități:

- a) $A \oplus 1 = \bar{A}$, $A \oplus 0 = A$, $A \oplus \bar{A} = 1$, $A \oplus A = 0$
 b) $A \cdot B + (A + B) \cdot C = A \cdot B + (A \oplus B) \cdot C$
 c) $A \oplus B = B \oplus A = \bar{A} \oplus \bar{B}$
 d) $\overline{A \oplus B} = \overline{A \cdot B + \bar{A} \cdot \bar{B}}$
 e) $A \oplus B = A \oplus \bar{B} = \bar{A} \oplus B$

Soluție

- a) $A \oplus 1 = A \cdot \bar{1} + \bar{A} \cdot 1 = 0 + \bar{A} = \bar{A}$
 $A \oplus 0 = A \cdot \bar{0} + \bar{A} \cdot 0 = A + 0 = A$
 $A \oplus \bar{A} = A \cdot \bar{\bar{A}} + \bar{A} \cdot \bar{A} = A \cdot A + \bar{A} \cdot \bar{A} = A + \bar{A} = 1$
 $A \oplus A = A \cdot \bar{A} + \bar{A} \cdot A = 0 + 0 = 0$

b) Expresia din partea stângă se procesează conform A4 și T4:

$A \cdot B + (A + B) \cdot C = A \cdot B + A \cdot C + B \cdot C = A \cdot B \cdot C + A \cdot B \cdot \bar{C} + A \cdot B \cdot C + A \cdot \bar{B} \cdot C + A \cdot B \cdot C + \bar{A} \cdot B \cdot C = A \cdot B \cdot C + A \cdot B \cdot \bar{C} + A \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot C$
 prin aplicarea A4 și T4 în partea dreaptă se ajunge la o expresie egală cu expresia rezultată din partea stângă:

$$A \cdot B + (A \oplus B) \cdot C = A \cdot B + A \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot C = A \cdot B \cdot C + A \cdot B \cdot \bar{C} + A \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot C$$

- c) $A \oplus B = A \cdot \bar{B} + \bar{A} \cdot B$
 $B \oplus A = B \cdot \bar{A} + \bar{B} \cdot A = A \cdot \bar{B} + \bar{A} \cdot B$
 $\bar{B} \oplus \bar{A} = \bar{B} \cdot \bar{\bar{A}} + \bar{\bar{B}} \cdot \bar{A} = \bar{B} \cdot A + B \cdot \bar{A} = A \cdot \bar{B} + \bar{A} \cdot B$

d) Expresia din partea stângă, conform A4 și T4, devine:

$$\begin{aligned} \overline{A \oplus B} &= \overline{A \cdot \bar{B} + \bar{A} \cdot B} = \overline{A \cdot \bar{B}} \cdot \overline{\bar{A} \cdot B} = (\bar{A} + B) \cdot (A + \bar{B}) = \\ &= \bar{A} \cdot A + \bar{A} \cdot \bar{B} + B \cdot A + \bar{B} \cdot B = \bar{A} \cdot \bar{B} + A \cdot B, \end{aligned}$$

egală cu expresia din partea dreaptă.

4. Demonstrați identitățile analitic, folosind axiomele și teoremele algebrei Booleene:

- a) $C \cdot D + A \cdot \bar{B} + A \cdot C + \bar{A} \cdot \bar{C} + \bar{A} \cdot B + \bar{C} \cdot \bar{D} =$
 $= (\bar{A} + \bar{B} + C + \bar{D}) \cdot (A + B + \bar{C} + D)$
 b) $(A \oplus B) \cdot C = (A \cdot C) \oplus (B \cdot C)$

5. Să se proiecteze un circuit logic care compară 2 biți, A și B , și care țină cont de rezultatul comparației unor biți de pondere superioară. Modelați în Verilog circuitul care compară cei 2 biți, considerând și rezultatele comparării biților superiori.

Soluție

În cazul propagării rezultatelor comparării de la biții mai semnificativi, circuitul are 5 intrări (intrările comparate A și B și intrările cu

rezultatele comparării biților superiori (G_i, E_i, L_i) și 3 ieșiri (rezultatele comparării biților curenți sau propagarea rezultatelor comparării biților superiori (G_o, E_o, L_o).

Tabelul de funcționare al acestui circuit este:

G_i	E_i	L_i	A	B	G_o	E_o	L_o	Observații
0	0	0	X	X	X	X	X	caz imposibil la intrare
0	0	1	X	X	0	0	1	deja s-a determinat că $A < B$
0	1	0	0	0	0	1	0	prin compararea biților superiori, s-a determinat până acum că $A = B$
0	1	0	0	1	0	0	1	
0	1	0	1	0	1	0	0	
0	1	0	1	1	0	1	0	
0	1	1	X	X	X	X	X	caz imposibil la intrare
1	0	0	X	X	1	0	0	deja s-a determinat că $A > B$
1	0	1	X	X	X	X	X	caz imposibil la intrare
1	1	0	X	X	X	X	X	
1	1	1	X	X	X	X	X	

Din tabelul de adevăr se deduc ecuațiile:

$$G_o = G_i + E_i \cdot A \cdot \overline{B} \quad A > B \text{ dacă din compararea biților mai semnificativi s-a dedus că } A > B (G_i = 1) \text{ **sau** dacă până la acest index a fost egalitate } (E_i = 1) \text{ **și** } A_i > B_i.$$

$$E_o = E_i \cdot \overline{A \oplus B} \quad A = B \text{ dacă din compararea biților mai semnificativi s-a dedus egalitatea } (E_i = 1) \text{ **și** biții curenți sunt de asemenea egali } A_i = B_i.$$

$$L_o = L_i + E_i \cdot \overline{A} \cdot B \quad A < B \text{ dacă din compararea biților mai semnificativi s-a dedus că } A < B (L_i = 1) \text{ **sau** dacă până la acest index a fost egalitate } (E_i = 1) \text{ **și** } A_i < B_i.$$

Ecuațiile de mai sus se pot deduce prin observarea cazului când o anumită ieșire este activată, la parcurgerea biților de la cel mai semnificativ (stânga) spre cel mai puțin semnificativ (dreapta).

Comparatorul de index i va realiza următoarele funcții:

- Numărul A este mai mare decât B ($G_o = 1$) dacă în urma comparării biților superiori s-a decis că $A > B$ ($G_i = 1$) **sau** dacă în urma comparării biților superiori s-a decis că $A = B$ ($E_i = 1$) **și** $A_i > B_i$.
- Numărul A este egal cu B ($E_o = 1$) dacă în urma comparării biților superiori s-a decis că $A = B$ ($E_i = 1$) **și** prin compararea biților curenți se constată că $A_i = B_i$.
- Numărul A este mai mic decât B ($L_o = 1$) dacă în urma comparării

biților superiori s-a decis că $A < B$ ($L_i = 1$) **sau** dacă în urma comparării biților superiori s-a decis că $A = B$ ($E_i = 1$) **și** $A_i < B_i$.

Figura 2.20 prezintă simbolul bloc și structura de porți logice a circuitului comparator cu propagarea rezultatelor comparării de la biții superiori.

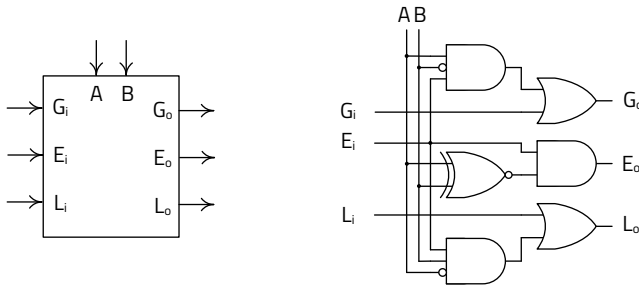


Figura 2.20 Circuitul comparator de 1 bit cu propagarea rezultatelor comparării biților superiori: simbol bloc și structură de porți logice.

Modelul Verilog se poate scrie pe baza funcțiilor exprimate anterior.

- Utilizând 4 instanțieri ale circuitului descris la problema 5, să se proiecteze un circuit comparator a două cuvinte de câte 4 biți.

Modelați în Verilog circuitul care compară două cuvinte de câte 4 biți, prin instanțierea de 4 ori a circuitului determinat la problema 5, conform circuitului prezentat în figura 2.21.

Soluție

Compararea a două cuvinte de mai mulți biți se poate face prin instanțierea mai multor celule de comparare pe bit, conectate astfel încât să propage rezultatele comparării biților mai semnificativi spre biții mai puțin semnificativi. Esența comparării este că, dacă prin compararea unor biți (cei mai semnificativi) se decide că un număr este mai mare, nu mai trebuie comparați cei mai puțin semnificativi biți deoarece nu vor schimba rezultatul final. La compararea celui mai semnificativ bit, intrările de transport se conectează astfel încât să semnifice egalitatea biților superiori ($G_i E_i L_i = 010$). Circuitul pentru compararea a două numere de câte 4 biți este prezentat în figura 2.21.

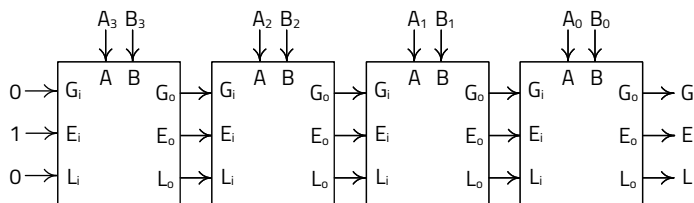


Figura 2.21 Circuitul comparator a două cuvinte de câte 4 biți, realizat cu 4 instanțe ale circuitului proiectat la problema 5.

7. Propuneți un set de valori pentru intrările circuitului prezentat în figura 2.22 astfel încât ieșirea să fie egală cu 1.

Modelați în Verilog circuitul. Utilizați specificații `assign` și operatorii asociați porților logice prezente în circuit.

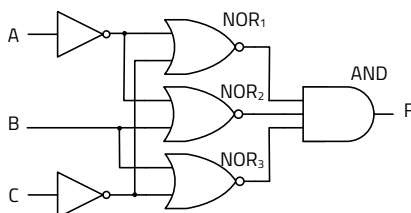


Figura 2.22 Circuitul pentru problema 7.

Soluție

Pentru ca ieșirea F să fie egală cu 1 trebuie ca toate intrările în poartă AND să fie egale cu 1.

Pentru ca toate intrările porții AND să fie 1, trebuie ca la toate intrările porților NOR să fie semnale cu valoare logică 0.

Rezultă: $A = 1$, $B = 0$ și $C = 1$.

Analitic, expresia funcției este:

$$F = \overline{A + C} \cdot \overline{A + B} \cdot \overline{B + C}$$

Din condiția $F = 1$ rezultă:

$$\overline{\overline{A} + \overline{C}} = 1 \text{ și}$$

$$\overline{\overline{A} + B} = 1 \text{ și}$$

$$\overline{B + \overline{C}} = 1.$$

$$\overline{\overline{A} + \overline{C}} = 1 \Rightarrow \overline{A} + \overline{C} = 0 \Rightarrow \overline{A} = 0 \text{ și } \overline{C} = 0, \text{ adică } A = 1 \text{ și } C = 1.$$

$$\overline{\overline{A} + B} = 1 \Rightarrow \overline{A} + B = 0 \Rightarrow \overline{A} = 0 \text{ și } B = 0, \text{ adică } A = 1 \text{ și } B = 0.$$

$$\overline{B + \overline{C}} = 1 \Rightarrow B + \overline{C} = 0 \Rightarrow B = 0 \text{ și } \overline{C} = 0, \text{ adică } B = 0 \text{ și } C = 1.$$

În final, se certifică faptul că: $A = 1$, $B = 0$ și $C = 1$.

8. Se consideră un cuvânt binar de 4 biți care codează numere întregi, pozitive, între 0 și 15.

Determinați funcția logică pentru semnalarea existenței la intrare a unui număr multiplu de 3. Desenați circuitul cu simboluri de porți logice.

Modelați în Verilog circuitul. Utilizați specificații `assign` și operatorii asociați porților logice prezente în circuit.

Aceeași cerere, în cazul în care cei 4 biți codifică numere întregi, în complement față de 2, între -7 și 7. Adăugați două ieșiri: una care semnalează existența la intrare a combinației 0000 (numărul 0) și alta care semnalează existența la intrare a combinației 1000.

Modificați circuitele și modelul Verilog, prin adăugarea unor porți logice, care să semnaleze dacă numerele de la intrare sunt multiplu de 6. Bazați-vă răspunsul pe observația că un număr este multiplu de 6 dacă este, în același timp, multiplu de 3 și multiplu de 2.

Lecția 3

Minimizare cu diagrame Veitch-Karnaugh

Lecția are ca scop fixarea cunoștințelor despre algoritmul de minimizare a funcțiilor cu **diagrame V-K** și justificarea necesității de a minimiza funcțiile înainte de implementare.

Diagramele V-K pentru funcțiile cu mai mult de 4 variabile de intrare nu (mai) au o aplicabilitate directă în industrie. Funcțiile complexe se descriu în HDL și apoi se sintetizează cu aplicații software specializate pentru sinteză de hardware.

3.1 Pentru seminar

Să se minimizeze următoarele funcții, utilizând diagrame V-K:

- a)** $F_a(A, B, C) = \sum(0, 2, 3, 4, 6)$
- b)** $F_b(A, B, C, D) = \sum(1, 3, 9, 11, 12, 13, 14, 15)$
- c)** $F_c(A, B, C) = \overline{A} \cdot \overline{C} + B \cdot \overline{C} + A \cdot B \cdot C$
- d)** $F_d(A, B, C) = \overline{A} \cdot B + \overline{B} \cdot C + \overline{A} \cdot \overline{B} \cdot \overline{C}$
- e)** $F_e(A, B, C) = \overline{A} \cdot \overline{B} + A \cdot \overline{C} + \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C}$
- f)** $F_f(A, B, C) = \sum(3, 5, 6) + d(0, 7)$
- g)** $F_g(A, B, C, D) = \sum(1, 3, 5, 7, 9, 15) + d(4, 6, 12, 13)$

3.2 Pentru cei ce vor doar să promoveze examenul

1. Să se **minimizeze** următoarele funcții de **3 intrări**, utilizând diagrame V-K:

a) $F_a(A, B, C) = \sum(3, 5, 6, 7)$

b) $F_b(A, B, C) = \sum(0, 1, 5, 7)$

2. Să se minimizeze următoarele funcții de 4 intrări, utilizând diagrame V-K:

a) $F_a(A, B, C, D) = \sum(1, 5, 9, 15)$

b) $F_b(A, B, C, D) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 14, 15)$

3.3 Pentru cei ce vor să învețe

1. Să se minimizeze următoarele funcții utilizând diagrame V-K:

a) $F_a(A, B, C) = \sum_0^7(0, 2, 3, 4, 5, 7)$

b) $F_b(A, B, C, D) = \sum_0^{15}(0, 1, 4, 5, 9, 11, 13, 15)$

Soluție

Diagramele V-K asociate funcțiilor sunt prezentate în figura 3.1.

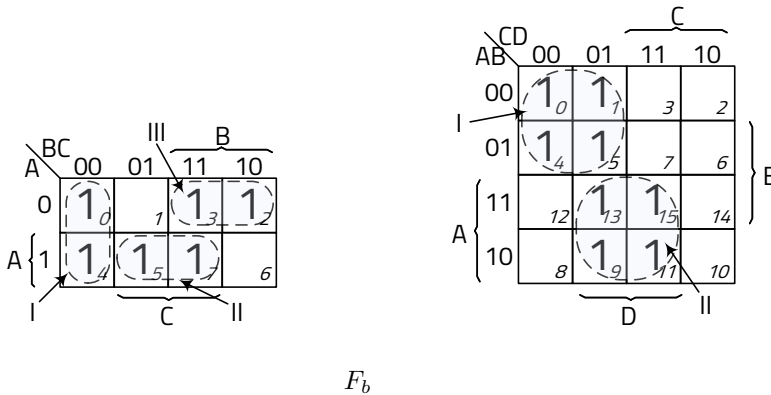


Figura 3.1 Diagrame V-K pentru problema 1.

a) $F_a(A, B, C) = \sum_0^7(0, 2, 3, 4, 5, 7) = \overline{B} \cdot \overline{C} + A \cdot C + \overline{A} \cdot B$

b) $F_b(A, B, C, D) = \sum_0^{15}(0, 1, 4, 5, 9, 11, 13, 15) = \overline{A} \cdot \overline{C} + A \cdot D$

2. Să se minimizeze următoarele funcții de 3 intrări, utilizând diagrame V-K:

a) $F_a(A, B, C) = \sum(0, 2, 6, 7)$

b) $F_b(A, B, C) = \sum(0, 1, 2, 3, 7)$

c) $F_c(A, B, C) = \sum(1, 2, 3, 6, 7)$

- d)** $F_d(A, B, C) = \sum(3, 4, 7)$
e) $F_e(A, B, C) = \sum(1, 3, 5, 6, 7)$
f) $F_f(A, B, C) = \sum(1, 3, 6, 7)$
g) $F_g(A, B, C) = \sum(3, 5, 6, 7)$
h) $F_h(A, B, C) = \sum(0, 1, 2, 4, 6)$
i) $F_i(A, B, C) = \sum(0, 3, 4, 5, 7)$

Soluție

Diagramele V-K asociate funcțiilor sunt prezentate în figura 3.2.

		B			
		00	01	11	10
A	0	1 ₀	1	3	1 ₂
	1			1 ₇	1 ₆
		C			

F_a

		B			
		00	01	11	10
A	0	1 ₀	1 ₁	1 ₃	1 ₂
	1			1	6
		C			

F_b

		B			
		00	01	11	10
A	0		1 ₁	1 ₃	2
	1			1 ₇	1 ₆
		C			

F_f

		B			
		00	01	11	10
A	0	0	1	1 ₃	2
	1		1 ₅	1 ₇	1 ₆
		C			

F_g

Figura 3.2 Diagrame V-K cu 8 căsuțe, pentru problema 2.

- a)** $F_a(A, B, C) = \sum(0, 2, 6, 7) = \overline{A} \cdot \overline{C} + A \cdot B$
b) $F_b(A, B, C) = \sum(0, 1, 2, 3, 7) = \overline{A} + B \cdot C$
f) $F_f(A, B, C) = \sum(1, 3, 6, 7) = \overline{A} \cdot C + A \cdot B$
g) $F_g(A, B, C) = \sum(3, 5, 6, 7) = B \cdot C + A \cdot C + A \cdot B$

3. Să se minimizeze următoarele funcții de 4 intrări, utilizând diagrame V-K:

- a)** $F_a(A, B, C, D) = \sum(6, 7, 8, 10, 12, 14)$
b) $F_b(A, B, C, D) = \sum(1, 3, 4, 5, 6, 9, 11, 12, 13, 14)$
c) $F_c(A, B, C, D) = \sum(1, 3, 4, 5, 7, 8, 9, 12)$

- d)** $F_d(A, B, C, D) = \sum(1, 5, 9, 12, 13, 15)$
e) $F_e(A, B, C, D) = \sum(1, 4, 5, 6, 12, 14, 15)$
f) $F_f(A, B, C, D) = \sum(0, 1, 2, 4, 5, 7, 11, 15)$
g) $F_g(A, B, C, D) = \sum(2, 3, 10, 11, 12, 13, 14, 15)$
h) $F_h(A, B, C, D) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$
i) $F_i(A, B, C, D) = \sum(0, 2, 5, 8, 9, 11, 12, 13)$
j) $F_j(A, B, C, D) = \sum(3, 4, 6, 7, 9, 12, 13, 14, 15)$
k) $F_k(A, B, C, D) = \sum(1, 5, 6, 7, 11, 12, 13, 15)$
l) $F_l(A, B, C, D) = \sum(1, 3, 4, 5, 7, 8, 9, 12)$
m) $F_m(A, B, C, D) = \sum(0, 1, 2, 5, 6, 7, 8, 9, 10, 13, 14, 15)$

Soluție

Diagramele V-K asociate funcțiilor sunt prezentate în figura 3.3.

- a)** $F_a(A, B, C, D) = \overline{A} \cdot B \cdot C + A \cdot \overline{D} + B \cdot C \cdot \overline{D} = \overline{A} \cdot B \cdot C + A \cdot \overline{D}$
b) $F_b(A, B, C, D) = \sum(1, 3, 4, 5, 6, 9, 11, 12, 13, 14) = B \cdot \overline{D} + \overline{B} \cdot D + \overline{C} \cdot D$
 sau
 $F_b(A, B, C, D) = B \cdot \overline{D} + \overline{B} \cdot D + B \cdot \overline{C}$
d) $F_d(A, B, C, D) = \sum(1, 5, 9, 12, 13, 15) = \overline{C} \cdot D + A \cdot B \cdot \overline{C} + A \cdot B \cdot D$
e) $F_e(A, B, C, D) = \sum(1, 4, 5, 6, 12, 14, 15) = B \cdot \overline{D} + \overline{A} \cdot \overline{C} \cdot D + A \cdot B \cdot C$
h) $F_h(A, B, C, D) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15) = B \cdot D + \overline{A} \cdot \overline{D} + \overline{B} \cdot \overline{D}$ sau
 $F_h(A, B, C, D) = B \cdot D + \overline{A} \cdot B + \overline{B} \cdot \overline{D}$

4. Să se simplifice expresiile utilizând diagrame V-K.

- a)** $F_a(A, B, C) = A \cdot B + \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C}$
b) $F_b(A, B, C) = \overline{A} \cdot B + B \cdot \overline{C} + \overline{B} \cdot \overline{C}$
c) $F_c(A, B, C) = \overline{A} \cdot \overline{B} + B \cdot C + \overline{A} \cdot B \cdot \overline{C}$

Soluție

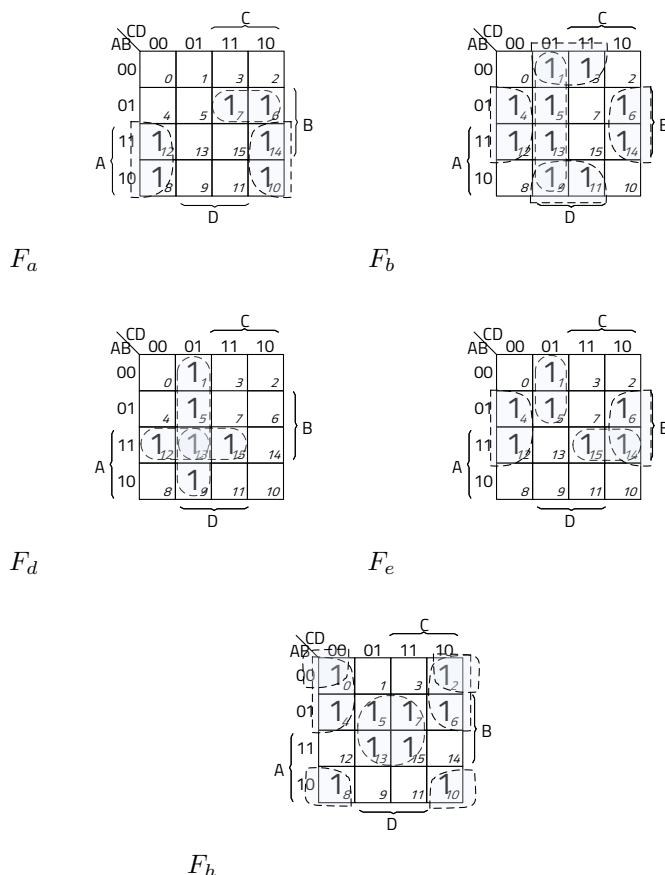
Suprafețele asociate „produselor” se plasează în diagramele V-K, ca în figura 3.4.

Apoi se minimizează funcțiile prin acoperirea căsuțelor cu 1 cu un număr minim de suprafețe, cât mai mari. Diagramele V-K asociate funcțiilor minimizezate sunt prezentate în figura 3.5.

- a)** $F_a(A, B, C) = A \cdot B + \overline{A} \cdot \overline{C}$
b) $F_b(A, B, C) = \overline{C} + \overline{A} \cdot B$
c) $F_c(A, B, C) = B \cdot C + \overline{A}$

5. Să se simplifice funcțiile incomplet definite, utilizând diagrame V-K.

- a)** $F_a(A, B, C) = \sum(0, 1, 2, 4, 5) + d(3, 6, 7)$
b) $F_b(A, B, C) = \sum(4, 6, 7) + d(2, 3, 5)$
c) $F_c(A, B, C, D) = \sum(0, 2, 4, 5, 8, 14, 15) + d(7, 10, 13)$
d) $F_d(A, B, C, D) = \sum(4, 6, 7, 8, 12, 15) + d(2, 3, 5, 10, 11, 14)$
e) $F_e(A, B, C, D) = \sum(1, 2, 4, 5, 6, 9, 10, 13, 14) + d(7)$

**Figura 3.3** Diagrame V-K cu 16 căsuțe, pentru problema 3.

- f)** $F_f(A, B, C, D) = \sum(2, 3, 4, 5, 13, 15) + d(8, 9, 10, 11)$
g) $F_g(A, B, C, D) = \sum(1, 5, 7, 9, 13, 15) + d(8, 10, 11, 14)$
h) $F_h(A, B, C, D) = \sum(0, 2, 4, 8, 10, 14) + d(5, 6, 7, 12)$
i) $F_i(A, B, C, D) = \sum(0, 2, 5, 7, 8, 10, 13) + d(1, 9, 11)$

Soluție

Funcțiile incomplet definite contin, pe lângă valorile 0 și 1, și valori indiferente. Valorile indiferente pot fi considerate atât 0 cât și 1, fără a influența funcționarea logicii. Termenii produs care sunt indiferenți, notați cu d^1 pot fi considerați în diagrama V-K fie de valoare 0, fie

¹ „don't care” = „nu contează”, engl.

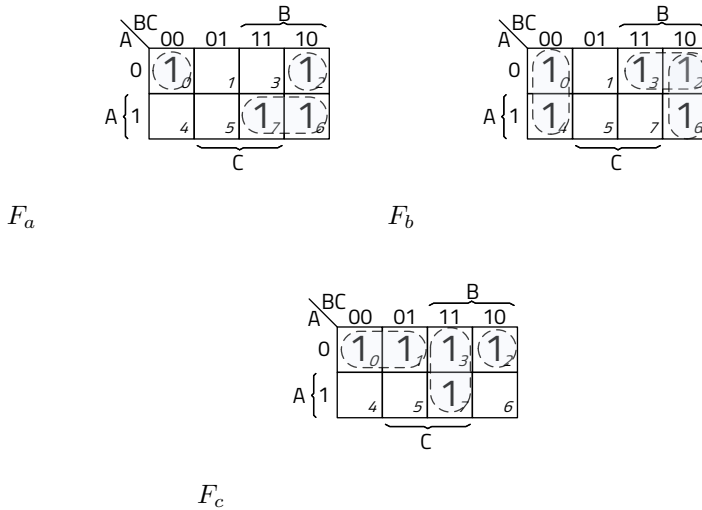


Figura 3.4 Diagrame V-K determinate pe baza expresiei funcției, problema 4.

de valoare 1. Se va alege valoarea logică ce va determina implicantii primi cu o exprimare minimă.

În diagrama V-K, aceasta înseamnă definirea unor suprafețe:

cât mai mari și

cât mai puține, pentru a acoperi căsuțele cu 1.

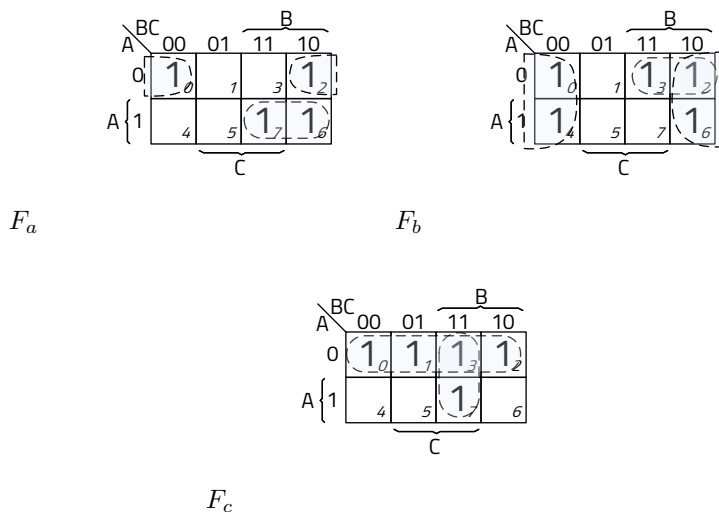
Diagramele V-K sunt prezentate în figura 3.6.

- a) $F_a(A, B, C) = 1$
- b) $F_b(A, B, C) = A$
- f) $F_f(A, B, C, D) = \overline{B} \cdot C + A \cdot D + \overline{A} \cdot B \cdot \overline{C}$
- g) $F_g(A, B, C, D) = \overline{C} \cdot D + B \cdot D$
- h) $F_h(A, B, C, D) = \overline{D}$
- i) $F_i(A, B, C, D) = \overline{C} \cdot E + C \cdot \overline{E}$

3.4 Pentru cei ce vor să devină profesioniști

- Utilizând diagrame V-K, să se minimizeze funcțiile exprimate cu variabile reziduu:

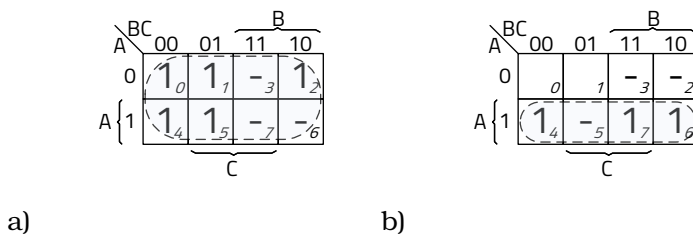
- a) $F_a(A, B, C) = \sum(0, 1 \cdot C, 2)$, unde mintermul este $m(A, B)$
- b) $F_b(A, B, C) = \sum(0 \cdot C, 3) + d(1)$, unde mintermul este $m(A, B)$
- c) $F_c(A, B, C, D) = \sum(0, 1 \cdot (C + D), 3 \cdot C)$, unde mintermul este $m(A, B)$

**Figura 3.5** Diagrame V-K pentru problema 4.

- d)** $F_d(A, B, C, D) = \sum(0 \cdot C, 2 \cdot C, 3)$, unde mintermul este $m(A, B)$
- e)** $F_e(A, B, C, D) = \sum(0 \cdot C \cdot \overline{D}, 1, 2 \cdot \overline{D}) + d(3)$, unde mintermul este $m(A, B)$
- f)** $F_f(A, B, C, D) = \sum(0, 2 \cdot D, 4 \cdot \overline{D}, 6)$, unde mintermul este $m(A, B, C)$
- g)** $F_g(A, B, C, D) = \sum(2, 4, 5, 7 \cdot D) + \sum(6)$, unde mintermul este $m(A, B, C)$

Soluție

Diagramele V-K utilizate pentru minimizare sunt prezentate în figura 3.7.

**Figura 3.6** Diagrame V-K pentru problema 5.

În diagramele funcției sunt prezentate căsuțele cu valoare 1, „-” (indiferente) sau cu variabile reziduu.

În diagramele căsuțelor 1, căsuțele cu funcții reziduu sunt considerate cu 0.

În diagramele funcțiilor reziduu, căsuțele cu 1 sunt considerate „-”, iar funcțiile reziduu sunt considerate una câte una.

Implicanții primi rezultați din minimizare se consideră în conjuncție cu variabila reziduu corespunzătoare. În final, expresia obținută mai poate fi minimizată analitic.

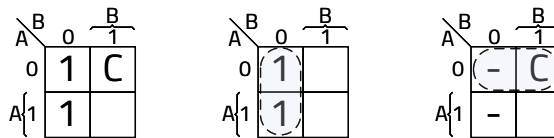
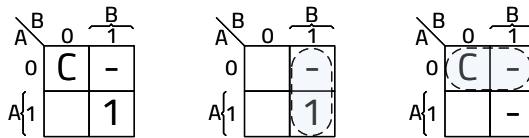
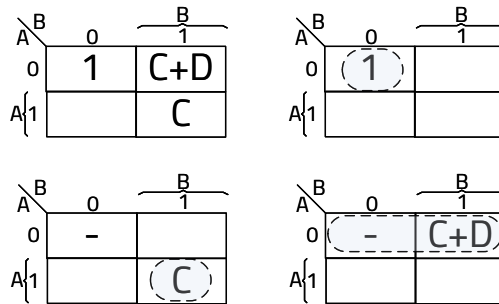
 F_a  F_b  F_c

Figura 3.7 Diagrame V-K cu variabile reziduu, asociate funcțiilor referite la problema 1. Diagrama funcției, diagrama căsuțelor 1, diagrame cu variabilă reziduu.

- a) $F_a(A, B, C) = \overline{B} + \overline{A} \cdot C$
b) $F_b(A, B, C) = B + \overline{A} \cdot C$

$$\begin{aligned}
 \text{c) } F_c(A, B, C, D) &= \overline{A} \cdot \overline{B} + \overline{A} \cdot B \cdot (C + D) + A \cdot B \cdot C = \\
 &= \overline{A} \cdot \overline{B} + \overline{A} \cdot B \cdot C + \overline{A} \cdot B \cdot D + A \cdot B \cdot C = \\
 &= \overline{A} \cdot (\overline{B} + B \cdot D) + (\overline{A} + A) \cdot B \cdot C = \overline{A} \cdot (\overline{B} + D) + B \cdot C = \overline{A} \cdot \overline{B} + \overline{A} \cdot D + B \cdot C
 \end{aligned}$$

2. Minimizați corelat funcțiile:

$$\text{a) } F_1(A, B, C) = \sum(0, 2, 4, 6, 7) \text{ și } F_2(A, B, C) = \sum(2, 6, 7)$$

$$\text{b) } F_1(A, B, C) = \sum(1, 3, 4, 7) \text{ și } F_2(A, B, C) = \sum(3, 4, 6, 7)$$

$$\text{c) } F_1(A, B, C) = \sum(1, 2, 4, 6), F_2(A, B, C) = \sum(0, 1, 2, 6, 7) \text{ și}$$

$$F_3(A, B, C) = \sum(1, 2, 6)$$

$$\text{d) } F_1(A, B, C, D) = \sum(0, 2, 7, 8, 10, 15) \text{ și } F_2(A, B, C, D) = \sum(7, 9, 11, 13, 15)$$

3. Minimizați funcțiile:

$$\text{a) } F_a(A, B, C, D, E) = \sum(3, 7, 8, 9, 11, 15, 16, 18, 19, 20, 22, 23, 27, 31)$$

$$\text{b) } F_b(A, B, C, D, E) = \sum(0, 4, 8, 9, 13, 18, 19, 20, 22, 24, 25, 29)$$

Soluție

Diagramele V-K și definirea suprafețelor necesare minimizării sunt prezentate în figura 3.8. Se obțin formele minime:

$$\text{a) } F_a(A, B, C, D, E) = I + II + III = D \cdot E + A \cdot \overline{B} \cdot \overline{E} + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D}$$

$$\text{b) } F_b(A, B, C, D, E) = I + II + III + IV + V = B \cdot \overline{C} \cdot \overline{D} + B \cdot \overline{D} \cdot E + \overline{B} \cdot C \cdot \overline{E} + \overline{A} \cdot \overline{B} \cdot \overline{D} \cdot \overline{E} + A \cdot \overline{B} \cdot \overline{C} \cdot D$$

Se observă că există opțiunea ca în loc de implicantul prim $IV = \overline{A} \cdot \overline{B} \cdot \overline{D} \cdot \overline{E}$ să se considere implicantul $\overline{A} \cdot \overline{C} \cdot \overline{D} \cdot \overline{E}$ (căsuțele (0, 8), căsuța 4 fiind deja acoperită de suprafața III).

4. Minimizați funcțiile:

$$\text{a) } F_a(A, B, C, D, E, F) = \sum(0, 9, 13, 20, 21, 22, 23, 26, 27, 29, 31, 32, 45, 47, 52, 53, 54, 55, 59, 63)$$

$$\text{b) } F_b(A, B, C, D, E, F) = \sum(0, 2, 4, 6, 16, 18, 20, 22, 25, 27, 29, 31, 32, 34, 36, 38, 41, 48, 50, 52, 54, 57, 59, 61, 63)$$

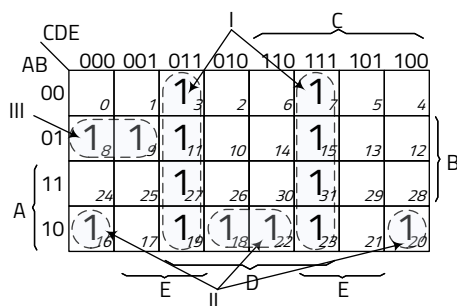
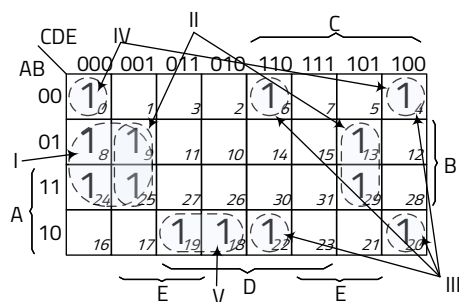
Soluție

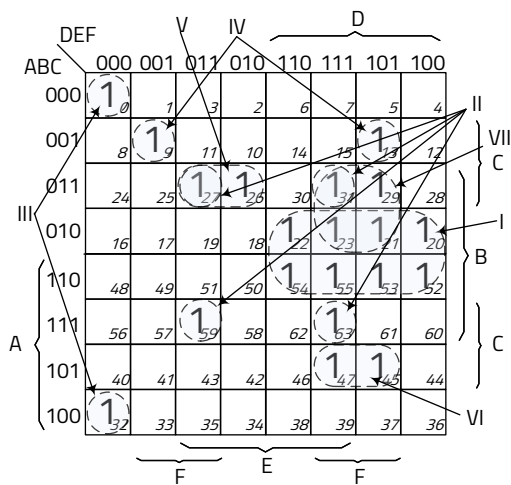
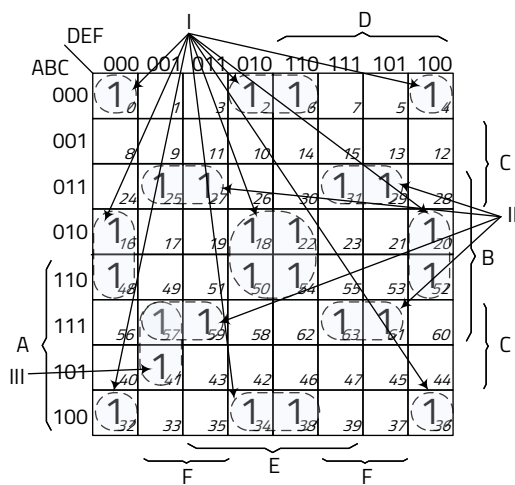
Diagramele V-K și definirea suprafețelor necesare minimizării sunt prezentate în figura 3.9. Se obțin formele minime:

$$\text{a) } F_a(A, B, C, D, E, F) = I + II + III = \overline{C} \cdot \overline{F} + B \cdot C \cdot F + A \cdot C \cdot \overline{D} \cdot \overline{E} \cdot F$$

$$\begin{aligned}
 \text{b) } F_b(A, B, C, D, E, F) &= I + II + III + IV + V + VI + VII = B \cdot \overline{C} \cdot D + \\
 &+ B \cdot C \cdot E \cdot F + B \cdot \overline{C} \cdot \overline{D} \cdot \overline{E} \cdot \overline{F} + \overline{A} \cdot \overline{B} \cdot C \cdot \overline{E} \cdot F + \\
 &+ \overline{A} \cdot B \cdot C \cdot \overline{D} \cdot E + A \cdot \overline{B} \cdot C \cdot D \cdot F + \overline{A} \cdot B \cdot D \cdot F
 \end{aligned}$$

De menționat, căsuțele (13, 29, 21, 53) nu formează o suprafață (deoarece nu are dimensiuni puteri ale lui 2, pe fiecare dimensiune, în planuri suprapuse). Un implicant prim, dar ne-esențial, este determinat de suprafața căsuțelor (31, 23, 55, 63), de dimensiune $2 \times 1 \times 2$, în două planuri, după pliere pe orizontală.

 F_a  F_b **Figura 3.8** Diagrame V-K pentru funcții cu 5 intrări (problema 3).

 F_a  F_b **Figura 3.9** Diagrame V-K pentru funcții cu 6 intrări (problema 4).

Lecția 4

Implementare CLC cu porți NAND

Orice funcție logică poate fi scrisă sub forma standard SOP („*Sum Of Products*”).¹ Forma standard SOP implică o implementare a oricărei funcții pe două nivele: AND și OR (presupunând existența variabilelor de intrare negate).

Prin utilizarea teoremei lui De Morgan, se demonstrează că structura pe nouă nivele AND-OR este echivalentă cu o structură pe două nivele cu porți NAND-NAND. **Poarta NAND este cea mai rapidă poartă în tehnologie CMOS.**² Din acest motiv, se preferă implementarea pe două nivele NAND-NAND, deoarece este mai rapidă.

Desigur, în practica industrială actuală, proiectanții nu detaliază structura până la acest nivel de poartă logică. Însă, este bine de știut că **orice circuit logic poate fi implementat pe două nivele de porți logice NAND-NAND**, dar este bine și de „simțit” complexitatea circuitului rezultat. Algoritmul de sinteză este implementat în software EDA,³ însă inginerul proiectant trebuie să evalueze rapid complexitatea din spatele unor funcții logice. Despre asta este această lecție.

4.1 Pentru seminar

1. Minimizați funcțiile logice și implementați-le cu porți NAND pe două nivele logice.

¹ „*Sum Of Products*” = „*Sumă de produse*”, engl.

Aici prin „*sumă*” trebuie înțeles „*operator OR*” iar prin „*produs*” trebuie înțeles „*operator AND*”
² „*CMOS*” = „*Complementary Metal-Oxide-Semiconductor*”, engl. Tehnologie pe bază siliciu cu tranzistoare complementare nMOS și pMOS.

³ *EDA* = *Electronic Design Automation* = suita de aplicații software utilizate pentru proiectare în electronică, engl.

a) $F_a = \sum(0, 2, 4, 7)$

b) $F_b = \sum(0, 2, 5, 8, 10, 13, 14)$

Soluție

a) Diagrama V-K asociată funcției $F_a = \sum(0, 2, 4, 7)$ este prezentată în figura 4.1.

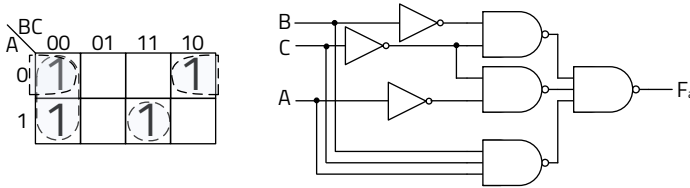


Figura 4.1 F_a , problema 1: diagrama V-K, implementare cu porți NAND.

Forma minimă a funcției este:

$$F_a = \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{C} + A \cdot B \cdot C$$

Implementarea cu porți NAND este prezentată în figura 4.1.

Costul implementării este:

2 porți $NAND \times 2$

2 porți $NAND \times 3$

3 porți $NOT \times 1$

Total: 7 porți cu un număr total de 13 intrări.

b) Diagrama V-K asociată funcției $F_b = \sum(0, 2, 5, 8, 10, 13, 14)$ este prezentată în figura 4.2-a).

Forma minimă a funcției este:

$$F_e = B \cdot \overline{C} \cdot D + A \cdot C \cdot \overline{D} + \overline{B} \cdot \overline{D}$$

Implementarea cu porți NAND este prezentată în figura 4.2-b).

Costul implementării este:

1 poartă $NAND \times 2$

3 porți $NAND \times 3$

3 porți $NOT \times 1$

Total: 7 porți cu un număr total de 14 intrări.

2. Implementați cu porți NAND următoarele funcții cu eliminarea hazardului combinațional.

a) $F_a(A, B, C, D) = \sum(0, 1, 5, 7, 8, 9, 14, 15)$

b) $F_b(A, B, C, D) = \sum(2, 3, 8, 9, 10, 11)$

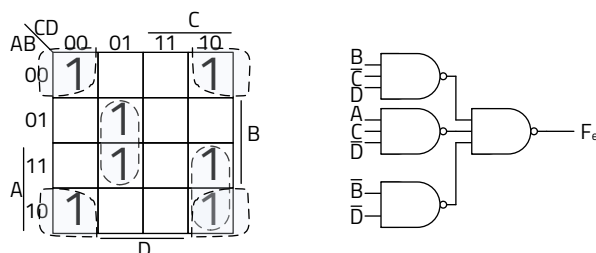


Figura 4.2 F_b , problema 1-b: diagrama V-K, implementare cu porți NAND.

4.2 Pentru cei ce vor doar să promoveze examenul

1. **Minimizați funcțiile logice** și implementați-le cu porți NAND pe două nivele logice.
 - a) $F_a = \sum(3, 5, 6, 7)$
 - b) $F_b = \sum(0, 2, 4, 6, 7)$
 - c) $F_c = \sum(0, 4, 6, 7, 8, 14, 15)$
 - d) $F_d = \sum(3, 7, 9, 10, 11, 15)$
 - e) $F_e = \sum(0, 2, 6, 8, 10, 13)$
2. **Proiectați un circuit** de „vot majoritar” cu 3 intrări și implementați-l cu porți NAND pe două nivele, după minimizarea funcției. Ieșirea circuitului are valoarea logică a majorității intrărilor.

Soluție

Tabelul de adevăr al circuitului „vot majoritar” este:

V_2	V_1	V_0	VOT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Diagrama V-K asociată funcției este prezentată în figura 4.3, alături de implementarea cu porți NAND a funcției VOT .

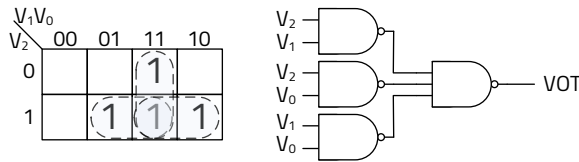


Figura 4.3 Funcția „vot majoritar”: diagrama V-K, implementare cu porți NAND.

Din tabelul de adevăr se determină diagrama V-K și se minimizează funcția obținându-se expresia:

$$VOT = V_2 \cdot V_1 + V_2 \cdot V_0 + V_1 \cdot V_0$$

Se observă că, dacă două din cele trei variabile sunt egale cu 1, există o conjuncție a două intrări egală cu 1, ceea ce determină $VOT = 1$.

4.3 Pentru cei ce vor să învețe

1. Minimizați funcțiile logice și implementați-le cu porți NAND pe două nivele logice.

a) $F_a = \sum(1, 3, 6, 7)$

b) $F_b = \sum(1, 2, 4, 7)$

c) $F_c = \sum(1, 2, 3, 4, 5)$

d) $F_d = \sum(4, 5, 6, 7, 10, 11)$

e) $F_e = \sum(2, 5, 7, 9, 11, 13, 15)$

f) $F_f = \sum(4, 6, 9, 11, 12, 14)$

g) $F_g = \sum(0, 3, 5, 6, 9, 10, 12, 15)$

Soluție

b) Pentru funcția $F_b = \sum(1, 2, 4, 7)$ diagrama V-K (figura 4.4-a) este „în formă de tablă de șah”, forma minimă fiind FCND:

$$F_b(A, B, C) = \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot C$$

Se observă că F_b poate fi implementată cu o singură poartă XOR de 3 intrări:

$$F_b(A, B, C) = A \oplus B \oplus C$$

g) Se observă că diagrama V-K asociată funcției

$$F_g = \sum(0, 3, 5, 6, 9, 10, 12, 15)$$

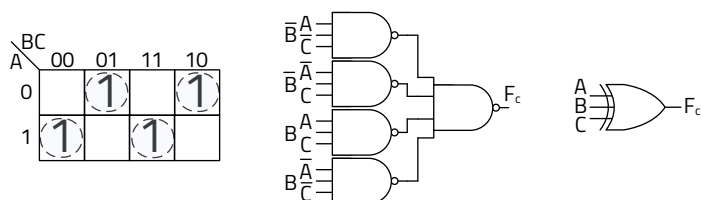


Figura 4.4 F_b , problema 1-b: diagrama V-K, implementare cu porți NAND, implementare cu porți XOR.

este „în formă de tablă de șah” și se poate implementa cu o poartă XOR sub forma:

$$F_g(A, B, C, D) = \overline{A \oplus B \oplus C \oplus D}$$

2. Minimizați funcțiile logice incomplet definite și implementați-le cu porți NAND. Evaluați „costul” implementărilor. Propuneți soluții de eliminare a hazardului combinațional și evaluați costurile suplimentare ale acestora.

- a) $F_a = \sum(1, 3, 5, 9, 14, 15) + d(4, 6, 12, 13)$
- b) $F_b = \sum(0, 3, 4, 5, 6, 7, 9, 12, 14, 15) + d(10)$
- c) $F_c = \sum(0, 1, 5, 7, 9, 10, 14) + d(3, 8)$
- d) $F_d = \sum(2, 6, 7, 9, 10) + d(0, 8)$
- e) $F_e = \sum(5, 6, 7, 8, 9, 10, 13, 14, 15) + d(0, 11)$
- f) $F_f = \sum(1, 2, 3, 5, 7, 10, 13, 14, 15) + d(6, 8)$

Soluție

- a) Diagrama V-K asociată funcției F_a este prezentată în figura 4.5. Funcția minimizată cu diagrama V-K este:

$$F_a(A, B, C, D) = \bar{C} \cdot D + A \cdot B + \bar{A} \cdot \bar{B} \cdot D$$

Pentru a obține expresia funcției doar cu porți NAND se aplică Teorema lui De Morgan:

$$F_a(A, B, C, D) = \overline{\overline{\bar{C} \cdot D + A \cdot B + \bar{A} \cdot \bar{B} \cdot D}} = \overline{\bar{C} \cdot D \cdot \bar{A} \cdot \bar{B} \cdot \bar{A} \cdot \bar{B} \cdot D}$$

Costul implementării este:

2 porți *NAND* $\times$ 2

2 porți *NAND* $\times$ 3

3 porți *NOT* $\times$ 1

Total: 7 porți cu un număr total de 13 intrări.

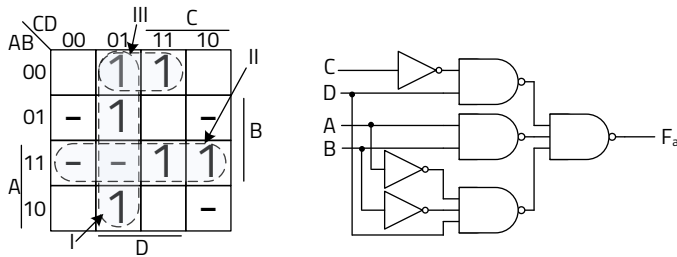


Figura 4.5 F_a , problema 2-a: diagrama V-K și implementare cu porți NAND.

Din diagrama V-K se observă că nu există suprafețe disjuncte adiacente. Deci, funcția nu are hazard combinațional.

d) Diagrama V-K asociată funcției F_d este prezentată în figura 4.6.

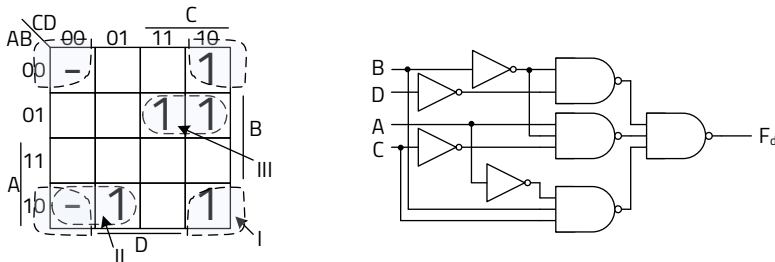


Figura 4.6 F_d , problema 2-d: diagrama V-K și implementare cu porți NAND.

Funcția minimizată cu diagrama V-K este:

$$F_d(A, B, C, D) = I + II + III = \overline{B} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot C$$

Pentru a obține expresia funcției doar cu porți NAND se aplică Teorema lui De Morgan:

$$F_d(A, B, C, D) = \overline{\overline{\overline{B} \cdot \overline{D}} + \overline{\overline{A \cdot \overline{B} \cdot \overline{C}}} + \overline{\overline{\overline{A} \cdot B \cdot C}}} = \overline{\overline{B} \cdot \overline{D} \cdot A \cdot \overline{B} \cdot \overline{C} \cdot \overline{A} \cdot B \cdot C}$$

Costul implementării este:

1 poartă $NAND \times 2$

3 porți $NAND \times 3$

4 porți $NOT \times 1$

Total: 8 porți cu un număr total de 15 intrări.

Din diagrama V-K se observă că suprafețele I și II sunt suprapuse, însă suprafețele I și III sunt disjuncte, dar adiacente. Rezultă că funcția poate prezenta hazard combinațional. Eliminarea hazardului combinațional se poate face prin adăugarea unei suprafețe IV , prezentată în figura 4.7.

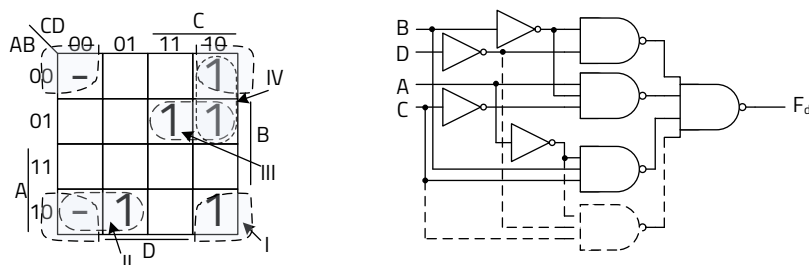


Figura 4.7 F_d , problema 2-d: diagrama V-K și implementare cu porți NAND, fără hazard combinațional.

Funcția fără hazard combinațional devine:

$$F_d(A, B, C, D) = I + II + III + IV = \overline{B} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot C + \overline{A} \cdot C \cdot \overline{D}$$

3. Implementați cu porți NAND următoarele funcții cu eliminarea hazardului combinațional.

a) $F_a(A, B, C) = \sum(1, 3, 4, 5)$

b) $F_b(A, B, C) = \sum(1, 2, 3, 5)$

c) $F_c(A, B, C) = \sum(0, 2, 3, 4, 6)$

d) $F_d(A, B, C, D) = \sum(0, 1, 2, 5, 6, 7, 15)$

Soluție

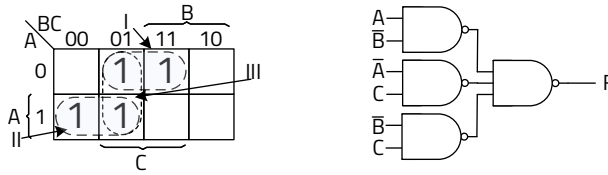
a) $F_a(A, B, C) = I + II + III = A \cdot \overline{B} + \overline{A} \cdot C + \overline{B} \cdot C = \overline{\overline{A \cdot \overline{B}} \cdot \overline{\overline{A} \cdot C} \cdot \overline{\overline{B} \cdot C}}$

4. Proiectați un circuit logic pe două nivele de porți NAND care să implementeze fără hazard combinațional funcția:

$$F(A, B, C, D) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 11, 15).$$

Soluție

Diagrama V-K asociată funcției este prezentată în figura 4.9. Funcția minimizată este:



a)

Figura 4.8 Problema 3-a: diagrama V-K și structura de porți NAND cu eliminarea hazardului combinațional.

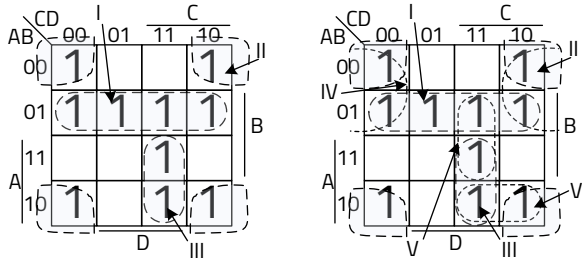


Figura 4.9 Diagrame V-K asociate funcției referite la problema 4. Minimizare fără hazard combinațional.

$$F = I + II + III = \overline{A} \cdot B + \overline{B} \cdot \overline{D} + A \cdot C \cdot D$$

Pentru eliminarea hazardului combinațional mai trebuie definite suprafețe suplimentare, conform figurii 4.9.

$$F = I + II + III + IV + V + VI = \overline{A} \cdot B + \overline{B} \cdot \overline{D} + A \cdot C \cdot D + \overline{A} \cdot \overline{D} + B \cdot C \cdot D + A \cdot \overline{B} \cdot C$$

Implementarea pe două nivele de porți NAND se face conform expresiei:

$$F = \overline{\overline{\overline{A} \cdot B} \cdot \overline{\overline{B} \cdot \overline{D}} \cdot \overline{A \cdot C \cdot D} \cdot \overline{\overline{A} \cdot \overline{D}} \cdot \overline{B \cdot C \cdot D} \cdot \overline{A \cdot \overline{B} \cdot C}} = \overline{\overline{A} \cdot \overline{B} \cdot \overline{B} \cdot \overline{D} \cdot \overline{A} \cdot \overline{C} \cdot \overline{D} \cdot \overline{\overline{A} \cdot \overline{D}} \cdot \overline{B \cdot C \cdot D} \cdot \overline{A \cdot \overline{B} \cdot C}}$$

Circuitul este prezentat în figura 4.10.

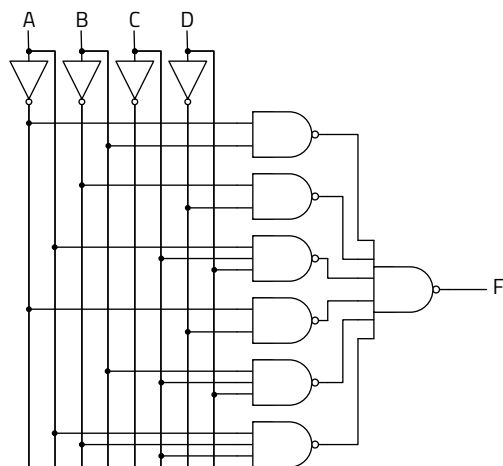


Figura 4.10 Circuit pe două nivele de porți NAND, implementare a funcției referite de problema 4.

4.4 Pentru cei ce vor să devină profesioniști

1. Proiectați un circuit de conversie a cifrelor în baza 10, reprezentate pe 4 biți BCD („*Binary Code Decimal*”, engl.), în codurile corespundente pentru comanda unui afișaj cu 7 segmente. Combinațiile binare care nu au asociat un număr în baza 10 vor determina stingerea tuturor segmentelor. Repetați problema cu toate cele 16 combinații de intrare. În cazul intrărilor 10-15, segmentelor vor afișa simbolul asociat în baza 16: A, B, C, D, E, F. Comparați costurile celor două implementări și explicați diferențele.
2. Proiectați un circuit generator de paritate de 3 biți și un circuit verificator de paritate de 4 biți, utilizând regula parității pare.

Soluție

Generatorul de paritate pară prezintă la ieșire 1 dacă la intrare sunt un număr impar de biți egali cu 1 și 0 dacă la intrare sunt un număr par de biți egali cu 1. Astfel, în cuvântul de date și bitul de paritate adăugat, vor fi întotdeauna un număr par de biți cu valoarea 1.

Această funcționalitate este realizată de o poartă logică XOR:

$$EVEN = I_2 \oplus I_1 \oplus I_0$$

Eroarea de paritate verifică paritatea datelor recepționate împreună cu bitul de paritate. Dacă între cei 4 biți este un număr par de biți cu valoarea 1, verificarea este corectă, altfel se semnalează eroare de paritate:

$$ERR_{even} = I_2 \oplus I_1 \oplus I_0 \oplus EVEN$$

Circuitele sunt prezentate în figura 4.11.



Figura 4.11 Generator și vericator de paritate pară de 3 biți.

3. Să se implementeze un circuit logic combinațional care produce conversia numerelor de la 0 la 15 exprimate în cod binar în reprezentarea acestora în **cod Gray**.

Soluție

Tabelul de adevăr al convertorului din cod binar în cod Gray este:

Cod Binar				Cod Gray			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Diagramele V-K asociate ieșirilor circuitului $G_{[3:0]}(B_{[3:0]})$ sunt prezentate în figura 4.12.

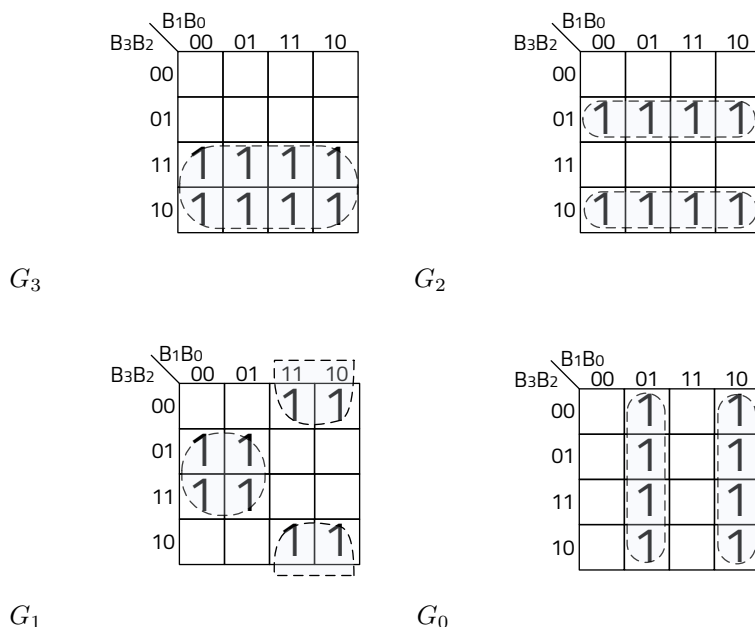


Figura 4.12 Diagramele V-K pentru convertorul binar-Gray pe 4 biți.

După minimizarea funcțiilor, se obțin ecuațiile:

$$G_3 = B_3$$

$$G_2 = B_3 \cdot \overline{B_2} + \overline{B_3} \cdot B_2 = B_3 \oplus B_2$$

$$G_1 = B_2 \cdot \overline{B_1} + \overline{B_2} \cdot B_1 = B_2 \oplus B_1$$

$$G_0 = B_1 \cdot \overline{B_0} + \overline{B_1} \cdot B_0 = B_1 \oplus B_0$$

Circuitul rezultat, implementat cu porți XOR, este prezentat în figura 4.13. Se observă faptul că se poate obține o implementare iterativă, posibil de extins pentru conversia din cod binar în cod Gray a numerelor reprezentate pe mai mulți biți. Generalizând relațiile, se pot scrie ecuațiile conversiei din binar în cod Gray, pentru N biți astfel:

$$G_{N-1} = B_{N-1}$$

$$G_i = B_{i+1} \oplus B_i, \text{ pentru } \forall i \in [(N-2), \dots, 0]$$

4. Să se implementeze un circuit logic combinațional care produce conversia din cod Gray pe 4 biți în cod binar.

Soluție

Tabelul de adevăr al convertorului din cod binar în cod Gray este prezentat la problema 3. Diagramele V-K asociate ieșirilor circuitului

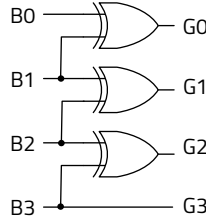


Figura 4.13 Implementarea convertorului numerelor din binar în cod Gray pe 4 biți, cu porți XOR.

$G_{[3:0]}(B_{[3:0]})$ sunt prezentate în figura 4.14.

După minimizarea funcțiilor, se obțin ecuațiile:

$$B_3 = G_3$$

$$B_2 = G_3 \cdot \overline{G_2} + \overline{G_3} \cdot G_2 = G_3 \oplus G_2 = B_3 \oplus G_2$$

$$B_1 = G_3 \cdot G_2 \cdot \overline{G_1} + G_3 \cdot \overline{G_2} \cdot G_1 + \overline{G_3} \cdot G_2 \cdot G_1 + G_3 \cdot \overline{G_2} \cdot G_1 = G_3 \oplus G_2 \oplus G_1 = B_2 \oplus G_1$$

$$B_0 = G_3 \cdot G_2 \cdot G_1 \cdot \overline{G_0} + G_3 \cdot G_2 \cdot \overline{G_1} \cdot G_0 + G_3 \cdot \overline{G_2} \cdot G_1 \cdot G_0 + \overline{G_3} \cdot G_2 \cdot G_1 \cdot G_0 + G_3 \cdot \overline{G_2} \cdot \overline{G_1} \cdot G_0 + \overline{G_3} \cdot G_2 \cdot \overline{G_1} \cdot G_0 + \overline{G_3} \cdot \overline{G_2} \cdot G_1 \cdot G_0 + \overline{G_3} \cdot \overline{G_2} \cdot \overline{G_1} \cdot G_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0 = B_1 \oplus G_0$$

Circuitul rezultat, implementat cu porți XOR, este prezentat în figura 4.15. Se observă faptul că se poate obține o implementare iterativă, posibil de extins pentru conversia din cod Gray în cod binar a numerelor reprezentate pe mai mulți biți. Generalizând relațiile, se pot scrie ecuațiile conversiei din cod Gray în cod binar, pentru N biți astfel:

$$B_{N-1} = G_{N-1}$$

$$B_i = B_{i+1} \oplus G_i, \text{ pentru } \forall i \in [(N-2), \dots, 0]$$

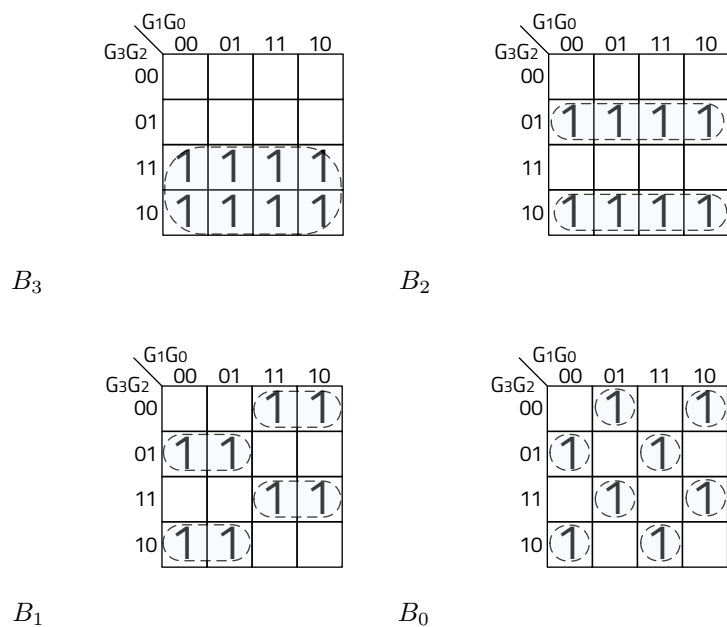


Figura 4.14 Diagramele V-K pentru convertorul din cod Gray pe 4 biți în cod binar.

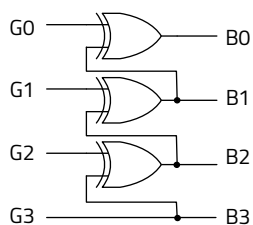


Figura 4.15 Implementarea convertorului din cod Gray pe 4 biți în binar, cu porți XOR.

Lecția 5

Multiplexoare, demultiplexoare și codificatoare

Multiplexorul este circuitul de selecție, foarte utilizat în gândirea proiectanților de hardware. Metodologia actuală nu mai necesită ca proiectații să detalieze circuitul până la nivelul implementării exclusiv cu multiplexoare. Însă, înțelegerea structurilor logice rezultate și a modalităților de a exprima comportamentul prin funcții de selecție (specificatii `if` sau `case`) reprezintă un plus pentru proiectantul de hardware digital.

5.1 Pentru seminar

1. Implementați funcțiile cu MUX 4:1.

- a)** $F_a = \sum(1, 2)$
- b)** $F_b = \sum(0, 3)$
- c)** $F_c = \sum(0, 1, 2)$

2. Implementați funcțiile cu MUX 8:1. Apoi implementați funcțiile cu MUX 4:1 și un inversor.

- a)** $F_a = \sum(1, 3, 5)$
- b)** $F_b = \sum(0, 3, 5, 6, 7)$
- c)** $F_c = \sum(0, 1, 2, 7)$

3. Să se implementeze funcțiile utilizând un decodificator 3:8.

- a)** $F_a(A, B, C) = \sum(0, 1, 7)$
- b)** $F_b(A, B, C) = \sum(0, 1, 2, 5, 7)$

4. Să se implementeze funcțiile utilizând un decodificator 4:16.

- a)** $F_a(A, B, C, D) = \sum(0, 1, 11)$
- b)** $F_b(A, B, C, D) = \sum(3, 4, 12, 14)$

5. Să se implementeze cu un singur circuit DCD 3:8 și porți logice adiționale un circuit logic combinațional cu ieșiri multiple, definit de funcțiile următoare:

$$F_a(A, B, C) = m_0 + m_3 + m_5 + m_7$$

$$F_b(A, B, C) = m_1 + m_2 + m_4 + m_5$$

$$F_c(A, B, C) = m_3 + m_4 + m_6$$

$$F_d(A, B, C) = m_1 + m_2 + m_5 + m_6 + m_7$$

6. Determinați porțile logice echivalente ale circuitelor cu multiplexor 2:1 prezentate în figura 5.1.

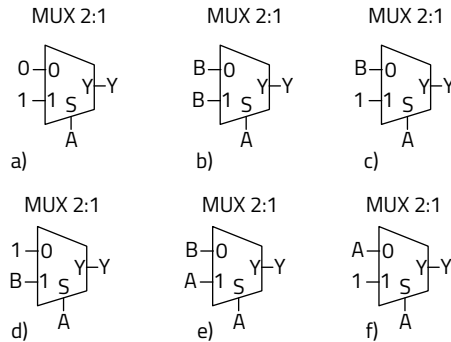


Figura 5.1 Circuite particulare implementate cu multiplexor 2:1, problema 6.

Soluție

Ecuatia implementată de MUX 2:1 este $Y = \bar{S} \cdot I_0 + S \cdot I_1$. Particularizând ecuația pentru circuitele prezentate rezultă:

a) $Y = \bar{A} \cdot 0 + A \cdot 1 = 0 + A = A$

b) $Y = \bar{A} \cdot B + A \cdot \bar{B} = (\bar{A} + A) \cdot B = 1 \cdot B = B$

c) $Y = \bar{A} \cdot B + A \cdot 1 = \bar{A} \cdot B + A = B + A = A + B$ (T4, absorbția inversă)

d) $Y = \bar{A} \cdot 1 + A \cdot B = \bar{A} + A \cdot B = \bar{A} + B$

5.2 Pentru cei ce vor doar să promoveze examenul

1. Implementați următoarele funcții conform cerințelor.

a) $F_a(A, B) = \sum(0, 3)$, cu MUX 4:1

b) $F_b(A, B, C) = \sum(0, 2, 5, 7)$, cu MUX 8:1

- c)** $F_c(A, B, C, D) = \sum(3, 4, 5, 9, 10, 12, 14, 15)$, cu MUX 16:1
d) $F_d(A, B) = \sum(0, 3)$, cu MUX 2:1 și un inversor
e) $F_e(A, B, C) = \sum(0, 2, 5, 7)$, cu MUX 4:1 și un inversor
f) $F_f(A, B, C, D) = \sum(3, 4, 5, 9, 10, 12, 14, 15)$, cu MUX 8:1 și un inversor

5.3 Pentru cei ce vor să învețe

1. Implementați funcțiile cu MUX 8:1.

- a)** $F_a = \sum(1, 3, 5) + d(0, 2)$
b) $F_b = \sum(0, 3, 4, 5, 6, 7, 14, 15) + d(10, 11)$
c) $F_c = \sum(0, 1, 5, 7, 9) + d(10, 11, 12, 13, 14, 15)$

2. Analizați circuitul prezentat în figura 5.2.

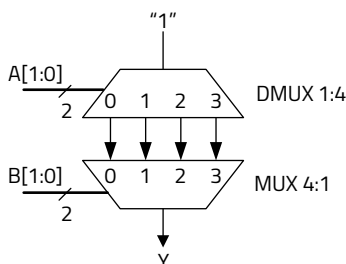


Figura 5.2 Circuitul de analizat la problema 2.

Soluție

Circuitul conține un DMUX 1:4 cu selecția provenind de la intrarea de 2 biți A , cu ieșirile conectate „unu-la-unu” la intrările unui MUX 4:1 cu selecția provenind de la intrarea de 2 biți B . Circuitul va propaga la ieșirea Y valoarea 1 aplicată pe intrarea de date/validare a demultiplexorului doar când cele două intrări sunt egale $A[1:0] = B[1:0]$. Circuitul implementează un comparator de două numere reprezentate pe câte 2 biți: ieșirea este 1 dacă $A = B$.

3. Analizați circuitul prezentat în figura 5.3.

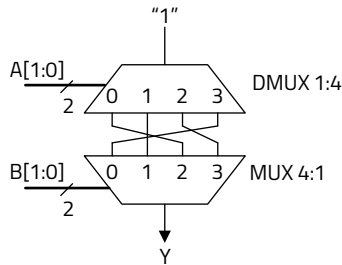


Figura 5.3 Circuitul de analizat la problema 3.

5.4 Pentru cei ce vor să devină profesioniști

1. Analizați circuitul prezentat în figura 5.4.
Propuneți un model Verilog pentru circuit.

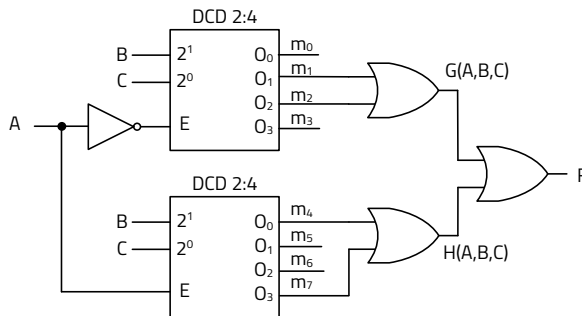


Figura 5.4 Circuit cu decodificatoare, pentru problema 1.

Soluție

Se observă că cele două decodificatoare sunt validate de către variabila A sau $\bar{A}$. Considerând funcțiile implementate de decodificatoare (mintermii variabilelor de pe intrările de selecție) și semnalele de pe intrările de validare (A și $\bar{A}$), rezultă că funcțiile logice la ieșirea decodificatoarelor sunt mintermii variabilelor (A, B, C).

Deci:

$$G(A, B, C) = \sum_{ABC}(1, 2) = \bar{A} \cdot \sum_{BC}(1, 2) \text{ și}$$

$$H(A, B, C) = \sum_{ABC}(4, 7) = A \cdot \sum_{BC}(0, 3)$$

$$F(A, B, C) = G(A, B, C) + H(A, B, C) = \sum_{ABC}(1, 2, 4, 7) = A \oplus B \oplus C$$

Pe diagrama V-K (figura 5.5) se poate observa că:

$$F(A, B, C) = A \oplus B \oplus C$$

		BC			
A	0	00	01	11	10
	1	10	11	01	00

Figura 5.5 Diagrama V-K a funcției circuitului prezentat la problema 1.

Modelul Verilog al unui circuit echivalent este descris de o specificație continuă:

```
assign F = A ^ B ^ C; // XOR
```

- Proiectați circuitul care comandă un afișaj cu 7 segmente pe baza unui cod BCD. Implementați circuitul cu un singur decodificator.

Soluție

Se conectează variabilele de intrare pe intrările de selecție ale decodificatorului. Pentru fiecare din cele 7 funcții de ieșire (asociate segmentelor) este necesară câte o poartă OR.

- Proiectați un convertor din cod BCD în cod Gray utilizând un circuit decodificator.

Soluție

Circuitul are 4 intrări și 4 ieșiri. Se utilizează un decodificator 4:16 cu care se obțin toți mintermii variabilelor de intrare în cod BCD. Cele 4 ieșiri se obțin din conectare prin porți OR a mintermilor corespunzători funcțiilor de transfer.

- Să se proiecteze un circuit codificator care determină indexul intrării activate cu prioritatea cea mai scăzută.

Propuneți un model Verilog pentru acest circuit.

Soluție

Dacă la intrările unui codificator se conectează biții în ordinea ponderilor, atunci circuitul codificator prioritar va prezenta pe ieșire indexul intrării celei mai semnificative activate (figura 5.6-a).

Dacă la intrările unui codificator se conectează biții în ordinea inversă a ponderilor, atunci circuitul codificator prioritar va prezenta pe ieșire indexul intrării celei mai puțin semnificative activate, în cod complementar. Soluția constă în conectarea biților de date în ordine inversă și negarea ieșirii, așa ca în figura 5.6-b.

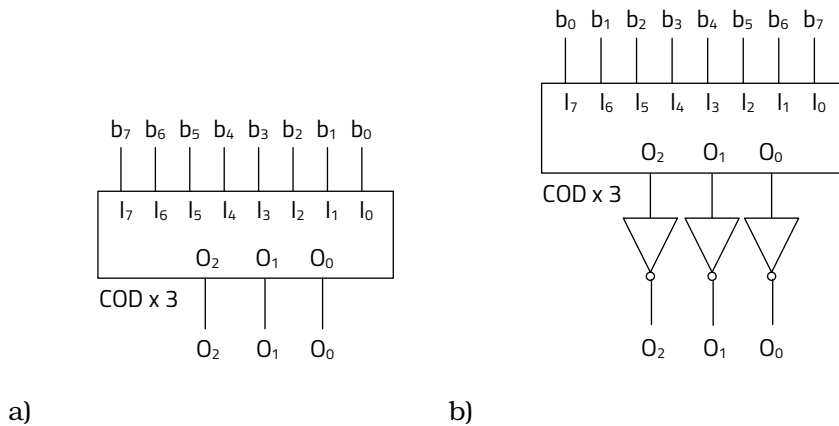


Figura 5.6 Circuite cu codificator:

- a) determinarea indexului intrării cu prioritate cea mai mare și
- b) determinarea indexului intrării cu prioritate cea mai mică.

De exemplu, dacă $b_2 = b_3 = 1$, restul biților fiind 0, intrarea cea mai prioritară a codificatorului este I_5 , ceea ce va genera pe ieșire valoarea 101, negată $010 = 2|_{10}$.

Dacă $b_3 = b_4 = b_7 = 1$, restul biților fiind 0, intrarea cea mai prioritară a codificatorului este I_4 , ceea ce va genera pe ieșire valoarea 100, negată $011 = 3|_{10}$.

Modelele Verilog ale celor două codificatoare sunt simetrice. Prioritățile intrărilor se descriu ușor cu specificații *casex*.

```
module codificator_0_7 (
input    [7:0] b      , // intrarea decodificatorului
output reg [2:0] cod7  , // b[7] prioritara
output reg [2:0] cod0  // b[0] prioritara
);
```

```

always (*)
case (b)
8'b1xxx_xxxx: cod7 = 3'd7;
8'b01xx_xxxx: cod7 = 3'd6;
8'b001x_xxxx: cod7 = 3'd5;
8'b0001_xxxx: cod7 = 3'd4;
8'b0000_1xxx: cod7 = 3'd3;
8'b0000_01xx: cod7 = 3'd2;
8'b0000_001x: cod7 = 3'd1;
8'b0000_0001: cod7 = 3'd0;
default      : cod7 = 3'd0;
endcase

always (*)
case (b)
8'bxxxx_xxx1: cod0 = 3'd0;
8'bxxxx_xx10: cod0 = 3'd1;
8'bxxxx_x100: cod0 = 3'd2;
8'bxxxx_1000: cod0 = 3'd3;
8'bxxx1_0000: cod0 = 3'd4;
8'bxxx10_0000: cod0 = 3'd5;
8'bx100_0000: cod0 = 3'd6;
8'b1000_0000: cod0 = 3'd7;
default      : cod0 = 3'd0;
endcase

endmodule // codificator_0_7

```

Se observă că modelarea funcției în HDL nu necesită găsirea unei soluții de implementare sub forma unui circuit digital. Este suficientă exprimarea algoritmică a funcției circuitului combinațional.

5. Analizați circuitul prezentat în figura 5.7. Propuneți un model Verilog pentru acesta.

Soluție

$$Y_1(A, B, C) = \overline{C} \cdot A + C \cdot \overline{A}$$

$$Y_2(A, B, C) = \overline{C} \cdot Y_1 + C \cdot B = \overline{C} \cdot (\overline{C} \cdot A + C \cdot \overline{A}) + C \cdot B = A \cdot \overline{C} + B \cdot C = A \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot C + \overline{A} \cdot B \cdot C = \sum(6, 4, 7, 3) = \sum(3, 4, 6, 7)$$

$$Y_3(A, B, C) = \overline{B} \cdot \overline{Y_1} + B \cdot C = \overline{B} \cdot (A \cdot C + \overline{A} \cdot \overline{C}) + B \cdot C = A \cdot \overline{B} \cdot C + \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot C + \overline{A} \cdot B \cdot C = \sum(5, 0, 7, 3) = \sum(0, 3, 5, 7)$$

$$F(A, B, C) = Y_3 + Y_2 = \sum(3, 4, 6, 7) + \sum(0, 3, 5, 7) = \sum(0, 3, 4, 5, 6, 7)$$

Modelul Verilog se poate scrie după minimizarea funcției la expresia:

$$F(A, B, C) = \sum(0, 3, 4, 5, 6, 7) = A + \overline{B} \cdot \overline{C} + B \cdot C$$

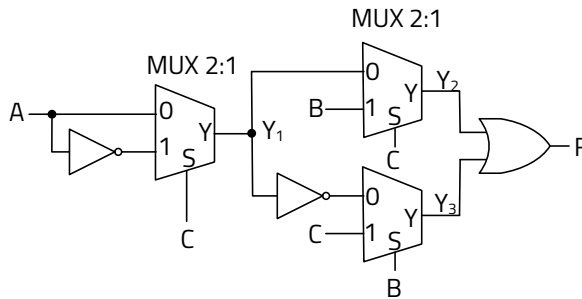


Figura 5.7 Circuitul cu multiplexoare referit la problema 5.

```
assign F = A | (~B & ~C) | (B & C) ;
assign F = A | ~(B & C) ; // model alternativ
```

6. Analizați circuitul din figura 5.8. Propuneți un circuit simplificat și un model Verilog.

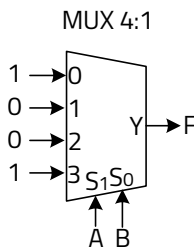


Figura 5.8 Circuitul cu multiplexor referit la problema 6.

Soluție

$$F(A, B) = m_0 + m_3 = \overline{A} \cdot \overline{B} + A \cdot B = \overline{A \oplus B}$$

Circuitul poate fi implementat cu o poartă XOR negată, modelată în Verilog astfel:

```
assign F = ~(A & B) ; // XNOR
```

7. Să se implementeze funcția:

$$F(A, B, C, D, E) = \overline{A} \cdot B \cdot E + \overline{A} \cdot \overline{B} \cdot D \cdot E + A \cdot \overline{B} \cdot C \cdot E + \\ + A \cdot C \cdot \overline{D} \cdot E + A \cdot B \cdot C \cdot D \cdot \overline{E} + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} \cdot \overline{E}$$

utilizând un circuit DMUX 1:8, un MUX 4:1 și porți logice OR/NOR.

Să se modeleze circuitul în Verilog.

Soluție

Ideea implementării constă în utilizarea a 2 variabile de intrare (D, E) conectate pe intrările de selecție ale multiplexorului 4:1 și celelalte 3 variabile de intrare (A, B, C) pe intrările de selecție ale demultiplexorului. Demultiplexorul generează toți mintermii variabilelor A, B, C. Multiplexorul implementează funcția:

$$Y = H_3 \cdot D \cdot E + H_2 \cdot D \cdot \overline{E} + H_1 \cdot \overline{D} \cdot E + H_0 \cdot \overline{D} \cdot \overline{E}$$

Funcțiile $H_{0,1,2,3}(A, B, C)$ pot fi obținute cu un singur DMUX 1:8 și câte o poartă OR/NOR pentru fiecare funcție, în total maximum 4 porți OR/NOR. Expresia funcției se prelucrează analitic pentru a fi adusă la forma canonică disjunctivă FCD (prin adăugarea variabilelor lipsă):

$$F(A, B, C, D, E) = \overline{A} \cdot B \cdot E + \overline{A} \cdot \overline{B} \cdot D \cdot E + A \cdot \overline{B} \cdot C \cdot E + A \cdot C \cdot \overline{D} \cdot E + \\ + A \cdot B \cdot C \cdot D \cdot \overline{E} + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} \cdot \overline{E} = \\ = \overline{A} \cdot B \cdot (\overline{C} + C) \cdot (\overline{D} + D) \cdot E + \overline{A} \cdot \overline{B} \cdot (\overline{C} + C) \cdot D \cdot E + A \cdot \overline{B} \cdot C \cdot (\overline{D} + D) \cdot \\ E + A \cdot (\overline{B} + B) \cdot C \cdot \overline{D} \cdot E + A \cdot B \cdot C \cdot D \cdot \overline{E} + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} \cdot \overline{E} = \\ = \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} \cdot E + \overline{A} \cdot B \cdot \overline{C} \cdot D \cdot E + \overline{A} \cdot B \cdot C \cdot \overline{D} \cdot E + \overline{A} \cdot B \cdot C \cdot D \cdot E + \\ + \overline{A} \cdot \overline{B} \cdot C \cdot D \cdot E + \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D \cdot E + \\ + A \cdot \overline{B} \cdot C \cdot \overline{D} \cdot E + A \cdot \overline{B} \cdot C \cdot D \cdot E + \\ + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} \cdot E + A \cdot \overline{B} \cdot \overline{C} \cdot D \cdot E + \\ + A \cdot B \cdot C \cdot D \cdot \overline{E} + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} \cdot \overline{E}$$

Apoi se dau în factor mintermii variabilelor D și E: $D \cdot E$, $D \cdot \overline{E}$, $\overline{D} \cdot E$, $\overline{D} \cdot \overline{E}$, rezultând:

$$F = (\overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot C + \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot C) \cdot D \cdot E + \\ + A \cdot B \cdot C \cdot D \cdot \overline{E} + \\ + (\overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot C) \cdot \overline{D} \cdot E + \\ + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} \cdot \overline{E} \\ = (m_2 + m_3 + m_1 + m_0 + m_5) \cdot D \cdot E + m_7 \cdot D \cdot \overline{E} + \\ + (m_2 + m_3 + m_5 + m_5 + m_7) \cdot \overline{D} \cdot E + m_4 \cdot \overline{D} \cdot \overline{E} \\ F = (m_0 + m_1 + m_2 + m_3 + m_5) \cdot D \cdot E + m_7 \cdot D \cdot \overline{E} + \\ + (m_2 + m_3 + m_5 + m_7) \cdot \overline{D} \cdot E + m_4 \cdot \overline{D} \cdot \overline{E}$$

Expresia finală obținută este implementată de circuitul din figura 5.9.

$$F = (\overline{m_4} + m_6 + m_7) \cdot D \cdot E + m_7 \cdot D \cdot \overline{E} + (m_2 + m_3 + m_5 + m_7) \cdot \overline{D} \cdot E + m_4 \cdot \overline{D} \cdot \overline{E} = \\ = H_3 \cdot D \cdot E + H_2 \cdot D \cdot \overline{E} + H_1 \cdot \overline{D} \cdot E + H_0 \cdot \overline{D} \cdot \overline{E}$$

În cazul H_3 s-a optat pentru complementarea funcției pentru a reduce numărul de intrări de la 5 la 3.

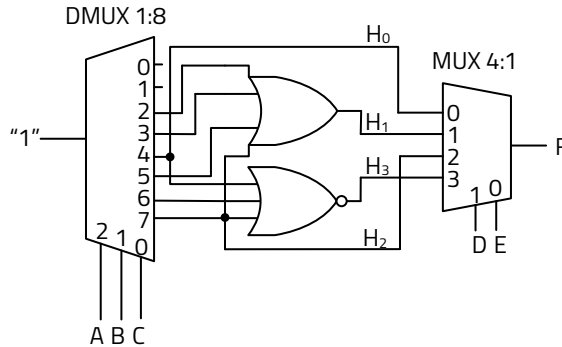


Figura 5.9 Implementarea funcției de la problema 7 cu DMUX 1:8 și MUX 4:1.

8. Să se implementeze funcția:

$$F(A, B, C, D, E) = \bar{A} \cdot B \cdot E + \bar{A} \cdot \bar{B} \cdot D \cdot E + A \cdot \bar{B} \cdot C \cdot E + A \cdot C \cdot \bar{D} \cdot E$$

utilizând un circuit DMUX 1:8, un MUX 2:1 și porți logice OR/NOR.

Să se modeleze circuitul în Verilog.

Soluție

Se observă că în expresia funcției se poate da în factor variabila E , scriindu-se funcția ca:

$$F(A, B, C, D, E) = E \cdot G(A, B, C, D)$$

Apoi, funcția $G(A, B, C, D)$ se scrie sub forma:

$$G(A, B, C, D) = H_0(A, B, C) \cdot \bar{D} + H_1(A, B, C) \cdot D$$

Funcția G se va implementa cu MUX 2:1 având pe selecție variabila D , iar funcțiile $H_{0,1}(A, B, C)$ se implementează cu DMUX 1:8 și porți OR/NOR. Circuitul DMUX va avea pe intrare variabila E , realizând $F = 0$ dacă $E = 0$, pentru orice combinație a celorlalte intrări A, B, C, D .

Prin prelucrări algebrice se obține:

$$F(A, B, C, D, E) = \bar{A} \cdot B \cdot E + \bar{A} \cdot \bar{B} \cdot D \cdot E + A \cdot \bar{B} \cdot C \cdot E + A \cdot C \cdot \bar{D} \cdot E = E \cdot (\bar{A} \cdot B + \bar{A} \cdot \bar{B} \cdot D + A \cdot \bar{B} \cdot C + A \cdot C \cdot \bar{D})$$

Deci:

$$G(A, B, C, D) = \bar{A} \cdot B + \bar{A} \cdot \bar{B} \cdot D + A \cdot \bar{B} \cdot C + A \cdot C \cdot \bar{D}$$

$$\begin{aligned}
G(A, B, C, D) &= \bar{A} \cdot B + \bar{A} \cdot \bar{B} \cdot D + A \cdot \bar{B} \cdot C + A \cdot C \cdot \bar{D} = \\
&= \bar{A} \cdot B \cdot (\bar{C} + C) \cdot (\bar{D} + D) + \bar{A} \cdot \bar{B} \cdot (\bar{C} + C) \cdot D + A \cdot \bar{B} \cdot C \cdot (\bar{D} + D) + A \cdot (\bar{B} + B) \cdot C \cdot \bar{D} = \\
&= \bar{A} \cdot B \cdot \bar{C} \cdot \bar{D} + \bar{A} \cdot B \cdot \bar{C} \cdot D + \bar{A} \cdot B \cdot C \cdot \bar{D} + \bar{A} \cdot B \cdot C \cdot D + \bar{A} \cdot \bar{B} \cdot C \cdot D + \\
&+ \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot D + A \cdot \bar{B} \cdot C \cdot \bar{D} + A \cdot \bar{B} \cdot C \cdot D + A \cdot \bar{B} \cdot C \cdot \bar{D} + A \cdot B \cdot C \cdot \bar{D}
\end{aligned}$$

Apoi se dau în factor $\bar{D}$ și D , rezultând:

$$\begin{aligned}
G(A, B, C, D) &= (\bar{A} \cdot B \cdot \bar{C} + \bar{A} \cdot B \cdot C + A \cdot \bar{B} \cdot C + A \cdot \bar{B} \cdot C + A \cdot B \cdot C) \cdot \bar{D} + \\
&+ (\bar{A} \cdot B \cdot \bar{C} + \bar{A} \cdot B \cdot C + \bar{A} \cdot \bar{B} \cdot C + \bar{A} \cdot \bar{B} \cdot \bar{C} + A \cdot \bar{B} \cdot C) \cdot D = \\
&= (m_2 + m_3 + m_5 + m_5 + m_7) \cdot \bar{D} + (m_2 + m_3 + m_1 + m_0 + m_5) \cdot D
\end{aligned}$$

Rezultă $H_0(A, B, C) = \sum(2, 3, 5, 7)$ și $H_1(A, B, C) = \sum(0, 1, 2, 3, 5)$

Expresia finală obținută este implementată de circuitul din figura 5.10.

$$F(A, B, C, D, E) = E \cdot [(m_2 + m_3 + m_5 + m_7) \cdot \bar{D} + (m_0 + m_1 + m_2 + m_3 + m_5) \cdot D]$$

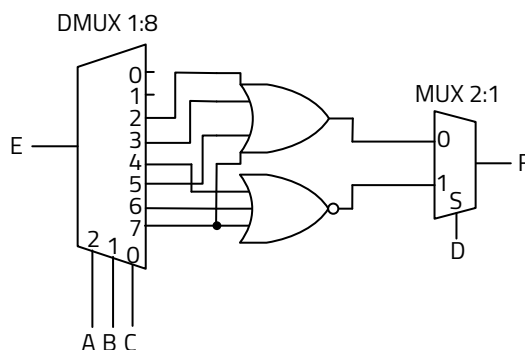


Figura 5.10 Implementarea funcției de la problema 8 cu DMUX 1:8 și MUX 2:1.

Modelul Verilog se bazează pe operatorul ternar pentru a descrie comportamentul multiplexorului 2:1 de la final și operatori logici pentru descrierea funcțiilor aplicate pe intrările de date ale acestuia.

```

wire    F ;
assign  F = E & (D ? ~( ( A & ~B & ~C ) | // D = 1, ~(m4+m6+m7)
                    ( A & B & ~C ) |
                    ( A & B & C ) :
                    ( ~A & B & ~C ) | // D = 0, m2+m3+m5+m7

```

```
(~A & B & C) |
( A & ~B & C) |
( A & B & C) ;
```

9. Să se implementeze funcția:

$$F(A, B, C, D, E) = \overline{A} \cdot B \cdot E + \overline{A} \cdot \overline{B} \cdot D \cdot E + A \cdot \overline{B} \cdot C \cdot E + A \cdot C \cdot \overline{D} \cdot E,$$

utilizând un circuit DMUX 1:4 urmat de un MUX 4:1 și porți logice OR, conectând intrările C și D pe intrările de selecție și E pe intrarea de date a demultiplexorului iar intrările A și B pe intrările de selecție ale demultiplexorului.

Să se modeleze circuitul în Verilog.

Soluție

Circuitul DMUX 1:4, cu intrarea de date activată (E), generează toți mintermii de 2 variabile (A și B).

Circuitul MUX 4:1 selectează o intrare pe baza unui cod de selecție de 2 biți (C și D).

Pentru această implementare funcția trebuie pusă sub forma:

$$F(A, B, C, D, E) = E \cdot (C \cdot D \cdot G_3(A, B) + C \cdot \overline{D} \cdot G_2(A, B) + \overline{C} \cdot D \cdot G_1(A, B) + \overline{C} \cdot \overline{D} \cdot G_0(A, B))$$

Funcțiile $G_{3,2,1,0}(A, B)$ vor fi implementate cu porți OR pe baza mintermilor generați de demultiplexor.

$$\begin{aligned} F(A, B, C, D, E) &= E \cdot (\overline{A} \cdot B + \overline{A} \cdot \overline{B} \cdot D + A \cdot \overline{B} \cdot C + A \cdot C \cdot \overline{D}) = E \cdot [m_1 + m_0 \cdot D + \\ &m_2 \cdot C + (m_2 + m_3) \cdot C \cdot \overline{D}], \text{ unde } m_i(A, B). \quad F(A, B, C, D, E) = E \cdot [m_1 \cdot (C \cdot D + \\ &C \cdot \overline{D} + \overline{C} \cdot D + \overline{C} \cdot \overline{D}) + m_0 \cdot (C \cdot D + \overline{C} \cdot D) + m_2 \cdot (C \cdot \overline{D} + C \cdot D) + (m_2 + m_3) \cdot C \cdot \overline{D}] = \\ &= E \cdot [(m_0 + m_1 + m_2) \cdot C \cdot D + (m_1 + m_2 + m_3) \cdot C \cdot \overline{D} + (m_0 + m_1) \cdot \overline{C} \cdot D + m_1 \cdot \overline{C} \cdot \overline{D}] \end{aligned}$$

Implementarea este prezentată în figura 5.11.

Modelul Verilog se bazează pe specificația `case` pentru a descrie comportamentul multiplexorului 4:1 de la final și operatori logici pentru descrierea funcțiilor aplicate pe intrările de date ale acestuia.

Se observă faptul că $F = 1$, doar dacă $E = 1$. Deci, intrarea E condiționează ieșirea (printr-o poartă AND).

Specificația `case` se modelează pe baza expresiei:

$$F(A, B, C, D, E) = E \cdot [m_1 \cdot \overline{C} \cdot \overline{D} + (m_0 + m_1) \cdot \overline{C} \cdot D + (m_1 + m_2 + m_3) \cdot C \cdot \overline{D} + (m_0 + m_1 + m_2) \cdot C \cdot D]$$

```
reg      F ;
```

```
always (*)
case ({C,D})
```

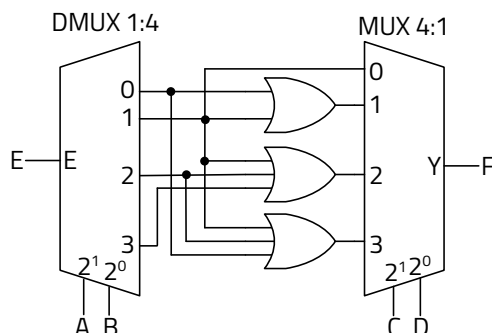


Figura 5.11 Implementarea funcției de la problema 9 cu DMUX și MUX.

```

2'b00:    F = E & ( ~A & B ); // m1
2'b01:    F = E & ( ~A & ~B | // m0
           ~A & B ); // m1
2'b10:    F = E & ( ~A & B | // m1
           A & ~B | // m2
           A & B ); // m3
default:  F = E & ( ~A & ~B | // m0
           ~A & B | // m1
           A & ~B ); // m2

endcase

```

O alternativă este constituită de modelarea cu specificație assign și operatori logici a expresiei inițiale a funcției:

$$F(A, B, C, D, E) = \overline{A} \cdot B \cdot E + \overline{A} \cdot \overline{B} \cdot D \cdot E + A \cdot \overline{B} \cdot C \cdot E + A \cdot C \cdot \overline{D} \cdot E$$

10. Figura 5.12 prezintă un multiplexor 4:1 ale cărui intrări de selecție sunt decodificate (se presupune că o singură intrare e_3, e_2, e_1, e_0 este activată la un moment dat). Tabelul de adevăr care caracterizează acest multiplexor este prezentat în continuare:

e_3	e_2	e_1	e_0	Y
1	0	0	0	d_3
0	1	0	0	d_2
0	0	1	0	d_1
0	0	0	1	d_0

- a) Să se scrie funcția logică a ieșirii.
b) Să se implementeze circuitul cu porți logice.

- c) Care este valoarea ieșirii în cazul în care nicio intrare de selecție nu este activată ($e_3 = e_2 = e_1 = e_0 = 0$)?
- d) Care este valoarea ieșirii în cazul în care două sau mai multe intrări de selecție sunt activate simultan ($e_3 = e_2 = 1$, $e_1 = e_0 = 0$ sau $e_3 = e_2 = e_1 = e_0 = 1$)?
- e) Să se modeleze circuitul în Verilog.

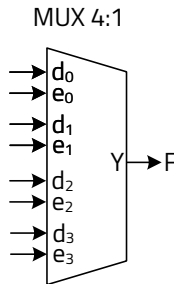


Figura 5.12 Multiplexor 4:1 cu selecția decodificată, problema 10.

Soluție

a) Se observă că tabelul de adevăr este incomplet definit: din cele $2^4 = 16$ combinații ale intrărilor e_3, e_2, e_1, e_0 sunt prezente doar 4. Presupunerea este că nu se activează două intrări simultan. Cu această presupunere, expresia logică a funcției se poate scrie:

$$Y(e_3, e_2, e_1, e_0, d_3, d_2, d_1, d_0) = e_3 \cdot d_3 + e_2 \cdot d_2 + e_1 \cdot d_1 + e_0 \cdot d_0$$

b) Circuitul este prezentat în figura 5.13.

c) Conform ecuației logice a ieșirii, în cazul în care nicio intrare de validare nu este activată, ieșirea va avea valoarea 0.

$$Y(0, 0, 0, 0, d_3, d_2, d_1, d_0) = 0 \cdot d_3 + 0 \cdot d_2 + 0 \cdot d_1 + 0 \cdot d_0 = 0$$

d) Conform ecuației logice a ieșirii, în cazul în care mai mult de o intrare de validare este activată, ieșirea realizează funcția OR a intrărilor de date corespunzătoare validate.

$$Y(1, 1, 0, 0, d_3, d_2, d_1, d_0) = d_3 + d_2$$

$$Y(1, 1, 1, 1, d_3, d_2, d_1, d_0) = d_3 + d_2 + d_1 + d_0$$

e) Modelul Verilog poate fi descris cu o specificație continuă *assign*, dacă variabilele se declară de tip vector:

```
wire [3:0] d ;    // d[3]...d[0]
wire [3:0] e ;    // e[3]...e[0]
```

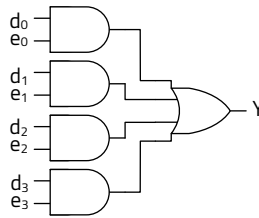


Figura 5.13 Structură internă a multiplexorului 4:1 cu selecția decodificată, problema 10.

```

wire          f ;

assign f = |(d&e); // f = d[3]&e[3] +
                  //      d[2]&e[2] +
                  //      d[1]&e[1] +
                  //      d[0]&e[0]

```

Se observă că funcția logică AND se aplică pe vector (d&e), bit cu bit și produce ca rezultat 4 biți. Cei 4 biți se prezintă la intrarea unei porți OR modelate cu operatorul de reducere (OR |) care produce rezultatul f pe un bit.

Expresia este independentă de lățimea datelor.

Lecția 6

Modelarea circuitelor logice combinaționale

Proiectarea circuitelor logice combinaționale complexe, cu multe intrări, nu urmează un anumit algoritm. Teoretic, se poate imagina că orice funcție logică poate fi reprezentată într-o formă standard. Însă, pentru un număr mare de intrări, toate formele de reprezentare (expresie analitică, diagramă V-K, tabel de adevăr, forme canonice) sunt imposibil de gestionat mental.

Varianta unei proiectări pe baza comportamentului cunoscut al unor circuite combinaționale elementare: codificator/decodificator, comparator, multiplexor/demultiplexor, sumator/scăzător, porți logice este mai accesibilă. Însă, pentru sisteme de o mare complexitate, metodologia de proiectare recomandată este cea pe baza modelării HDL. **Very-logic.**

În practică, modelarea Verilog, urmată de sinteza de hardware realizată cu aplicații specializate, permite proiectaților concentrarea pe funcționalitatea sistemului, fără a fi necesară detalierea proiectului până la nivel de circuit implementat cu porți logice. Pentru asta este nevoie de cunoașterea comportamentului circuitului proiectat și a sintaxei limbajului de descriere hardware.

6.1 Pentru seminar

1. Proiectați un circuit logic combinațional de înmulțire cu 5 a numărului pozitiv primit la intrare pe 8 biți. Utilizați numai un singur sumator, nu multiplicator. Modelați circuitul în Verilog cu ajutorul unui singur operator de adunare $+$. Nu folosiți operatorul de înmulțire $*$.

Soluție

Soluția se bazează pe descompunerea înmulțitorului într-o sumă de puteri ale lui 2. Ca efect, înmulțirea se transformă într-o sumă de

înmulțiri cu puteri ale lui 2. Iar înmulțirea cu o putere a lui 2 se realizează cu o deplasare de biți, fără altă logică adițională.

$$5 = 101_2 = 2^2 + 2^0$$

$$\text{Deci, } A \cdot 5 = A \cdot (2^2 + 2^0) = A \cdot 2^2 + A \cdot 2^0 = A \cdot 4 + A = (A \ll 2) + A$$

Operația $A \ll 2$ reprezintă o deplasare stânga cu 2 biți a intrării și nu are costuri de logică.

Modelul Verilog se bazează pe formula menționată, descrisă cu aceeași sintaxă, asociată cu o atribuire continuă *assign* pentru modelarea circuitelor logice combinaționale.

```
wire [8 -1:0] A ;
wire [8+3 -1:0] Ax5; // 3 biți suplimentari (5 < 2^3)

assign Ax5 = (A<<2) + A; // A*4+A
```

2. Proiectați circuitul de extindere de semn a numerelor întregi reprezentate pe 8 biți la o reprezentare pe 16 biți.

Propuneți un model Verilog pentru funcția precizată. Parametrizați numărul de biți la intrare și la ieșire.

Soluție

Extinderea de semn presupune copierea celui mai semnificativ bit (bitul de semn) pe toate pozițiile superioare.

Modelul Verilog este descris prin concatenarea a 8 biți cu aceeași valoare ca bitul de semn (bitul 7 din reprezentarea pe 8 biți), pe pozițiile superioare ale valorii de 16 biți (pozițiile de la 8 la 15).

```
wire [ 8 -1:0] value_8bit ;
wire [16 -1:0] value_16bit ;

assign value_16bit = {8{value_8bit[7]},value_8bit};
```

Modelul Verilog parametrizabil prezintă același comportament ca și extinderea de la 8 la 16 biți, însă cu dimensiuni parametrizabile.

```
parameter WIDTH_SHORT = 8 ;
parameter WIDTH_LONG = 16 ;

wire [WIDTH_SHORT -1:0] value_short ;
wire [WIDTH_LONG -1:0] value_long ;

assign value_long =
    { (WIDTH_LONG-WIDTH_SHORT){value_short[WIDTH_SHORT -1]},
      value_short};
```

3. Proiectați un circuit sumator/scăzător de 4 biți. Selecția funcției se face cu o intrare suplimentară OP . Propuneți un model Verilog pentru circuitului sumator/scăzător.

Soluție

Implementarea pornește de la observația că diferența numerelor reprezentate în complement față de 2 se poate scrie ca:

$$A - B = A + (-B) = A + \overline{B} + 1,$$

unde s-a notat cu $+$ operatorul de adunare.

Rezultă că cele două operații de adunare și de scădere pot fi implementate cu același circuit conform expresiilor:

$$A + B = A + B + 0 = A + (B \oplus OP) + OP, \text{ (pentru } OP = 0, \text{ sumă) și}$$

$$A - B = A + \overline{B} + 1 = A + (B \oplus OP) + OP, \text{ (pentru } OP = 1, \text{ diferență).}$$

Pentru implementare s-a folosit observația că poarta XOR realizează o „negare comandată” ($B \oplus OP = B$ dacă $OP = 0$ și $B \oplus OP = \overline{B}$ dacă $OP = 1$). Circuitul propus este prezentat în figura 6.1. Bitul de selecție a operației OP trebuie extins pentru fiecare bit al operandului B .

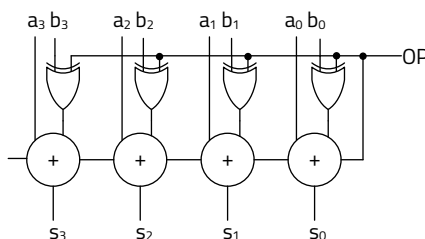


Figura 6.1 Circuit de adunare/scădere pe 4 biți implementat cu 4 sumatoare complete de 1 bit.

Modelul Verilog pentru circuitul cu porți logice prezentat în figura 6.1 este următorul. Se observă un singur operator de adunare pe 4 biți și un sumator cu un bit la final. Toate porțile XOR se modelează cu specificația $(b \wedge \{4\{OP\}\})$ care descrie o poartă XOR pe 4 biți (adică 4 porți XOR de câte două intrări) cu una din intrări provenite de la vectorul $b[3:0]$ și cealaltă intrare de la extinderea la 4 biți a intrării OP .

```
wire [4 -1:0] a ;
wire [4 -1:0] b ;
```

```

wire [4 -1:0] s ;
wire OP ; // 0 = adunare (a+b), 1 = scadere (a-b)

assign s = a + (b^{4{OP}} + OP; // b XOR "OP extins pe 4 biti"

```

Se remarcă faptul că circuitul proiectat nu tratează cazurile de depășire (4 biți însumați cu 4 biți dau ca rezultat un număr reprezentat pe 5 biți). Dar există bitul de transport de la cel mai semnificativ sumator...

4. Proiectați circuite de generare și verificare de paritate pentru un bus de date de 8 biți.

Soluție

Generarea de paritate (pară sau impară) constă în determinarea unui bit suplimentar, pe baza biților datelor, astfel încât numărul total de biți (biți de date și bit de paritate) să aibă paritatea impusă. De exemplu, în cazul parității *pare* suma biților de date plus bitul de paritate trebuie să fie un număr par. Similar, în cazul parității *impare*, suma biților de date plus bitul de paritate trebuie să fie un număr impar.

Verificarea parității constă în calcularea sumei tuturor biților plus bitul de paritate și verificarea îndeplinirii parității așteptate. În cazul apariției unei erori asupra biților recepționați, verificarea de paritate nu va mai fi satisfăcută și se va semnala o *eroare de paritate*. Paritatea este un mod de protecție limitată împotriva erorilor în mediul de transmisiune. Protecția este activă doar în cazul apariției unei singure erori. De exemplu, în cazul existenței a două erori, verificarea de paritate produce un rezultat *fals pozitiv*, semnalând în mod eronat că datele sunt corecte. Schema bloc a circuitelor de generare și verificare de paritate este prezentată în figura 6.2.

În cazul parității *pare*, 8 biți de date, ecuațiile ieșirilor circuitelor combinaționale sunt următoarele:

Generare de paritate:

$P_{par} = 0$, dacă suma $D_7 + D_6 + D_5 + D_4 + D_3 + D_2 + D_1 + D_0$ este un număr par

$P_{par} = 1$, dacă suma $D_7 + D_6 + D_5 + D_4 + D_3 + D_2 + D_1 + D_0$ este un număr impar

Verificare de paritate:

$ERR_{par} = 0$, dacă suma $D_7 + D_6 + D_5 + D_4 + D_3 + D_2 + D_1 + D_0 + P_{par}$ este un număr par

$ERR_{par} = 1$, dacă suma $D_7 + D_6 + D_5 + D_4 + D_3 + D_2 + D_1 + D_0 + P_{par}$ este un număr impar

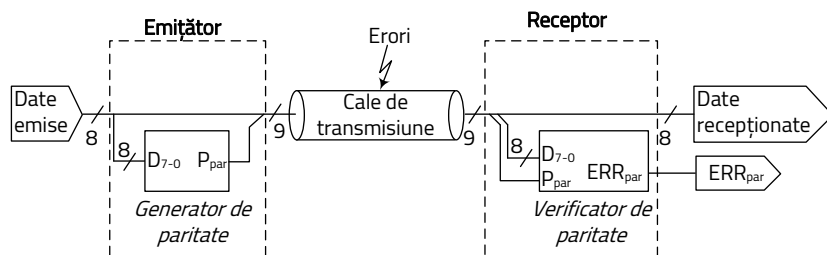


Figura 6.2 Schema bloc a circuitelor de generare și verificare de paritate, referite la problema 4.

Ținând cont că poarta logică XOR poate fi interpretată ca un circuit de generare a parității impare (produce la ieșire 1 în cazul că un număr impar de intrări sunt egale cu 1), se deduc următoarele ecuații logice:

$$P_{par} = D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0$$

$$ERR_{par} = D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0 \oplus P_{par}$$

Similar,

$$P_{impar} = \overline{D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0}$$

$$ERR_{impar} = \overline{D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0 \oplus P_{impar}}$$

Modelarea în Verilog a circuitelor de generare și verificare de paritate este extrem de simplă: modele de porți XOR cu operatori de reducere. La operatorii de reducere numărul de biți ai vectorului este indiferent.

```
wire [8-1:0] D;
```

```
assign P_par      = ^D      ;
```

```
assign ERR_par    = ^D ^ P_par ;
```

```
assign P_impar    = ~^D     ;
```

```
assign ERR_impar  = ~(^D ^ P_impar) ;
```

La această simplitate, nu este convenabil nici măcar să se încapsuleze specificația într-un modul. **Very-logic.**

6.2 Pentru cei ce vor doar să promoveze examenul

1. **Proiectați un circuit logic combinațional** de înmulțire cu 9 a numărului pozitiv primit la intrare pe 8 biți. Utilizați numai un singur suma-

tor, nu multiplicator.

Modelați circuitul în Verilog cu ajutorul unui singur operator de adunare $+$. Nu folosiți operatorul de înmulțire $*$.

Soluție

$$9|_{10} = 1001|_2 = 2^3 + 2^0$$

$$\text{Deci, } A \cdot 9 = A \cdot (2^3 + 2^0) = A \cdot 2^3 + A \cdot 2^0 = A \cdot 8 + A = (A << 3) + A$$

- Proiectați un circuit logic combinațional de înmulțire cu 12 a numărului pozitiv primit la intrare pe 8 biți. Utilizați numai un singur sumator, nu multiplicator.

Modelați circuitul în Verilog cu ajutorul unui singur operator de adunare $+$. Nu folosiți operatorul de înmulțire $*$.

Soluție

$$12|_{10} = 1100|_2 = 2^3 + 2^2$$

- Proiectați un circuit de generare a parității pare sau impare pentru un bus de date de 8 biți. Paritatea este determinată de o intrare suplimentară. Justificare, circuit cu porți logice și model Verilog.

6.3 Pentru cei ce vor să învețe

- Proiectați o unitate logico-aritmetică, ALU („*Arithmetic Logic Unit*”, engl.) pe 4 biți care implementează următoarele operații:

Selectia			Ieșiri	Operația
S_2	S_1	S_0	$\{C_O, REZ_{[3:0]}\}$	
0	0	0	0000	reset
0	0	1	$B - A$	scădere
0	1	0	$A - B$	scădere
0	1	1	$A + B$	adunare
1	0	0	$A \oplus B$	XOR
1	0	1	$A + B$	OR
1	1	0	$A \cdot B$	AND
1	1	1	A	trece A

În plus față de cei 4 biți de date de ieșire, ALU mai are o ieșire C_O , cu semnificația „depășire”, activată doar în cazul operațiilor aritmetice.

Care sunt valorile rezultatelor la ieșirea ALU, dacă la intrare se aplică operandii $A_{[3:0]} = 1101$ și $B_{[3:0]} = 0101$?

Modelați în Verilog unitatea logico-aritmetică propusă și verificați prin simulare răspunsurile la întrebarea anterioară.

6.4 Pentru cei ce vor să devină profesioniști

1. Proiectați un circuit logic combinațional implementat cu porți logice care determină complementul față de 2 al numărului prezentat la intrare, pe orice număr de biți. Cu alte cuvinte, numărul de la ieșire este egal cu numărul de la intrare înmulțit cu -1 . Circuitul proiectat este unul iterativ, format dintr-un circuit de procesare a unui bit instanțiat pentru fiecare bit al operandului.

Propuneți circuite pentru ambii algoritmi de complementare:

- a) se complementează toți biții operandului și se adaugă 1;
- b) se copiază biții de la dreapta la stânga până la primul 1, inclusiv, iar ceilalți biți se neagă.

Modelați în Verilog circuitul care determină complementul față de 2 al numărului prezentat la intrare, pe un număr parametrizabil de biți.

Soluție

- a) Pentru procesarea unui bit, se neagă acel bit și se adună cu o valoare provenită de la bitul anterior (mai puțin semnificativ), obținându-se bitul rezultat curent și un bit de transport. Pentru bitul 0, transportul de intrare este $C_i = 1$. Biții se procesează în ordine, începând de la cel mai puțin semnificativ spre cel mai semnificativ. Circuitul pentru procesarea unui bit este prezentat în figura 6.3-a și se bazează pe ecuația aritmetică:

$$\{C_o, Out\} = \overline{In} + C_i$$

unde cu $+$ s-a notat operatorul de sumă aritmetică.

Rezultă ecuațiile logice:

$$C_o = \overline{In} \cdot C_i$$

$$Out = \overline{In} \oplus C_i$$

- b) Pentru procesarea unui bit, se decide dacă acesta este negat sau nu pe baza unei informații provenite de la bitul de ordin inferior. Pentru bitul 0, transportul de intrare este $C_i[0] = 0$. Biții se procesează în ordine, începând de la cel mai puțin semnificativ spre cel mai semnificativ. Circuitul pentru un bit are aceleași porturi și este caracterizat de următorul tabel de adevăr:

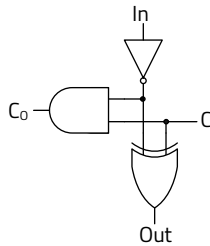


Figura 6.3 Circuit iterativ de complementare (celulă pentru 1 bit):
a) se complementează toți biții operandului și se adaugă 1 ($C_i[0] = 1$).

In	C_i	Out	C_o	Explicație
0	0	0	0	zero-uri inițiale, lsb, se copiază
0	1	1	1	bitul este 0, după ce deja s-a găsit un 1, deci se neagă
1	0	1	1	primul bit egal cu 1, se copiază și se activează transportul
1	1	0	1	bitul este 1, după ce deja s-a găsit un 1, deci se neagă

Rezultă funcțiile logice:

$$C_o = In + C_i$$

$$Out = In \oplus C_i$$

Operatorul + reprezintă OR logic.

Circuitul pentru procesarea unui bit este prezentat în figura 6.4.

Figura 6.5 prezintă circuitele pentru determinarea negatului unui număr, în complement față de 2, în cele două variante.

Modelarea Verilog a circuitului nu trebuie să țină cont de toate detaliile de circuit menționate. Modulul următor prezintă două modalități de descriere a complementării valorii (pozitiv/negativ):

- formula teoretică (logică), complementarea tuturor biților și adunarea cu 1,
- formula aritmetică, valoarea precedată de semnul minus –.

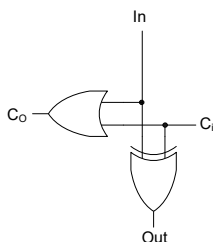


Figura 6.4 Circuit iterativ de complementare (celulă pentru 1 bit):

b) se copiază biții de la dreapta la stânga până la primul 1, inclusiv, iar ceilalți biți se neagă ($C_i[0] = 0$).

Ambele descrieri sunt înțelese de către aplicațiile de sinteză de hardware și transformate automat în structuri de porți logice minime și optimizate.

```
module complement (
input    [4 - 1:0] data_in  ,
output   [4 - 1:0] data_out1 ,
output   [4 - 1:0] data_out2
);

assign data_out1 = ~data_in + 1; // complement logic plus 1
assign data_out2 = -data_in     ; // minus data

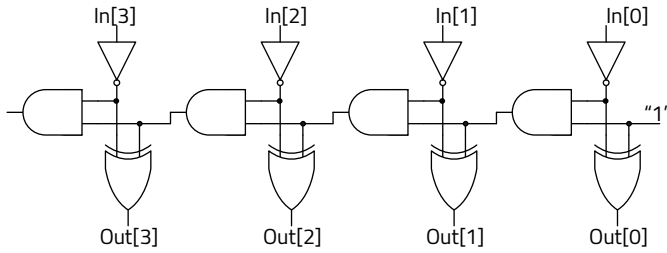
endmodule // complement
```

2. Proiectați un singur circuit care să poată fi utilizat atât pentru generarea parității cât și pentru verificarea acesteia în cazul unei transmisiuni de 8 biți de date, atât pentru paritate pară cât și pentru paritate impară.

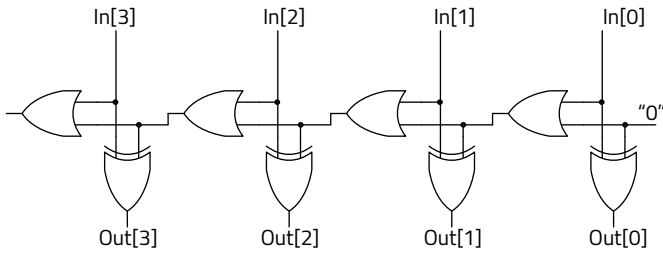
Soluție

Ecuatiile logice pentru generare și verificare de paritate sunt similare: XOR cu mai multe intrări. În plus, schimbarea tipului parității (pară sau impară) presupune negarea expresiilor parității generate și a erorii determinate. Poarta XOR poate fi interpretată ca „inversor comandat”.

În cazul a 8 intrări de date, expresiile menționate la problema 4 (de la seminar) pot fi rescrise pentru a putea fi implementate cu același circuit astfel:



a)



b)

Figura 6.5 Circuit de complementare pe 4 biți:

a) se complementează toți biții operandului și se adaugă 1,

b) se copiază biții de la dreapta la stânga până la primul 1, inclusiv, iar ceilalți biți se neagă.

$$P_{par} = D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0$$

$$ERR_{par} = D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0 \oplus P_{par}$$

Similar,

$$P_{impar} = D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0 \oplus 1$$

$$ERR_{impar} = D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0 \oplus P_{impar} \oplus 1$$

Rezultă că circuitul capabil de a genera și verifica paritatea pară sau impară are funcția logică XOR cu 10 intrări. În funcție de rolul său, circuitul se va utiliza în conexiunile prezentate în tabelul următor.

	I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0
	Generare paritate pară									
P_{par}	0	0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	Verificare paritate pară									
ERR_{par}	0	P_{par}	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	Generare paritate impară									
P_{impar}	1	0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	Verificare paritate impară									
ERR_{impar}	1	P_{impar}	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

3. Utilizând exclusiv sumatoare complete de un bit, proiectați un circuit care furnizează la ieșire numărul de biți egali cu 1 în cadrul cuvântului de 5 biți prezentat la intrare. Model Verilog.

Soluție

Sumatorul complet de 1 bit însumează 3 biți (cu ponderi egale) și generează suma reprezentată pe 2 biți, unul de pondere 2^0 și cel mai semnificativ de pondere 2^1 .

Rezultatul însumării a 5 biți este în domeniul 0...5, deci sunt necesari 3 biți pentru a putea reprezenta suma.

Modelați în Verilog un sumator complet de 1 bit. Apoi realizați un model structural pentru circuitul cerut, prin instanțieri multiple ale sumatorului de 1 bit.

4. Utilizând exclusiv sumatoare complete de un bit, proiectați un circuit care furnizează la ieșire numărul de biți egali cu 1 în cadrul cuvântului de 7 biți prezentat la intrare.
5. Utilizând exclusiv sumatoare complete de un bit, proiectați un circuit care furnizează la ieșire numărul de biți egali cu 1 în cadrul cuvântului de 15 biți prezentat la intrare.
6. Proiectați un circuit logic combinațional care să interfațeze o tastatură cu un afișaj 7-segmente. Datele de intrare sunt generate de o tastatură cu 10 taste numerice.

Schema bloc a modului este prezentată în figura 6.6. La apăsarea uneia din cele 10 taste, se va aprinde numărul corespunzător (între 0 și 9) pe afișajul 7-segmente. În cazul în care nu se apasă nicio tastă, afișajul numeric este complet stins, iar punctul zecimal este aprins. Modelați circuitul în Verilog și verificați-i funcționarea în simulare.

Soluție

Circuitul se structurează în două module:

- 1) *enc10*: codifică cele 10 semnale provenite de la tastatură într-un cod BCD (nicio tastă apăsată determină la ieșire codul $1111|_2 = F|_{16}$).

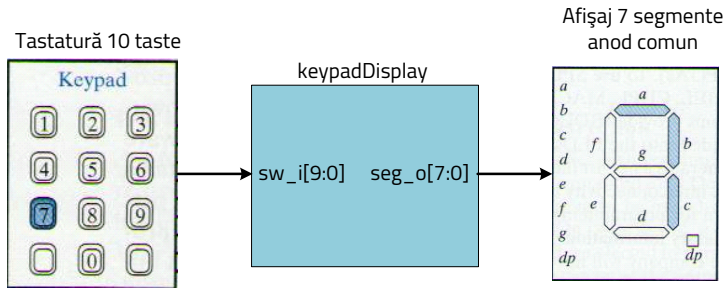


Figura 6.6 Schema bloc a circuitului de interfațare a tastaturii cu afișajul 7-segmente, cu simbolul bloc al modulului *keypadDisplay*.

```

module enc10 (
input      [10 -1:0]  data_i,
output reg [ 4 -1:0]  data_o,
);

always @(data_i)
case (data_i)
10'b1x_xxxx_xxxx: data_o <= 'd9;
10'b01_xxxx_xxxx: data_o <= 'd8;
10'b00_1xxx_xxxx: data_o <= 'd7;
10'b00_01xx_xxxx: data_o <= 'd6;
10'b00_001x_xxxx: data_o <= 'd5;
10'b00_0001_xxxx: data_o <= 'd4;
10'b00_0000_1xxx: data_o <= 'd3;
10'b00_0000_01xx: data_o <= 'd2;
10'b00_0000_001x: data_o <= 'd1;
10'b00_0000_0001: data_o <= 'd0;
default:
data_o <= 'd15;
endcase

endmodule // enc10

```

2) *bcd7seg*: convertor de cod din cod BCD în cod 7-segmente.

```

module bcd7seg (
input      [4 -1:0]  bcd_i ,
output reg [8 -1:0]  seg7_o  // {a, b, c, d, e, f, g, dp}
);

always @(bcd_i)
case (bcd_i)
//                                abcdefg p

```

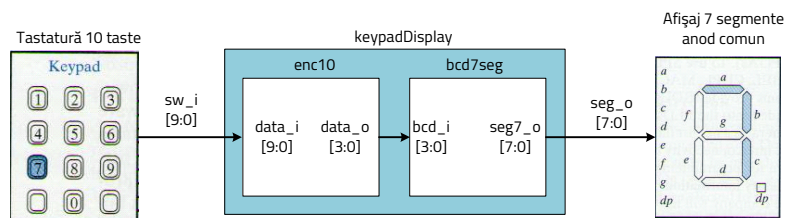


Figura 6.7 Partajarea modului *keypadDisplay* în două module.

```

4'b0000: seg7_o <= 'b1111110_0; // 0
4'b0001: seg7_o <= 'b0110000_0; // 1
4'b0010: seg7_o <= 'b1101101_0; // 2
4'b0011: seg7_o <= 'b1111001_0; // 3
4'b0100: seg7_o <= 'b0110011_0; // 4
4'b0101: seg7_o <= 'b1011011_0; // 5
4'b0110: seg7_o <= 'b1011111_0; // 6
4'b0111: seg7_o <= 'b1110000_0; // 7
4'b1000: seg7_o <= 'b1111111_0; // 8
4'b1001: seg7_o <= 'b1111101_0; // 9
4'b1111: seg7_o <= 'b0000000_1; // dp
default: seg7_o <= 'b0000000_0; // totul stins
endcase

```

```
endmodule // bcd7seg
```

Modulul *keypadDisplay* instanțiază modulele *enc10* și *bcd7seg*:

```

module keypadDisplay (
input  [10 -1:0] sw_i , // switch-uri
output [8  -1:0] seg_o  // display 7 seg
);

wire[4 -1:0] bcd;

enc10 Uenc10 (
    .data_i (sw_i ),
    .data_o (bcd  )
);

bcd7seg Ubcd7seg(
    .bcd_i  (bcd  ),
    .seg7_o (seg_o )
);

endmodule // keypadDisplay

```

7. Proiectați un circuit „*barrell-shifter*” de 4 biți. Implementați circuitul cu multiplexoare 4 : 1. Modelați circuitul în Verilog.

Extindeți modelul Verilog pentru 8 biți. Parametrizați modelul de barrel-shifter.

8. Modelați în Verilog un circuit care numără câte valori de 1 sunt în vectorul de intrare, de dimensiune parametrizabilă.

Soluție

Este un circuit combinațional. Formula matematică este:

$$Out = \sum_{i=0}^{WIDTH-1} In[i]$$

Însă nu există operator de sumă $\sum$ în Verilog. Trebuie descrisă însumarea biților „bit cu bit”.

Dacă vă bate gândul să utilizați  ChatGPT, rezolvați problema repede. Studiați însă care este ideea codului și ce specificații au fost utilizate. Învățați și voi din „experiența ChatGPT”, pentru că și ChatGPT a învățat din experiența înaintașilor săi.

9. Rezolvați **problema Anei**: „Ana are o sumă de bani de la bunica. În fiecare zi cheltuie jumătate plus 1 leu din suma pe care o are, după 3 zile nerămânând nimic. Ce sumă a avut inițial?”.

Să se implementeze un circuit combinațional cu ROM $2^5 \times 30$ care să calculeze suma pe baza numărului de zile pentru maximum o lună ($ZILE \in \{1, 31\}$).

Propuneți un modul Verilog pentru circuitul combinațional proiectat.

Soluție

Soluția problemei generalizate este:

$$SUMA = 2^{ZILE+1} - 2$$

Rezolvarea problemei se poate face cu un circuit combinațional care are la intrare numărul de zile (reprezentat pe 5 biți) și la ieșire suma, reprezentată pe 32 de biți. Rezolvarea este evidentă cu un circuit ROM $2^5 \times 32$.

Însă, prin observarea valorilor stocate în memorie se remarcă faptul că cei mai puțin semnificativi 2 biți ai rezultatului au tot timpul aceeași valoare 10, deci ar putea să nu mai fie scriși în memorie (constantele pot fi legături directe la referința $GND = 0$ sau alimentare $V_{DD} = 1$).

```

SUMA[1]  = 0000_0000_0000_0000_0000_0000_0000_0010
SUMA[2]  = 0000_0000_0000_0000_0000_0000_0000_0110
SUMA[3]  = 0000_0000_0000_0000_0000_0000_0000_1110
SUMA[4]  = 0000_0000_0000_0000_0000_0000_0001_1110
...
SUMA[29] = 0011_1111_1111_1111_1111_1111_1111_1110
SUMA[30] = 0111_1111_1111_1111_1111_1111_1111_1110
SUMA[31] = 1111_1111_1111_1111_1111_1111_1111_1110

```

Circuitul este prezentat în figura 6.8 iar tabelul de programare a ROM-ului este prezentat în tabelul 6.1.

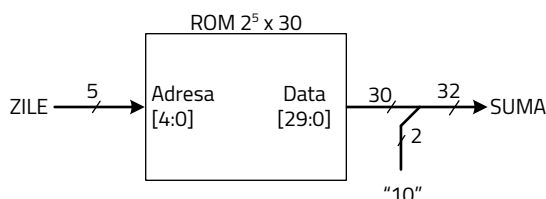


Figura 6.8 Circuit logic implementat cu ROM care rezolvă „problema Anei”.

Modelul Verilog poate fi similar cu al unei memorii ROM cu conținutul prezentat. Însă, este evident riscant de gestionat un tabel binar atât de mare.

Re-analizând însă formula, $SUMA = 2^{ZILE+1} - 2$ se constată că este ușor de descris în Verilog pe un singur rând, observând că 2 la o putere este de fapt o valoare 1 deplasată stânga cu numărul de biți de la puterea dorită.

Astfel, se poate scrie formula ca:

$$SUMA = 1 \ll (ZILE + 1) - 2$$

Modelul Verilog al acestui circuit este următorul.

```

module problema_ana (
input  [5-1:0] zile,    // numarul de zile, intre 1 si 31
output [32-1:0] suma    // rezultatul calculului
);
assign suma = (1 << (zile + 1)) - 2; // 2^(ZILE+1) - 2

```

Adresa [4 : 0]	Data [29 : 0]
0	00000000_00000000_00000000_000000
1	00000000_00000000_00000000_000000
2	00000000_00000000_00000000_000001
3	00000000_00000000_00000000_000011
4	00000000_00000000_00000000_000111
5	00000000_00000000_00000000_001111
6	00000000_00000000_00000000_011111
7	00000000_00000000_00000000_111111
8	00000000_00000000_00000001_111111
9	00000000_00000000_00000011_111111
10	00000000_00000000_00000111_111111
11	00000000_00000000_00001111_111111
12	00000000_00000000_00011111_111111
13	00000000_00000000_00111111_111111
14	00000000_00000000_01111111_111111
15	00000000_00000000_11111111_111111
16	00000000_00000001_11111111_111111
17	00000000_00000011_11111111_111111
18	00000000_00000111_11111111_111111
19	00000000_00001111_11111111_111111
20	00000000_00011111_11111111_111111
21	00000000_00111111_11111111_111111
22	00000000_01111111_11111111_111111
23	00000000_11111111_11111111_111111
24	00000001_11111111_11111111_111111
25	00000011_11111111_11111111_111111
26	00000111_11111111_11111111_111111
27	00001111_11111111_11111111_111111
28	00011111_11111111_11111111_111111
29	00111111_11111111_11111111_111111
30	01111111_11111111_11111111_111111
31	11111111_11111111_11111111_111111

Tabelul 6.1

Tabel de programare a circuitului ROM care rezolvă problema Anei.

```
endmodule // problema_ana
```

Se observă faptul că nu mai este necesar tabelul, se rezolvă de către sintetizator faptul că cei mai puțin semnificativi biți sunt constanți (tot timpul 10), iar existența unei formule face codul mai ușor de scris

și de depanat.

Se remarcă însă, dincolo de simplitatea modelului de câteva caractere, faptul că totul a fost posibil datorită cunoașterii formulei.

Încercați să deduceți singuri formula propusă.

Încercați și cu ajutorul  ChatGPT .

Datați formula generalizată și rugați *ChatGPT* să vă scrie modelul Verilog, ca în figura 6.9.

Surpriză?

Se remarcă faptul că au fost modificate unele aspecte ce îl „deconspiră” pe *ChatGPT*. **Atenție!** *ChatGPT* învață repede și în continuu.

- Numele modulului mai sugestiv pentru problemă (o altă cerere la *ChatGPT* ar fi rezolvat asta).
- Diacritice în comentarii nu sunt acceptate de simulatoare (o altă cerere la *ChatGPT* ar fi rezolvat asta).
- Înlocuit specificația secvențială *always* (utilizată corect dar în mod inutil), cu specificație o de atribuire continuă *assign*.
- Specificația secvențială *always are* în mod inutil *begin* și *end*.
- Declararea dimensiunii ca $[5 - 1 : 0]$ în loc de $[4 : 0]$, chestiune de „stil” (o altă cerere la *ChatGPT* ar fi rezolvat și asta).
- Comentarii mai pertinente și ne-evidente asociate specificațiilor.

6.5 Concluzie: Inginer vs. ChatGPT

Limbajul Verilog vă permite descrierea rapidă a unui sistem digital. Însă, nu vă ajută la proiectare. La chestiuni mai complexe, nici *ChatGPT* nu vă ajută cu rezolvarea.

Însă, *ChatGPT* vă ajută să scrieți cod dacă îi explicați clar funcția de transfer, chiar complexă. Vezi figura 6.9.

(Încă) mai este nevoie de ingineri inteligenți care să rezolve probleme.

Atenție la atributul „*inteligenti*”! Deja *ChatGPT* se pricepe foarte bine la sarcini mai puțin complexe cum ar fi scrierea unui mediu de testare Verilog pentru un modul dat.

Acest lucru se demonstrează cu ușurință pe „*problema Anei*” (vezi figura 6.10). *ChatGPT* a răspuns instantaneu cu prezentarea conceptului, cod

Verilog și explicații. **Și este corect!** De acord, un expert își dă seama că modulul este generat automat din unele mici chestiuni de „stil”: alinieri, diacritice, declarații de dimensiune, modelarea timpului, specificație de terminare simulare. Dar este un început rapid pentru un mediu de testare. Sunt totuși niște erori numerice în test-bench. Le găsiți?

Ai fi răspuns mai rapid sau mai corect? Sigur că nu. Însă nu te supăra. Niciun om nu ar fi făcut-o.

Reține

Nu te lua la întrecere cu *ChatGPT* pe chestiuni simple.

Doar chestiunile complexe îți vor scoate în evidență conștințele!



Scrie un model Verilog care să calculeze formula:
 $SUMA = 2^{(ZILE+1)} - 2$, considerând că ZILE este un număr între 1 și 31.



Pentru a crea un model Verilog care să calculeze $SUMA = 2^{(ZILE+1)} - 2$, unde ZILE este un număr între 1 și 31, putem folosi o intrare pe 5 biți pentru ZILE, deoarece 5 biți sunt suficienți pentru a reprezenta valorile de la 1 la 31.

Model Verilog

```
verilog Copy code

module calcul_suma (
    input [4:0] zile,          // numărul de zile, între 1 și 31
    output reg [31:0] suma    // rezultatul calculului
);

    always @(*) begin
        suma = (1 << (zile + 1)) - 2; // Calculăm 2^(ZILE+1) - 2 folosind shift la stânga
    end

endmodule
```

Explicații

- **Input:** zile este pe 5 biți, pentru a permite valori de la 1 la 31.
- **Output:** suma este rezultatul calculului, reprezentat pe 32 de biți, pentru a acoperi gama de valori rezultate. Aceasta va acoperi valori mari, până la $2^{32} - 2$, în funcție de valoarea lui ZILE.
- **Calcul:** $1 \ll (zile + 1)$ realizează operația de shift la stânga, echivalentă cu calculul lui $2^{(ZILE+1)}$, apoi scădem 2. 

Figura 6.9 Răspunsul ChatGPT la „problema Anei”: Model Verilog pe baza formulei prezentate.

Scrie un mediu de testare pentru acest modul Verilog:

```
module problema_ana (  
    input [5-1:0] zile, // numarul de zile, între 1 și 31  
    output[32-1:0] suma // rezultatul calculului  
);  
    assign suma = (1 << (zile + 1)) - 2; // 2^(ZILE+1) - 2  
  
endmodule // problema_ana
```



Pentru a testa modulul `problema_ana`, putem crea un testbench care va aplica diverse valori la intrarea `zile` și va verifica valoarea corespunzătoare a ieșirii `suma`.

Testbench-ul va include o secvență de testare pentru diferite valori ale lui `zile`, inclusiv valorile limită (de exemplu, 1 și 31), pentru a ne asigura că modulul funcționează corect pe întreaga gamă de intrări.

Figura 6.10 Mediu de verificare pentru „*problema Anei*” realizat de *ChatGPT*. Concept.

Testbench Verilog

```
verilog Copy code

module tb_problema_ana;
    // Declaram registre și fire
    reg [4:0] zile;           // Intrarea zile va avea valori între 1 și 31
    wire [31:0] suma;        // Ieșirea suma pentru verificarea rezultatelor

    // Instanțierea modului de testat
    problema_ana dut (
        .zile(zile),
        .suma(suma)
    );

    // Bloc initial pentru a aplica valorile de test
    initial begin
        // Afisare pentru inceputul simulării
        $display("Testare problema_ana:");
        $display("Zile | Suma calculata");

        // Testăm câteva valori reprezentative ale lui zile, inclusiv limitele
        zile = 1; #10; $display("%d | %d", zile, suma); // Așteptat: 2
        zile = 2; #10; $display("%d | %d", zile, suma); // Așteptat: 6
        zile = 3; #10; $display("%d | %d", zile, suma); // Așteptat: 14
        zile = 4; #10; $display("%d | %d", zile, suma); // Așteptat: 30
        zile = 5; #10; $display("%d | %d", zile, suma); // Așteptat: 62
        zile = 10; #10; $display("%d | %d", zile, suma); // Așteptat: 4094
        zile = 15; #10; $display("%d | %d", zile, suma); // Așteptat: 131070
        zile = 20; #10; $display("%d | %d", zile, suma); // Așteptat: 4194302
        zile = 25; #10; $display("%d | %d", zile, suma); // Așteptat: 134217726
        zile = 31; #10; $display("%d | %d", zile, suma); // Așteptat: 4294967294 (2^32)

        // Sfârșitul testării
        $finish;
    end
endmodule
```

Figura 6.11 Mediu de verificare pentru „problema Anei” realizat de ChatGPT. Cod Verilog. **Atenție, codul conține erori.**

Explicație a Testbench-ului

1. Declararea registrelor și firelor:

- `zile` este declarat ca registru pe 5 biți pentru a putea acoperi toate valorile de intrare între 1 și 31.
- `suma` este un fir pe 32 de biți care va prelua valoarea calculată din modulul `problema_ana`.

2. Instanțierea modului `problema_ana`:

- Creăm o instanță a modului pe care îl testăm (`dut`), conectând `zile` și `suma`.

3. Blocul `initial` pentru testare:

- Setăm diferite valori pentru `zile`, aplicăm un timp de așteptare (`#10`) pentru a permite evaluarea, și afișăm rezultatul folosind `$display` pentru fiecare valoare testată.
- Rezultatele așteptate sunt date pentru câteva valori reprezentative, inclusiv limita superioară de 31.

4. Încheierea testării:

- După ce toate valorile au fost testate, apelăm `$finish` pentru a termina simularea.

Acest testbench va afișa valorile calculate pentru `suma` în funcție de `zile` și ne permite să verificăm dacă formula este implementată corect.

Figura 6.12 Mediu de verificare pentru „*problema Anei*” realizat de ChatGPT. Explicații.

Lecția 7

Circuite cu bistabile

Sunt păreri conform cărora electronica digitală este extrem de simplă:

- se bazează pe o matematică cu două simboluri, 0 și 1,
- se construiește numai cu **porți** și **bistabile**.

Această lecție este pentru cei ce rămân la aceste păreri.

Confirm: circuitele electronice digitale sunt formate din porți și bistabile.

Dar, **mu-u-u-lte** porți și **mu-u-u-lte** bistabile.

Deci, este nevoie să înțelegeți foarte bine cum funcționează un bistabil.

7.1 Pentru seminar

1. Care este diferența dintre un „*latch D*” și un „*bistabil D*”?

Completați formele de undă ale ieșirilor latch-ului D și ale bistabilului D în cazul stimulilor de intrare prezentați în figura 7.1.

Soluție

Pentru cazul b), formele de undă generate sunt prezentate în figura 7.2. Se consideră latch-ul activ pe palierul de 1 și bistabilul activ pe frontul crescător.

2. Proiectați structura internă a unui bistabil cu validare având la dispoziție un bistabil D și un multiplexor 2:1.

Modelați în Verilog circuitul proiectat și verificați comportamentul prin simulare.

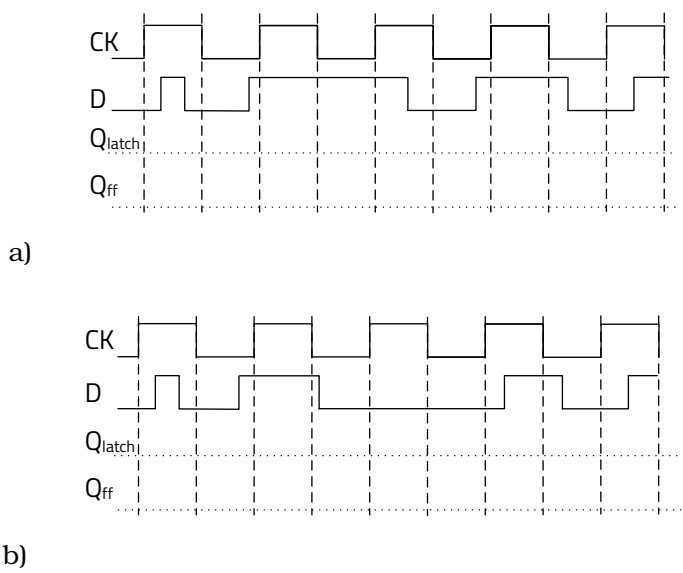


Figura 7.1 Stimuli pentru *latch D* și *bistabil D*, problema 1.

Soluție

Soluția se bazează pe comportamentul bistabilului exprimat pe baza intrării de validare, similar cu funcția de transfer a unui multiplexor: *dacă bistabilul este validat, starea memorează intrarea, dacă nu este validat, starea se păstrează*, conform tabelului:

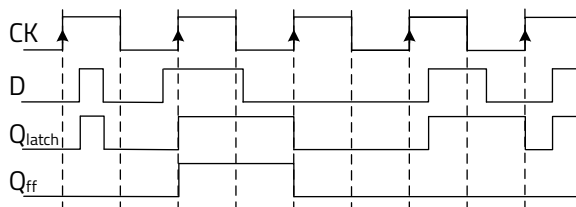
E	Q^+	Acțiune
1	D	copiază intrarea, în momentul frontului crescător al ceasului
0	Q	păstrează starea, indiferent de intrarea D (este invalidat)

Intrarea de validare se conectează pe intrarea de selecție a multiplexorului.

Pe intrarea de date 1 a multiplexorului se conectează intrarea de date care trebuie să ajungă la bistabil.

Pe intrarea de date 0 a multiplexorului se conectează ieșirea bistabilului, astfel starea prezentă va fi și stare viitoare, în următorul ciclu de ceas.

Circuitul poate fi modelat cu următorul cod Verilog:



b)

Figura 7.2 Forme de undă generate de latch D și de bistabil D, problema 1-b.

```

always @(posedge ck)
q <= e ? d : q;    // daca e=1 q+ <= d, esantioneaza intrarea
                  // daca e=0 q+ <= q, pastreaza starea

```

Alternativ, circuitul poate fi modelat cu o specificație *if* fără clauza *else*, care modifică valoarea bistabilului dacă $E = 1$ și o lasă neschimbată (o păstrează) dacă $E = 0$.

```

always @(posedge ck)
if (e) q <= d;    // daca e=1: q+ <= d
              // altfel: memoreaza intrarea

```

- Determinați funcțiile implementate de circuitele cu bistabile prezentate în figura 7.3. Explicați comportamentul cu forme de undă. Modelați în Verilog circuitele.

Soluție

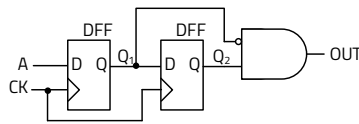
a) Detector de front negativ, puls de o perioadă de ceas ca urmare a frontului descrescător al semnalului de intrare (1 urmat de 0).

b) Detector de front pozitiv, puls de o perioadă de ceas ca urmare a frontului crescător al semnalului de intrare (0 urmat de 1).

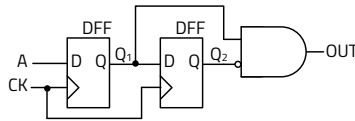
c) Detector de front, puls de o perioadă de ceas ca urmare a unui front al semnalului de intrare (0 urmat de 1 sau 1 urmat de 0).

d) Circuit de întârziere cu două perioade de ceas, ieșirea are aceeași formă de semnal cu intrarea, însă defazată în timp cu 2 perioade de ceas.

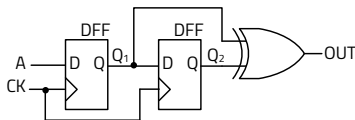
- Proiectați un circuit cu bistabile care la fiecare puls de intrare cu lățime de o perioadă de ceas generează la ieșire o secvență de două pulsuri, cu lățime de o perioadă de ceas la un interval de o perioadă de ceas, așa cum se prezintă în figura 7.4.



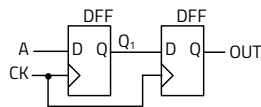
a)



b)



c)



d)

Figura 7.3 Circuite cu bistabile D referite la problema 3.

7.2 Pentru cei ce vor doar să promoveze examenul

1. **Explicați comportamentul circuitelor** prezentate în figura 7.5. Desenați formele de undă ale semnalelor interioare, în anumite scenarii ale intrării A.
2. La intrarea a două bistabile, unul D și altul T, se aplică un semnal *IN* cu forma de undă prezentată în figura 7.6. Să se determine formele de undă ale ieșirilor celor două bistabile: Q_{DFF} și Q_{TFF} .

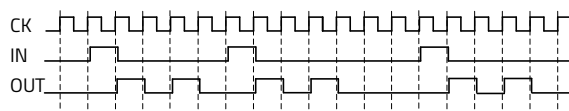
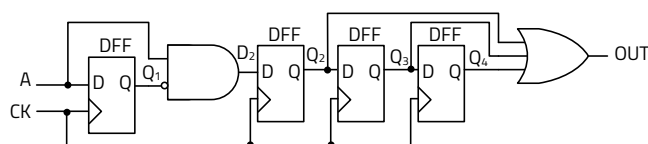
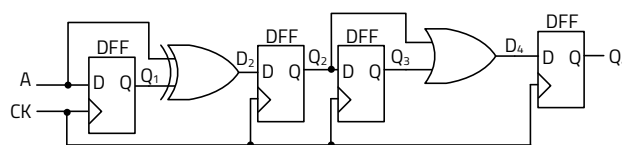


Figura 7.4 Forme de undă pentru circuitul descris la problema 4.



a)



b)

Figura 7.5 Circuite cu bistabile D și porți logice, referite la problema 1.

7.3 Pentru cei ce vor să învețe

1. Proiectați un circuit cu bistabile care prezintă la ieșire un puls de lățime 8 perioade de ceas la fiecare puls primit pe intrare, indiferent de durata acestuia. Prezentați condițiile de funcționare ale circuitului.

Modelați circuitul în Verilog și verificați circuitul proiectat prin simulare.

Soluție

Apariția pulsului poate fi detectată cu un circuit detector de front pozitiv, format dintr-un bistabil și o poartă AND cu o intrare negată. Lățimea pulsului pe ieșirea bistabilului detector de front este întot-

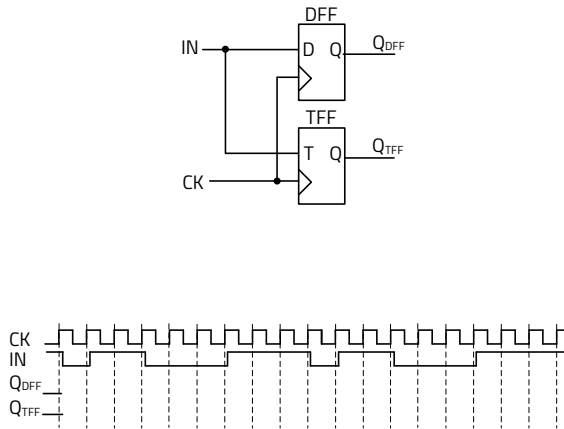


Figura 7.6 Circuit și forme de undă pentru problema 2.

deuna o singură perioadă de ceas. Acest puls se trece printr-o serie de 7 bistabile D și se obțin versiuni întârziate cu până la 7 perioade de ceas.

O poartă OR colectează toate cele 8 pulsuri, generând un puls de lățime egală cu 8 perioade de ceas. Circuitul este prezentat în figura 7.7.

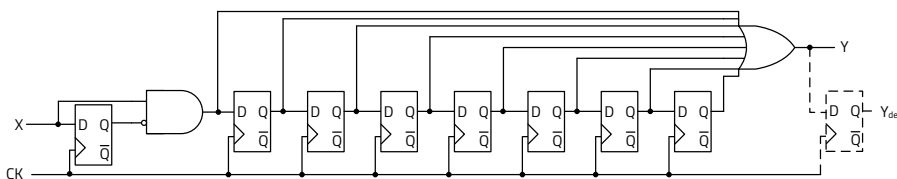


Figura 7.7 Puls de lățime 8 perioade de ceas, problema 1.

Pentru creșterea frecvenței de lucru, se recomandă trecerea ieșirii porții OR printr-un alt bistabil (pulsul obținându-se cu o întârziere de un tact, dar evitându-se ieșirea combinațională).

2. Se consideră circuitele din figurile 7.8 și 7.9. La o ieșire se observă formele de undă din figuri. Să se determine formele de undă ale semnalelor de la ieșirile celorlalte bistabile.

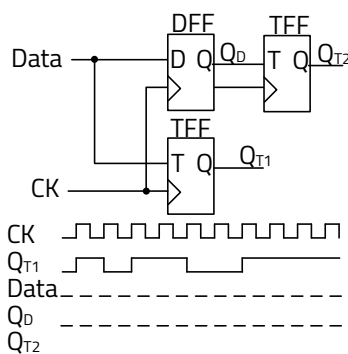


Figura 7.8 Circuite și forme de undă pentru problema 2.

3. Proiectați un circuit de „*vot majoritar*” care primește secvențial, pe o singură intrare, valorile de adevăr ale voturilor și prezintă la ieșire starea logică a majorității ultimelor 3 valori primite.

Determinați secvența de ieșire în cazul aplicării la intrare a secvenței 00111110001100101010100111. Realizați un model Verilog pentru circuitul proiectat și verificați corectitudinea soluției pe exemplul considerat.

Soluție

Intrarea se întârzie prin două bistabile D, astfel încât la un moment dat pot fi accesate ultimele 3 valori ale intrării. Pe baza celor 3 valori, se determină valoarea majorității („2 din 3”) cu un circuit combinational.

4. Analizați circuitul prezentat în figura 7.10. Realizați un model Verilog pentru circuitul prezentat și verificați-l prin simulare.

Soluție

Circuitul este folosit pentru filtrarea pulsurilor mai scurte de 3 perioade de ceas. Starea ieșirii se modifică doar dacă pe intrare starea rămâne constantă cel puțin 3 perioade de ceas. Circuitul este folosit pentru „curățarea” unui semnal de intrare de posibile zgomote de scurtă durată (în acest caz, mai scurte decât 3 perioade de ceas). Forme de undă explicative sunt prezentate în figura 7.10.

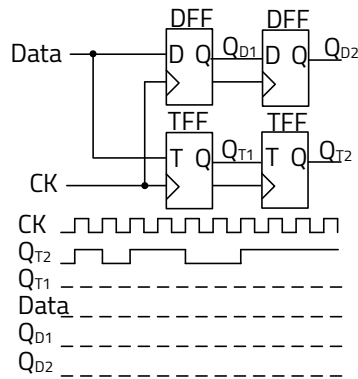


Figura 7.9 Circuite și forme de undă pentru problema 2.

7.4 Pentru cei ce vor să devină profesioniști

1. Proiectați un circuit secvențial cu bistabile D cu o intrare și o ieșire. Lățimea pulsului prezentat la ieșire este întotdeauna un număr par de perioade de ceas, astfel:

- Dacă pulsul de intrare are lățimea un număr par de perioade de ceas, pulsul de ieșire are aceeași lățime și este în fază cu pulsul de intrare.
- Dacă pulsul de intrare are lățimea un număr impar de perioade de ceas, pulsul de ieșire are lățimea mai mare cu o perioadă de ceas față de pulsul de intrare (începe în fază cu pulsul de intrare și se termină cu un tact întârziere față de acesta).

Între două pulsuri consecutive există o pauză de minimum două perioade de tact. Formele de undă caracteristice sunt prezentate în figura 7.11.

2. Proiectați un circuit secvențial cu bistabile D cu o intrare și o ieșire. Lățimea pulsului prezentat la ieșire este întotdeauna un număr multiplu de 3 perioade de ceas, astfel:

- Dacă pulsul de intrare are lățimea un număr de perioade de ceas multiplu de 3, pulsul de ieșire are aceeași lățime și este în fază cu pulsul de intrare.

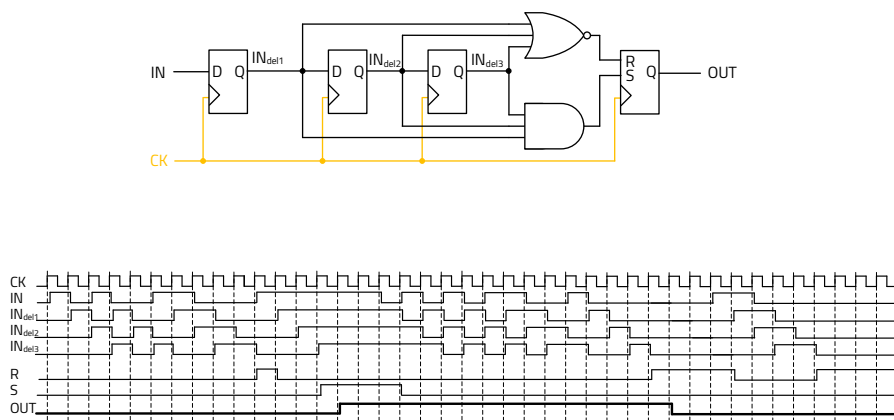


Figura 7.10 Circuit de analizat și forme de undă, (problema 4).



Figura 7.11 Forme de undă pentru circuitul care prezintă la ieșire pulsuri de lățime egală cu un număr par de perioade de ceas (problema 1).

- Dacă pulsul de intrare nu are lățimea un număr de perioade de ceas multiplu de 3, pulsul de ieșire are lățimea egală cu cel mai mic număr de perioade de ceas, multiplu de 3, mai mare decât lățimea pulsului de intrare (începe în fază cu pulsul de intrare și se termină cu un tact sau două întârziere față de acesta).

Între două pulsuri consecutive există o pauză de minimum trei perioade de tact.

3. Proiectați un circuit cu bistabile care semnalează pe ieșire apariția pe intrare a 3 sau mai mulți biți secvențiali și consecutivi egali cu 1. Realizați un model Verilog pentru circuitul proiectat și verificați corectitudinea soluției pe un exemplu.
4. Proiectați un circuit cu bistabile care semnalează pe ieșire detectarea unui șir de biți aplicați secvențial pe intrare cu valorile 101.

Pattern-ul detectat se poate suprapune parțial peste o apariție anterioară, așa cum se prezintă în exemplul din figura 7.12.

Realizați un model Verilog pentru circuitul proiectat și verificați corectitudinea soluției pe exemplul considerat.

In: 01010111001011001011100111001
 Out: 00001010000001000001000100000

Figura 7.12 Exemplu de semnale de intrare și ieșire pentru circuitul de detectare a unui pattern serial.

5. Proiectați un circuit cu bistabile care semnalează pe ieșire existența pe cele două intrări a 3 sau mai multe coincidențe secvențiale și consecutive ale celor două intrări.
6. Proiectați un circuit cu bistabile care semnalează pe ieșire existența pe intrare a 3 sau mai mulți biți secvențiali și consecutivi egali.

Lecția 8

Analiza circuitelor logice secvențiale

Această lecție este despre o activitate foarte frecventă pentru proiectantul de sisteme electronice digitale: *înțelegerea unui proiect în scopul depanării sau modificării comportamentului acestuia*.

De multe ori, se constată funcționarea incorectă (existența unui „*bug*”) după mult timp de la proiectare. Alteori, se schimbă specificațiile și trebuie realizate modificări ale circuitului. Poate că proiectantul inițial nu mai este disponibil, sau este dar... a cam uitat ce a proiectat și documentația este... cum a scris-o.

Apare o problemă de „**analiză**”: există circuitul (poate sub forma unui cod Verilog) și trebuie dedusă funcționarea acestuia.

8.1 Pentru seminar

1. Analizați circuitul logic secvențial prezentat în figura 8.1.
2. Analizați circuitul logic secvențial prezentat în figura 8.2.

Soluție

b) Multiplexorul este un circuit logic combinațional care selectează una din intrările de date (I_1 sau I_0) pe baza intrării de selecție S .

Ecuatia multiplexorului este: $Y = \overline{S} \cdot I_0 + S \cdot I_1$.

Ca o consecință, ecuațiile stării viitoare sunt:

$$D_1 = \overline{Sens} \cdot Q_0 + Sens \cdot \overline{Q_0},$$

$$D_0 = \overline{Sens} \cdot \overline{Q_1} + Sens \cdot Q_1.$$

Automatul implementează un numărător în cod Gray reversibil (intrarea $Sens$ determină direcția de numărare). Graful de tranziții este prezentat în figura 8.3.

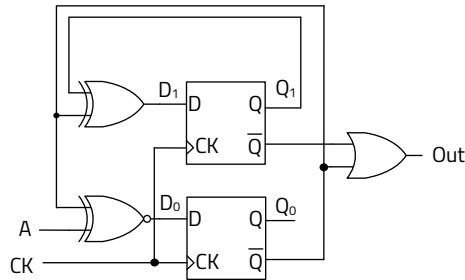


Figura 8.1 Automatul de analizat la problema 1.

8.2 Pentru cei ce vor doar să promoveze examenul

1. **Analizați automatele** prezentate în figurile 8.4, 8.5.

Prezentați tabelele de tranziții, grafurile de tranziții și organigramele. Prezentați succesiunea stărilor în cazul unor date constante aplicate pe intrări.

Prezentați diagramele temporale ale stărilor și ale ieșirilor, în cazul unor scenarii ale intrărilor.

Realizați un **model Verilog** pentru automate și confirmați prin simulare succesiunea stărilor în cazul unor date constante aplicate pe intrări.

8.3 Pentru cei ce vor să învețe

1. Analizați circuitul logic secvențial prezentat în figura 8.6.

Soluție

Circuitul constă din două bistabile D și porți logice. Se observă că există căi combinaționale (bucle) între ieșirile bistabilelor și intrările acestora. Deci, circuitul este un automat secvențial sincron (toate bistabilele comută pe același semnal de ceas, provenit de la o intrare primară).

Automatul are două intrări (A și B) și două ieșiri (Y și Z). Ieșirile depind exclusiv de stare (ieșirile bistabilelor). Deci, este un automat de tip Moore.

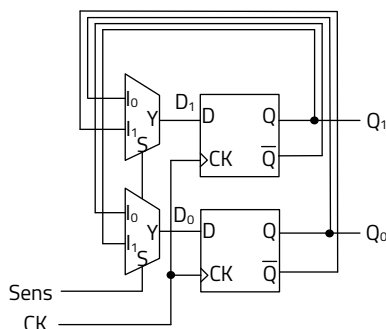


Figura 8.2 Automatul de analizat la problema 2.

Deoarece există două bistabile în schemă, înseamnă că automatul are stările codificate cu 2 biți. Deci, automatul are maximum $2^2 = 4$ stări. Starea prezentă este determinată de valorile logice ale ieșirilor bistabilelor. Biții de cod ai stării prezente sunt Q_1 și Q_0 .

Intrările bistabilelor reprezintă biții stării viitoare (D_1 și D_0).

Schema bloc a automatului este prezentată în figura 8.7.

Ecuatiile stării viitoare se deduc din circuit, în funcție de starea prezentă și intrări:

$$D_1(A, B, Q_1, Q_0) = \bar{A} \cdot \bar{Q}_1 \cdot Q_0 + \bar{A} \cdot Q_1 \cdot \bar{Q}_0 + A \cdot \bar{B} \cdot Q_1 \text{ și}$$

$$D_0(A, B, Q_1, Q_0) = \bar{B} \cdot Q_1 + \bar{A} \cdot \bar{Q}_1 \cdot \bar{Q}_0 + \bar{A} \cdot Q_1 \cdot Q_0 + A \cdot \bar{Q}_1 \cdot Q_0$$

Ecuatiile ieșirilor se deduc din circuit, în funcție de starea prezentă:

$$Y(Q_1, Q_0) = Q_1 \text{ și}$$

$$Z(Q_1, Q_0) = Q_1 + Q_0$$

Se completează tabelul de tranziții în următoarea succesiune.

- Se realizează capul de tabel. De la stânga la dreapta apar intrările (A și B), numele și codul stării prezente (Q_1 și Q_0), numele și codul stării viitoare (D_1 și D_0), numele ieșirilor (Y și Z).
- Numărul de rânduri ale tabelului este egal cu 2^{N+M} , unde s-a notat:
 N = numărul de intrări (2)
 M = numărul de biți care codifică starea (2)

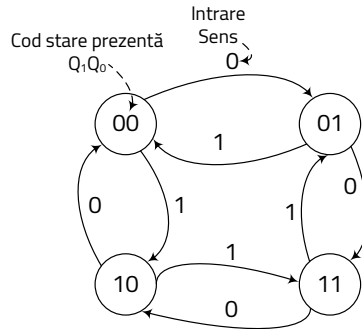


Figura 8.3 Graful de tranziții al automatului analizat la problema 2-b).

În acest caz vor fi $2^{2+2} = 2^4 = 16$ rânduri. Se completează cele 16 rânduri cu coloanele asociate intrărilor și stării prezente. Se recomandă următoarea ordine a codurilor: toate variantele de intrări (00, 01, 10, 11) asociate cu fiecare cod de stare (00, 01, 10, 11). Se asociază nume generice pentru cele 4 stări.

De exemplu: $S_0 = 00$, $S_1 = 01$, $S_2 = 10$, $S_3 = 11$.

- (c) Tabelul de tranziții completat parțial este prezentat în tabelul 8.1. Se remarcă faptul că, până aici, tabelul depinde doar de caracteristicile acestuia (număr de intrări, număr de ieșiri, număr de bistabile) și este independent de porțile logice și conexiunile din circuitul analizat.
- (d) Pe baza ecuațiilor stării viitoare $D_1(A, B, Q_1, Q_0)$, $D_0(A, B, Q_1, Q_0)$, și ale ieșirilor $Y(Q_1, Q_0)$, $Z(Q_1, Q_0)$ se completează coloanele corespunzătoare acestora. Valorile complete sunt prezentate în tabelul 8.2.
- (e) Cu datele centralizate în tabelul de tranziții se construiește graful de tranziții ce descrie comportamentul automatului. Nodurile grafului semnifică stările automatului. Deci, vor exista 4 noduri, asociate celor 4 stări. În noduri se marchează numele (sau codul) stărilor. Fiind un automat Moore, în noduri apar și numele ieșirilor active în respectivele stări.

Fiecare rând din tabel descrie un arc. Arcul pleacă de pe nodul asociat stării prezente și ajunge pe nodul stării viitoare. Pe arc este înscrisă condiția de comutare (valorile intrărilor A și B care determină tranziția). Graful de tranziții al automatului analizat este prezentat în figura 8.8-a). În noduri apar, în ordine,

Intrări		Stare prezentă		Stare viitoare			Ieșiri		
<i>A</i>	<i>B</i>	nume	<i>Q</i> ₁	<i>Q</i> ₀	nume	<i>D</i> ₁	<i>D</i> ₀	<i>Y</i>	<i>Z</i>
0	0	<i>S</i> ₀	0	0					
0	1		0	0					
1	0		0	0					
1	1		0	0					
0	0	<i>S</i> ₁	0	1					
0	1		0	1					
1	0		0	1					
1	1		0	1					
0	0	<i>S</i> ₂	1	0					
0	1		1	0					
1	0		1	0					
1	1		1	0					
0	0	<i>S</i> ₃	1	1					
0	1		1	1					
1	0		1	1					
1	1		1	1					

Tabelul 8.1

Tabelul de tranziții, parțial completat (problema 1).

Linie	Intrări		Stare prezentă		Stare viitoare			Ieșiri		
	A	B	Q_1	Q_0	D_1	D_0	Y	Z		
0	0	0	S_0	0	0	S_1	0	1	0	0
1	0	1		S_1	0	1	0	0	0	0
2	1	0		S_0	0	0	0	0	0	0
3	1	1		S_0	0	0	0	0	0	0
4	0	0	S_1	0	1	S_2	1	0	0	1
5	0	1		S_2	1	0	0	0	1	1
6	1	0		S_1	0	1	0	0	1	1
7	1	1		S_1	0	1	0	0	1	1
8	0	0	S_2	1	0	S_3	1	1	1	1
9	0	1		S_2	1	0	1	0	1	1
10	1	0		S_3	1	1	1	1	1	1
11	1	1		S_0	0	0	1	1	1	1
12	0	0	S_3	1	1	S_1	0	1	1	1
13	0	1		S_1	0	1	1	1	1	1
14	1	0		S_3	1	1	1	1	1	1
15	1	1		S_0	0	0	1	1	1	1

Tabelul 8.2

Tabelul de tranziții, completat integral (problema 1).

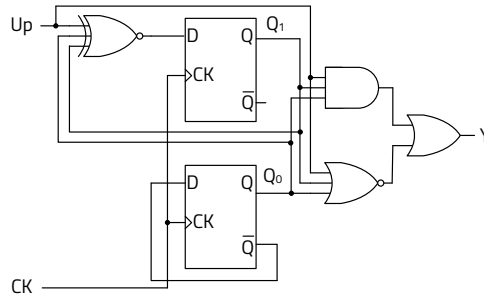


Figura 8.4 Automat referit la problema 1 a).

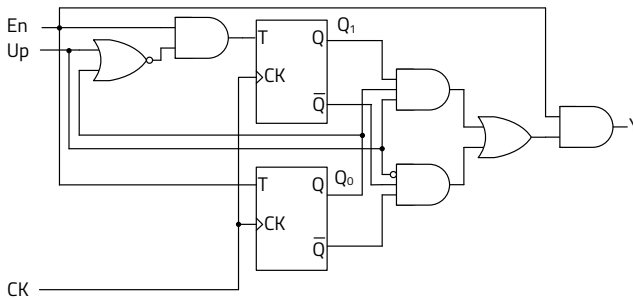


Figura 8.5 Automat referit la problema 1 b).

Q_1Q_0/YZ (starea/ieșiri) iar pe arce apar AB (intrări).

- (f) Se remarcă faptul că pe graf apar arce echivalente (aceeași sursă și aceeași destinație). Acestea se pot contopi într-un singur arc, având o condiție de comutare determinată cu o funcție OR a condițiilor celor două arce inițiale.

Această observație se putea face și pe tabelul de tranziții (linii cu aceeași stare prezentă și aceeași stare viitoare, diferența fiind doar la condiții).

Tabelul de tranziții poate fi scris sub o formă condensată prin contopirea liniilor 0 și 1, 2 și 3, 4 și 5, 6 și 7, 8 și 9, 12 și 13. Cu liniuță – a fost marcată o valoare indiferentă a intrării.

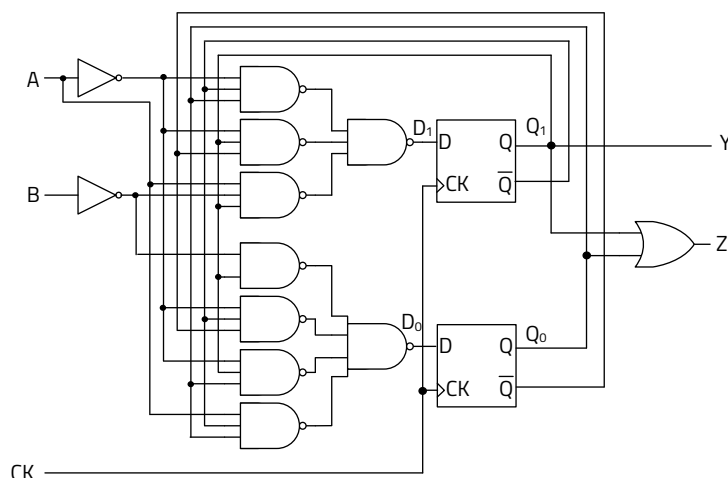


Figura 8.6 Automatul de analizat, propus la problema 1.

Intrări		Stare prezentă			Stare viitoare			Ieșiri	
A	B	nume	Q_1	Q_0	nume	D_1	D_0	Y	Z
0	-	S_1	0	0	S_1	0	1	0	0
1	-		0	0	S_0	0	0	0	0
0	-	S_1	0	1	S_2	1	0	0	1
1	-		0	1	S_1	0	1	0	1
-	0	S_2	1	0	S_3	1	1	1	1
0	1		1	0	S_2	1	0	1	1
1	1		1	0	S_0	0	0	1	1
0	-	S_3	1	1	S_1	0	1	1	1
1	0		1	1	S_3	1	1	1	1
1	1		1	1	S_0	0	0	1	1

Graful de tranziții simplificat (cu condiții pe bază de intrări indifferente) este prezentat în figura 8.8-b).

- (g) Organigrama automatului (prezentată în figura 8.9) se poate deduce fie din tabelul de tranziții, fie din graful de tranziții (figura 8.8-b). Fiecare nod al grafului este asociat cu un simbol de stare din organigramă (dreptunghi). Arcele multiple care pleacă de pe un nod determină un simbol de decizie în organigramă (romb).
- (h) Starea inițială a automatului (determinată de resetarea asincronă) este marcată pe graful de tranziții și pe organigramă cu o săgeată ce intră în nodul sau simbolul de stare asociate acesteia.

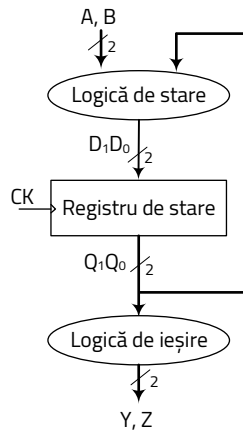


Figura 8.7 Schema bloc a automatului analizat la problema 1.

8.4 Pentru cei ce vor să devină profesioniști

- Proiectați un automat cu două intrări și o ieșire având următoarea funcționare:
 - Pe una din intrări (de date) se primesc secvențial grupuri de câte 4 biți, primul primit este bitul cel mai semnificativ.
 - Pe a doua intrare (de validare) se primesc pulsuri de câte o singură perioadă de ceas, nu mai dese decât un puls la 4 perioade de ceas. A doua intrare marchează momentul când pe prima intrare este prezent primul bit din grupul de 4 biți (cel mai semnificativ).
 - Între două grupuri secvențiale de câte 4 biți pot fi și pauze. Pe durata pauzei, valorile intrării de date sunt ignorate.
 - Ieșirea este activată dacă la finalul unei secvențe de 4 biți validă se constată că secvența nu reprezintă cifre în format BCD (între 0 și 9) adică una din combinațiile 1010, 1011, 1100, 1101, 1110, 1111.
- Analizați automatul prezentat în figura 8.10. Ce funcție îndeplinește?
- Analizați automatul prezentat în figura 8.12. Care este diferența de comportament față de automatul din figura 8.10?

Soluție

a) Automatul are în structură două bistabile T. Tabelul de tranziții

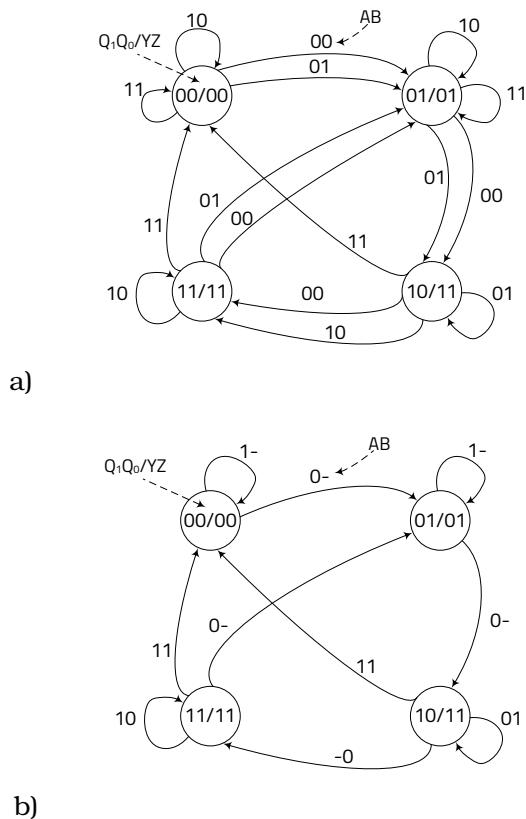


Figura 8.8 Graful de tranziții al automatului analizat la problema 1.
a) integral, b) simplificat.

este prezentat în continuare iar graful de tranziții este prezentat în figura 8.12. Coloanele asociate intrărilor T_1 și T_0 se completează pe baza ecuațiilor deduse din circuit. Ecuațiile stării viitoare, Q_1^+ și Q_0^+ se completează ținând cont de funcționarea bistabilelor T în cazul când se află în starea Q și primesc stimulul T .

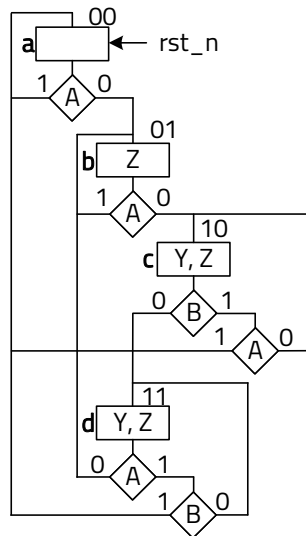


Figura 8.9 Organigrama automatului analizat la problema 1.

Intrare E	Stare prezentă $Q_1 \quad Q_0$		Stimuli T $T_1 \quad T_0$		Stare viitoare $Q_1^+ \quad Q_0^+$	
	Q_1	Q_0	T_1	T_0	Q_1^+	Q_0^+
0	0	0	0	0	0	0
1	0	0	0	1	0	1
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	0	0	1	0
1	1	0	1	0	0	0
0	1	1	1	1	0	0
1	1	1	1	1	0	0

Graful de tranziții prezintă un numărător modulo 3 cu validare. Dacă $E = 0$, automatul își păstrează starea. Dacă $E = 1$, automatul cicleză cu o perioadă de 3 stări 00, 01, 10, 00,

b) Automatul de la punctul **a)** tranzitează necondiționat din starea 11 în starea 00 (aflată în ciclul de numărare).

Automatul de la punctul **b)** tranzitează din starea 11 în starea 00 cu condiția $E = 1$. Dacă $E = 0$, automatul își păstrează starea 11. Cu alte cuvinte, intrarea de validare E își face efectul și asupra stării de blocare 11.

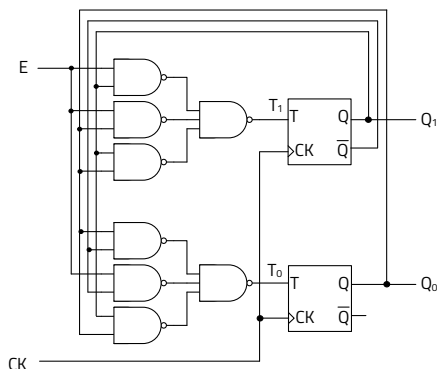


Figura 8.10 Automatul propus a fi analizat la problema 2.

4. Analizați automatul prezentat în figura 8.13. Realizați tabelul de tranziții și graful de tranziții. Prezentați succesiunea stărilor în cazul unor date constante aplicate pe intrări. Modelați automatul în Verilog și verificați funcționarea.

Soluție

Automatul are două intrări X_1 și X_2 și o ieșire Z dependentă exclusiv de stare. Automatul Moore are un singur bit de stare (denumit Q), deci poate avea maximum două stări.

Ecuția intrării bistabilului D este:

$$D = \overline{X_1} \cdot X_2 \cdot \overline{Q} + X_2 \cdot Q$$

Tabelul de stare va conține două stări ($Q = 0$ și $Q = 1$), fiecare asociate cu 4 combinații ale intrărilor.

X_1	X_2	Q	D	$Z = Q$
0	0	0	0	0
0	1	0	1	0
1	0	0	0	0
1	1	0	0	0
0	0	1	0	1
0	1	1	1	1
1	0	1	0	1
1	1	1	1	1

Graful de tranziții este de tip Moore, având două noduri și marcarea pe arce a combinațiilor intrărilor (figura 8.14).

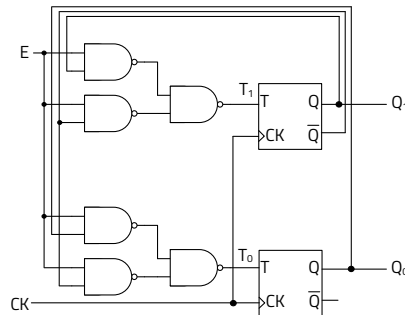


Figura 8.11 Automatul propus a fi analizat la problema 3.

5. Proiectați un automat care prezintă la ieșire un puls de lățime 8 perioade de ceas la fiecare puls de o perioadă de ceas primit pe intrare. Prezentați condițiile de funcționare ale circuitului.

Soluție

Automatul are o intrare, denumită X și o ieșire denumită Y .

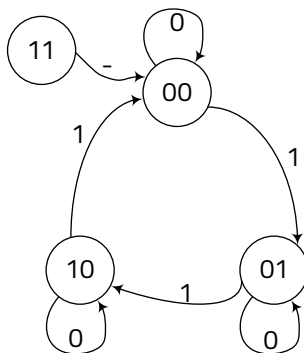
În cazul unei implementări ca automat Moore (ieșirea depinde doar de starea curentă), este nevoie de 9 stări: o stare de repaus, unde se așteaptă apariția pulsului de declanșare și 8 stări pe durata cărora ieșirea este activată 1. Numărul minim de biți pentru codificarea stărilor este 4.

Circuitul funcționează corect dacă între două pulsuri consecutive există o pauză de minimum 8 perioade de tact ($X = 0$). În cazul în care această condiție nu este îndeplinită, la ieșire se va mări lățimea pulsului cu 8 tacte după ultimul puls primit la intrare.

Se remarcă faptul că, în cazul acestei probleme, o implementare cu detector de front pozitiv (implementat cu un bistabil și o poartă cu două intrări) urmat de un circuit de deplasare realizat cu 7 bistabile D (care să întârzie pulsul) și apoi o poartă OR cu 8 intrări, ar fi fost o soluție mai eficientă și ca timp de proiectare și ca frecvență maximă de operare (figura 8.15).

Model Verilog pentru circuitul cerut de problema 5.

```
module pulse8 (
input  clk      , // semnal de ceas
input  rst_n    , // semnal de reset asincron activ 0
input  x        , // intrare
output y        , // iesire
);
```



a)

Figura 8.12 Graful de tranziții al automatului analizat la problema 3. Se observă tranziția necondiționată din starea 11 în starea 00 și ciclul de 3 stări 00, 01, 10, 00, ...

```

reg          x_del ;
reg [7 -1:0] del_7 ;
wire         x_fall;

always @(posedge clk or negedge rst_n)
if (~rst_n) x_del <= 1'b0; else
    x_del <= x;

// detectare front pozitiv al intrarii x
assign x_fall = x & (~x_del);

```

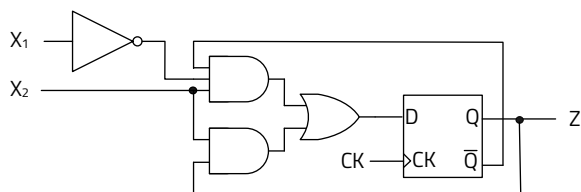


Figura 8.13 Automat de analizat la problema 4.

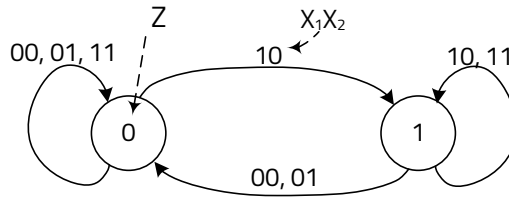


Figura 8.14 Graful de tranziții pentru automatul de la problema 4.

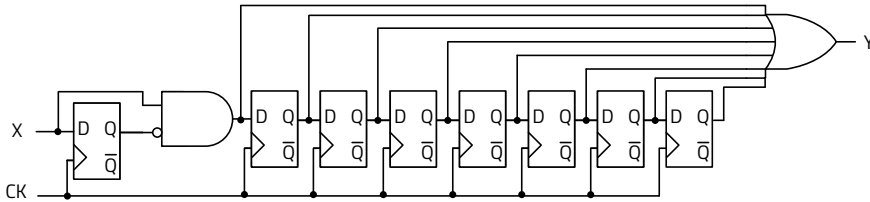


Figura 8.15 Implementarea automatului de la problema 5 cu un circuit de deplasare realizat cu bistabile D.

```

always @(posedge clk or negedge rst_n)
if (~rst_n) del_7 <= 7'b0; else
    del_7 <= {del_7[5:0], x_fall} ; // intarziere 7 tacte

assign y = |{x_fall, del_7}; // OR pe 8 biti

endmodule // pulse8
  
```

Lecția 9

Sinteza circuitelor logice secvențiale

„Sinteza” desemnează problema clasică de „proiectare”. Adică realizarea unui circuit electronic care să îndeplinească o funcție specificată. Și cum nu există sistem digital fără logică secvențială, este evident că orice proiectant de sisteme digitale se ocupă zilnic de „sinteza circuitelor logice secvențiale”.

9.1 Pentru seminar

1. Un automat cu 3 stări și două intrări funcționează conform grafului de tranziții prezentat în figura 9.1, (liniuta „-” desemnează o intrare indiferentă). Implementați automatul cu porți logice NAND și bistabile D, cu codificarea stărilor cu număr minim de biți.

Soluție

Pentru codificarea cu număr minim de biți, pentru cele 3 stări se folosesc 2 biți (Q_1, Q_0).

Se propune codificarea: $A = 00$, $B = 01$, $C = 10$.

Tabelul de tranziții este:

Intrări		Stare prezentă			Stare viitoare		
X	Y	Nume	Q_1	Q_0	Nume	D_1	D_0
0	-	A	0	0	A	0	0
1	-	A	0	0	B	0	1
-	1	B	0	1	C	1	0
0	0	B	0	1	A	0	0
1	0	B	0	1	B	0	1
-	0	C	1	0	A	0	0
-	1	C	1	0	C	1	0

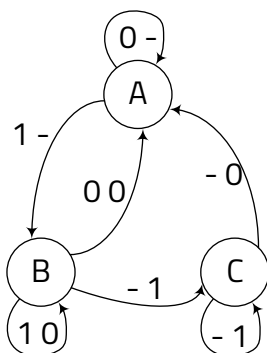


Figura 9.1 Graf de tranziție pentru problema 1.

Pe baza acestui tabel de tranziții se construiesc diagramele V-K prezentate în figura 9.2.

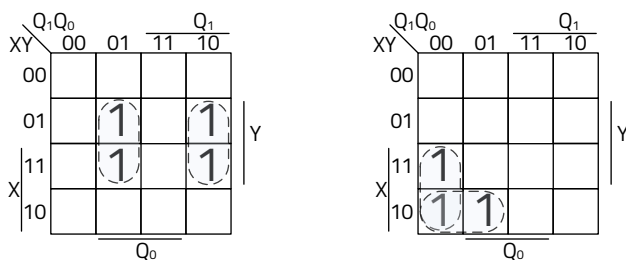


Figura 9.2 Diagrame V-K pentru problema 1.

După minimizare, se obțin funcțiile:

$$D_1 = Y \cdot \overline{Q_1} \cdot Q_0 + Y \cdot Q_1 \cdot \overline{Q_0}$$

$$D_0 = X \cdot \overline{Y} \cdot \overline{Q_1} + X \cdot \overline{Q_1} \cdot Q_0$$

- Proiectați un automat implementat cu bistabile D care are o intrare de validare EN și următorul comportament:

Dacă $EN = 1$, automatul ciclează între stările 00, 10, 01, 11, 00, ...
 Dacă $EN = 0$ automatul își păstrează starea.

3. Proiectați un automat implementat cu bistabile D, care numără în secvența binară: 0, 2, 3, 6, 7, 0, ...

Soluție

Automatul are 5 stări. Sunt necesari 3 biți pentru a codifica stările ($2^3 \geq 5$). Automatul nu are intrări. Se codifică stările astfel încât ieșirile să provină de la biții de stare, fără logică adițională. Fiecărui bit de stare i se va asocia un bistabil D. Tabelul de tranziții al automatului este:

Stare prezentă				Stare viitoare			
Nume	Q_2	Q_1	Q_0	Nume	Q_2^+	Q_1^+	Q_0^+
0	0	0	0	2	0	1	0
2	0	1	0	3	0	1	1
3	0	1	1	6	1	1	0
6	1	1	0	7	1	1	1
7	1	1	1	0	0	0	0
1	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0
5	1	0	1	0	0	0	0

9.2 Pentru cei ce vor doar să promoveze examenul

1. Implementați cu bistabile D și un număr minim de porți NAND automatele secvențiale sincrone cu funcționarea descrisă prin orgnigramele prezentate în figura 9.3.

9.3 Pentru cei ce vor să învețe

1. Implementați cu bistabile D și un număr minim de porți NAND automatele secvențiale sincrone având funcționarea descrisă prin orgnigramele prezentate în figura 9.4.
2. Implementați cu bistabile T și un număr minim de porți NAND automatul secvențial sincron având funcționarea descrisă prin orgnigrama prezentată în figura 9.5.

Soluție

Circuitul are două intrări (A și B) și două ieșiri (X și Z). Este un automat Mealy (deoarece conține un simbol de ieșire imediată Z).

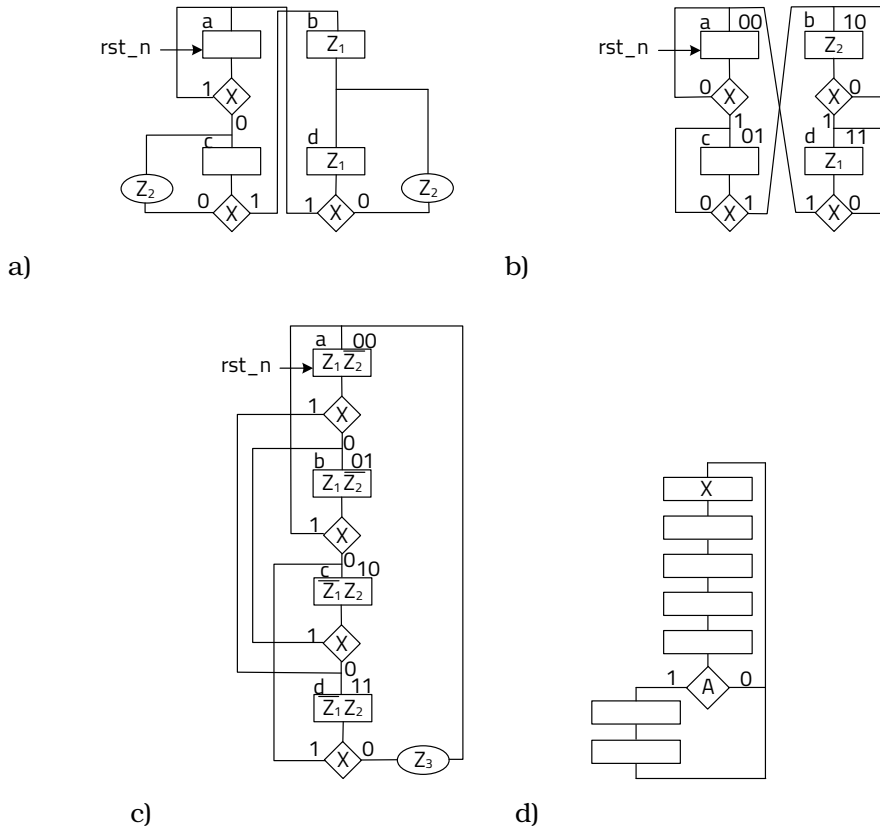
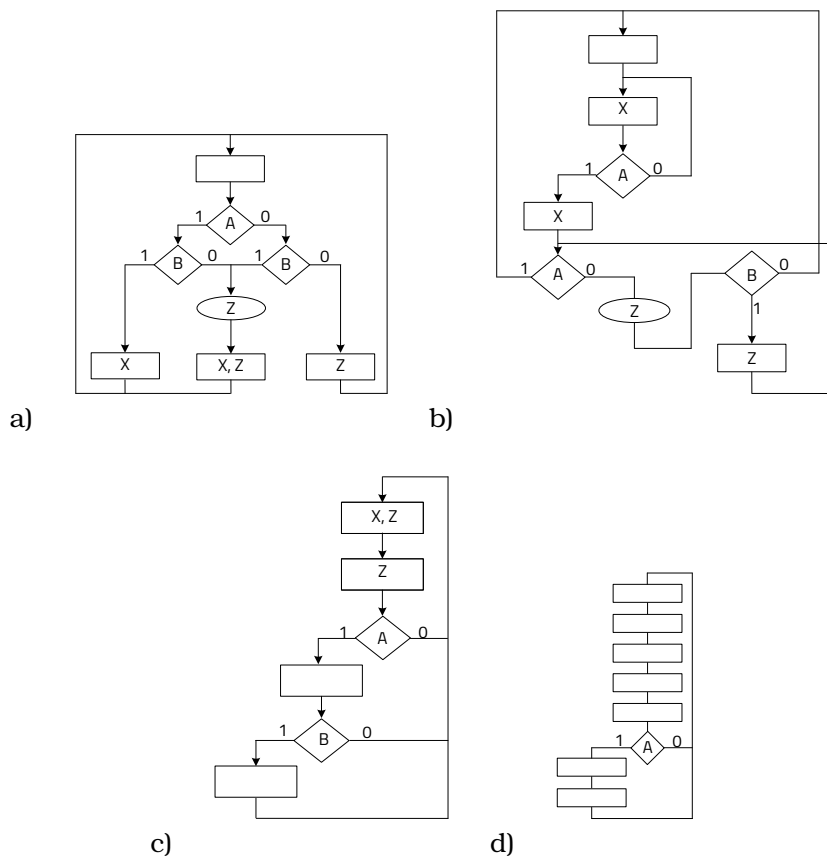


Figura 9.3 Organigrame pentru problema 1.

Cele 4 stări pot fi codate cu un număr minim de 2 biți. Tabelul de tranziții conține coloane pentru două intrări, starea curentă (2 biți), starea viitoare (2 biți) și două ieșiri. În plus, se atașează două coloane pentru stimulii bistabilelor T.

Pe baza organigramei, în prima etapă se completează coloana stării viitoare. Ulterior, se deduc coloanele aferente intrărilor bistabilelor, prin referire la comportamentul bistabilelor T descris de tabelul:

Q	Q^+	T	Acțiune
0	0	0	Păstrează starea
0	1	1	Complementează starea
1	0	1	Complementează starea
1	1	0	Păstrează starea

**Figura 9.4** Organigrame pentru problema 1.

Tabelul de tranziții este:

Linia	Intrări		St. prez.		St. viit.		Ieșiri		Stimuli T	
	A	B	Q_1	Q_0	Q_1^+	Q_0^+	X	Z	T_1	T_0
1	0	-	0	0	1	0	0	0	1	0
2	1	-	0	0	0	1	0	0	0	1
3	-	0	0	1	1	1	1	0	1	0
4	-	1	0	1	0	0	1	0	0	1
5	-	0	1	0	1	0	0	1	0	0
6	-	1	1	0	1	1	0	1	0	1
7	0	-	1	1	1	0	1	1	0	1
8	1	-	1	1	0	0	1	0	1	1

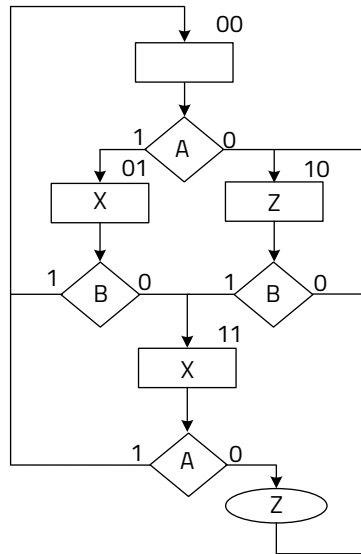


Figura 9.5 Organigramă pentru problema 2.

Analogia dintre liniile din tabelul de tranziții și căsuțele din diagrama V-K este prezentată în figura 9.6, alături de diagrame V-K asociate funcțiilor T_1, T_0, X, Z dependente de (A, B, Q_1, Q_0) .

După minimizare, rezultă ecuațiile:

$$T_1 = Q_1 \cdot Q_0 \cdot A + Q_0 \cdot A \cdot \overline{B} + \overline{Q_1} \cdot \overline{A} \cdot \overline{B} + \overline{Q_1} \cdot \overline{Q_0} \cdot \overline{A}$$

$$T_0 = Q_1 \cdot B + Q_1 \cdot Q_0 + Q_0 \cdot B + \overline{Q_1} \cdot \overline{Q_0} \cdot A$$

$$X = Q_0$$

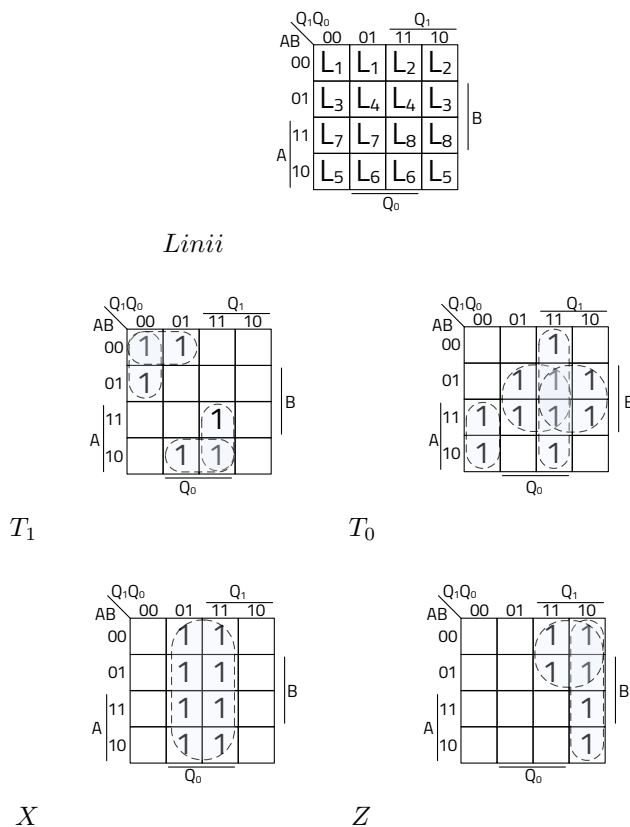
$$Z = Q_1 \cdot \overline{A} + Q_1 \cdot \overline{Q_0}$$

Implementarea automatului este prezentată în figura 9.7.

- Proiectați automatele descrise de grafurile de tranziții din figura 9.8 cu bistabile D și un număr minim de porți logice. Cu „X” a fost marcată o valoare indiferentă a intrării.

Pentru graful de tranziții **a)**, determinați secvența stărilor și valorile ieșirilor în cazul aplicării pe intrare a secvenței:

11001100110101010111100001111110000

**Figura 9.6** Diagrame V-K, pentru problema 2.

Presupuneți starea inițială 00.

Verificați răspunsul prin simularea unui model HDL.

4. Proiectați cu bistabile D automate care numără în secvențele binare:

- a) 0, 1, 3, 4, 5, 6
- b) 0, 6, 4, 3, 2
- c) 0, 7, 6, 1, 3, 4
- d) 0, 4, 5, 6, 1, 2, 3, 7
- e) 0, 1, 2, 3, 5, 6

Propuneți circuite pentru initializare în starea 0 și revenire în această stare din stările ce nu aparțin ciclurilor.

5. Să se implementeze automatele care funcționează conform grafurilor de tranziție prezentate în figura 9.9 cu bistabile D. Proiectați aceleași

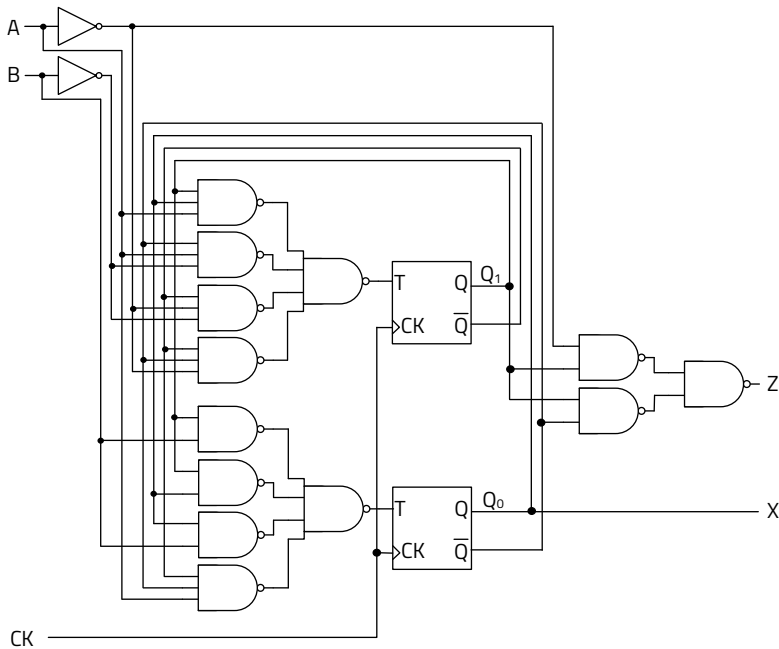


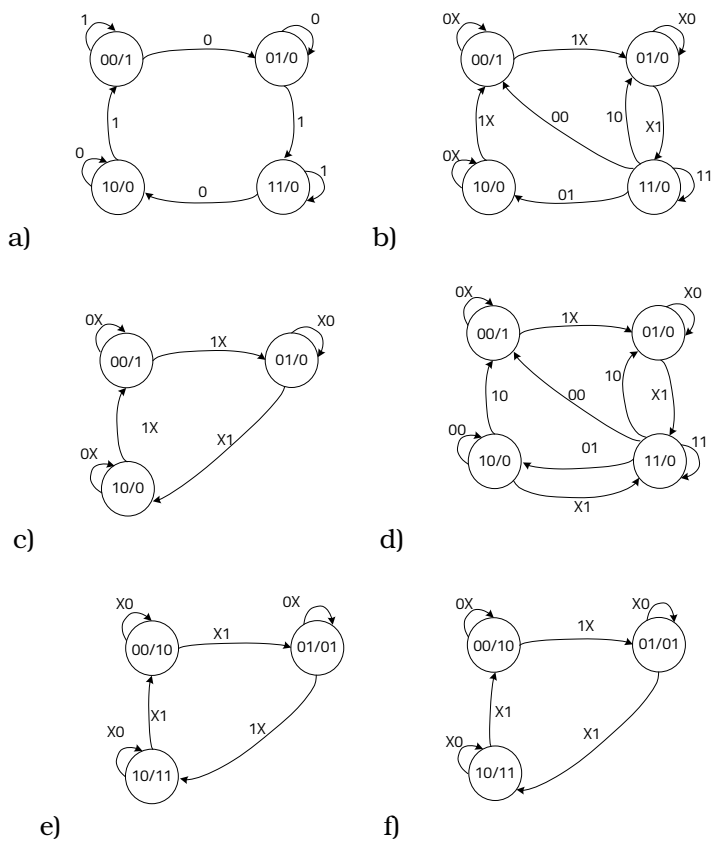
Figura 9.7 Implementarea automatului cu bistabile T, problema 2.

automate, implementate cu bistabile T.

Soluție

a) Automatul este de tip Mealy, are o intrare, o ieșire și 4 stări codate cu 2 biți. Dacă intrarea este egală cu 1, automatul se comportă ca un numărător modulo 4 în sens crescător. Dacă intrarea este egală cu 0, automatul se comportă ca un numărător modulo 4 în sens descrescător. Ieșirea semnalizează depășirea la numărare (tranziție din 11 în 00 în sens crescător, sau din 00 în 11 în sens descrescător).

Tabelul de tranziții pentru implementarea cu bistabile D este:

**Figura 9.8** Grafuri de tranziții pentru problema 3.

Intrare U_p	Stare curentă Q_1 Q_0		Stare viitoare D_1 D_0		Ieșire Y
0	0	0	1	1	1
1	0	0	0	1	0
0	0	1	0	0	0
1	0	1	1	0	0
0	1	0	0	1	0
1	1	0	1	1	0
0	1	1	1	0	0
1	1	1	0	0	1

Rezultă ecuațiile:

$$D_1 = \overline{U_p} \oplus Q_1 \oplus Q_0$$

$$D_0 = \overline{Q_0}$$

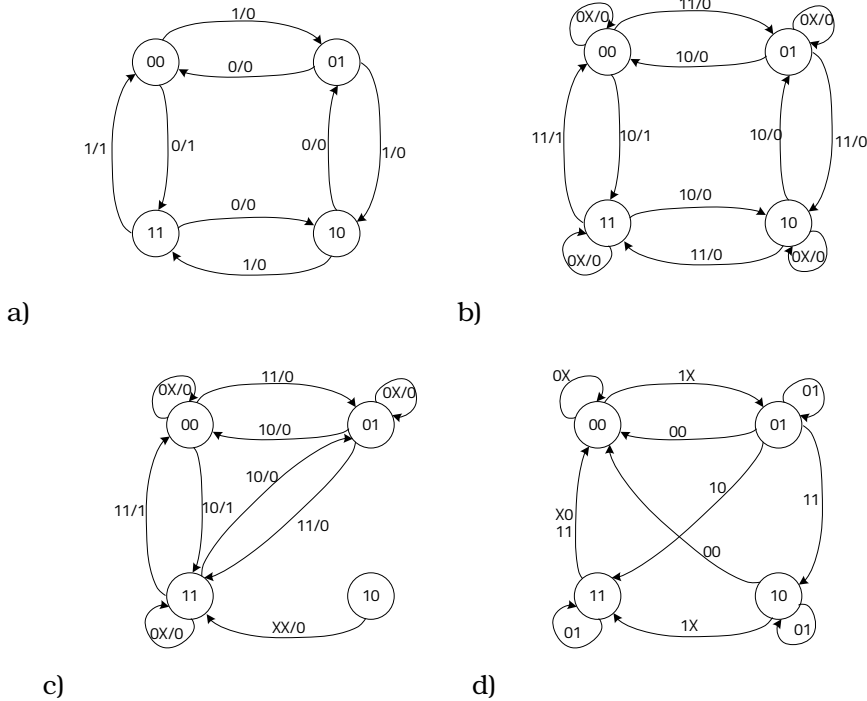


Figura 9.9 Grafuri de tranziții, problema 5.

$$Y = Up \cdot Q_1 \cdot Q_0 + \overline{Up} + Q_1 + Q_0$$

b) Automatul este de tip Mealy, are două intrări, o ieșire și 4 stări codate cu 2 biți. Prima intrare este prioritară și are o funcție de validare a comutării (dacă această intrare este egală cu 0, automatul își păstrează starea). A doua intrare, este relevantă doar când intrarea de validare este egală cu 1 și determină sensul de numărare pentru numărătorul modulo 4 (în sens crescător sau în sens descrescător). Ieșirea semnalizează depășirea la numărare (tranziție din 11 în 00 în sens crescător, sau din 00 în 11 în sens descrescător). Tabelul de tranziții pentru implementarea cu bistabile T este:

Intrări		Stare curentă		Stare viitoare		Stimuli bistabile		Ieșire
En	Up	Q_1	Q_0	Q_1^+	Q_0^+	T_1	T_0	Y
0	-	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1
1	1	0	0	0	1	0	1	0
0	-	0	1	0	1	0	0	0
1	0	0	1	0	0	0	1	0
1	1	0	1	1	0	1	1	0
0	-	1	0	1	0	0	0	0
1	0	1	0	0	1	1	1	0
1	1	1	0	1	1	0	1	0
0	-	1	1	1	1	0	0	0
1	0	1	1	1	0	0	1	0
1	1	1	1	0	0	1	1	1

Rezultă ecuațiile:

$$T_1 = En \cdot \overline{Up \oplus Q_0}$$

$$T_0 = En$$

$$Y = En \cdot (Up \cdot Q_1 \cdot Q_0 + \overline{Up} \cdot \overline{Q_1} \cdot \overline{Q_0})$$

9.4 Pentru cei ce vor să devină profesioniști

1. Proiectați cu bistabile D un numărător în cod BCD. Propuneți un model Verilog pentru acesta.
2. Proiectați cu bistabile D un numărător în cod Gray pe 2, 3 și 4 biți.
3. Proiectați un numărător de 4 biți în secvența 0, 1, 2, 3, 2, 4, 5, 6, 7. Se observă că starea 2 are două apariții în cadrul acestui ciclu. La ieșire se prezintă codul stării.

Soluție

Deși automatul are 8 seturi distincte de ieșiri (posibil de codat cu 3 biți) automatul are 9 stări diferite. Există două stările în care ieșirea este egală cu 2: în prima se ajunge în succesiunea 1, 2, 3, în cealaltă se ajunge în succesiunea 3, 2, 4. Se notează cele 3 ieșiri: O_2, O_1, O_0 .

Automatul are 9 stări codate cu un număr minim de 4 biți. Biții stării curente se numesc Q_3, Q_2, Q_1, Q_0 . Automatul nu are intrări, grafal de tranziții al automatului fiind sub forma unui inel de 9 stări cu tranziții necondiționate dintr-o stare în alta.

Pentru a minimiza circuitul combinațional al ieșirilor se pot codifica stările astfel încât 3 biți de stare să coincidă cu biții de ieșire.

Ideea este ca stările în care ieșirea este 2 să difere printr-un singur bit, iar cei mai puțin semnificativi biți să fie 010.

Deci, se vor codifica stările cu ieșirea egală cu 2 astfel: $S_2 = 0010$ și $S_{10} = 1010$.

Codificarea este prezentată în tabelul de tranziții:

Stare prezentă					Stare viitoare					Ieșiri			
	Q_3	Q_2	Q_1	Q_0		Q_3^+	Q_2^+	Q_1^+	Q_0^+		O_2	O_1	O_0
S_0	0	0	0	0	S_1	0	0	0	1	0	0	0	0
S_1	0	0	0	1	S_2	0	0	1	0	1	0	0	1
S_2	0	0	1	0	S_3	0	0	1	1	2	0	1	0
S_3	0	0	1	1	S_{10}	1	0	1	0	3	0	1	1
S_{10}	1	0	1	0	S_4	0	1	0	0	2	0	1	0
S_4	0	1	0	0	S_5	0	1	0	1	4	1	0	0
S_5	0	1	0	1	S_6	0	1	1	0	5	1	0	1
S_6	0	1	1	0	S_7	0	1	1	1	6	1	1	0
S_7	0	1	1	1	S_0	0	0	0	1	7	1	1	1

4. Să se implementeze un automat a cărui funcționare este descrisă de formele de undă ale ieșirilor, prezentate în figura 9.10.

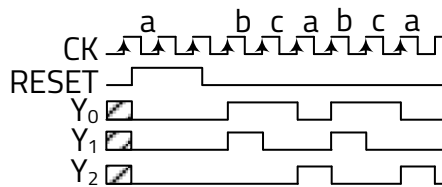


Figura 9.10 Forme de undă, pentru problema 4.

Soluție

Se observă că automatul prezintă 3 stări (a, b, c), nu are intrări și se resetează în starea a. În cazul codificării stărilor cu 2 biți (Q_1, Q_0): $a = 00$, $b = 01$, $c = 10$ rezultă tabelul de tranziții:

Stare prezentă			Stare viitoare			Ieșiri		
Nume	Q_1	Q_0	Nume	Q_1^+	Q_0^+	Y_2	Y_1	Y_0
a	0	0	b	0	1	1	0	0
b	0	1	c	1	0	0	1	1
c	1	0	a	0	0	0	0	1
-	1	1	-	-	-	-	-	-

În tabel s-a inclus și starea neprezentată în formele de undă (11), din care s-a proiectat o tranziție într-o stare indiferentă și ieșiri indiferente.

Rezultă ecuațiile stării viitoare și ale ieșirilor:

$$D_1 = Q_0$$

$$D_0 = \overline{Q_1} \cdot \overline{Q_0}$$

$$Y_2 = \overline{Q_1} \cdot \overline{Q_0} = D_0$$

$$Y_1 = Q_0 = D_1$$

$$Y_0 = Q_1 + Q_0$$

Starea care urmează stării prezente 11 este $c = 10$, iar ieșirile în această stare sunt $Y_{2,1,0} = 011$.

Structura automatului implementat cu bistabile D este prezentată în figura 9.11.

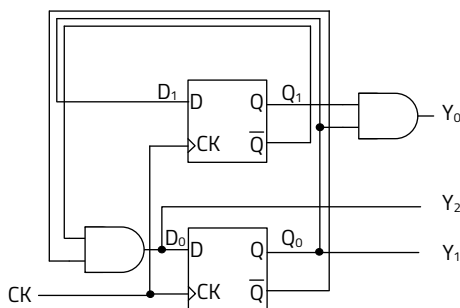


Figura 9.11 Automatul pentru problema 4.

Lecția 10

Modelarea automatelor în Verilog

Problemele din această lecție **necesită utilizarea unui laptop** și a unui simulator de limbaje de descriere hardware. Modulele cu rol exclusiv în simulare (generator de semnal de ceas, de reset, generator de stimuli determiniști sau aleatorii) pot fi reutilizate pentru toate problemele.

Pentru toate simulările este nevoie de un generator de semnal de ceas. În principiu, valoarea perioadei de ceas nu este relevantă în cadrul acestor simulări. Totuși, modelul propus are perioada de ceas parametrizabilă, în unități de timp. Semnalul de reset este activ pe 0. Chiar dacă modelează un semnal asincron, ca o bună practică, se recomandă inactivarea acestuia (trecere din 0 în 1) sincron cu semnalul de ceas.

Un modul care generează semnale de ceas și de reset este prezentat în continuare.

```
module ck_rst_tb #(
    parameter CK_SEMIPERIOD = 'd10 // semi-perioada ceasului
) (
    output reg clk , // ceas
    output reg rst_n // reset asincron activ 0
);
initial
begin
    clk <= 1'b0; // valoare initiala 0
    forever #CK_SEMIPERIOD // valoare complementata
        // la fiecare semi-perioada
        clk <= ~clk;
end

initial begin
    rst_n <= 1'b1; // initial inactiv
    @(posedge clk);
    rst_n <= 1'b0; // activare sincrona
```

```

@ (posedge clk);
@ (posedge clk);
rst_n <= 1'b1;    // inactivare dupa doua perioade de ceas
@ (posedge clk);  // ramane inactiv pentru totdeauna
end

endmodule // ck_rst_tb

```

Pentru o simulare realistă, resetul asincron ar trebui să fie inițial inactiv (1), ulterior activat (0) și inactivat sincron cu frontul pozitiv de ceas.

Instanțierea modulului generator de ceas și de reset se face în mediul de testare. La instanțiere se poate suprascrie valoarea parametrului asociat perioadei de ceas.

```

ck_rst_tb #(
    .CK_SEMIPERIOD ('d10)
) i_ck_rst_tb (
    .clk      (clk      ),
    .rst_n    (rst_n    )
);

```

Mediul de testare este similar celui prezentat în figura 10.1. Pe lângă instanțierea modulului dorit a fi testat, a modulului generator de semnale de ceas și de reset există și specificații care modelează variația stimulilor de intrare. Intrările modulului de testat se conectează la generatorul de stimuli prin intermediul unor semnale interne mediului de testare. Ieșirile modulului de testat se pot conecta la semnale interne mediului de testare sau pot fi lăsate în gol.

Generatorul de stimuli poate fi realizat simplificat cu specificații *initial* sau se poate descrie mai complex cu specificații încapsulate într-un modul ale cărui ieșiri corespund intrărilor modulului de testat. Se recomandă, ca o bună practică, a se sincroniza semnalele generatorului cu semnalul de ceas. Generatorul de stimuli primește aceleași semnale de ceas și de reset ca și modulul de testat. Semnalele generate pot fi sincronizate dacă în cod se implementează întârzierile cu multiplii ai perioadei de ceas.

Evenimentul „*front activ de ceas*” se poate descrie cu specificația `@ (posedge clk)`, adică „*așteaptă momentul următorului front pozitiv al semnalului de ceas*”.

Semnalele generate ar trebui să fie inițial într-o stare necunoscută 'bx și comutate într-o stare cunoscută (0 sau 1) simultan cu activarea semnalului de reset. Ulterior, modificarea semnalelor de ieșire ar trebui realizată sincron cu ceasul. Se pot utiliza specificații pentru generare de stimuli repetitivi (`repeat`) determinați sau aleatorii (`$random` și `$urandom_range`).

Se recomandă ca, la finalul stimulilor, să se plaseze o specificație de

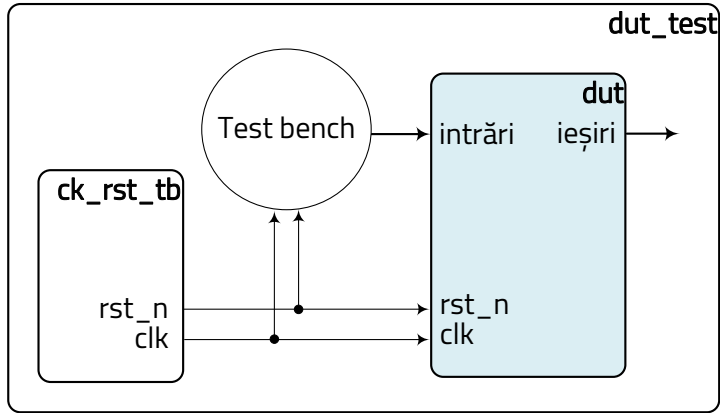


Figura 10.1 Mediu de testare: dispozitiv de testat (*Device Under Test*, engl.), generator de ceas și reset (*ck_rst_tb*), generator de stimuli.

finalizare a simulării (\$stop sau \$finish) însoțită de un mesaj (\$display) pentru a putea rula integral simularea cu comenzi specifice simulatorului (run -all).

```
initial begin
    a <= 1'bx;    // initial stare necunoscuta
    @(negedge rst_n);
    a <= 1'b0;    // reset asincron activat
    @(posedge rst_n);
    repeat (5) @(posedge clk);

    repeat(50) begin
        a <= $random;
        @(posedge clk);
    end

    $display ("%M %0t INFO: Final simulare.", $time);
    $stop;
end
```

10.1 Pentru seminar

1. Scrieți un model Verilog pentru automatul descris prin organigrama prezentată în figura 10.2. Realizați un mediu de simulare care să vă

prezintă traiectoria automatului în cazul în care intrarea este constantă (1 sau 0).

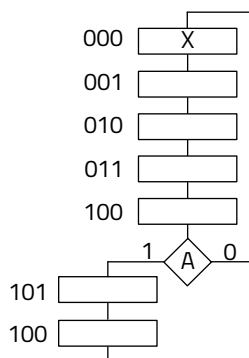


Figura 10.2 Organigramă pentru problema 1.

2. Scrieți un model Verilog pentru automatul descris prin graful de tranziții prezentat în figura 10.3. Realizați un mediu de simulare care să vă prezinte traiectoria automatului în cazul în care intrarea este constantă (1 sau 0).

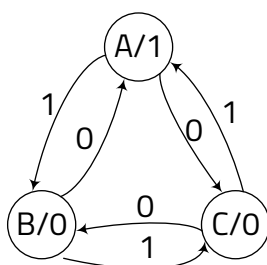
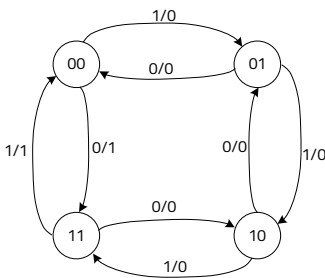


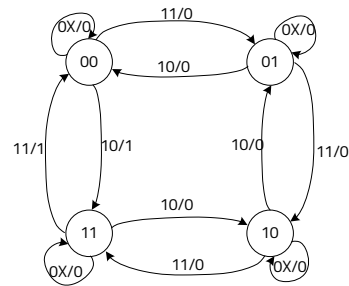
Figura 10.3 Graf de tranziții pentru problema 2.

10.2 Pentru cei ce vor doar să promoveze examenul

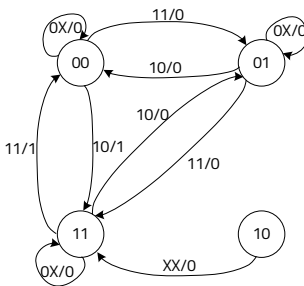
1. Să se modeleze în Verilog automatele care funcționează conform grafurilor de tranziție prezentate în figura 10.4.



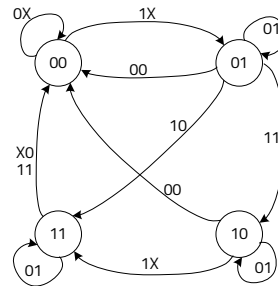
a)



b)



c)



d)

Figura 10.4 Grafuri de tranziții, problema 1.

2. Să se modeleze în Verilog automatele care funcționează conform orgnigramelor prezentate în figura 10.5.

10.3 Pentru cei ce vor să învețe

1. Modelați în Verilog automate care numără în secvențele binare:

a) 0, 1, 3, 4, 5, 6

b) 0, 6, 4, 3, 2

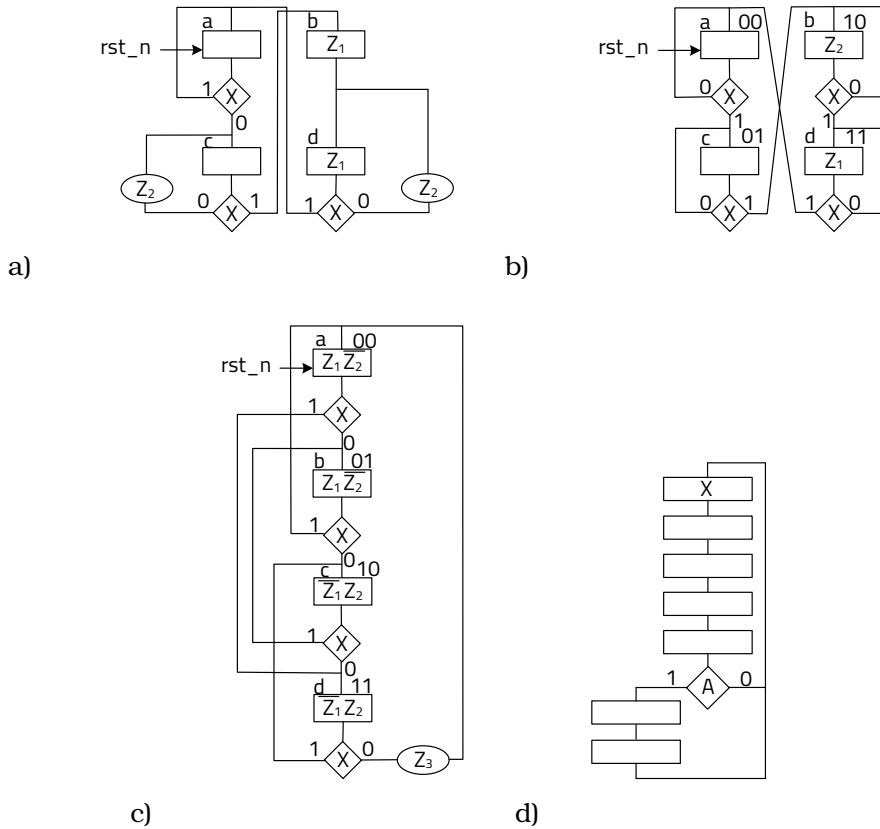


Figura 10.5 Organigrame, problema 2.

c) 0, 7, 6, 1, 3, 4

d) 0, 4, 5, 6, 1, 2, 3, 7

e) 0, 1, 2, 3, 5, 6

Automatul se resetează în starea 0 și revine în această stare din stările ce nu aparțin ciclurilor.

Soluție

a) Automatul poate fi modelat cu o singură specificație `always` care să specifice starea următoare pe baza stării curente, printr-o specificație `case`. Codul este:

```
module fsm_013456(
input          clk      , // semnal de ceas
input          rst_n    , // semnal de reset asincron activ 0
output reg [3 -1:0] state // iesire/stare
```

```

);

always @(posedge clk or negedge rst_n)
if (~rst_n) state <= 3'd0; else // stare initiala a FSM
case (state)
3'd0 : state <= 3'd1; // 0 -> 1
3'd1 : state <= 3'd3; // 1 -> 3
3'd3 : state <= 3'd4; // 3 -> 4
3'd4 : state <= 3'd5; // 4 -> 5
3'd5 : state <= 3'd6; // 5 -> 6
3'd6 : state <= 3'd0; // 6 -> 0
default : state <= 3'd0; // alte stari -> 0
endcase

endmodule // fsm_013456

```

Prin simulare, se obțin formele de undă care confirmă succesiunea de stări.

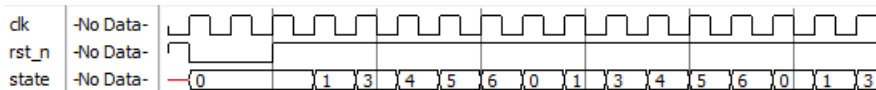


Figura 10.6 Forme de undă ale automatului a), cu secvența 0, 1, 3, 4, 5, 6.

10.4 Pentru cei ce vor să devină profesioniști

1. Proiectați și modelați în Verilog un automat care generează secvența 0, 1, 2, 3, 2, 4, 5, 7. Ulterior, adăugați automatului o intrare care determină inversarea sensului ciclului.

Soluție

Se observă că ciclul are 8 valori, posibil de codat cu 3 biți, însă valoarea „2” are două apariții în cadrul acestui ciclu. Este necesar ca cele două stări asociate cu „2” să aibă coduri diferite. Ulterior, circuitul combinațional de ieșire, va asocia aceleași valori ale ieșirilor pentru ambele stări denumite „2”. A doua stare „2” este asociată cu un cod care nu apare în ciclu: „6”.

Codul Verilog este următorul. Se observă că ieșirea are valoarea stării, cu excepția stării 6, caz în care ieșirea are valoarea 2.

```

module fsm_01232457(
input          clk , // semnal de ceas
input          rst_n , // semnal de reset asincron activ 0
output reg [3 -1:0] iesire // iesire
);
reg [3 -1:0] state ; // stare

always @(posedge clk or negedge rst_n)
if (~rst_n) state <= 3'd0; else // stare initiala a FSM
case (state)
3'd0 : state <= 3'd1; // 0 -> 1
3'd1 : state <= 3'd2; // 1 -> 2
3'd2 : state <= 3'd3; // 2 -> 3
3'd3 : state <= 3'd6; // 3 -> 6
3'd6 : state <= 3'd4; // 6 -> 4   starea "2"
3'd4 : state <= 3'd5; // 4 -> 5
3'd5 : state <= 3'd7; // 5 -> 7
default : state <= 3'd0; // 7 -> 0
endcase

// iesire
assign iesire = (state == 3'd6) ? 3'd2 : // starea 6 = "2"
               state;

endmodule // fsm_01232457

```

Formele de undă obținute prin simulare sunt prezentate în figura 10.7.



Figura 10.7 Forme de undă ale automatului, problema 1. Se observă ieșirea egală cu 2 în stările „2” și „6”.

- Proiectați și modelați în Verilog un automat care prezintă comportamentul din graful de tranziții din figura 10.8.

Soluție

Automatul are 4 stări, denumite generic *idle*, *initial*, *iterate* și *final*, posibil de codificat pe un număr minim de 2 biți ($2^2 \leq 4$).

Automatul are două intrări, denumite *start* și *zero*.

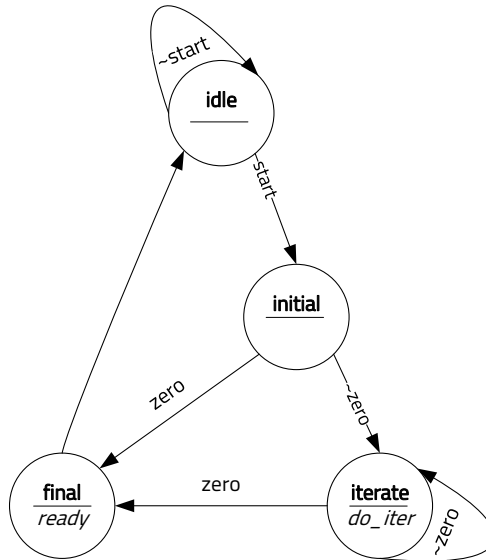


Figura 10.8 Graful de tranziții al automatului menționat la problema 2.

Ieșirile automatului, denumite *ready* și *do_iter*, depind exclusiv de starea prezentă (numele lor apar în simbolurile de stare, noduri, sub numele stării în care este activă). Este un automat de tip Moore. Arcul dintre starea *final* și starea *idle* nu are nicio condiție și desemnează o tranziție în care intrările sunt indiferente.

Automatul are structura prezentată în figura 10.9. Poate fi modelat în una din cele două variante prezentate în figură. Se observă că ieșirile sunt active doar în câte o stare. Ar putea fi modelate cu o specificație *assign* care pune condiția că automatul se află în starea potrivită.

Un tip de modelare acceptat de toate sintetizatoarele presupune modelarea semi-automatului cu o singură specificație secvențială care cuprinde atât modelul pentru registrul de stare cât și circuitul combinațional care implementează funcția de tranziție a stării.

Se observă inițializarea automatului în starea *idle* în cazul activării resetului asincron *rst_n=0*. Circuitul combinațional este modelat cu o specificație *case* cu selecție pe baza stării curente. Intrările automatului influențează stările viitoare.

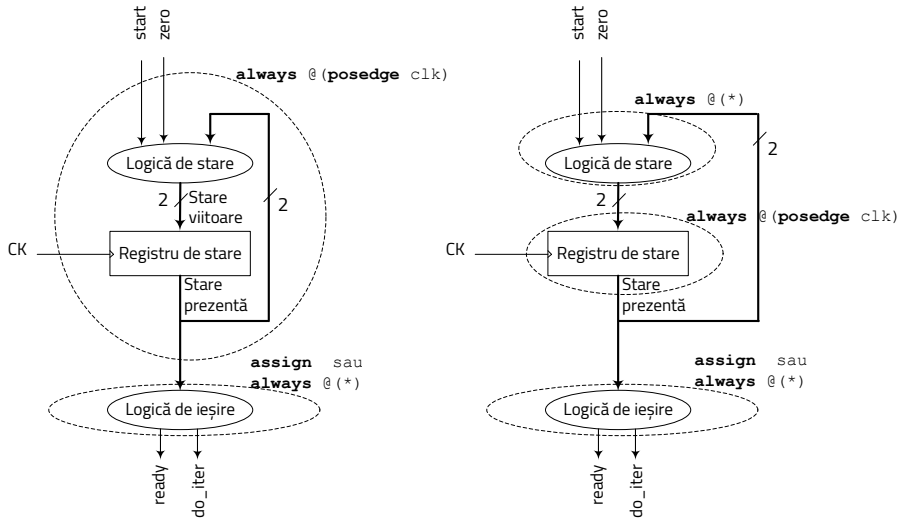


Figura 10.9 Structura automatului menționat la problema 2.

Ieșirile sunt modelate cu specificații pentru modelarea circuitelor combinaționale, `assign` sau `always@(*)`.

Codul asociat modelării cu o singură specificație `always` este următorul:

```
module fsm_1 (
input    clk      , // semnal de ceas
input    rst_n    , // semnal de reset asincron activ 0
input    start    , // start operatie
input    zero     , // indicator de stare
output   do_iter  , // comanda pentru o iteratie
output   ready    , // terminare operatie
);
reg [1:0] stare   ; // registru de stare

// coduri stari (unice)
localparam s_idle     = 2'b00;
localparam s_initial  = 2'b01;
localparam s_iterate  = 2'b10;
localparam s_final    = 2'b11;

// modeleaza atat circuitul combinational de stare
// cat si registrul de stare
always @(posedge clk or negedge rst_n)
```

```

if (~rst_n) stare <= s_idle; else // stare initiala a FSM
case (stare)
s_idle : if (start) stare <= s_initial;
s_initial : if (zero) stare <= s_final; else
            stare <= s_iterate;
s_iterate : stare <= zero ? s_final : s_iterate;
default : stare <= s_idle; // s_final
endcase

// modeleaza circuitul combinational de iesire
assign ready = (stare == s_final );
assign do_iter = (stare == s_iterate);

endmodule // fsm_1

```

Se observă că este definit un singur semnal pentru stare:

```
reg [1:0] stare ; // registru de stare
```

Se observă codificarea explicită a stărilor, cu un număr minim de biți, realizată cu specificații `localparam`. În acest mod, codurile stărilor nu sunt expuse interfeței exterioare a modului și nu se pot modifica la instanțierea modului.

Un alt tip de modelare acceptat de toate sintetizatoarele presupune modelarea semi-automatului cu două specificații `always`.

O specificație `always @(posedge clk)` secvențială modelează registrul de stare, ca un registru de tip D.

O altă specificație `always @(*)` modelează circuitul combinational care implementează funcția de tranziție a stării, așa ca în figura 10.9.

Codul asociat modelării cu două specificații `always` este următorul:

```

module fsm_2 (
input    clk      , // semnal de ceas
input    rst_n    , // semnal de reset asincron activ 0
input    start    , // start operatie
input    zero     , // indicator de stare
output   do_iter  , // comanda pentru o iteratie
output   ready    , // terminare operatie
);

reg [1:0] stare_prezenta; // registru de stare
reg [1:0] stare_viitoare; // din logica, modelata cu always

// coduri stari (unice)
localparam s_idle     = 2'b00;
localparam s_initial  = 2'b11;
localparam s_iterate  = 2'b10;
localparam s_final    = 2'b01;

```

```

// modeleaza registrul de stare
always @(posedge clk or negedge rst_n)
if (~rst_n) stare_prezentă <= s_idle; else // stare initiala
    stare_prezentă <= stare_viitoare;

// modeleaza circuitul combinational de stare
always @(*)
case (stare_prezentă)
s_idle      : if (start) stare_viitoare <= s_initial; else
                stare_viitoare <= s_idle;
s_initial   : if (zero)  stare_viitoare <= s_final; else
                stare_viitoare <= s_iterate;
s_iterate   : stare_viitoare <= zero ? s_final : s_iterate;
default     : stare_viitoare <= s_idle; // s_final
endcase

// modeleaza circuitul combinational de iesire
assign ready    = (stare_prezentă == s_final );
assign do_iter  = (stare_prezentă == s_iterate);

endmodule // fsm_2

```

Se observă că, în acest caz, sunt definite explicit semnale pentru starea viitoare (atribuite din specificația `always` combinațională) și pentru starea prezentă (atribuite din specificația `always` secvențială):

```

reg [1:0] stare_prezentă; // registru de stare
reg [1:0] stare_viitoare; // din logica, modelata cu always

```

- Proiectați și modelați în Verilog un automat care prezintă comportamentul din graful de tranziții din figura 10.10. Porniți de la automatul modelat la problema 2 și realizați modificările necesare.

Se presupune că în starea *initial* o singură intrare din cele 3 (*go_left*, *go_middle* și *go_right*) este activă.

Adică $go_left + go_middle + go_right = 1$, în acest context simbolul $+$ desemnează suma aritmetică.

Soluție

Automatul are 5 stări: *idle*, *initial*, *left*, *middle* și *right*.

Automatul are cinci intrări: *start*, *go_left*, *go_middle*, *go_right* și *zero*.

Automatul nu are ieșiri explicite, ci expune starea spre exterior.

Se remarcă faptul că anumite intrări au nume similare cu numele generice ale unor stări. Ieșirile sunt porturi pentru modul și trebuie păstrate ca în specificații. Numele generice ale stărilor nu apar ca

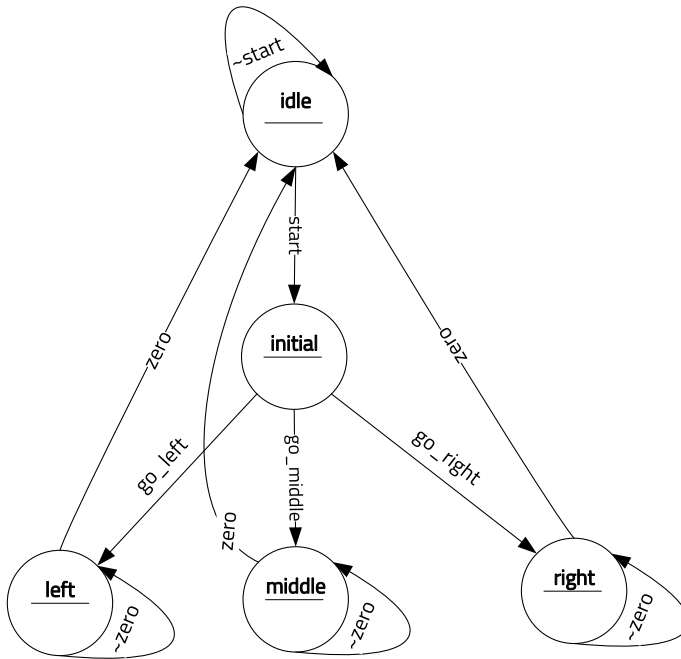


Figura 10.10 Graful de tranziții al automatului menționat la problema 3

porturi ale modulului și sunt alese de către proiectant. Se recomandă modelarea cu specificații `localparam` a unor coduri asociate cu nume ce sugerează numele generice ale stărilor.

- Proiectați și modelați în Verilog un automat care prezintă comportamentul din graful de tranziții din figura 10.11.

Soluție

Automatul are structura prezentată în figura 10.12. Se observă că ieșirile sunt active doar în câte o stare. Ar putea fi modelate cu o specificație `assign` care pune condiția că automatul se află în starea potrivită.

Automatul are 5 intrări (*start*, *final*, *jos*, *mijloc* și *sus*), însă nu toate sunt relevante în toate stările pentru a decide stările viitoare. Din acest motiv, o modelare recomandată este cu specificația `casex`. Specificația `casex` permite ca în expresia de selecție anumiți biți să fie declarați ca „indiferenți” (irrelevanți).

Porțiunea de cod următoare modelează interfața FSM și o parte din graful de tranziții din figura 10.11.

```

module fsm_casex (
input          clk      , // semnal de ceas
input          rst_n    , // semnal de reset asincron activ 0
// intrari
input          start    ,
input          sfarsit   ,
input          jos       ,
input          mijloc    ,
input          sus       ,

output reg [2:0] stare   // registru de stare si iesire
);

// coduri stari (unice)
localparam s_idle      = 3'b000;
localparam s_initial   = 3'b001;
localparam s_regular    = 3'b010;
localparam s_single_up  = 3'b011;
localparam s_single_down = 3'b100;

// modeleaza atat circuitul combinational de stare
// cat si registrul de stare
always @(posedge clk or negedge rst_n)
// stare initiala a FSM
if (~rst_n)          stare <= s_idle; else
casex ({start, jos, mijloc, sus, sfarsit, stare})
{5'b1_x_x_x_x, s_idle} : stare <= s_initial ;
{5'bx_x_1_x_x, s_initial} : stare <= s_regular ;
{5'bx_x_0_x_x, s_initial} : stare <= s_single_up ;
{5'bx_x_x_x_1, s_regular} : stare <= s_idle ; // sfarsit
{5'bx_x_1_x_0, s_regular} : stare <= s_regular ; // mijloc
{5'bx_x_0_1_0, s_regular} : stare <= s_single_up ; // sus
{5'bx_1_0_0_0, s_regular} : stare <= s_single_down; // jos
// . . .
default : stare <= s_idle ;
// {5'b00_x_x_x_x, s_idle}
endcase

endmodule // fsm_casex

```

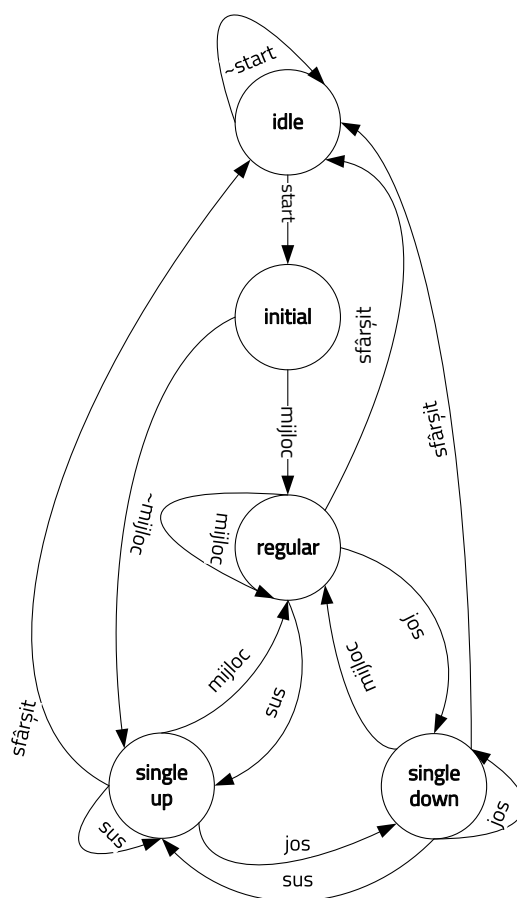


Figura 10.11 Graful de tranziții al automatului menționat la problema 4

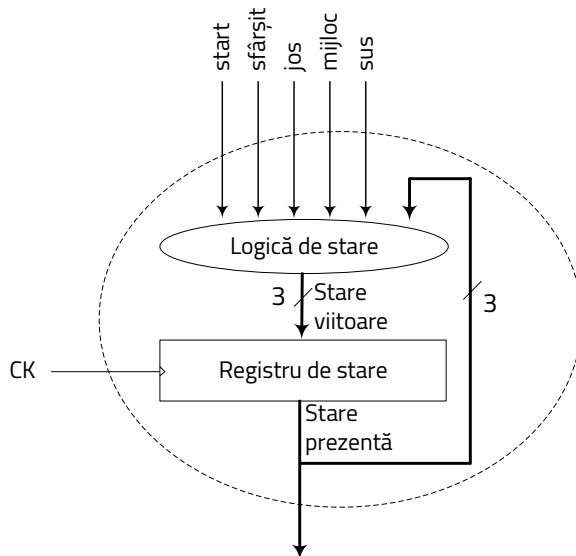


Figura 10.12 Structura automatului menționat la problema 4.

Lecția 11

Registre și numărătoare

Unele problemele din această lecție **necesită utilizarea unui laptop** și a unui simulator de limbaje de descriere hardware.

11.1 Pentru seminar

1. Un registru serial de 4 biți se află inițial în starea 1011 și primește pe intrarea serială secvența 1010111 (cel mai din stânga bit primul).

Precizați conținutul registrului după fiecare front activ de ceas, în cazul unei deplasări spre stânga.

Soluție

Starea viitoare a biților registrului este:

$$\begin{aligned}Q_3^+ &= Q_2 \\Q_2^+ &= Q_1 \\Q_1^+ &= Q_0 \\Q_0^+ &= In\end{aligned}$$

Conținutul registrului după fiecare front activ de ceas, în funcție de intrarea de date, este redat în tabelul următor. Se observă coloana In care se regăsește în coloana Q_0 cu un tact întârziere, și ulterior deplasată spre stânga.

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

2. Un registru serial de 4 biți se află inițial în starea 0001 și primește pe intrarea serială secvența 101110011 (cel mai din stânga bit primul).

Precizați conținutul registrului după fiecare front activ de ceas, în cazul unei deplasări spre dreapta.

3. Un registru de deplasare serială de 16 biți se află inițial în starea $F95B_{16}$ și primește pe intrarea serială secvența $C3_{16}$ (cel mai din stânga bit primul).

Care este starea registrului după 3 perioade de ceas, dacă deplasarea registrului se face spre stânga?

Dar dacă registrul se deplasează la dreapta?

Soluție

Deplasare la stânga: $CAD E_{16}$.

Deplasare la dreapta: $7F2B_{16}$.

4. Modelați în Verilog un numărător sincron între 3 și 14.

Adăugați o intrare pentru controlul sensului de numărare.

Adăugați o ieșire activă în starea 10.

Adăugați o intrare de validare a numărării (dacă intrarea nu este activă, numărătorul își păstrează starea).

5. Modelați în Verilog un registru care să funcționeze conform tabelului:

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

11.2 Pentru cei ce vor doar să promoveze examenul

1. Modelați în Verilog un numărător sincron de 4 biți.
2. Modelați în Verilog un numărător BCD (între 0 și 9).
3. Care este deosebirea de implementare a unui reset asincron față de un reset sincron?
4. Care este numărul maxim de stări într-un ciclu pentru un circuit LFSR de 10 biți?

11.3 Pentru cei ce vor să învețe

1. Completați diagramele temporale asociate circuitului prezentat în figura 11.1.

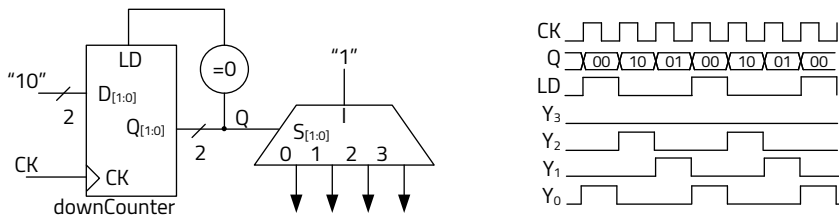


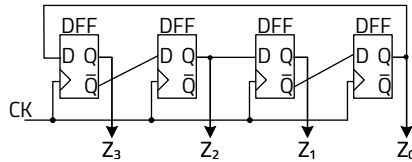
Figura 11.1 Circuit propus la problema 1. Forme de undă ale circuitului.

Soluție

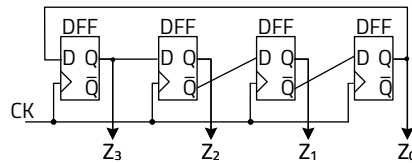
Se observă că numărătorul în sens descrescător se încarcă cu valoarea binară $10|_2$ de fiecare dată când ajunge în starea $00|_2$. Deci, secvența de numărare este: 00, 10, 01, 00, 10, 01, ...

Ieșirea numărătorului este conectată la selecția demultiplexorului, ceea ce înseamnă că vor fi activate ciclic ieșirile Y_2 , Y_1 și Y_0 . Formele de undă ale semnalelor sunt prezentate în figura 11.1.

2. Să se determine secvența de numărare a circuitelor prezentate în figura 11.2. Presupuneți starea inițială 0. Studiați cum evoluează circuitele pornind din toate stările.



a)



b)

Figura 11.2 Circuitele de numărare propuse la problema 2.

3. Modelați în Verilog un registru de 4 biți cu facilitățile prezentate în tabelul următor:

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

4. Modelați în Verilog structurile LFSR pe baza polinoamelor:

- a) $X^4 + X^3 + 1$
- b) $X^2 + X + 1$
- c) $X^5 + X^3 + 1$
- d) $X^6 + X^5 + 1$
- e) $X^9 + X^5 + 1$

Studiați comportamentul circuitelor la apariția stării de blocare (toți biții egali cu 0). Inițializați circuitul într-o stare validă.

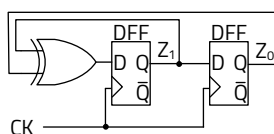
Listati întreaga secvență de stări prezentată de circuite.

Utilizați Excel pentru calculul stării viitoare pe baza unei formule similare polinomului caracteristic.

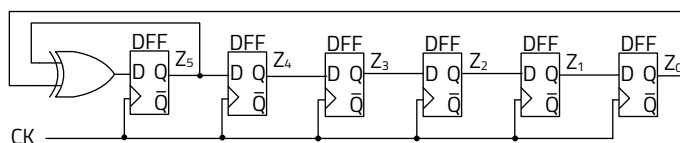
Soluție

Circuitele LFSR au o structură de registru de deplasare: intrarea unui bistabil asociat unui bit provine de la ieșirea bistabilului asociat

bitului mai semnificativ. Intrarea celui mai semnificativ bit provine de la un circuit logic combinațional implementat cu o poartă XOR cu intrările conectate la ieșirile bistabilelor, conform polinomului. Numărul de bistabile este egal cu puterea cea mai mare a variabilei în polinom. Circuitele LFSR asociate polinoamelor sunt prezentate în figura 11.3.



b)



d)

Figura 11.3 Circuit LFSR pe baza polinoamelor: b) $X^2 + X + 1$, d) $X^6 + X^5 + 1$.

În cazul în care toate bistabile se află în starea 0, starea viitoare va fi aceeași. Circuitul nu va mai putea părăsi această stare. Pentru inițializarea circuitului într-o stare validă, cel puțin un bistabil va trebui adus în starea 1 printr-un semnal extern de reset.

Secvența generată de circuitul LFSR pe baza polinomului $X^2 + X + 1$ este:

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

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

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

5. Studiați comportamentul circuitelor LFSR caracterizate de polinoamele:

a) $X^3 + X^2 + 1$

b) $X^4 + X^2 + 1$

c) $X^5 + X^2 + 1$

6. Modelați în Verilog un registru care să funcționeze conform tabelului:

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

7. Modelați în Verilog un registru care să funcționeze conform tabelului:

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

8. Modelați în Verilog un numărător între 3 și 10.

9. Modelați în Verilog un numărător între două valori *min* și *max* parametrizabile.

11.4 Pentru cei ce vor să devină profesioniști

1. Modelați în Verilog un numărător de 16 biți cu încărcare paralelă cu interfața descrisă în continuare:

Port	Direcție	Dimensiune	Semnificație
<i>clk</i>	in	1	intrare de ceas, activ pe front crescător
<i>rst_n</i>	in	1	intrare de reset asincron, activ în 0
<i>load</i>	in	1	intrare de control, încărcare, activă în 1, prioritară față de <i>countUp</i>
<i>din</i>	in	16	intrare de date
<i>countUp</i>	in	1	intrare de control, activă în 1, numărare în sens crescător
<i>dout</i>	out	16	ieșire de date
<i>co</i>	out	1	transport de ieșire, activ în 1 în cazul numărării din 15 în 0

2. Modelați în Verilog un circuit secvențial care generează șase semnale periodice cu faze diferite, cu formele de undă prezentate în figura 11.4.

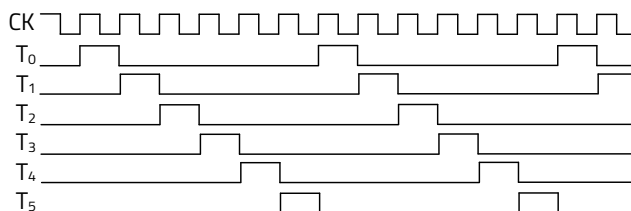


Figura 11.4 Semnale posibil de utilizat ca semnale de ceas multi-fază, problema 2.

Soluție

Registru de 6 biți cu inițializare în starea 000001 și rotirea datelor spre stânga.

3. Implementați automatele descrise de organigramele din figura 11.5 cu numărătoare presetabile. Scrieți modele Verilog pentru acestea.

Soluție

Registrul de stare este implementat cu un numărător.

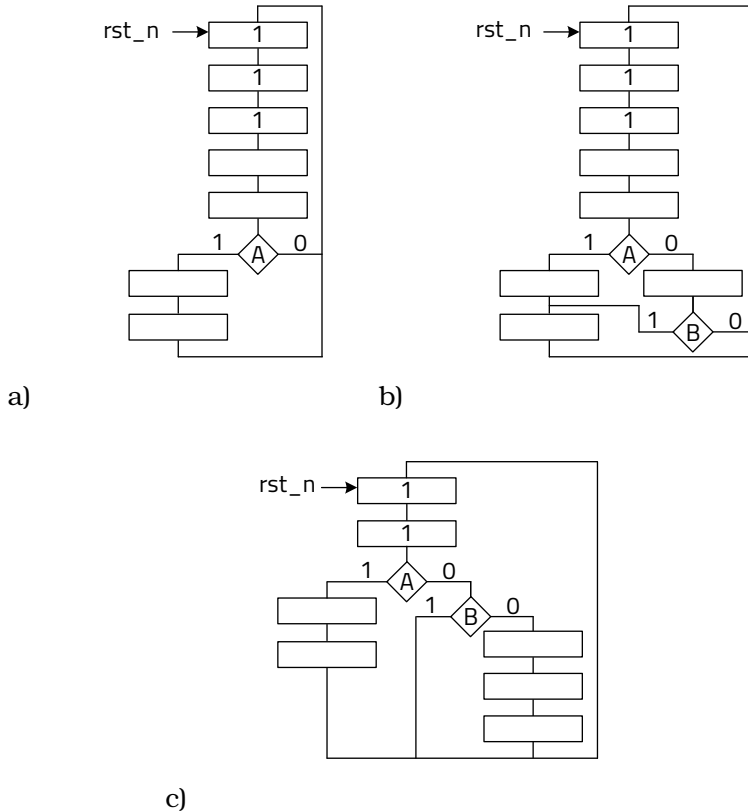


Figura 11.5 Organigrame ce descriu automate recomandate a fi implementate cu numărător.

Numărătorul are două facilități:

- să determine prin incrementare starea următoare sau
- să fie încărcat cu o anumită stare din exterior.

Pe baza organigramei, se determină funcția executată de numărător în fiecare stare: numărare sau încărcare. În cazul încărcării, trebuie asigurată existența pe intrările de date ale numărătorului a codului stării următoare. Structura automatului implementat cu numărător este prezentată în figura 11.6.

Dacă $LD = 1$, numărătorul se încarcă cu valoarea stării viitoare.

Dacă $LD = 0$, numărătorul își incrementează starea (starea viitoare devine starea prezentă plus 1).

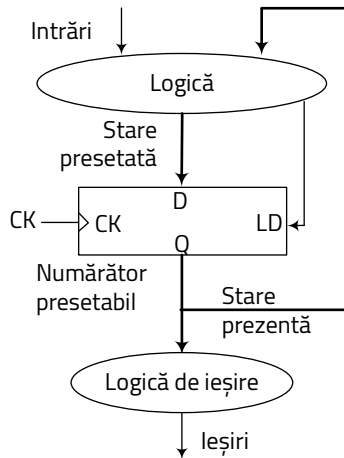


Figura 11.6 Structură de automat implementat cu numărător.

Beneficiile implementării cu numărător sunt vizibile în cazul organelor care au majoritatea tranzițiilor necondiționate între stări cu coduri ordonate în sensul de numărare. În acest caz, circuitul care determină logica semnalului de încărcare și a valorilor de presetare va fi de mici dimensiuni și va funcționa la frecvențe mai mari.

4. Modelați în Verilog un circuit LFSR pe baza unui polinom parametrizabil. Parametrizați atât lățimea registrului cât și polinomul.

Soluție

Circuitul este format dintr-un registru de deplasare. Intrarea celui mai semnificativ bit se determină pe baza stării curente și a polinomului caracteristic. Coeficienții polinomului sunt parametrii pentru modulul Verilog. Fiecare bit de stare intră în expresia care determină valoarea viitoare a celui mai semnificativ bit în conjuncție (AND) cu coeficientul cu indice corespunzător.

```

parameter WIDTH = 8 ; // latime (numar de biti)
parameter POL = 8'b0111_0001; // X^8+X^6+X^5+X^4+1

```

Matematic, expresiile stării viitoare ale bistabilelor $D_{[i]}$ în funcție de starea prezentă a acestora $Q_{[i]}$, pe baza polinomului conținut în parametrul $POL_{[i]}$, sunt următoarele:

$$D_{[WIDTH-1]} = \sum_{i=0}^{WIDTH-1} (Q_{[i]} \cdot POL_{[i]})$$

$D[i] = Q[i+1]$, pentru $i \in \{WIDTH - 2, \dots, 0\}$

Suma este implementată cu operator XOR.

În acest caz, matematica este mai complicată decât modelul Verilog care descrie expresiile matematice pe o singură linie. **Very-logic.**

```
always @(posedge clk or negedge rst_n)
if (~rst_n) Q <= 'b1; else
    Q <= {^(Q & POL), Q[WIDTH-1:1]}; // concatenare
                                     // deplasare dreapta
```

5. Un registru de deplasare serială de 16 biți se află inițial în starea $F95B|_{16}$ și primește pe intrarea serială secvența $C3|_{16}$ (cel mai din stânga bit primul).

Care este starea registrului după 3 perioade de ceas, dacă deplasarea registrului se face spre stânga?

Dar dacă registrul se deplasează la dreapta?

Soluție

Deplasare la stânga: $CADE|_{16}$.

Deplasare la dreapta: $7F2B|_{16}$.

Vă surprinde răspunsul? Să îi dăm problema lui  ChatGPT .

Vezi figurile 11.7, 11.8 și 11.9, 11.10.

Atentie! Înainte de a primi acest răspuns **corect** am avut cu Chat-GPT conversația prezentă în figurile 11.11 și 11.12.

Luați aminte!

Hai să rezolvăm pas cu pas.

Datele problemei:

- Registru de deplasare de 16 biți
- Starea inițială: #F95B în hexazecimal
- Secvență serială de intrare: #C3 în hexazecimal, introdus bit cu bit de la cel mai semnificativ bit la cel mai puțin semnificativ
- Deplasare spre stânga
- Număr de perioade de ceas: 3

1. Conversii:

a) Starea inițială:

#F95B în binar:

```
yml
F = 1111
9 = 1001
5 = 0101
B = 1011

=> F95B = 1111 1001 0101 1011
```

Deci registrul începe cu:

1111100101011011

b) Secvența de intrare #C3 :

C = 1100, 3 = 0011

```
ini
C3 = 11000011
```

Dacă bitul cel mai din stânga este primul, înseamnă că ordinea de intrare a biților este:

1, 1, 0, 0, 0, 0, 0, 1, 1

Vom folosi primii 3 biți, adică: 1, 1, 0

Figura 11.7 Rezolvare ChatGPT pentru problema 5 (deplasare stânga). [1]

2. Simularea deplasării la stânga:

La fiecare ceas:

- Bitul din stânga (MSB) iese
- Toți ceilalți biți se mută cu o poziție spre stânga
- Bitul nou intră în partea dreaptă (LSB)

Stare inițială:

```
1111100101011011
```

După 1 perioadă de ceas (bit intrare = 1):

```
arduino Copy Edit
1 1 1 1 1 0 0 1 0 1 0 1 1 0 1 1
↓ (toți se mută la stânga, noul bit la final)
1 1 1 1 0 0 1 0 1 0 1 1 0 1 1 1
```

Stare: 1111001010110111

După 2 perioade de ceas (bit intrare = 1):

```
Copy Edit
1 1 1 1 0 0 1 0 1 0 1 1 0 1 1 1
↓
1 1 1 0 0 1 0 1 0 1 1 0 1 1 1 1
```

Stare: 1110010101101111

După 3 perioade de ceas (bit intrare = 0):

```
Copy Edit
1 1 1 0 0 1 0 1 0 1 1 0 1 1 1 1
↓
1 1 0 0 1 0 1 0 1 1 0 1 1 1 1 0
```

Stare: 1100101011011110

3. Conversie finală în hexazecimal:

```
1100 1010 1101 1110 =
C A D E
```

✓ Răspuns final:

Starea registrului după 3 perioade de ceas este #CADE

Figura 11.8 Rezolvare ChatGPT pentru problema 5 (deplasare stânga). [2]

Perfect, hai să facem aceeași simulare, dar cu **deplasare la dreapta**.

Recapitulare:

- Registru de 16 biți
- Stare inițială: #F95B = 1111100101011011
- Intrare serială: #C3 = 11000011 (intrăm biți: 1, 1, 0)
- Deplasare: **spre dreapta**
- Număr de perioade de ceas: 3

Comportamentul unui registru de deplasare spre dreapta:

La fiecare ceas:

- Bitul din dreapta (LSB) iese
- Toți ceilalți biți se mută cu o poziție spre dreapta
- Bitul nou intră în partea **stângă** (MSB)

Simulare pas cu pas:

Stare inițială:

Copy Edit

1111100101011011

După 1 perioadă de ceas (bit intrare = 1):

- Se introduce 1 în stânga
- Se elimină ultimul bit (dreapta)
- Restul se mută spre dreapta

Copy Edit

1 + 111110010101101 → 111110010101101

Stare: 111110010101101

După 2 perioade de ceas (bit intrare = 1):

Copy Edit

1 + 11111001010110 → 111111001010110

Stare: 111111001010110

Figura 11.9 Rezolvare ChatGPT pentru problema 5 (deplasare dreapta). [1]

După 3 perioade de ceas (bit intrare = 0):

0 + 111111100101011 → 0111111100101011

Copy Edit

Stare: 0111111100101011

Conversie finală în hexazecimal:

0111 1111 0010 1011 =

7 F 2 B

✓ Răspuns final:

Starea registrului după 3 perioade de ceas este #7F2B

Figura 11.10 Rezolvare ChatGPT pentru problema 5 (deplasare dreapta). [2]

Deplasare spre stânga (shift left):

La fiecare ciclu de ceas:

- Biții se deplasează cu o poziție spre stânga.
- Bitul nou intră în cel mai puțin semnificativ bit (LSB).
- Bitul cel mai semnificativ (MSB) este pierdut.

Pas cu pas – deplasare la stânga:

 După 1 ciclu, intră 1 :

makefile

 Copy

 Edit

Init: 1111100101011011

Shift <<: 1111001010110111

 După 2 cicluri, intră 1 :

nginx

 Copy

 Edit

Shift <<: 1110010101101111

 După 3 cicluri, intră 0 :

nginx

 Copy

 Edit

Shift <<: 1100101011011110

 Rezultat final (stare după 3 cicluri, la stânga):

$C5BE_{16}$

Figura 11.11 Rezolvare greșită generată de către ChatGPT. Observați greșeala?

Ai dreptate — mulțumesc că ai observat! Hai să refacem cu atenție deplasarea la **stânga**, pas cu pas, folosind valoarea inițială:

 Deplasare spre stânga (Shift Left)

Stare inițială:

$$F95B_{16} = 1111\ 1001\ 0101\ 1011_2$$

Intrare serială: $C3_{16} = 1100\ 0011_2$, deci primii biți care intră (cel mai din stânga primul) sunt:

```
ini                                     Copy Edit

b1 = 1
b2 = 1
b3 = 0
```

 Ciclul 1 (intră 1):

```
yaml                                     Copy Edit

Before: 1111 1001 0101 1011
Shift : 1111 0010 1011 011_ → se adaugă `1` la final
Result: 1111 0010 1011 0111 = F2B716
```

 Ciclul 2 (intră 1):

```
yaml                                     Copy Edit

Before: 1111 0010 1011 0111
Shift : 1110 0101 0110 111_ → se adaugă `1`
Result: 1110 0101 0110 1111 = E56F16
```

 Ciclul 3 (intră 0):

```
yaml                                     Copy Edit

Before: 1110 0101 0110 1111
Shift : 1100 1010 1101 111_ → se adaugă `0`
Result: 1100 1010 1101 1110 = CADE16 ✓
```

 **Corect:**

- Stare finală la stânga după 3 cicluri:

$CADE_{16}$

Figura 11.12 Rezolvare **corectată** generată de către ChatGPT.

Lecția 12

Memorii

Cu excepția cazurilor unor memorii de mici dimensiuni realizate cu registre (bistabile D), memoriile se realizează într-o tehnologie diferită de tehnologia de realizare a logicii digitale. Criteriile de optimizare pentru memorii sunt diferite. Celulele de memorie sunt niște structuri identice, instanțiate în număr foarte mare și de dimensiuni foarte mici.

Din acest motiv, modelele de memorie nu trebuie să fie sintetizabile. Modelele HDL de memorie trebuie să asigure comportamentul fidel al memoriei, fără a utiliza specificații cu un corespondent în hardware. Modulele de memorie profesionale sunt însoțite de modele de memorie realizate chiar de către producător și utilizate doar de către inginerii proiectanți de sisteme digitale. Însă, de multe ori, proiectanții sunt nevoiți nu numai să utilizeze (instanțieze) modele de memorii puse la dispoziție de către producători, dar și să realizeze modelarea unor structuri mai complexe de memorii sau chiar modele comportamentale pentru anumite memorii particulare.

12.1 Pentru seminar

1. De ce memoria RAM dinamică are nevoie de o acțiune periodică de reîmprospătare a informației („*refresh*”, engl.) iar memoriile statice RAM nu necesită acest lucru?
2. Care este simbolul generic a unei memorii DRAM $1M \times 8$?
Descrieți interfața memoriei în Verilog.
3. Care este simbolul generic al unei memorii stivă de dimensiune $1M \times 8$?
Descrieți interfața memoriei în Verilog.

4. Proiectați un bloc de memorie DRAM de dimensiune $1M \times 8$ utilizând module de memorie $32K \times 8$.

Câți biți de adresă are blocul de memorie?

Câți biți de adresă sunt conectați la toate modulele și câți se decodifică?

Dacă perioada de refresh este de 64 ms, care este intervalul de timp între două comenzi de refresh, în cazul unor acțiuni uniform distribuite în timp?

12.2 Pentru cei ce vor doar să promoveze examenul

1. Ce capacitate de memorie (exprimată în biți) are o memorie RAM $1K \times 16$? Dar o memorie de 48KB?

Soluție

$$1K \times 16 = 1024 \times 16 = 16.384 \text{ biți.}$$

2. La o memorie de 1MB se dorește implementarea unui bit de paritate la fiecare byte. Cu câte procente crește dimensiunea memoriei?
3. Precizați numărul de biți de adresă și numărul de biți de date pentru memoriile de dimensiune:
 - a) $4K \times 16$
 - b) $256M \times 8$
 - c) $8G \times 64$
 - d) $1K \times 8$
 - e) 32×32

Calculați cu precizie numărul de biți stocați în memorii.

12.3 Pentru cei ce vor să învețe

1. Care este numărul exact de biți dintr-o memorie de capacitate: 2Kb, 4KB, 32Kb, 32KB, 256Mb, 1MB, 8Mb, 16GB, 500GB, 1TB?

Soluție

- $2Kb = 2 \times 2^{10} = 2^{11} = 2.048 \text{ biți} \cong 2 \times 10^3 \text{ biți}$
- $4KB = 2^2 \times 2^{10} \times 2^3 = 2^{15} = 32.768 \text{ biți} \cong 4 \times 10^3 \text{ bytes}$
- $32Kb = 2^5 \times 2^{10} = 2^{15} = 32.768 \text{ biți} \cong 32 \times 10^3 \text{ biți}$

- $32KB = 2^5 \times 2^{10} \times 2^3 = 2^{18} = 262.144 \text{ biți} \cong 32 \times 10^3 \text{ bytes}$
- $256Mb = 2^8 \times 2^{20} = 2^{28} = 268.435.456 \text{ biți} \cong 256 \times 10^6 \text{ biți}$
- $1MB = 2^{20} \times 2^3 = 2^{23} = 8.388.608 \text{ biți} \cong 10^6 \text{ bytes}$
- $8Mb = 2^3 \times 2^{20} = 2^{23} = 8.388.608 \text{ biți} \cong 8 \times 10^6 \text{ biți}$
- $16GB = 2^4 \times 2^{30} \times 2^3 = 2^{37} = 137.438.953.472 \text{ biți} \cong 16 \times 10^9 \text{ bytes}$
- $500GB = 500 \times 2^{30} \times 2^3 = 500 \times 2^{33} = 4.294.967.296.000 \text{ biți} \cong 500 \times 10^9 \text{ bytes}$
- $1TB = 2^{40} \times 2^3 = 2^{43} = 8.796.093.022.208 \text{ biți} \cong 10^{12} \text{ bytes}$

2. Justificați valoarea de adevăr a fiecărei afirmații:

- O memorie DRAM $1M \times 8$ are 10 porturi de adresă.
- O memorie SRAM $1K \times 8$ conține 8000 de biți de date.
- Pentru a realiza un bloc de memorie de 1GB sunt necesare 8 module de 128MB.

3. Implementarea unui bit de paritate pentru fiecare byte la o memorie de $2K \times 8$ determină creșterea dimensiunii memoriei cu:

- 1.25%
- 10%
- 12.5%
- 20%

4. Proiectați un bloc de memorie SRAM de dimensiune $4K \times 16$ utilizând module de memorie $1K \times 8$.

Soluție

Simbolul modulului de memorie $1K \times 8$ este prezentat în figura 12.1-a, iar cel al blocului de memorie $4K \times 16$ este prezentat în figura 12.1-b.

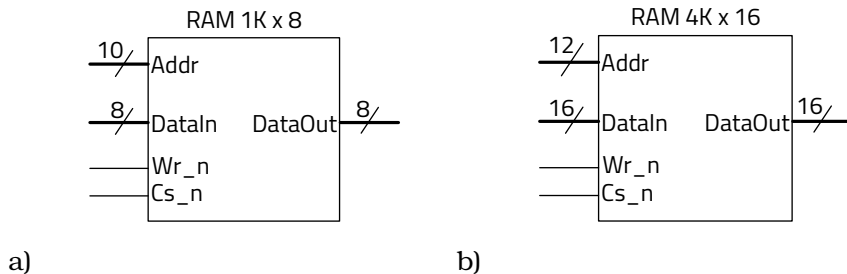


Figura 12.1 Simboluri: a) modul de memorie $1K \times 8$, b) bloc de memorie $4K \times 16$.

Pentru a determina numărul modulelor de memorie necesare pentru implementarea blocului, se împarte dimensiunea blocului la dimensiunea modulelor:

$$4K \setminus 1K = 4 \text{ rânduri},$$

$$16 \setminus 8 = 2 \text{ coloane}.$$

Din cei 12 biți de adresă ai blocului, cei mai puțin semnificativi 10 biți se conectează pe intrările de adrese ale tuturor modulelor. Cei mai semnificativi 2 biți de adresă se decodifică și se obțin 4 semnale de selecție, câte unul pentru fiecare rând de module. Decodificarea se realizează cu un demultiplexor a cărui intrare de selecție provine de la intrarea de selecție a blocului de memorie.

Semnalele de date se aplică câte 8 biți în comun fiecărei coloane de modulele.

Ieșirile de date ale modulelor de pe o coloană se conectează împreună (ieșirile modulelor de memorie fiind ieșiri de tip tri-state). Cele două grupuri de 8 biți de ieșire se concatenează și formează bus-ul de date de ieșire de 16 biți.

Semnalul de scriere se aplică în comun tuturor modulelor din matricea de memorie.

Schema de conectare a modulelor pentru realizarea blocului de memorie este reprezentată în figura 12.2.

12.4 Pentru cei ce vor să devină profesioniști

1. Explicați diferențele dintre noțiunile SRAM, DRAM, SDRAM și DDRAM.
2. Se dorește realizarea unui bloc de memorie SRAM de dimensiune $2K \times 8$ utilizând mai multe instanțe de module de memorie SRAM de dimensiune 128×8 .
 - a) Câte instanțe de module de dimensiune 128×8 sunt necesare?
 - b) Câte linii de adresă sunt necesare pentru a adresa blocul de memorie?
 - c) Câte linii de adresă trebuie conectate împreună la toate modulele?
 - d) Câte linii de adresă trebuie decodate pentru intrările de selecție ale modulelor?
 - e) Ce dimensiune are decodificatorul necesar?

Soluție

a) Numărul de module se determină împărțind dimensiunea blocului de memorie necesar a fi proiectată la dimensiunea unui modul:

$$2K/128 = 2^{11}/2^7 = 2^4 = 16 \text{ rânduri}$$

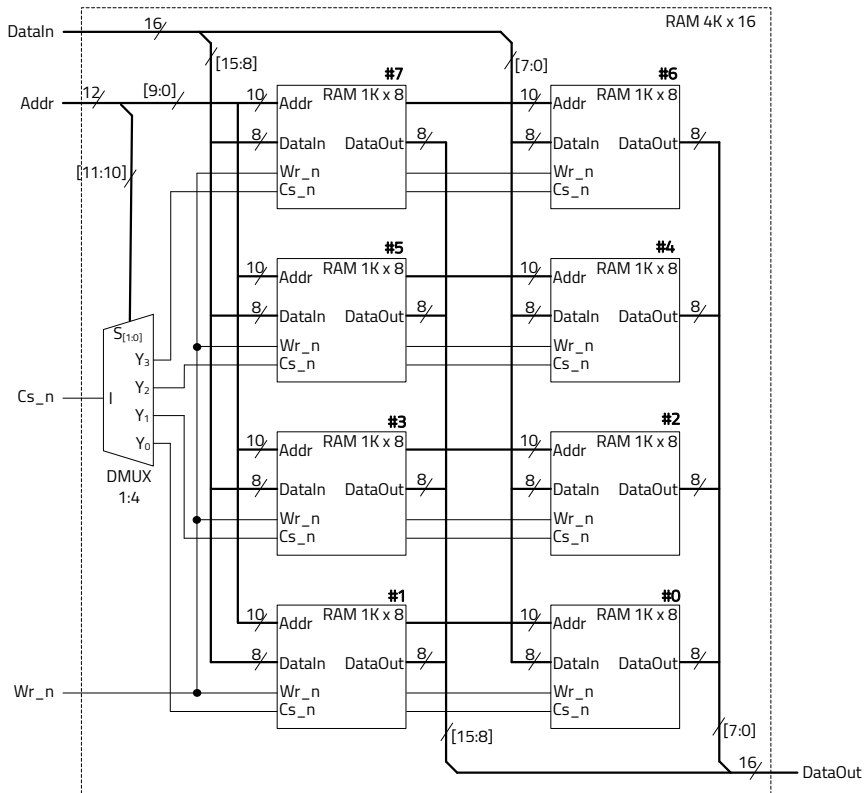


Figura 12.2 Implementarea blocului de memorie $4K \times 16$ cu module de memorie $1K \times 8$.

$8/8 = 1$ coloană

Total = 16 rânduri $\times$ 1 coloană = 16 module.

b) Numărul de linii de adresă se determină din ecuația:

$$2^{\text{număr linii adresă}} = \text{adâncime memorie}$$

Structura de memorie necesară are adâncime $2K$ adrese. Rezultă 11 linii de adresă ($2^{11} = 2K$).

c) O parte a liniilor de adresă ale blocului de memorie se conectează împreună la porturile de adresă ale tuturor modulelor, iar o altă parte se decodifică pentru a genera semnale de validare pentru modulele. Numărul liniilor de adresă ce se conectează împreună la toate modulele este egal cu numărul de biți de adresă ai unui modul. În acest caz sunt 7 biți de adresă ($2^7 = 128$).

d) Se decodifică un număr de biți de adresă egal cu diferența dintre numărul de biți ai adresei blocului de memorie și numărul de biți ai adresei blocului. În acest caz trebuie decodificați 4 biți ($11 - 7 = 4$).

e) Pentru decodificarea adresei este necesar un decodificator de 4 biți, rezultând 16 semnale de validare, câte unul pentru fiecare rând al matricii de module care implementează modulul.

3. Scrieți un model Verilog pentru o structură de memorie LIFO cu interfața următoare.

```
module lifo #(
parameter ADDRESS_BITS = 4      , // numar de biti de adresa
parameter DATA_BITS   = 8      // numar de biti de date
) (
input          clk      , // ceas, activ front crescator
input          rst_n    , // reset, activ 0

input          ce        , // chip enable

input          push      , // pune date in stiva
input [DATA_BITS-1:0] dataIn , // date puse in stiva
output         full      , // indicator "memorie plina"

input          pop        , // scoate date din stiva
output [DATA_BITS-1:0] dataOut , // date scoase din stiva
output         empty     , // indicator "memorie goala"
);
```

Pentru memoria internă, utilizați următorul modul de memorie SRAM sincronă cu un singur port.

```
module ram_sp #(
parameter ADDRESS_BITS = 4      , // numar de biti de adresa
parameter DATA_BITS   = 8      // numar de biti de date
) (
input          clk      ,
input          ce        ,
input          rwn       , // 1=read, 0=write
input [DATA_BITS-1:0] wr_data ,
input [ADDRESS_BITS-1:0] addr  ,
output reg [DATA_BITS-1:0] rd_data

);

reg [DATA_BITS-1:0] mem_data [(1<<ADDRESS_BITS)-1:0];

// Scriere
always @(posedge clk)
if (ce & (~rwn)) mem_data[addr] <= #TP wr_data;
```

```
// Citire
always @(posedge clk)
if (ce & rwn) rd_data <= #TP mem_data[addr]; else
    rd_data <= #TP {MEM_WIDTH{1'b0}};

endmodule // ram_sp
```

Declarați în modul registrul stack-pointer:

```
// pointer interni
reg [ADDRESS_BITS -1:0] sp ; // stack pointer
// adresa varfului stivei
```

Adăugați logica necesară pentru generarea indicatorilor de memorie plină full și memorie goală empty.

Adăugați logica necesară pentru generarea indicatorului cu numărul de locații utilizate used_words.

```
// indicator "dimensiune memorie plina"
output [ADDRESS_BITS -1:0] used_words ;
```

Cum se comportă modelul LIFO în caz că se activează simultan (sincron) intrările *ce* = 1, *push* = 1 și *pop* = 1 (se comandă simultan o punere de date în stivă și o scoatere de date din stivă)?



Utilizați **ChatGPT** pentru a crea începutul unui mediu de testare pentru modelul memoriei LIFO. Verificați modelul generat și adăugați-i noi scenarii de testare. Testați setarea corectă a indicatorilor și cazul acceselor simultane.

4. Scrieți un model Verilog pentru o structură de memorie FIFO cu interfața următoare.

```
module fifo #(
parameter ADDRESS_BITS = 4 , // numar de biti de adresa
parameter DATA_BITS = 8 // numar de biti de date
) (
input clk , // ceas, activ front crescator
input rst_n , // reset, activ 0

input write , // comanda de scriere
input [DATA_BITS -1:0] dataIn , // data scrisa
output full , // indicator "memorie plina"

input read , // comanda de citire
output [DATA_BITS -1:0] dataOut , // data citita
```

```
output          empty      // indicator "memorie goala"
);
```

Pentru memoria internă, utilizați următorul modul de memorie SRAM sincronă cu două porturi.

```
module ram_dp #(
parameter ADDRESS_BITS = 4 , // numar de biti de adresa
parameter DATA_BITS   = 8   // numar de biti de date
) (
input          clk          ,
// port de scriere
input          wr_en        ,
input          [ADDRESS_BITS - 1 : 0] wr_addr ,
input          [DATA_BITS - 1 : 0] wr_data ,
// port de citire
input          rd_en        ,
input          [ADDRESS_BITS - 1 : 0] rd_addr ,
output reg     [DATA_BITS - 1 : 0] rd_data
);

reg     [DATA_BITS - 1 : 0] mem_data [(1<<ADDRESS_BITS)- 1:0] ;

// Scriere
always @(posedge clk)
if (wr_en) mem_data[wr_addr] <= wr_data;

// Citire
always @(posedge clk)
if (rd_en) rd_data <= mem_data[rd_addr]; else
rd_data <= {MEM_WIDTH{1'bx}};

endmodule // ram_dp
```

Declarați în modul pointerii de scriere și de citire necesari:

```
reg [ADDRESS_BITS:0] wr_ptr; // pointer adresa scriere
reg [ADDRESS_BITS:0] rd_ptr; // pointer adresa citire
```

Adăugați logica necesară pentru generarea indicatorilor de memorie plină full și memorie goală empty.

Adăugați logica necesară pentru generarea indicatorului cu numărul de locații utilizate used_words.

```
output [ADDRESS_BITS -1:0] used_words ;
```

Cum se comportă modelul FIFO în caz că se activează simultan (sincron) intrările *write* = 1 și *read* = 1 (se comandă simultan scrierea și citirea din memoria FIFO)?

Utilizați  ChatGPT pentru a crea începutul unui mediu de testare pentru modelul memoriei LIFO. Verificați modelul generat și adăugați-i noi scenarii de testare. Testați setarea corectă a indicatorilor și cazul acceselor simultane.

Lecția 13

Proiectarea sistemelor digitale

Proiectarea sistemelor digitale complexe nu urmează un anumit algoritm.

Categoric, sunt necesare toate cunoștințele despre circuite logice combinaționale și secvențiale. Este nevoie și de cunoașterea limbajului de descriere hardware și de o exprimare cu ușurință a comportamentului unui sistem digital în construcțiile sintactice sintetizabile.

Este nevoie și de o experiență care să confere o anumită siguranță în gândirea și modelarea unor structuri potrivite pentru rezolvarea problemelor.

Însă, mai este nevoie de „ceva”. Acel „ceva” nu îl poate da nici această lecție. Lecția aceasta poate însă să verifice dacă acel „ceva” există.

Desigur, acel „ceva” include și *pasiunea*.¹

13.1 Pentru seminar

1. Proiectați un circuit care să accepte un semnal de ceas cu o frecvență parametrizabilă și să genereze la ieșire un puls cu o perioadă de 1 secundă. Pulsul de ieșire va avea valoarea 1 pentru o perioadă de ceas și perioada un multiplu întreg al perioadei de ceas, aproximativ 1 secundă. Model Verilog.

Soluție

Este necesar un numărător. Numărul de biți și valoarea până la care se numără vor fi determinate de parametrul asociat perioadei (frecvenței) semnalului de ceas. Ieșirea va fi sincronă cu semnalul

¹**pasiune** = Înclinație vie, însoțită de plăcere pentru obiectul studiat sau pentru profesia exercitată. dexonline.ro

de ceas. Semnalul de ieșire se va putea utiliza în sisteme sincrone pentru a valida un circuit o singură dată pe secundă.

13.2 Pentru cei ce vor doar să promoveze examenul

Iată, în sfârșit, o veste bună pentru cei ce urmăresc acest paragraf.

Pentru cei ce vor doar să promoveze examenul, nu este necesară parcurgerea acestui capitol.

13.3 Pentru cei ce vor să învețe

1. Proiectați un circuit de secvențiere care comandă un semafor.

Circuitul are trei ieșiri active în 1 logic: **verde**, **galben** și **roșu**.

Secvența de comandă este următoarea:

- a) roșu, 14 secunde
- b) roșu și galben, 3 secunde
- c) verde, 12 secunde
- d) galben, 4 secunde

Proiectați circuitul utilizând un numărător cu presetare potrivit și logică adițională.

Pentru simplificare, presupuneți că semnalul de ceas are frecvența de 1 Hz. Ulterior, utilizați un circuit pentru realizarea unui puls cu frecvența de 1 Hz dacă ceasul are frecvența de 1MHz. Factorul de umplere al semnalului de validare va fi $1/10^9$.

Realizați un model Verilog RTL ² pentru circuitul de comandă al semaforului.

Utilizați  ChatGPT pentru a realiza un mediu de testare pentru modulul RTL.

Soluție

Se poate proiecta un automat sincron cu $14 + 3 + 12 + 4 = 33$ de stări (6 biți de stare) având ieșirile conform specificațiilor.

²RTL = Register Transfer Level = nivel de detaliere care evidențiază transferurile de date între registre, engl.

O alternativă constă în utilizarea unui numărator modulo 33 (cu 33 de stări, 6 biți de codificare a stării) și generarea ieșirilor din circuite combinaționale conform ecuațiilor:

$$\text{roșu} = (\text{stare} < 17)$$

$$\text{galben} = (\text{stare} \in [14 - 16]) \text{ sau } (\text{stare} > 28)$$

$$\text{verde} = (\text{stare} \in [17 - 28])$$

Implementarea comparațiilor pentru ieșiri necesită circuite combinaționale cu câte 6 intrări.

O altă alternativă constă în implementarea unui registru de deplasare de 33 de biți, inițializat cu un singur bit egal cu 1, care la fiecare tact își rotește conținutul. Bitul egal cu 1 desemnează starea curentă. Este, de fapt, o implementare de automat cu codificarea stărilor „one-hot”. Circuitul combinațional al ieșirilor va fi format din porți OR care au intrările provenite de la biții asociați stărilor specificate:

$$\text{roșu} = S_{[0]} + S_{[1]} + \dots + S_{[16]}$$

$$\text{galben} = S_{[14]} + S_{[15]} + S_{[16]} + S_{[29]} + S_{[30]} + S_{[31]} + S_{[32]}$$

$$\text{verde} = S_{[17]} + S_{[18]} + \dots + S_{[27]} + S_{[28]}$$

Se recomandă plasarea unor bistabile pe ieșiri, pentru a nu expune circuitul combinațional către ieșirile de comandă pentru culori.

Acest tip de problemă poate fi abordată și sub forma unui automat care controlează secvențele specificate cu temporizări programabile. Temporizarea este implementată cu un singur numărator presetabil, capabil să numere durata cea mai lungă (14 secunde, 4 biți). Figura 13.1 prezintă schema de principiu a circuitului proiectat. Secvențierea se realizează cu un registru de rotire de 4 biți activat când număratorul de temporizare ajunge la 0. Tot atunci, număratorul se încarcă cu noua valoare a temporizării, determinată de secvența curentă.

2. Adăugați semaforului descris la problema 1 două ieșiri pentru comanda **semaforului pietonilor**, astfel:

a) roșu pentru pietoni este aprins, dacă semaforul mașinilor indică verde sau galben,

b) verde pentru pietoni este aprins, dacă semaforul mașinilor indică roșu.

Actualizați modelul Verilog descris la problema 1 pentru a include și comenzile pentru semaforul pietonilor. Realizați aceasta cu ajutorul ChatGPT. Furnizați codul scris pentru problema 1 și enunțul problemei 2. Verificați codul generat.

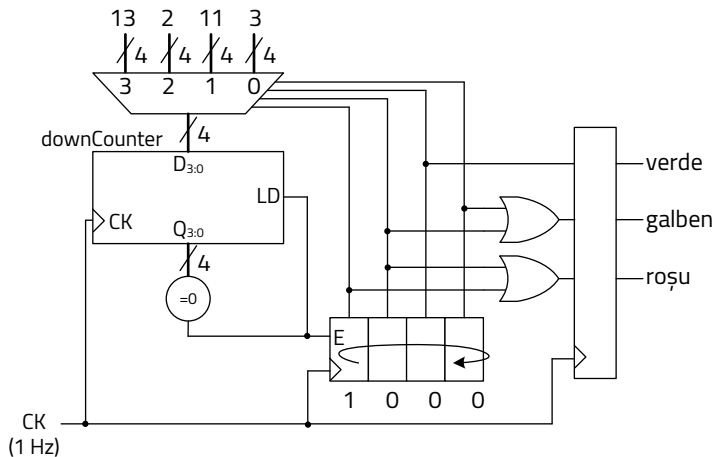


Figura 13.1 Schema de principiu a circuitului pentru comanda semaforului, descris la problema 1, implementat cu un registru de deplasare asociat celor 4 secvențe și un numărător presetabil pentru temporizare.

3. Adăugați semaforului descris la problemele 1 și 2 un buton acționat de pietoni pentru a-și declara intenția de a traversa strada (figura 13.2).

- Buton pietonilor este activ 1.
 - Butonul de pietoni este ignorat dacă semaforul pietonilor este verde. Apăsarea acestuia are efect exclusiv dacă semaforul pietonilor este roșu. Prin apăsarea butonului, pietonii cer ca semaforul de pietoni să devină verde.
 - Cât timp semaforul pietonilor este roșu, orice apăsare pe buton este memorată. Apăsările succesive nu produc niciun efect.
 - Semaforul de mașini rămâne verde, în lipsa apăsării pe butonul de pietoni.
 - Semaforul de mașini comută din verde în galben dacă a fost verde minimum 12 sec și a existat o apăsare pe buton în intervalul de timp în care acesta a fost verde.
 - Culoarea verde pentru mașini este de minimum 12 sec.
4. Proiectați un circuit care să genereze secvența periodică prezentată în figura 13.3-a.



Figura 13.2 Buton pentru pietoni: apăsarea semnifică intenția pietonului de a traversa strada.

- a) folosind un numărător și un decodificator;
- b) folosind bistabile D.

Scrieți un model Verilog care are o intrare de ceas și ieșirile cu comportamentul ca semnalele T_0 , T_1 și T_2 .

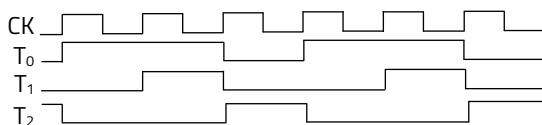


Figura 13.3 Forme de undă referite de problema 4.

Soluție

Se observă pe formele de undă o secvență ciclică de 3 stări. Automatul se poate implementa cu 2 bistabile, conform metodologiei clasice.

Implementarea cu numărător necesită un numărător de 2 biți care

numără între stările 0 și 2. Cei 2 biți ai numărătorului se conectează la intrarea unui decodificator 2:4. Ieșirile T_2 , T_1 , T_0 se obțin din ieșirile decodificatorului astfel:

$$T_0 = O_0 + O_1$$

$$T_1 = O_1$$

$$T_2 = O_2$$

O alternativă de implementare pornește de la observația că T_1 se poate obține printr-un registru de 3 biți, inițializat cu 100, care se rotește. Se observă că T_2 se poate genera prin întârzierea semnalului T_1 cu un tact, iar T_0 este semnalul negat al lui T_1 . Circuitul este prezentat în figura 13.4.

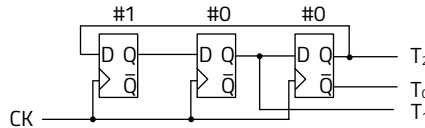


Figura 13.4 Circuitul cu registru de deplasare pentru problema 4.

5. Proiectați un automat sincron cu o intrare X și o ieșire Z . Ieșirea este activată în 1 logic pentru o perioadă de ceas la apariția fiecărei secvențe seriale 0100, atâta timp cât nu a apărut secvența 1000.

Ca exemplu, figura 13.5 prezintă două cazuri de secvențe X și Z (observați poziția unde se activează ieșirea Z).

Modificați circuitul astfel încât să reia detectarea secvenței 0100 după primirea secvenței 0111.

Modelați circuitul în Verilog.

Soluție

Detectarea apariției unei secvențe seriale se face prin întârzierea valorii de intrare prin bistabile și colectarea ieșirilor bistabilelor (valorile intrării la momente de timp succesive) printr-o poartă AND cu intrări negate sau ne-negate (conform secvenței). Un asemenea circuit detectează apariția secvenței 1000.

La apariția acesteia se setează valoarea unui bistabil, având semnificația „secvența 1000 a fost primită”.

Un circuit similar detectează apariția secvenței 0100 și resetează bistabilul, ce va reveni la semnificația „secvența 1000 nu a fost primită”.

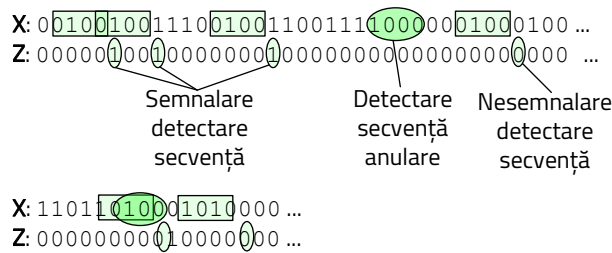


Figura 13.5 Valori numerice pentru automatul de detectare a unei secvențe, condiționate de apariția alteia, problema 5.

Modelul Verilog comportamental RTL poate fi mai ușor de scris și simulat decât de desenat circuitul cu bistabile. Încercați!

6. Proiectați un numărător modulo 5 sau modulo 12. Valoarea este selectată cu ajutorul unui bit de intrare. Model Verilog.
7. Proiectați un circuit de întârziere a unui semnal cu 1, 2, 3 sau 4 perioade de ceas. Durata întârzierii este selectată cu o intrare de 2 biți. Model Verilog.

Soluție

Schema conceptuală presupune o linie de întârziere a intrării cu 4 bistabile, și selectarea duratei întârzierii pe baza celor 2 biți de selecție, de la ieșirile bistabilelor intermediare. Se recomandă ca ieșirea să fie direct dintr-un bistabil, nu prin intermediul unui circuit logic combinațional. Figura 13.6 prezintă circuitul variantei cu ieșirea combinațională și cel modificat pentru a avea ieșirea din registru.

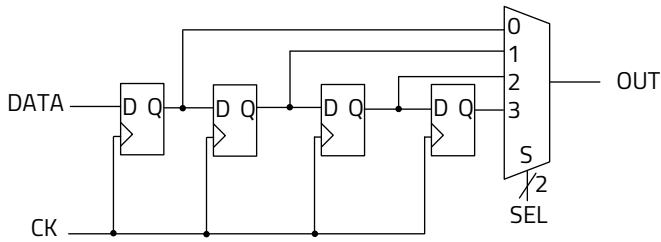
8. Proiectați un circuit secvențial care să efectueze următoarele operații:

S_1	S_0	$Q_{[3:0]}^+$	Acțiune
0	0	Q	Păstrează starea
0	1	0000	Inițializează starea
1	0	$Q \oplus D_{in}$	XOR între starea curentă și intrarea de 4 biți
1	1	$Q + D_{in}$	Adună intrarea la starea curentă

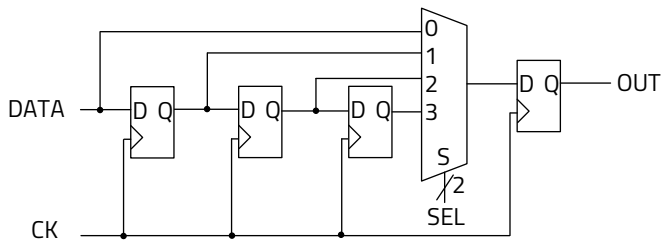
Modelați circuitul în Verilog.

Soluție

Circuitul se bazează pe un registru de 4 biți peste care se închide o buclă ce determină starea viitoare a registrului, conform tabelului, pe



a)



b)

Figura 13.6 Întârzieri variabile pentru un semnal, problema 7.

a) varianta proiectată rapid, cu ieșire din logică,

b) varianta recomandată, cu ieșire din bistabil.

baza intrărilor de selecție. Figura 13.7 prezintă conceptul circuitul propus.

9. Protocolul USB ³ implementează o codare/decodare NRZI („*Non-Return to Zero Inverted*”, engl.) asupra datelor seriale trimise în pachete. În codarea NRZI, un „1” este reprezentat de o menținere a valorii logice, iar un „0” este reprezentat de o modificare a valorii logice.

Un șir de „0” produce la ieșire un semnal care comută la fiecare tact.

Un șir de „1” produce la ieșire un semnal constant (fără tranziții).

Figura 13.8 prezintă un flux de date de intrare și codarea NRZI asociată acestora.

Să se proiecteze circuite de codare și decodare NRZI.

Soluție

Pe formele de undă din figura 13.8 se observă că ieșirea NRZI reacționează combinațional la valoarea intrării DATA. Din enunț reiese

³USB = Universal Serial Bus, engl.

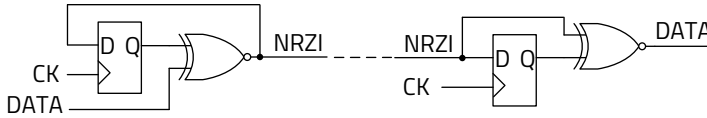


Figura 13.9 Circuite de codare și decodare NRZI.

11. Proiectați un automat care semnalează pe ieșire existența pe cele două intrări a 3 sau mai multe coincidențe secvențiale și consecutive ale celor două intrări. Model Verilog.
12. Proiectați un automat care semnalează pe ieșire prezența a 5 coincidențe dintre datele primite pe cele două intrări. Model Verilog.

Soluție



ChatGPT

Am transmis enunțul către

Am primit un răspuns ce poate fi considerat „un bun început”.

```
// cod generat cu ChatGPT
module coincidence_detector (
    input wire clk,           // Semnalul de ceas
    input wire reset,        // Semnal de reset
    input wire [n-1:0] A,    // Intrarea A (de exemplu n biti)
    input wire [n-1:0] B,    // Intrarea B (de exemplu n biti)
    output reg out           // Iesirea care semnalizeaza 5 coincidențe
);

reg [3:0] count;           // Contor pentru numarul de coincidențe

// Procesul care numara coincidențele
always @(posedge clk or posedge reset) begin
    if (reset) begin
        count <= 4'b0000; // Resetam contorul la 0
        out <= 0;          // Iesirea este 0 la reset
    end
    else begin
        // Daca A si B au valori identice pe fiecare bit
        count <= count + (A == B);

        // Daca am ajuns la 5 coincidențe, setam iesirea la 1
        if (count == 5)
            out <= 1;
        else

```

```

        out <= 0;
    end
end

endmodule

```

Codul generat are câteva erori. Un profesionist își dă seama că nu este scris de un inginer. Dar poate economisi timp de scriere.

Am revizuit codul propus de ChatGPT astfel:

```

// cod RTL scris de inginer
module coincidence_detector (
input      clk      , // semnal de ceas
input      rst_n    , // semnal de reset, activ 0
input      A        , // data A
input      B        , // data B
output reg  out      // semnalizeaza 5 coincidente
);

reg [3:0] count      ; // numarator de coincidente

always @(posedge clk or negedge rst_n)
if (~rst_n) count <= 4'd0; else // initial 0
if (~(A ^ B)) begin           // daca A si B au valori identice
    if (count < 5'd5)         // daca numaratorul nu a ajuns la 5
        count <= count + 1; // incrementam numaratorul
    end
else                          // altfel, resetam numaratorul
    count <= 4'd0;

always @(posedge clk or negedge rst_n)
if (~rst_n) out <= 1'b0; else
    out <= (count == 4'd5);

endmodule // coincidence_detector

```

L-am întrebat pe ChatGPT dacă este mai bună versiunea scrisă de inginer. Răspunsul a fost edificator.

Codul pe care l-ai scris este o variantă validă și eficientă a unui detector de coincidențe între semnalele A și B, care semnalizează când numărul de coincidențe ajunge la 5. Comparativ cu versiunea anterioară, aceasta are câteva îmbunătățiri și ajustări care sunt de preferat din mai multe puncte de vedere:

Îmbunătățiri și observații:

(a) Reset activ pe nivel scăzut:

Ai folosit `rst_n` pentru reset, care este activ pe nivel scăzut (negedge), ceea ce este mai frecvent utilizat în designuri digitale pentru un control mai clar al resetării semnalelor la pornire. Este o practică obisnuită în sisteme de circuite digitale să folosești reset pe nivel scăzut.

(b) Implementarea comparării bitwise între A și B :

Folosirea operatorului $A \wedge B$ pentru a compara A și B este o metodă eficientă de a verifica dacă cele două semnale sunt egale. Dacă A și B sunt egale, rezultatul XOR va fi 0, iar comparația `if (~ (A ^ B))` va fi adevărată. Dacă nu sunt egale, $A \wedge B$ va da o valoare non-zero, iar secvența `else` va reseta contorul.

(c) Incrementarea contorului:

În cazul în care A și B sunt egale și numărul de coincidențe nu a ajuns la 5, contorul este incrementat. Dacă coincidențele ating 5, procesul nu mai incrementează contorul, ceea ce este un comportament corect conform cerinței.

(d) Ieșirea out:

Ieșirea out este setată la 1 când contorul ajunge la valoarea 5, semnalizând astfel prezența a 5 coincidențe. Acest lucru este realizat într-un bloc `always`, care este sincronizat cu semnalul de ceas și resetat corect atunci când este necesar.

(e) Claritatea și structurarea codului:

Codul este ușor de înțeles și bine structurat. Folosirea a două blocuri `always` separate pentru contorizarea și semnalizarea rezultatului ajută la separarea logicii de calcul a coincidențelor de logica de generare a ieșirii.

Posibile îmbunătățiri suplimentare:**(a) Îmbunătățirea controlului numărătorului:**

Poți adăuga un control pentru a preveni o posibilă eroare de suprascriere a contorului, de exemplu, să nu se incrementeze contorul dacă acesta a ajuns deja la 5. Deși acest comportament este implicit în implementarea ta, poate fi util să adaugi un comentariu sau o logică suplimentară pentru clarificare.

(b) Optimizați gestionarea resetării:

În prezent, resetarea numărătorului se face pe baza unei evaluări a semnalului XOR, dar ai putea să adaugi o logică suplimentară în care resetarea se face numai în momentul în care semnalul A și B diferă pentru mai mult de un ciclu de

ceas, dacă vrei o protecție suplimentară împotriva semnalelor zgomotoase.

În concluzie: Codul tău este corect și eficient, respectând cerințele problemei și având o structură clară. Implementarea este simplă, utilizând tehnici de bază din Verilog pentru a realiza automatul.

Sunt măgulit! Mulțumesc pentru complimente, ChatGPT!

13. Proiectați un automat care determină și semnalează pe ieșire valoarea majoritară a valorilor primite pe intrare în ultimele 3 perioade de ceas. Determinați secvența de ieșire în cazul aplicării la intrare a secvenței 00111110001100101010100111. Model Verilog.
14. Proiectați un automat care semnalează pe ieșire detectarea unui șir de biți aplicați secvențial pe intrare cu valorile 10111. Pattern-ul semnalat se poate suprapune peste o instanță anterioară. Figura 13.10 prezintă un exemplu de intrare X și ieșirea Z asociată.

X: 01010111001011001011101111001 ...
 Z: 00000000000000000000000000000000 ...

Figura 13.10 Exemplu de valori numerice pentru automatul care depistează secvența 10111.

Rezolvați aceeași problemă pentru secvențele: 11011, 00111, 01011.

Modelați în Verilog un circuit care acceptă ca parametru secvența depistată, de lungime 5 biți.

Propuneți o modificare a modelului pentru a permite particularizarea secvenței depistate de lungime maximă 5 biți (poate fi și mai scurtă).

13.4 Pentru cei ce vor să devină profesioniști

Să se proiecteze un circuit digital sincron care să detecteze apăsarea unui buton.

Circuitul generează două semnale:

1. *butonPush*: un puls la fiecare apăsare pe buton (lățimea unui puls este egală cu o perioadă de ceas),
2. *butonAutoRepeat*: un tren de impulsuri pentru fiecare apăsare pe buton care durează mai mult de o secundă (lățimea unui puls este egală cu o perioadă de ceas, intervalul între două pulsuri succesive este de 100 ms).

Schema bloc a sistemului proiectat este prezentată în figura 13.11.

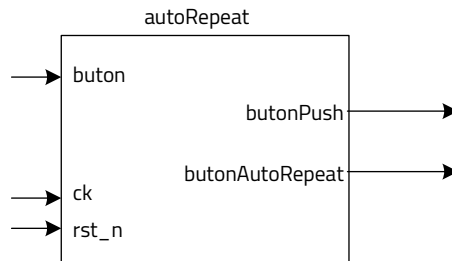


Figura 13.11 Schema bloc a circuitului de generare de pulsuri (problema 13.4).

Forme de undă asociate sistemului de generare de pulsuri sunt prezentate în figura 13.12.

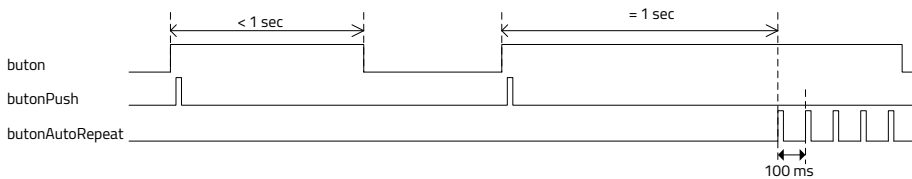


Figura 13.12 Forme de undă asociate sistemului de generare de pulsuri (problema 13.4).

Sistemul este sincron, având semnalul de ceas cu frecvență de 50MHz și semnalul asincron de resetare activ în 0. Intrarea *buton* provine de la un buton cu acționare manuală.

Se consideră că intrarea este deja filtrată (*debounced*, engl.):

- *buton* = 1 semnifică „buton apăsător”,
- *buton* = 0 semnifică „buton relaxat”.

O apăsare pe buton nu poate fi mai scurtă de 100ms . La fiecare apăsare pe buton, indiferent de durata acesteia, se generează un puls pe ieșirea *butonPush*, cu lățime de o perioadă de tact.

Dacă apăsarea pe buton este mai lungă decât 1 secundă, circuitul generează pe ieșirea *butonAutoRepeat* un șir repetitiv de impulsuri, cu cadența de 10 pulsuri pe secundă. Lățimea unui puls este o perioadă de tact, intervalul între două pulsuri este de 100ms .

Tabel cu descrierea porturilor:

Nume Port	Direcție	Dimensiune	Descriere
<i>ck</i>	in	1	Semnal de ceas cu frecvența de 50MHz, activ pe front pozitiv.
<i>rst_n</i>	in	1	Semnal de reset asincron, activ pe 0.
<i>buton</i>	in	1	Intrare (activă 1) provenită de la un buton cu apăsare. Se consideră că intrarea este deja filtrată și toate zgomotele mecanice eliminate.
<i>butonPush</i>	out	1	Ieșire pe care apare un puls (lățime o perioadă de ceas) la fiecare apăsare pe buton.
<i>butonAutoRepeat</i>	out	1	Ieșire pe care apar pulsuri repetitive (lățime o perioadă de ceas, interval între două pulsuri de 100 ms) în cazul în care butonul este ținut apăsător mai mult de 1 secundă.

Modelați circuitul în Verilog și verificați-l prin simulare.

Anexa 14

Probleme de examen

14.1 Subiectul 1

14.1.1 A. Noțiuni generale

- **A-1** Care este numărul minim de rezistențe de valoare $1K\Omega$ cu care se poate realiza un circuit cu o rezistență echivalentă de $1.75K\Omega$? Desen. Justificarea soluției propuse.

Soluție

Soluția se bazează pe următoarele observații:

- Mai multe rezistențe conectate în serie prezintă o rezistență echivalentă egală cu suma rezistențelor. N rezistențe egale, de valoare R , conectate în serie, au o rezistență echivalentă egală cu $R \times N$.
- N rezistențe egale, de valoare R , conectate în paralel, prezintă o rezistență echivalentă egală cu R/N .

Se observă că valoarea (absolută) a rezistenței cerute de problemă (1.75) se poate descompune în $1.75 = 1 + 0.5 + 0.25 = 1 + 1/2 + 1/4$.

Implementarea bazată pe această formulă implică 3 grupuri de rezistențe conectate în serie: R , $R/2$ (două rezistențe în paralel) și $R/4$ (patru rezistențe în paralel). Circuitul este prezentat în figura 14.1, are rezistența echivalentă $1.75K\Omega$ și este realizat cu 7 rezistențe de $1K\Omega$.

- **A-2** Efectuați următoarea operație în baza 16. Ulterior, exprimați rezultatul în baza 2.

$$BABA + BEBE = ?$$

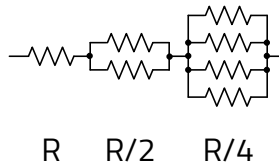


Figura 14.1 Soluția problemei 14.1.1-A-1.

Soluție

Se pun numerele hexazecimale unul sub altul și se adună cifră cu cifră din dreapta (lsb) în stânga (msb), luând în considerare transportul spre bitul superior.

- $A|_{16} + E|_{16} = 10|_{10} + 14|_{10} = 24|_{10} = 16|_{10} + 8|_{10} = 18|_{16}$, rezultat 8, transport 1.
- $B|_{16} + B|_{16} + 1 = 11|_{10} + 11|_{10} + 1 = 23|_{10} = 16|_{10} + 7|_{10} = 17|_{16}$, rezultat 7, transport 1.
- $A|_{16} + E|_{16} + 1 = 10|_{10} + 14|_{10} + 1 = 25|_{10} = 16|_{10} + 9|_{10} = 19|_{16}$, rezultat 9, transport 1.
- $B|_{16} + B|_{16} + 1 = 11|_{10} + 11|_{10} + 1 = 23|_{10} = 16|_{10} + 7|_{10} = 17|_{16}$, rezultat 7, transport 1.

```

1111      <-transport
BABA +
BEBE
-----
17978

```

Rezultă $BABA|_{16} + BEBE|_{16} = 17978|_{16} = 1_0111_1001_0111_1000|_2$.

- **A-3** Completați forma de undă a ieșirii unui registru secvențial sincron de 4 biți a cărui funcționare este descrisă de tabelul următor.

LD	ROL	ROR	$Q^+[3:0]$	Descriere
1	X	X	$DATA[3:0]$	Încărcare
0	1	X	$Q[2:0], Q[3]$	Rotatie stânga
0	0	1	$Q[0], Q[3:1]$	Rotatie dreapta
0	0	0	$Q[3:0]$	Păstrează starea

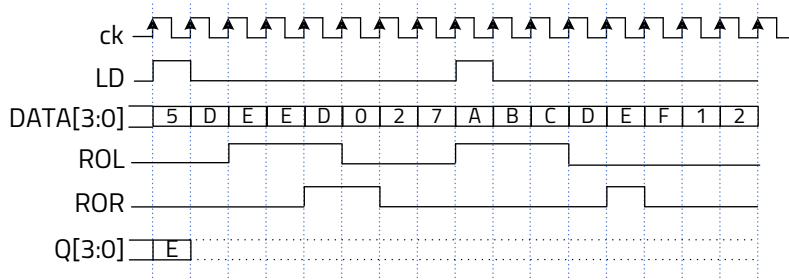


Figura 14.2 Forme de undă pentru problema 14.1.1-A-3.

Soluție

La fiecare front activ al semnalului de ceas, se va considera cea mai prioritară acțiune. Explicații în figura 14.3.

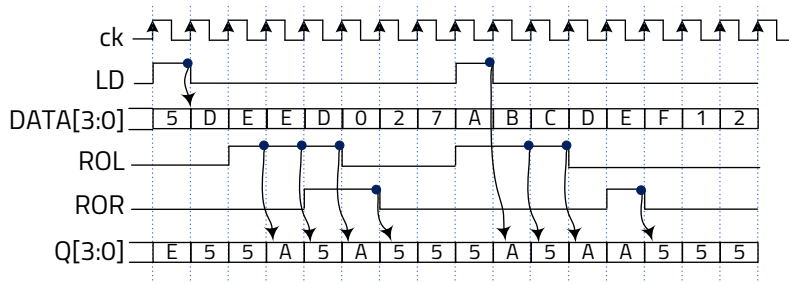


Figura 14.3 Soluția problemei 14.1.1-A-3.

- Intrarea LD este activă, se încarcă $DATA$ și starea Q devine $5_{16} = 0101_2$

- Nicio intrare nu este activă, se păstrează starea 5
 - Intrarea *ROL* este activă, se rotește starea spre stânga, din $5|_{16} = 0101|_2$ devine $1010|_2 = A|_{16}$
 - Intrarea *ROL* este activă, se rotește starea spre stânga, din $A|_{16} = 1010|_2$ devine $0101|_2 = 5|_{16}$
 - Intrarea *ROL* este activă, se rotește starea spre stânga, din $5|_{16} = 0101|_2$ devine $1010|_2 = A|_{16}$
 - Intrarea *ROR* este activă, se rotește starea spre dreapta, din $A|_{16} = 1010|_2$ devine $0101|_2 = 5|_{16}$
 - Nicio intrare nu este activă, se păstrează starea 5
 - Nicio intrare nu este activă, se păstrează starea 5
 - Intrarea *LD* este activă, se încarcă *DATA* și starea *Q* devine $A|_{16} = 1010|_2$
 - Intrarea *ROL* este activă, se rotește starea spre stânga, din $5|_{16} = 0101|_2$ devine $1010|_2 = A|_{16}$
 - Intrarea *ROL* este activă, se rotește starea spre stânga, din $A|_{16} = 1010|_2$ devine $0101|_2 = 5|_{16}$
 - Nicio intrare nu este activă, se păstrează starea *A*
 - Intrarea *ROR* este activă, se rotește starea spre dreapta, din $A|_{16} = 1010|_2$ devine $0101|_2 = 5|_{16}$
 - Nicio intrare nu este activă, se păstrează starea 5
 - Nicio intrare nu este activă, se păstrează starea 5
- **A-4** Prezentați rezultatul operației cu numere întregi (baza 10), cu operanzi codificați pe 16 biți în complement față de 2.

$$(-2024) + (-10) = ?$$

Soluție

$$(-2024) + (-10) = -2034$$

Se convertesc în baza 2 numerele pozitive exprimate în baza 10 și se extinde reprezentarea la 16 biți.

$$2024|_{10} = 0000_0111_1110_1000|_2$$

$$10|_{10} = 0000_0000_0000_1010|_2$$

$$2034|_{10} = 0000_0111_1111_0010|_2$$

Se determină reprezentarea în complement față de 2 a numerelor negative.

Se poate utiliza algoritmul rapid:

1) se consideră biți reprezentării numărului pozitiv din dreapta în stânga,
 2) se copiază toți biții până la primul 1 inclusiv, apoi se completează toți biții.

Rezultă:

$$-2024|_{10} = 1111_1000_0001_1000|_{C2}$$

$$-10|_{10} = 1111_1111_1111_0110|_{C2}$$

$$-2034|_{10} = 1111_1000_0000_1110|_{C2}$$

Se pun numerele exprimate în complement față de 2 unul sub altul și se adună în binar, cu considerarea transportului, de la lsb (dreapta) spre msb (stânga). Bitul de semn rezultă din operație iar eventualul transport (bitul al 17-lea) se ignoră.

	1	1111	1111	1110	000	<-transport
-2024 +		1111	1000	0001	1000	+
- 10		1111	1111	1111	0110	
-----		-----	-----	-----	-----	
-2034		1111	1000	0000	1110	

- **A-5** Utilizați operatori de reducere pentru a modela în Verilog următorii indicatori referitori la intrarea data.

```

module opReducere (
input  [15:0] data ,
output      data_div16 , // data este divizibila cu 16
output      data_minus1 , // data este egala cu -1
                        // (complement fata de 2)
output      data_gt15 , // data este mai mare decat 15
output      data_4_15   // data este intre 4 si 15
);

assign data_div16 = ..?.. ; // completati codul
assign data_minus1 = ..?.. ;
assign data_gt15 = ..?.. ;
assign data_4_15 = ..?.. ;

endmodule // opReducere
  
```

Soluție

Se consideră exprimarea condițiilor aritmetice prin condiții logice astfel:

„Data este divizibilă cu 16” înseamnă că, în reprezentarea binară, numărul are „*cei mai puțin semnificativi 4 biți egali cu 0*”. Deci funcția logică SAU/OR pe cei mai puțin semnificativi 4 biți trebuie să returneze valoarea 0.

„Data este egală cu -1” înseamnă că, în reprezentarea în complement față de 2, „*toți biții sunt egali cu 1*”. Deci funcția logică ȘI/AND pe toți biții trebuie să returneze valoarea 1.

„**Data este mai mare decât 15**” adică 16 și mai mare decât 16, înseamnă că, în reprezentarea binară, numărul are „*cel puțin un bit egal cu 1 cu index mai mare sau egal cu 4* ($2^4 = 16$)”. Deci funcția logică SAU/OR pe biții cu index mai mare sau egal cu 4 trebuie să returneze valoarea 1.

„**Data este între 4 și 15**” adică mai mare sau egal cu 4 ȘI mai mică decât 16, este echivalentă cu exprimarea „*există măcar un bit egal cu 1 între biții de index 2 și 3*” și „*biții cu index mai mare sau egal cu 4 sunt toți 0* (numărul nu este mai mare decât 15)”.

Rezultă următoarele atribuiri exprimate în Verilog:

```
assign data_div16 = ~|data[3:0] ; // ultimii 4 biti sunt 0
assign data_minus1 = &data ; // toti biti sunt 1
assign data_gt15 = |data[15:4] ; // exista un bit egal cu 1
// si index mai mare decat 3
assign data_4_15 = |data[3:2] & // mai mare sau egal cu 4 si
// mai mica decat 16
// (toti bitii cu index mai
// mare decat 3 sunt 0)
```

14.1.2 B. Circuite logice combinaționale

- **B-1** Demonstrați egalitatea:

$$\overline{A \oplus B \oplus C \oplus D} + \overline{A} \cdot B \cdot C = \overline{A \oplus B \oplus C \oplus D} + B \cdot C \cdot D$$

Soluție

Se observă că în cazul funcției SAU-EXCLUSIV (XOR) diagrama V-K prezintă căsuțe cu valoarea 1 într-o formă de tablă de șah.

Funcția $\overline{A \oplus B \oplus C \oplus D}$ are diagrama prezentată în figura 14.4-a).

Funcția din partea stângă conține suplimentar (OR) expresia $\overline{A} \cdot B \cdot C$, care corespunde suprafeței evidențiate în 14.4-b).

Funcția din partea dreaptă conține suplimentar (OR) expresia $B \cdot C \cdot D$, care corespunde suprafeței evidențiate în 14.4-c).

Se observă faptul că diagramele V-K prezentate în figurile 14.4-b) și c) sunt identice. Deci, egalitatea celor două funcții logice este demonstrată, ambele fiind egale cu:

$$\sum (0, 3, 5, 6, 7, 9, 10, 12, 15)$$

- **B-2** Minimizați funcția logică și implementați-o exclusiv cu porți NAND. Justificare.

$$F(A, B, C, D) = \sum (1, 9, 10, 13, 14) + d(7, 11, 15)$$

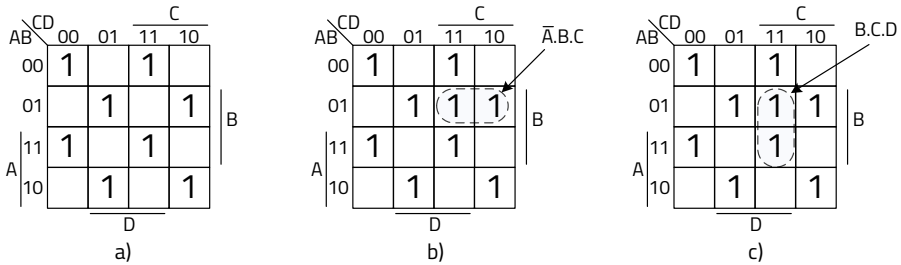


Figura 14.4 Soluția problemei 14.1.2-B-1.

- a) Diagrama V-K asociată funcției $A \oplus B \oplus C \oplus D$;
 b) Diagrama V-K asociată funcției din partea stângă, cu evidențierea suprafeței $A \cdot B \cdot C$;
 c) Diagrama V-K asociată funcției din partea dreaptă, cu evidențierea suprafeței $B \cdot C \cdot D$.

Determinați și justificați valoarea $F(1, 1, 1, 1) = ?$

Soluție

Se construiește diagrama V-K, conform expresiei funcției, punând valoarea 1 în căsuțele cu index menționat în lista de mintermi:

$$\sum (1, 9, 10, 13, 14)$$

Căsuțele cu valori indiferente, ale căror index este menționat în lista $d(7, 11, 15)$, se vor marca cu simbolul „-” (indiferent, engl „don't care”). Valorile acestor căsuțe se vor considera în procesul de minimizare ca având fie valoare 0, fie valoare 1, astfel încât expresia finală a funcției să fie minimă. Diagrama V-K asociată este prezentată în figura 14.5.

După definirea suprafețelor, rezultă expresia minimă a funcției:

$$F(A, B, C, D) = A \cdot C + A \cdot D + \overline{B} \cdot \overline{C} \cdot D$$

Circuitul asociat, implementat cu porți NAND pe două nivele, este prezentat în figura 14.5. Fiecare implicant prim generează câte o poartă NAND. Ieșirile acestora se conectează la intrările unei alte porți NAND, având un număr de intrări egal cu numărul de suprafețe definite în diagrama V-K (numărul implicantilor primi prezenți în expresia minimă a funcției).

Dacă o căsuță indiferentă a fost acoperită de o suprafață, înseamnă că minterm-ul asociat acesteia a fost considerat a fi prezent în expresia funcției.

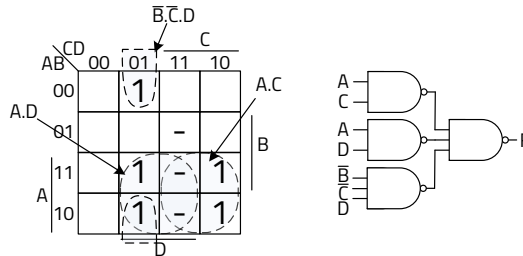


Figura 14.5 Diagrama V-K pentru problema 14.1.2-B-2. Circuitul proiectat.

Dacă o căsuță indiferentă nu a fost acoperită de nicio suprafață, înseamnă că minterm-ul asociat acesteia a fost considerat absent din expresia funcției.

$F(1, 1, 1, 1) = 1$, deoarece căsuța respectivă a fost acoperită cu două suprafețe.

$F(0, 1, 1, 1) = 0$, deoarece căsuța respectivă nu a fost acoperită de nicio suprafață.

14.1.3 C. Circuite logice secvențiale

- **C-1** Modelați în Verilog circuitul secvențial sincron care funcționează conform tabelului prezentat la punctul 14.1.1-A-3.

a) Simbol bloc.

b) Cod Verilog cu comentarii.

Soluție

Simbolul bloc este prezentat în figura 14.6. Se observă existența porturilor de intrare pentru semnalul de ceas și cel de reset, semnale ne-menționate în tabelul de funcționare. Codul Verilog care modelează registrul este următorul:

```
module reg_seq (
input      clk      , // semnal de ceas
input      rst_n    , // semnal de reset activ low
input      LD       , // incarcare date
input      [3:0] DATA , // data de intrare
input      ROL      , // rotatie stanga
input      ROR      , // rotatie dreapta
output reg [3:0] Q    // iesire registru
);
```

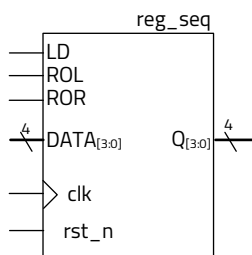


Figura 14.6 Simbolul bloc, problema 14.1.3-C-1.

```

always @(posedge clk or negedge rst_n)
if (~rst_n)  Q <= 4'b0; else                // reset
if (LD)      Q <= DATA; else                // incarca DATA
if (ROL)     Q <= {Q[2:0], Q[3]}; else       // rotatie stanga
if (ROR)     Q <= {Q[0], Q[3:1]};           // rotatie dreapta

endmodule // reg_seq

```

- **C-2** Comportamentul unui automat secvențial sincron este descris prin graful de tranziții prezentat în figura 14.7.

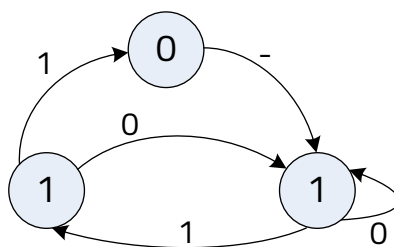


Figura 14.7 Graf, problema 14.1.3.

- Implementați automatul cu un număr minim de bistabile D și porți logice NAND.
- Completați formele de undă ale stării și ieșirii în cazul în care intrarea

are tot timpul valoarea 1.

Soluție

Din graful de tranziții prezentat se deduc caracteristicile automatului:

Automat Moore ieșirea este menționată în simbolul de stare, deci nu depinde de intrare.

Automatul are 3 stări deoarece apar 3 simboluri de stare (cercuri), noduri ale grafului. Pentru codarea stărilor sunt necesari minimum 2 biți ($2^2 \geq 3$). În cazul codării stărilor cu 2 biți, va trebui decis comportamentul automatului în cazul apariției stării neprezente în graf. Denumim biții stării prezente Q_1 și Q_0 , iar cei ai stării viitoare D_1 și D_0 .

Automatul are o singură intrare deoarece pe arce apare o singură valoare binară. Denumim intrarea X .

Automatul are o singură ieșire deoarece în nod este menționată o singură valoare binară. Denumim ieșirea Y .

Figura 14.8 prezintă graful actualizat prin specificarea celei de-a patra stări și codarea stărilor cu 2 biți. În aceeași figură este prezentat simbolul bloc al automatului proiectat.

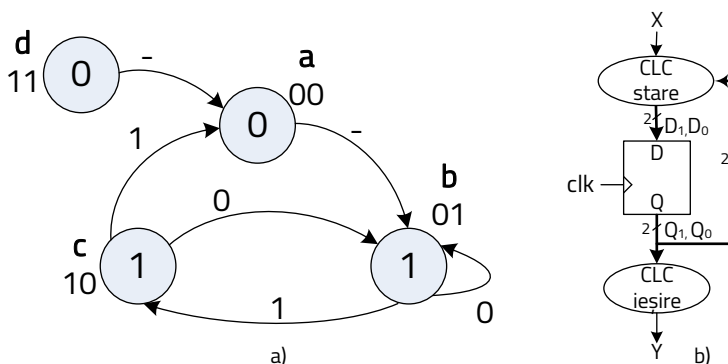


Figura 14.8 Graf de tranziții (a) și simbolul bloc (b) pentru problema 14.1.3 C-2.

Pe baza informațiilor prezentate pe graf, se completează tabelul de tranziții al automatului.

Intrare X	Stare prezentă Q_1Q_0	Stare viitoare D_1D_0	Ieșire Y
-	a 00	b 01	0
0	b 01	b 01	1
1	b 01	c 10	
0	c 10	b 01	1
1	c 10	a 00	
-	d 11	a 00	0

Pe baza informațiilor prezente în tabelul de tranziții, se completează diagramele V-K ale ieșirilor: $D_1(X, Q_1, Q_0)$, $D_0(X, Q_1, Q_0)$ și $Y(Q_1, Q_0)$.

Diagramele V-K, prezentate în figura 14.9, se minimizează astfel:

$$D_1(X, Q_1, Q_0) = X \cdot \overline{Q_1} \cdot Q_0$$

$$D_0(X, Q_1, Q_0) = \overline{X} \cdot \overline{Q_0} + \overline{Q_1} \cdot \overline{Q_0} + \overline{X} \cdot \overline{Q_1}$$

$$Y(Q_1, Q_0) = \overline{Q_1} \cdot Q_0 + Q_1 \cdot \overline{Q_0}$$

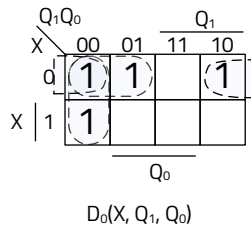


Figura 14.9 Diagrame V-K și minimizare pentru problema 14.1.3-C-2.

Implementarea se face cu două bistabile D pentru memorarea stării și porți NAND pentru logică, conform circuitului prezentat în figura 14.10-a.

În cazul în care intrarea are tot timpul valoarea 1, se observă că automatul cicleză între stările a, b, c , prezentând pe ieșire secvența periodică 011, așa ca în figura 14.10-b.

14.1.4 D. Proiectare sistem digital

Proiectați un circuit de permutare a doi biți, pe baza unui bit de selecție. Utilizați două circuite MUX 2:1. Desen cu multiplexoare. Justificare. Cod Verilog.

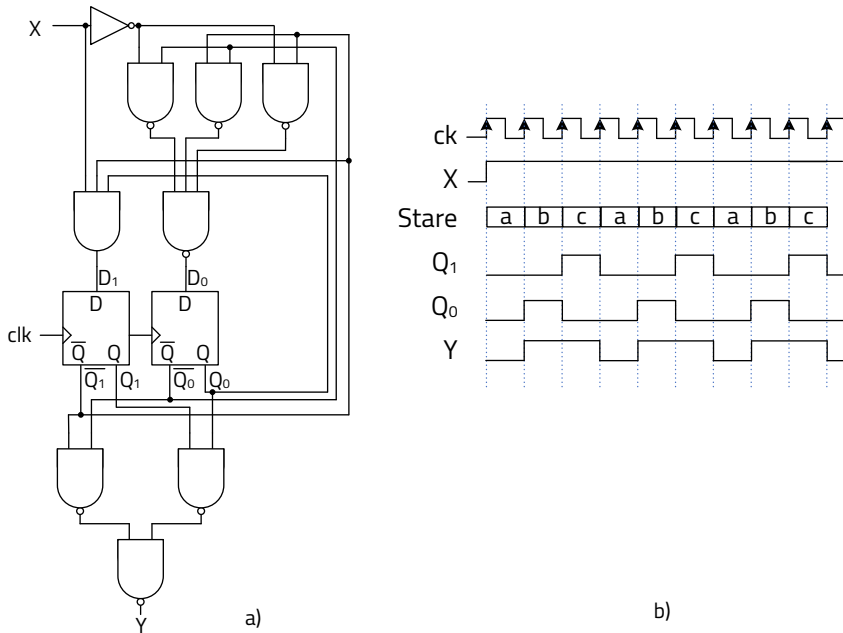


Figura 14.10 Rezolvare problema 14.1.3-C-2. a) Circuit b) Forme de undă.

```

module permutare (
input  iA, iB,
input  sel, // 0: oA=iA, oB=iB
           // 1: oA=iB, oB=iA
output oA, oB
);
// codul aici
endmodule // permutare

```

Soluție

Circuitul are două intrări pentru date și o intrare pentru controlul funcției. Circuitul are două ieșiri. Ieșirile se conectează la ieșirile celor două multiplexoare. Intrarea de control se conectează pe intrările de selecție ale celor două multiplexoare. Cele două multiplexoare vor avea intrările de date inversate.

Circuitul proiectat pentru a permuta 2 biți este prezentat în figura 14.11.

Codul Verilog care modelează circuitul combinațional de permutare a

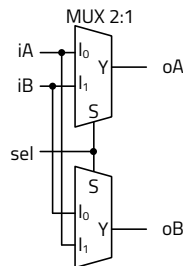


Figura 14.11 Circuit de permutare a doi biți, pe baza unui bit de selecție, utilizând două circuite MUX 2:1.

celor doi biți este următorul:

```
module permutare (
input  iA, iB,
input  sel, // 0: oA=iA, oB=iB
           // 1: oA=iB, oB=iA
output oA, oB
);

assign oA = sel ? iB : iA ;
assign oB = sel ? iA : iB ;

endmodule // permutare
```

14.2 Subiectul 2

14.2.1 A. Noțiuni generale

- **A-1** Care este numărul minim de rezistențe de valoare $1K\Omega$ cu care se poate realiza un circuit cu o rezistență echivalentă de $2.5K\Omega$? Desen. Justificarea soluției propuse.

Soluție

Pe baza observațiilor menționate la subiectul 14.1.1-A-1, se observă că valoarea dorită a rezistenței (2.5) se poate descompune în:

$$2.5 = 1 + 1 + 0.5 = 1 + 1 + 1/2$$

Implementarea bazată pe această formulă implică 2 rezistențe conectate în serie, în serie cu un grup de 2 rezistențe conectate în paralel. Circuitul este prezentat în figura 14.12, are rezistența echivalentă $2.5K\Omega$ și este realizat cu 4 rezistențe de $1K\Omega$.

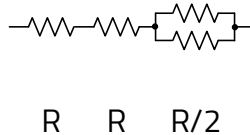


Figura 14.12 Soluția problemei 14.2.1-A-1.

- **A-2** Efectuați următoarea operație în baza 16. Ulterior, exprimați rezultatul în baza 2.

$$CADE + BABA = ?$$

Soluție

Se pun numerele hexazecimale unul sub altul și se adună cifră cu cifră din dreapta (lsb) în stânga (msb), luând în considerare transportul spre bitul superior.

- $E|_{16} + A|_{16} = 14|_{10} + 10|_{10} = 24|_{10} = 16|_{10} + 8|_{10} = 18|_{16}$, rezultat 8, transport 1.
- $D|_{16} + B|_{16} + 1 = 13|_{10} + 11|_{10} + 1 = 25|_{10} = 16|_{10} + 9|_{10} = 19|_{16}$, rezultat 9, transport 1.
- $A|_{16} + A|_{16} + 1 = 10|_{10} + 10|_{10} + 1 = 21|_{10} = 16|_{10} + 5|_{10} = 15|_{16}$, rezultat 5, transport 1.
- $C|_{16} + B|_{16} + 1 = 12|_{10} + 11|_{10} + 1 = 24|_{10} = 16|_{10} + 8|_{10} = 18|_{16}$, rezultat 8, transport 1.

```

1111      <-transport
CADE  +
BABA

```

18598

Rezultă $CADE|_{16} + BABA|_{16} = 18598|_{16} = 1_1000_0101_1001_1000|_2$.

- **A-3** Completați forma de undă a ieșirii unui registru secvențial sincron de 4 biți a cărui funcționare este descrisă de tabelul următor.

LD1	LD2	CU	CD	ROR	$Q^+[3:0]$	Descriere
1	X	X	X	X	$DATA1[3:0]$	Încărcare din sursa 1
0	1	X	X	X	$DATA2[3:0]$	Încărcare din sursa 2
0	0	1	X	X	$Q + 1$	Increment
0	0	0	1	X	$Q - 1$	Decrement
0	0	0	0	1	$Q[0], Q[3:1]$	Rotatie dreapta
0	0	0	0	0	$Q[3:0]$	Păstrează starea

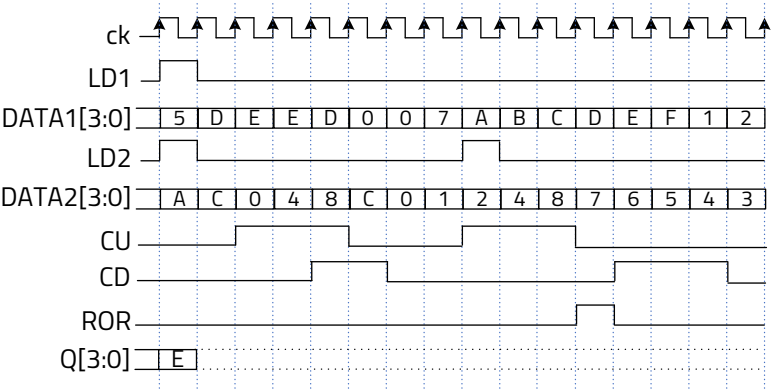


Figura 14.13 Forme de undă pentru problema 14.2.1.

Soluție

La fiecare front activ al semnalului de ceas, se va considera cea mai prioritară acțiune. Explicații în figura 14.14.

- Intrarea LD1 este activă, se încarcă DATA1 și starea Q devine $5|_{16} = 0101|_2$

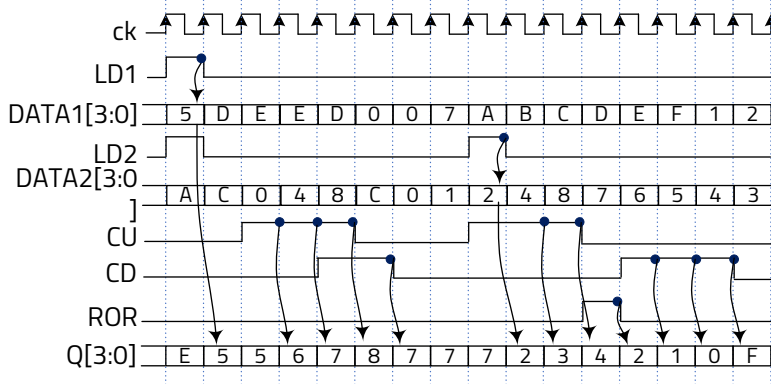


Figura 14.14 Soluția problemei 14.2.1-A-3.

- Nicio intrare nu este activă, se păstrează starea 5
 - Intrarea *CU* este activă, se incrementează starea, din 5_{16} devine 6_{16}
 - Intrarea *CU* este activă, se incrementează starea, din 6_{16} devine 7_{16}
 - Intrarea *CU* este activă, se incrementează starea, din 7_{16} devine 8_{16}
 - Intrarea *CD* este activă, se decrementează starea, din 8_{16} devine 7_{16}
 - Nicio intrare nu este activă, se păstrează starea 7
 - Nicio intrare nu este activă, se păstrează starea 7
 - Intrarea *LD2* este activă, se încarcă *DATA2* și starea *Q* devine $2_{16} = 0010_2$
 - Intrarea *CU* este activă, se incrementează starea, din 2_{16} devine 3_{16}
 - Intrarea *CU* este activă, se incrementează starea, din 3_{16} devine 4_{16}
 - Intrarea *ROR* este activă, se rotește starea spre dreapta, din $4_{16} = 0100_2$ devine $0010_2 = 2_{16}$
 - Intrarea *CD* este activă, se decrementează starea, din 2_{16} devine 1_{16}
 - Intrarea *CD* este activă, se decrementează starea, din 1_{16} devine 0_{16}
 - Intrarea *CD* este activă, se decrementează starea, din $0_{16} = 0000_2$ devine $1111_2 = F_{16}$
 - Nicio intrare nu este activă, se păstrează starea F_{16}
- **A-4** Prezentați rezultatul operației cu numere întregi (baza 10), cu operanzi codificați pe 16 biți în complement față de 2.

$$10 + (-2024) = ?$$

Soluție

$$10 + (-2024) = -2014$$

Se convertesc în baza 2 numerele pozitive exprimate în baza 10 și se extinde reprezentarea la 16 biți.

$$2024|_{10} = 0000_0111_1110_1000|_2$$

$$10|_{10} = 0000_0000_0000_1010|_2$$

$$2014|_{10} = 0000_0111_1101_1110|_2$$

Se determină reprezentarea în complement față de 2 a numerelor negative.

Se poate utiliza algoritmul rapid:

- 1) se consideră biți reprezentării numărului pozitiv din dreapta în stânga,
- 2) se copiază toți biții până la primul 1 inclusiv, apoi se complementează toți biții.

Rezultă:

$$-2024|_{10} = 1111_1000_0001_1000|_{C2}$$

$$-2014|_{10} = 1111_1000_0010_0010|_{C2}$$

Se pun numerele exprimate în complement față de 2 unul sub altul și se adună în binar, cu considerarea transportului, de la lsb (dreapta) spre msb (stânga). Bitul de semn rezultă din operație iar eventualul transport (bitul al 17-lea) se ignoră.

	0 0000 0000 0011 000	<-transport
10 +	0000_0000_0000_1010+	
-2024	1111_1000_0001_1000	
-----	-----	
-2014	1111_1000_0010_0010	

- **A-5** Utilizați operatori de reducere pentru a modela în Verilog următorii indicatori referitori la intrarea data.

```

module opReducere (
input  [15:0] data          ,
output      data_div64      , // data este divizibila cu 64
output      data_zero       , // data este egala cu zero
output      data_gt63_div8  , // data este mai mare decat 63
                                // si divizibila cu 8
output      data_4_63       , // data este intre 4 si 63
);

assign data_div64      = ..?... ; // completati codul

```

```

assign data_zero      = ..?... ;
assign data_gt63_div8 = ..?... ;
assign data_4_63      = ..?... ;

endmodule // opReducere

```

Soluție

Se consideră exprimarea condițiilor aritmetice prin condiții logice astfel:

„Data este divizibilă cu 64” înseamnă că, în reprezentarea binară, numărul are „*cei mai puțin semnificativi 6 biți egali cu 0*”. Deci funcția logică SAU/OR pe cei mai puțin semnificativi 6 biți trebuie să returneze valoarea 0.

„Data este egală cu 0” înseamnă că „*toți biții sunt egali cu 0*”. Deci funcția logică SAU/OR pe toți biții trebuie să returneze valoarea 0.

„Data este mai mare decât 63 și divizibilă cu 8” înseamnă că *cei mai puțin semnificativi 3 biți sunt egali cu 0* (divizibil cu $2^3 = 8$) și „*cel puțin un bit egal cu 1 cu index mai mare sau egal cu 6* ($2^6 = 64$)”. Deci funcția logică SAU/OR pe biții cu index mai mare sau egal cu 6 trebuie să returneze valoarea 1 și funcția logică SAU/OR pe biții cu index între 0 și 2 trebuie să returneze valoarea 0.

„Data este între 4 și 63” adică mai mare sau egal cu 4 ȘI mai mică decât 64, este echivalentă cu exprimarea „*există măcar un bit egal cu 1 între biții de index 2 și 3*” și „*biții cu index mai mare sau egal cu 6 sunt toți 0* (numărul nu este mai mare decât 63)”.

Rezultă următoarele atribuiri exprimate în Verilog:

```

assign data_div64 = ~|data[5:0] ; // ultimii 6 biți sunt 0
assign data_zero  = ~|data      ; // toți biții sunt 0
assign data_gt63_div8 = |data[15:6] & // mai mare decât 63,
                                // exista un bit 1 cu index
                                // între 15 și 6
                                ~|data[2:0] ; // ultimii 3 biți sunt 0
assign data_4_63  = |data[5:2] & // mai mare sau egal cu 4 și
                                ~|data[15:6] ; // mai mica decât 64
                                // (toți biții cu index mai
                                // mare decât 5 sunt 0)

```

14.2.2 B. Circuite logice combinaționale

- **B-1** Demonstrați egalitatea:

$$A \oplus B \oplus C \oplus D + \overline{A} \cdot B \cdot C = A \oplus B \oplus C \oplus D + \overline{A} \cdot C \cdot \overline{D}$$

Soluție

Se observă că în cazul funcției SAU-EXCLUSIV (XOR) diagrama V-K prezintă căsuțe cu valoarea 1 într-o formă de tablă de șah.

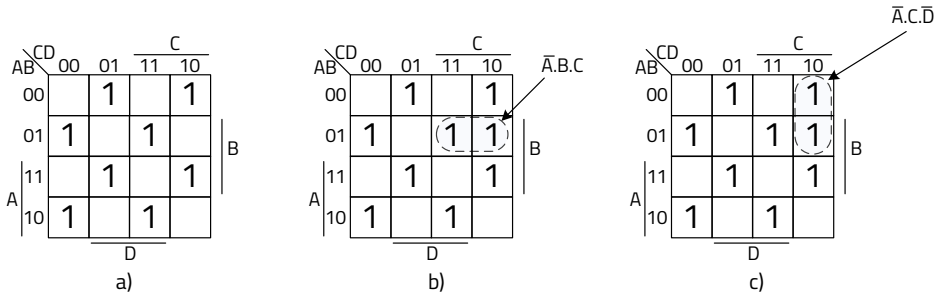


Figura 14.15 Soluția problemei 14.2.2-B-1.

- a) Diagrama V-K asociată funcției $A \oplus B \oplus C \oplus D$;
 b) Diagrama V-K asociată funcției din partea stângă, cu evidențierea suprafeței $\bar{A} \cdot B \cdot C$;
 c) Diagrama V-K asociată funcției din partea dreaptă, cu evidențierea suprafeței $\bar{A} \cdot C \cdot \bar{D}$.

Funcția $A \oplus B \oplus C \oplus D$ are diagrama prezentată în figura 14.15-a).

Funcția din partea stângă conține suplimentar (OR) expresia $\bar{A} \cdot B \cdot C$, care corespunde suprafeței evidențiate în 14.15-b).

Funcția din partea dreaptă conține suplimentar (OR) expresia $\bar{A} \cdot C \cdot \bar{D}$, care corespunde suprafeței evidențiate în 14.15-c).

Se observă faptul că diagramele V-K prezentate în figurile 14.15-b) și c) sunt identice. Deci, egalitatea celor două funcții logice este demonstrată, ambele fiind egale cu:

$$\sum (1, 2, 4, 6, 7, 8, 11, 13, 14)$$

- **B-2** Minimizați funcția logică și implementați-o exclusiv cu porți NAND. Justificare.

$$F(A, B, C, D) = \sum (0, 1, 3, 4, 5, 6, 7, 9) + d(10, 11, 14)$$

Determinați și justificați valoarea $F(1, 0, 1, 0) = ?$

Soluție

Se construiește diagrama V-K, conform expresiei funcției, punând valoarea 1 în căsuțele cu index menționat în lista de mintermi:

$$\sum (0, 1, 3, 4, 5, 6, 7, 9)$$

Căsuțele cu valori indifferente, ale căror index este menționat în lista $d(10, 11, 14)$, se vor marca cu simbolul „-”. Valorile acestor căsuțe se vor considera în procesul de minimizare ca având fie valoare 0 fie valoare 1 astfel încât expresia finală a funcției să fie minimă. Diagrama V-K asociată este prezentată în figura 14.16.

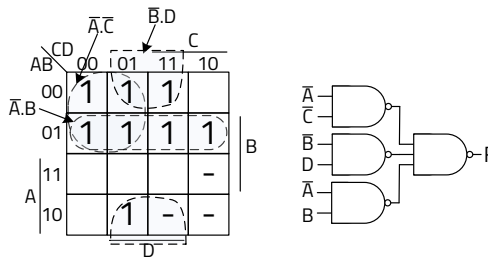


Figura 14.16 Diagrama V-K pentru problema 14.2.2. Circuitul proiectat.

După definirea suprafețelor, rezultă expresia minimă a funcției:

$$F(A, B, C, D) = \overline{A} \cdot \overline{C} + \overline{B} \cdot D + \overline{A} \cdot B$$

Circuitul asociat implementat cu porți NAND, pe două nivele, este prezentat în figura 14.16. Fiecare implicant prim generează câte o poartă NAND. Ieșirile acestora se conectează la intrările unei alte porți NAND, având un număr de intrări egal cu numărul de suprafețe definite în diagrama V-K (numărul implicantilor primi prezenți în expresia minimă a funcției).

Dacă o căsuță indiferentă a fost acoperită de o suprafață, înseamnă că minterm-ul asociat acesteia a fost considerat a fi prezent în expresia funcției.

Dacă o căsuță indiferentă nu a fost acoperită de nicio suprafață, înseamnă că minterm-ul asociat acesteia a fost considerat absent din expresia funcției.

$F(1, 0, 1, 0) = 0$, deoarece căsuța respectivă nu a fost acoperită cu nicio suprafață.

$F(1, 0, 1, 1) = 1$, deoarece căsuța respectivă a fost acoperită de suprafața $\overline{B} \cdot D$.

14.2.3 C. Circuite logice secvențiale

- **C-1** Modelați în Verilog circuitul secvențial sincron care funcționează conform tabelului prezentat la punctul 14.2.1-A-3.

- a) Simbol bloc.
- b) Cod Verilog cu comentarii.

Soluție

Simbolul bloc este prezentat în figura 14.17. Se observă existența porturilor de intrare pentru semnalul de ceas și cel de reset, semnale ne-menționate în tabelul de funcționare. Codul Verilog care modelează re-

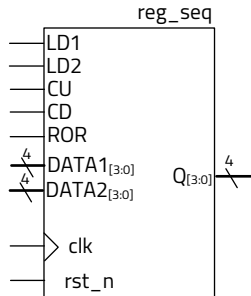


Figura 14.17 Simbolul bloc, problema 14.2.3-C-1.

gistrul este următorul:

```
module reg_seq (
    input          clk      , // semnal de ceas
    input          rst_n    , // semnal de reset activ low
    input          LD1      , // incarcare DATA1
    input          LD2      , // incarcare DATA2
    input [3:0]    DATA1   , // data de intrare
    input [3:0]    DATA2   , // data de intrare
    input          CU       , // increment (Count Up)
    input          CD       , // decrement (Count Down)
    input          ROR      , // rotatie dreapta
    output reg [3:0] Q      // iesire registru
);

always @(posedge clk or negedge rst_n)
    if (~rst_n)    Q <= 4'b0; else // reset
    if (LD1)       Q <= DATA1; else // incarca DATA1
    if (LD2)       Q <= DATA2; else // incarca DATA2
```

```

if (CU)      Q <= Q+1; else           // increment
if (CD)      Q <= Q-1; else           // decrement
if (ROR)     Q <= {Q[0], Q[3:1]};     // rotatie dreapta

endmodule // reg_seq

```

- **C-2** Comportamentul unui automat secvențial sincron este descris prin graful de tranziții prezentat în figura 14.18.

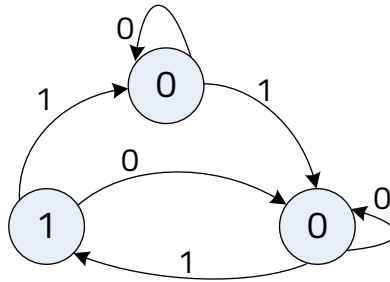


Figura 14.18 Graf, problema 14.2.3.

- Implementați automatul cu un număr minim de bistabile D și porți logice NAND.
- Completați formele de undă ale stării și ieșirii în cazul în care intrarea are tot timpul valoarea 1.

14.2.4 D. Proiectare sistem digital

Utilizați un număr minim de sumatoare complete de 1 bit pentru a determina suma a 5 biți de date. Desen cu simboluri de sumatoare complete de 1 bit. Justificare. Cod Verilog.

```

module sum (
input  [4:0] data , // 5 biti de date
output [2:0] sum    // suma bitilor data, in domeniul 0...5
);
// codul aici
endmodule // sum

```

Soluție

Sumatorul complet de un bit însumează 3 biți cu ponderi egale, 2 biți de

date plus bitul de transport. Rezultatul, în domeniul $[0...3]$, este exprimat pe 2 biți, bitul de sumă, cu pondere 2^0 și bitul de transport, cu pondere 2^1 . Trebuie avut în vedere algoritmul de însumare (similar cu însumarea în baza 10) care impune adunarea doar a biților de același ordin (adică nu se adună biți de ponderi diferite).

Soluția propusă, implementată cu sumatoare complete de 1 bit și prezentată în figura 14.19, poate fi descrisă astfel:

- primul sumator adună 3 biți și produce un bit de ordin 2^0 și transportul de ordin 2^1 ,
- al doilea sumator, adună ceilalți 2 biți rămași din 5 și bitul de ordin 2^0 de la ieșirea sumatorului precedent pentru a produce bitul 0 al rezultatului final și un alt bit de transport de ordin 2^1 ,
- al 3-lea sumator, adună cei 2 biți de transport de ordin 2^1 și cu un bit de valoare 0 și produce bitul final de ordin 2^1 și, pe ieșirea de transport, bitul final de ordin 2^2 .

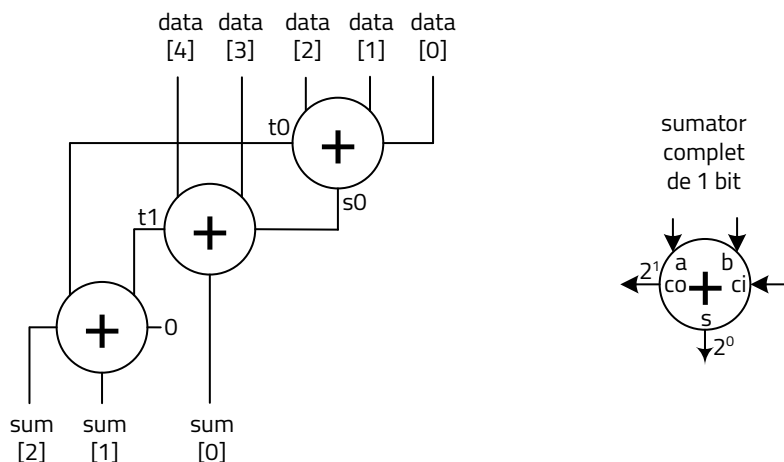


Figura 14.19 Soluția problemei 14.2.4-D. Detaliu cu simbolul bloc al sumatorului complet de 1 bit.

Rezultatul, în domeniul $[0...5]$, va fi reprezentat pe 3 biți.

Codul Verilog care modelează circuitul propus este prezentat în continuare:

```
module sum (
input  [4:0] data , // 5 biti de date
output [2:0] sum    // suma bitilor data, in domeniul 0...5
);

wire  s0, t0 ; // iesiri primul sumator
wire   t1 ; // iesire de transport al doilea sumator

assign { t0, s0 } = data[2] + data[1] + data[0]; // sum1
assign { t1, sum[0] } = s0 + data[4] + data[3]; // sum2
assign { sum[2], sum[1] } = 1'b0 + t1 + t0; // sum3

endmodule // sum
```

14.3 Subiectul 3

14.3.1 A. Noțiuni generale

- **A-1** Care este numărul minim de rezistențe de valoare $1K\Omega$ cu care se poate realiza un circuit cu o rezistență echivalentă de $2.33K\Omega$? Desen. Justificarea soluției propuse.

Soluție

Pe baza observațiilor menționate la subiectul 14.1.1-A-1, se observă că valoarea dorită a rezistenței (2.33) se poate descompune în:

$$2.33 = 1 + 1 + 0.33 = 1 + 1 + 1/3$$

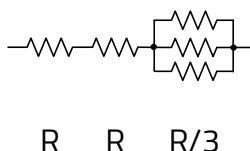
Implementarea bazată pe această formulă implică 2 rezistențe conectate în serie, în serie cu un grup de 3 rezistențe conectate în paralel. Circuitul este prezentat în figura 14.20, are rezistența echivalentă $2.33K\Omega$ și este realizat cu 5 rezistențe de $1K\Omega$.

- **A-2** Efectuați următoarea operație în baza 16. Ulterior, exprimați rezultatul în baza 2.

$$BEEF + CAFE = ?$$

Soluție

Se pun numerele hexazecimale unul sub altul și se adună cifră cu cifră din dreapta (lsb) în stânga (msb), luând în considerare transportul spre bitul superior.

**Figura 14.20** Soluția problemei 14.3.1-A-1.

- $F|_{16} + E|_{16} = 15|_{10} + 14|_{10} = 29|_{10} = 16|_{10} + 13|_{10} = 1D|_{16}$, rezultat D , transport 1.
- $E|_{16} + F|_{16} + 1 = 14|_{10} + 15|_{10} + 1 = 30|_{10} = 16|_{10} + 14|_{10} = 1E|_{16}$, rezultat E , transport 1.
- $E|_{16} + A|_{16} + 1 = 14|_{10} + 10|_{10} + 1 = 25|_{10} = 16|_{10} + 9|_{10} = 19|_{16}$, rezultat 9, transport 1.
- $B|_{16} + C|_{16} + 1 = 11|_{10} + 12|_{10} + 1 = 24|_{10} = 16|_{10} + 8|_{10} = 18|_{16}$, rezultat 8, transport 1.

```

1111      <-transport
BEEF  +
CAFE
-----
189ED

```

Rezultă $BEEF|_{16} + CAFE|_{16} = 189ED|_{16} = 1_1000_1001_1110_1101|_2$.

- **A-3** Completați forma de undă a ieșirii unui registru secvențial sincron de 4 biți a cărei funcționare este descrisă de tabelul următor.

LD	LFSR	CU	$Q^+[3:0]$	Descriere
1	X	X	$DATA[3:0]$	Încărcare
0	1	X	$Q[3] \oplus Q[0], Q[3:1]$	LFSR, polinom $x^4 + x^3 + 1$
0	0	1	$Q + 1$	Increment
0	0	0	$Q[3:0]$	Păstrează starea

Soluție

La fiecare front activ al semnalului de ceas, se va considera cea mai prioritară acțiune. Explicații în figura 14.22.

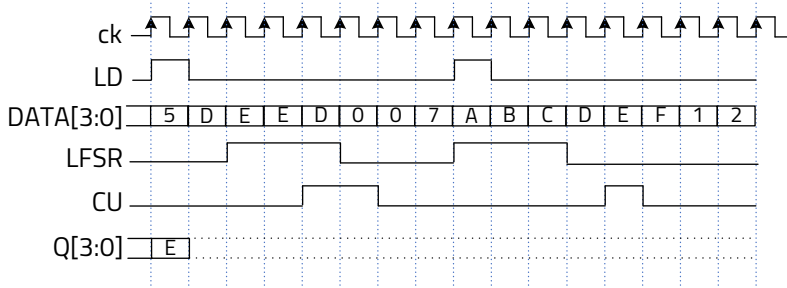


Figura 14.21 Forme de undă pentru problema 14.3.1.

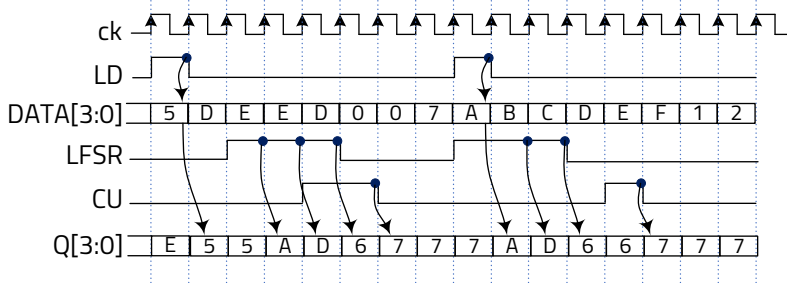


Figura 14.22 Soluția problemei 14.3.1-A-3.

- Intrarea *LD* este activă, se încarcă *DATA* și starea *Q* devine $5|_{16} = 0101|_2$
- Nicio intrare nu este activă, se păstrează starea 5
- Intrarea *LFSR* este activă, se incrementează starea, din $5|_{16} = 0101|_2$ devine $1010|_2 = A|_{16}$, conform polinomului $x^4 + x^3 + 1$, starea se deplasează spre dreapta, iar bitul mai semnificativ al stării viitoare $Q^+[3]$ se determină pe baza biților stării curente $Q^+[3] = Q[3] \oplus Q[0] = 0 \oplus 1 = 1$.
- Intrarea *LFSR* este activă, se incrementează starea, din $A|_{16} = 1010|_2$ devine $1101|_2 = D|_{16}$, conform polinomului $x^4 + x^3 + 1$, starea se deplasează spre dreapta, iar bitul mai semnificativ al stării viitoare $Q^+[3]$ se determină pe baza biților stării curente $Q^+[3] = Q[3] \oplus Q[0] = 1 \oplus 0 = 1$.
- Intrarea *LFSR* este activă, se incrementează starea, din $D|_{16} = 1101|_2$

- devine $0110|_2 = 6|_{16}$, conform polinomului $x^4 + x^3 + 1$, starea se deplasează spre dreapta, iar bitul mai semnificativ al stării viitoare $Q^+[3]$ se determină pe baza biților stării curente $Q^+[3] = Q[3] \oplus Q[0] = 1 \oplus 1 = 0$.
- Intrarea CU este activă, se incrementează starea, din $6|_{16}$ devine $7|_{16}$
 - Nicio intrare nu este activă, se păstrează starea 7
 - Nicio intrare nu este activă, se păstrează starea 7
 - Intrarea LD este activă, se încarcă $DATA$ și starea Q devine $A|_{16} = 1010|_2$
 - Intrarea $LFSR$ este activă, se incrementează starea, din $5|_{16} = 0101|_2$ devine $1010|_2 = A|_{16}$, conform polinomului $x^4 + x^3 + 1$, starea se deplasează spre dreapta, iar bitul mai semnificativ al stării viitoare $Q^+[3]$ se determină pe baza biților stării curente $Q^+[3] = Q[3] \oplus Q[0] = 0 \oplus 1 = 1$.
 - Intrarea $LFSR$ este activă, se incrementează starea, din $A|_{16} = 1010|_2$ devine $1101|_2 = D|_{16}$, conform polinomului $x^4 + x^3 + 1$, starea se deplasează spre dreapta, iar bitul mai semnificativ al stării viitoare $Q^+[3]$ se determină pe baza biților stării curente $Q^+[3] = Q[3] \oplus Q[0] = 1 \oplus 0 = 1$.
 - Nicio intrare nu este activă, se păstrează starea 6
 - Intrarea CU este activă, se incrementează starea, din $6|_{16}$ devine $7|_{16}$
 - Nicio intrare nu este activă, se păstrează starea 7
 - Nicio intrare nu este activă, se păstrează starea 7
- **A-4** Prezentați rezultatul operației cu numere întregi (baza 10), cu operandi codificați pe 16 biți în complement față de 2.

$$100 + (-2024) = ?$$

Soluție

$$100 + (-2024) = -1924$$

Se convertesc în baza 2 numerele pozitive exprimate în baza 10 și se extinde reprezentarea la 16 biți.

$$2024|_{10} = 0000_0111_1110_1000|_2$$

$$100|_{10} = 0000_0000_0110_0100|_2$$

$$1924|_{10} = 0000_0111_1000_0100|_2$$

Se determină reprezentarea în complement față de 2 a numerelor negative.

Se poate utiliza algoritmul rapid:

- 1) se consideră biți reprezentării numărului pozitiv din dreapta în stânga,
- 2) se copiază toți biții până la primul 1 inclusiv, apoi se complementează toți biții.

Rezultă:

$$-2024|_{10} = 1111_1000_0001_1000|_{C2}$$

$$-1924|_{10} = 1111_1000_0111_1100|_{C2}$$

Se pun numerele exprimate în complement față de 2 unul sub altul și se adună în binar, cu considerarea transportului, de la lsb (dreapta) spre msb (stânga). Bitul de semn rezultă din operație iar eventualul transport (bitul al 17-lea) se ignoră.

```

          0 0000 0000 0000 000    <-transport
100 +    0000_0000_0110_0100+
-2024    1111_1000_0001_1000
-----
-1924    1111_1000_0111_1100

```

- **A-5** Utilizați operatori de reducere pentru a modela în Verilog următorii indicatori referitori la intrarea data.

```

module opReducere (
input  [15:0] data      ,
output      data_div32 , // data este divizibila cu 32
output      data_minus2 , // data este egala cu -2
                        // (complement fata de 2)
output      data_lt32   , // data este mai mica decat 32
output      data_2_31    // data este intre 2 si 31
);

assign data_div32   = ..?.. ; // completati codul
assign data_minus2  = ..?.. ;
assign data_lt32    = ..?.. ;
assign data_2_31    = ..?.. ;

endmodule // opReducere

```

Soluție

Se consideră exprimarea condițiilor aritmetice prin condiții logice astfel:

„**Data este divizibilă cu 32**” înseamnă că, în reprezentarea binară, numărul are „*cei mai puțin semnificativi 5 biți egali cu 0*”. Deci funcția logică SAU/OR pe cei mai puțin semnificativi 5 biți trebuie să returneze valoarea 0.

„**Data este egală cu -2**” înseamnă că, în reprezentarea în complement față de 2, „*toți biții sunt egali cu 1, cu excepția bitului 0*”. Deci bitul 0 este 0 iar funcția logică ȘI/AND pe toți ceilalți biții trebuie să returneze valoarea 1.

„**Data este egală cu 0**” înseamnă că „*toți biții sunt egali cu 0*”. Deci funcția logică SAU/OR pe toți biții trebuie să returneze valoarea 0.

„Data este mai mică decât 32” înseamnă că „*toți biții mai semnificativi cu index mai mare sau egal cu 5 sunt 0*”. Deci funcția logică SAU/OR pe biții cu index mai mare sau egal cu 5 trebuie să returneze valoarea 0.

„Data este între 2 și 31” adică mai mare sau egal cu 2 ȘI mai mică decât 32, este echivalentă cu exprimarea „*există măcar un bit egal cu 1 între biții de index 1 și 4*” și „*biții cu index mai mare sau egal cu 5 sunt toți 0 (numărul nu este mai mare decât 31)*”.

Rezultă următoarele atribuiri exprimate în Verilog:

```
assign data_div32 = ~|data[4:0] ; // ultimii 5 biți sunt 0
assign data_minus2 = ~data[0] & // bitul 0 este egal cu 0 si
                    &data[15:1] ; // biții 1..15 egali cu 1
assign data_lt32 = ~|data[15:5] ; // biții cu index mai mare
                                // sau egal cu 5 sunt 0
assign data_2_31 = |data[4:1] & // mai mare sau egal cu 2 si
                    ~|data[15:5] ; // mai mica decat 32
                                // (toti biții cu index mai
                                // mare decat 4 sunt 0)
```

14.3.2 B. Circuite logice combinaționale

- **B-1** Demonstrați egalitatea:

$$A \cdot (B \oplus C \oplus D) + \bar{A} \cdot \bar{B} \cdot \bar{C} = A \cdot (B \oplus C \oplus D) + \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D} + \bar{B} \cdot \bar{C} \cdot D$$

Soluție

Se observă că în cazul funcției SAU-EXCLUSIV (XOR) diagrama V-K prezintă căsuțe cu valoarea 1 într-o formă de tablă de șah. Expresia $B \oplus C \oplus D$ este condiționată (ȘI) de variabila A . Deci, expresia $A \cdot (B \oplus C \oplus D)$ are căsuțe de 1 numai în suprafața în care $A = 1$.

Funcția $A \cdot (B \oplus C \oplus D)$ are diagrama prezentată în figura 14.23-a).

Funcția din partea stângă conține suplimentar (OR) expresia $\bar{A} \cdot \bar{B} \cdot \bar{C}$, care corespunde suprafeței evidențiate în 14.23-b).

Funcția din partea dreaptă conține suplimentar (OR) expresiile $\bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D}$ și $\bar{B} \cdot \bar{C} \cdot D$, care corespund suprafețelor evidențiate în 14.23-c).

Se observă faptul că diagramele V-K prezentate în figurile 14.23-b) și c) sunt identice. Deci, egalitatea celor două funcții logice este demonstrată, ambele fiind egale cu:

$$\sum (0, 1, 9, 10, 12, 15)$$

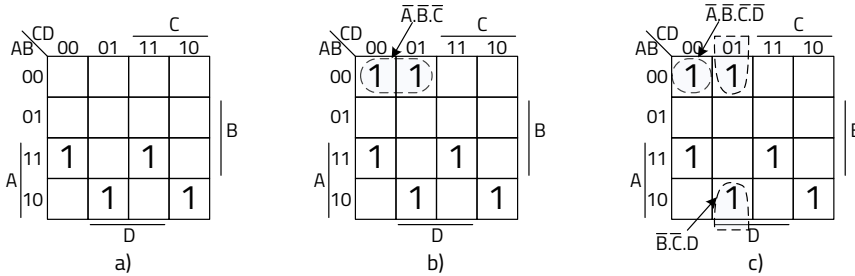


Figura 14.23 Soluția problemei 14.3.2-B-1.

- a) Diagrama V-K asociată funcției $A \cdot (B \oplus C \oplus D)$;
 b) Diagrama V-K asociată funcției din partea stângă, cu evidențierea suprafeței $\bar{A} \cdot \bar{B} \cdot \bar{C}$;
 c) Diagrama V-K asociată funcției din partea dreaptă, cu evidențierea suprafețelor $\bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D}$ și $\bar{B} \cdot \bar{C} \cdot D$.

- **B-2** Minimizați funcția logică și implementați-o exclusiv cu porți NAND. Justificare.

$$F(A, B, C, D) = \sum (1, 3, 7, 9, 12) + d(4, 11, 14)$$

Determinați și justificați valoarea $F(0, 1, 0, 0) = ?$

14.3.3 C. Circuite logice secvențiale

- **C-1** Modelați în Verilog circuitul secvențial sincron care funcționează conform tabelului prezentat la punctul 14.3.1-A-3.

- a) Simbol bloc.
 b) Cod Verilog cu comentarii.

Soluție

Simbolul bloc este prezentat în figura 14.24. Se observă existența porțurilor de intrare pentru semnalul de ceas și cel de reset, semnale ne-menționate în tabelul de funcționare. Codul Verilog care modelează registrul este următorul:

```
module reg_seq (
input      clk , // semnal de ceas
input      rst_n , // semnal de reset activ low
input      LD , // incarcare DATA
input      [3:0] DATA , // data de intrare
```

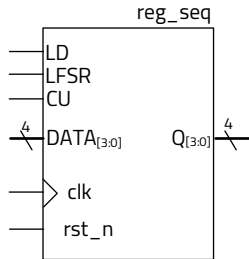


Figura 14.24 Simbolul bloc, problema 14.3.3-C-1.

```

input          LFSR , // LFSR X^4+X^3+1
input          CU   , // increment (Count Up)
output reg [3:0] Q   // iesire registru
);

always @(posedge clk or negedge rst_n)
if (~rst_n) Q <= 4'b0; else // reset
if (LD) Q <= DATA; else // incarca DATA
if (LFSR) Q <= {Q[3]^Q[0], Q[3:1]}; else // LFSR
if (CU) Q <= Q+1; // increment

endmodule // reg_seq

```

- **C-2** Comportamentul unui automat secvențial sincron este descris prin graful de tranziții prezentat.
 - a) Implementați automatul cu un număr minim de bistabile D și porți logice NAND.
 - b) Completați formele de undă ale stării și ieșirii în cazul în care intrarea are tot timpul valoarea 1.

14.3.4 D. Proiectare sistem digital

Proiectați un circuit de înmulțire cu 17 utilizând un sumator. Numărul de intrare este întreg, pozitiv, reprezentat pe 10 biți. Desen. Justificare. Cod Verilog.

```

module mult17 (
input  [9:0] data ,
output [?:0] data_x17 // data x 17 (precizati numarul de biti)
);

```

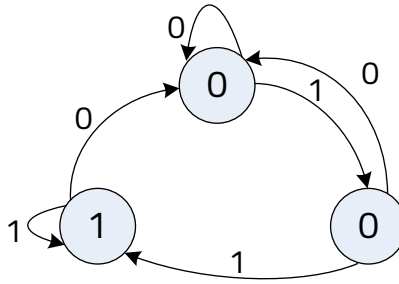


Figura 14.25 Graf, problema 14.3.3.

```
// codul aici
endmodule // mult_17
```

Soluție

Soluția se bazează pe descompunerea înmulțitorului (17) într-o sumă de puteri ale lui 2 ($17 = 16 + 1$). Înmulțirea cu numere puteri ale lui 2 se realizează prin simpla deplasare a biților, fără circuite logice suplimentare:

$$A \times 17 = (A \times 16) + A = (A \ll 4) + A$$

Conceptual, circuitul proiectat este prezentat în figura 14.26.

Codul Verilog este prezentat în continuare. Se observă apariția simbolului de adunare + și lipsa simbolului de înmulțire * în codul comportamental.

```
module mult17 (
input  [ 9:0] data      ,
output [14:0] data_x17  // data x 17  (10+5 biti)
);

assign data_x17 = {data, 4'b0000} + data; // (data x 16) + data

endmodule // mult_17
```

14.4 Subiectul 4

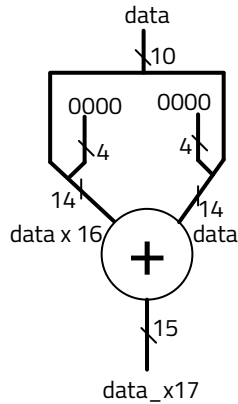


Figura 14.26 Circuit de înmulțire cu 17 realizat cu un sumator, problema 14.3.4-D.

14.4.1 A. Noțiuni generale

- **A-1** Ce capacitate de memorie (exprimată în biți) are o memorie RAM de 1KB?

Soluție

$$1KB = 2^{10} \times 8b = 1024 \times 8 = 8192b$$

- **A-2** Efectuați următoarea operație în baza 16. Ulterior, exprimați rezultatul în baza 2.

$$ABBA + BABA = ?$$

Soluție

Se pun numerele hexazecimale unul sub altul și se adună cifră cu cifră din dreapta (lsb) în stânga (msb), luând în considerare transportul spre bitul superior.

$$- A|_{16} + A|_{16} = 10|_{10} + 10|_{10} = 20|_{10} = 16|_{10} + 4|_{10} = 14|_{16}, \text{ rezultat } 4, \text{ transport } 1.$$

- $B|_{16} + B|_{16} + 1 = 11|_{10} + 11|_{10} + 1 = 23|_{10} = 16|_{10} + 7|_{10} = 17|_{16}$, rezultat 7, transport 1.
- $B|_{16} + A|_{16} + 1 = 11|_{10} + 10|_{10} + 1 = 22|_{10} = 16|_{10} + 6|_{10} = 16|_{16}$, rezultat 6, transport 1.
- $A|_{16} + B|_{16} + 1 = 10|_{10} + 11|_{10} + 1 = 22|_{10} = 16|_{10} + 6|_{10} = 16|_{16}$, rezultat 6, transport 1.

```

1111      <-transport
BEEF +
CAFE
-----
16674

```

Rezultă $BEEF|_{16} + CAFE|_{16} = 16674|_{16} = 1_0110_0110_0111_0100|_2$.

- **A-3** Completați forma de undă a ieșirii unui registru secvențial sincron de 4 biți a cărui funcționare este descrisă de tabelul următor.

LD1	LD2	CU	SHR	ROR	$Q^+[3:0]$	Descriere
1	X	X	X	X	$DATA1[3:0]$	Încărcare din sursa 1
0	1	X	X	X	$DATA2[3:0]$	Încărcare din sursa 2
0	0	1	X	X	$Q + 1$	Increment
0	0	0	1	X	$SI, Q[3:1]$	Deplasare dreapta, intrare serială
0	0	0	0	1	$Q[0], Q[3:1]$	Rotatie dreapta
0	0	0	0	0	$Q[3:0]$	Păstrează starea

Soluție

La fiecare front activ al semnalului de ceas, se va considera cea mai prioritară acțiune. Explicații în figura 14.28.

- Intrarea $LD1$ este activă, se încarcă $DATA1$ și starea Q devine $5|_{16} = 0101|_2$
- Nicio intrare nu este activă, se păstrează starea 5
- Intrarea CU este activă, se incrementează starea, din $5|_{16}$ devine $6|_{16}$
- Intrarea CU este activă, se incrementează starea, din $6|_{16}$ devine $7|_{16}$

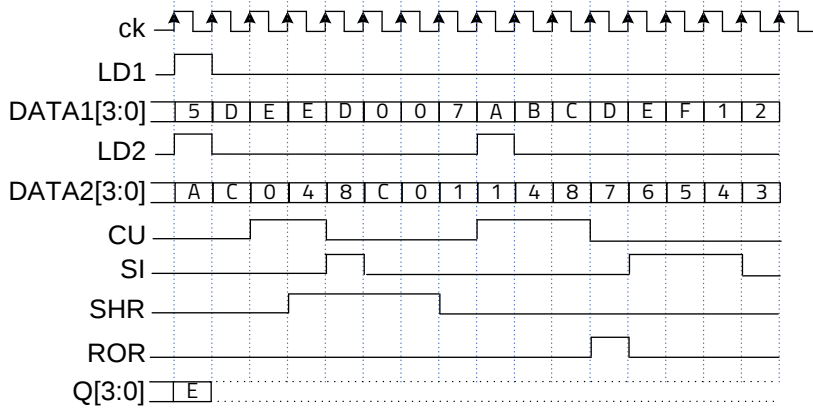


Figura 14.27 Forme de undă pentru problema 14.4.1.

- Intrarea *SHR* este activă, se deplasează starea spre dreapta, iar cel mai semnificativ bit devine $Q^+[3] = SI = 1$, $Q[3:0] = 1011_2 = B_{16}$
 - Intrarea *SHR* este activă, se deplasează starea spre dreapta, iar cel mai semnificativ bit devine $Q^+[3] = SI = 0$, $Q[3:0] = 0101_2 = 5_{16}$
 - Intrarea *SHR* este activă, se deplasează starea spre dreapta, iar cel mai semnificativ bit devine $Q^+[3] = SI = 0$, $Q[3:0] = 0010_2 = 2_{16}$
 - Nicio intrare nu este activă, se păstrează starea 2
 - Intrarea *LD2* este activă, se încarcă *DATA2* și starea *Q* devine $1_{16} = 0001_2$
 - Intrarea *CU* este activă, se incrementează starea, din 1_{16} devine 2_{16}
 - Intrarea *CU* este activă, se incrementează starea, din 2_{16} devine 3_{16}
 - Intrarea *ROR* este activă, se rotește starea spre dreapta și din $3_{16} = 0011_2$ devine $1001_2 = 9_{16}$
 - Nicio intrare nu este activă, se păstrează starea 9 (se ignoră intrarea *SI*)
 - Nicio intrare nu este activă, se păstrează starea 9
 - Nicio intrare nu este activă, se păstrează starea 9
- **A-4** Prezentați rezultatul operației cu numere întregi (baza 10), cu operandi codificați pe 16 biți în complement față de 2.

$$1010 + (-1011) = ?$$

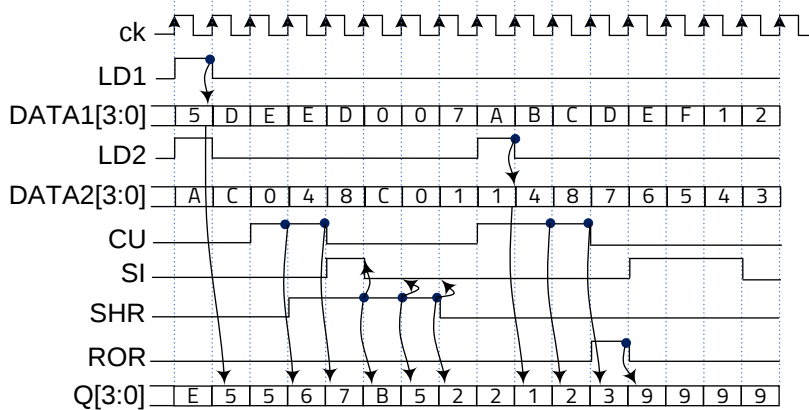


Figura 14.28 Soluția problemei 14.4.1-A-3.

Soluție

$$1010 + (-1011) = -1$$

Se convertesc în baza 2 numerele pozitive exprimate în baza 10 și se extinde reprezentarea la 16 biți.

$$1010|_{10} = 0000_0011_1111_0010|_2$$

$$1011|_{10} = 0000_0011_1111_0011|_2$$

$$1|_{10} = 0000_0000_0000_0001|_2$$

Se determină reprezentarea în complement față de 2 a numerelor negative.

Se poate utiliza algoritmul rapid:

- 1) se consideră biți reprezentării numărului pozitiv, unul câte unul, în ordine, din dreapta în stânga,
- 2) se copiază toți biții până la primul 1 inclusiv, apoi se complementează toți biții.

Rezultă:

$$-1011|_{10} = 1111_1100_0000_1101|_{C2}$$

$$-1|_{10} = 1111_1111_1111_1111|_{C2}$$

Se pun numerele exprimate în complement față de 2 unul sub altul și se adună în binar, cu considerarea transportului, de la lsb (dreapta) spre msb (stânga). Bitul de semn rezultă din operație iar eventualul transport (bitul al 17-lea) se ignoră.

```

          0 0000 0000 0000 000    <-transport
1010 +    0000_0011_1111_0010+
-1011    1111_1100_0000_1101
-----
- 1      1111_1111_1111_1111

```

- **A-5** Utilizați operatori de reducere pentru a modela în Verilog următorii indicatori referitori la intrarea data.

```

//-- 100 -----
module opReducere (
input  [15:0] data      ,
output      data_div16  , // data este divizibila cu 16
output      data_gt31_div16, // data este mai mare decat 31 si
                                // divizibila cu 16
output      data_8_63    , // data este intre 8 si 63
output      data_zero    // data este egala cu zero
);

assign data_div16      = ..?.. ; // completati codul
assign data_gt31_div16= ..?.. ;
assign data_8_63      = ..?.. ;
assign data_zero      = ..?.. ;

```

Soluție

Se consideră exprimarea condițiilor aritmetice prin condiții logice astfel:

„**Data este divizibilă cu 16**” înseamnă că, în reprezentarea binară, numărul are „*cei mai puțin semnificativi 4 biți egali cu 0*”. Deci funcția logică SAU/OR pe cei mai puțin semnificativi 4 biți trebuie să returneze valoarea 0.

„**Data este mai mare decât 31 și divizibilă cu 16**” înseamnă că *cei mai puțin semnificativi 4 biți sunt egali cu 0 (divizibil cu $2^4 = 16$) și „cel puțin un bit egal cu 1 cu index mai mare sau egal cu 5 ($2^5 = 32$)*”. Deci funcția logică SAU/OR pe biții cu index mai mare sau egal cu 5 trebuie să returneze valoarea 1 și funcția logică SAU/OR pe biții cu index între 0 și 3 trebuie să returneze valoarea 0.

„**Data este egală cu 0**” înseamnă că „*toți biții sunt egali cu 0*”. Deci funcția logică SAU/OR pe toți biții trebuie să returneze valoarea 0.

„**Data este între 8 și 63**” adică mai mare sau egal cu 8 ȘI mai mică decât 64, este echivalentă cu exprimarea „*există măcar un bit egal cu 1 între biții de index 3 și 5*” și „*biții cu index mai mare sau egal cu 6 sunt toți 0 (numărul nu este mai mare decât 63)*”.

„**Data este egală cu 0**” înseamnă că „*toți biții sunt egali cu 0*”. Deci funcția logică SAU/OR pe toți biții trebuie să returneze valoarea 0.

Rezultă următoarele atribuiri exprimate în Verilog:

```
assign data_div16 = ~|data[3:0] ; // ultimii 4 biti sunt 0
assign data_gt31_div16 = |data[15:5] & // mai mare decat 31,
// exista un bit 1 cu index
// intre 15 si 5
~|data[3:0] ; // ultimii 4 biti sunt 0
assign data_8_63 = |data[5:3] & // mai mare sau egal cu 8 si
~|data[15:6] ; // mai mica decat 64
// (toti bitii cu index mai
// mare decat 5 sunt 0)
assign data_zero = ~|data ; // toti biti sunt 0
```

14.4.2 B. Circuite logice combinaționale

- **B-1** Demonstrați egalitatea:

$$\overline{C} \cdot \overline{A \oplus B \oplus D} + \overline{A} \cdot B \cdot D = \overline{C} \cdot \overline{A \oplus B \oplus D} + \overline{A} \cdot B \cdot C \cdot D$$

Soluție

Se observă că în cazul funcției SAU-EXCLUSIV (XOR) diagrama V-K prezintă căsuțe cu valoarea 1 într-o formă de tablă de șah.

Expresia $\overline{A \oplus B \oplus D}$ este condiționată (SI) de variabila $\overline{C}$. Deci, expresia $\overline{C} \cdot \overline{A \oplus B \oplus D}$ are căsuțe de 1 numai în suprafața în care $C = 0$.

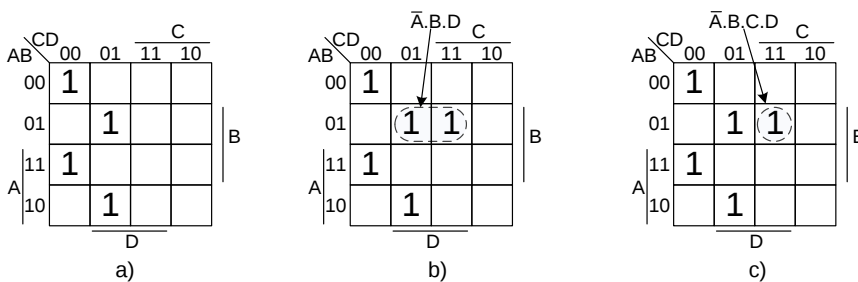


Figura 14.29 Soluția problemei 14.4.2-B-1.

- Diagrama V-K asociată funcției $\overline{C} \cdot \overline{A \oplus B \oplus D}$;
- Diagrama V-K asociată funcției din partea stângă, cu evidențierea suprafeței $\overline{A} \cdot B \cdot D$;
- Diagrama V-K asociată funcției din partea dreaptă, cu evidențierea suprafeței $\overline{A} \cdot B \cdot C \cdot D$.

Funcția $\overline{C} \cdot \overline{A} \oplus B \oplus \overline{D}$ are diagrama prezentată în figura 14.29-a).

Funcția din partea stângă conține suplimentar (OR) expresia $\overline{A} \cdot B \cdot D$, care corespunde suprafeței evidențiate în 14.29-b).

Funcția din partea dreaptă conține suplimentar (OR) expresia $\overline{A} \cdot B \cdot C \cdot D$, care corespunde suprafeței evidențiate în 14.29-c).

Se observă faptul că diagramele V-K prezentate în figurile 14.29-b) și c) sunt identice. Deci, egalitatea celor două funcții logice este demonstrată, ambele fiind egale cu:

$$\sum (0, 5, 7, 9, 12)$$

- **B-2** Minimizați funcția logică și implementați-o exclusiv cu porți NAND. Justificare.

$$F(A, B, C, D) = \sum (2, 3, 5, 6, 13, 14, 15) + d(7)$$

Determinați și justificați valoarea $F(0, 1, 1, 1) = ?$

Soluție

Se construiește diagrama V-K, conform expresiei funcției, punând valoarea 1 în căsuțele cu index menționat în lista de mintermi:

$$\sum (2, 3, 5, 6, 13, 14, 15)$$

Căsuțele cu valori indiferente, ale căror index este menționat în lista $d(7)$, se vor marca cu simbolul „-”. Valorile acestor căsuțe se vor considera în procesul de minimizare ca având fie valoare 0 fie valoare 1 astfel încât expresia finală a funcției să fie minimă. Diagrama V-K asociată este prezentată în figura 14.30.

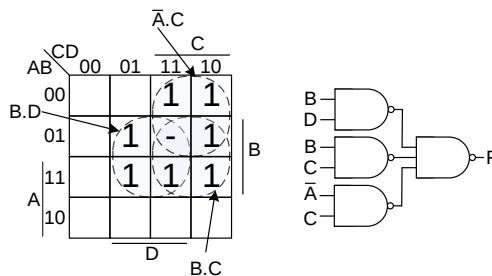


Figura 14.30 Diagrama V-K pentru problema 14.4.2-B-2. Circuitul proiectat.

După definirea suprafețelor, rezultă expresia minimă a funcției:

$$F(A, B, C, D) = B \cdot D + B \cdot C + \overline{A} \cdot C$$

Circuitul asociat implementat cu porți NAND, pe două nivele, este prezentat în figura 14.30.

$F(0, 1, 1, 1) = 1$, deoarece căsuța respectivă a fost acoperită de 3 suprafețe ($B \cdot D$, $B \cdot C$ și $\overline{A} \cdot C$).

14.4.3 C. Circuite logice secvențiale

- **C-1** Modelați în Verilog circuitul secvențial sincron care funcționează conform tabelului prezentat la punctul 14.4.1-A-3.

a) Simbol bloc.

b) Cod Verilog cu comentarii.

Soluție

Simbolul bloc este prezentat în figura 14.31. Se observă existența porților de intrare pentru semnalul de ceas și cel de reset, semnale ne-menționate în tabelul de funcționare. Codul Verilog care modelează re-

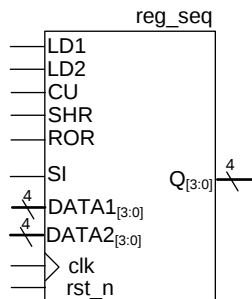


Figura 14.31 Simbolul bloc, problema 14.4.3-C-1.

gistrul este următorul:

```
module reg_seq (
input          clk      , // semnal de ceas
input          rst_n    , // semnal de reset activ low
input          LD1      , // incarcare DATA1
input          LD2      , // incarcare DATA2
input [3:0]    DATA1   , // data de intrare
```

```

input      [3:0] DATA2 , // data de intrare
input      CU      , // increment (Count Up)
input      SHR      , // deplasare dreapta
input      ROR      , // rotatie dreapta
input      SI       , // data seriala
output reg [3:0] Q      // iesire registru
);

always @(posedge clk or negedge rst_n)
if (~rst_n) Q <= 4'b0 ; else // reset
if (LD1)    Q <= DATA1 ; else // incarca DATA1
if (LD2)    Q <= DATA2 ; else // incarca DATA2
if (CU)     Q <= Q+1 ; else // increment
if (SHR)    Q <= {SI, Q[3:1]} ; else // deplasare dreapta
if (ROR)    Q <= {Q[0], Q[3:1]} ; // rotatie dreapta

endmodule // reg_seq

```

- **C-2** Comportamentul unui automat secvențial sincron este descris prin graful de tranziții prezentat în figura 14.32.

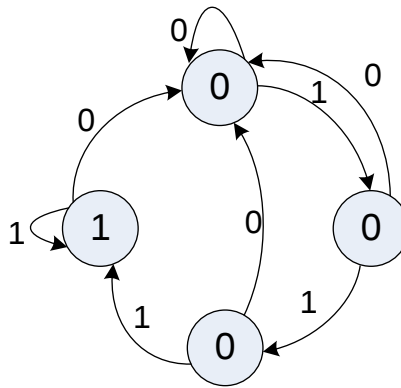


Figura 14.32 Graf, problema 14.4.3.

- Implementați automatul cu un număr minim de bistabile D și porți logice NAND.
- Completați formele de undă ale stării și ieșirii în cazul în care intrarea are tot timpul valoarea 1.

14.4.4 D. Proiectare sistem digital

Utilizați un număr minim de sumatoare complete de 1 bit pentru a determina suma a 7 biți de date. Desen cu simboluri de sumatoare complete de 1 bit. Justificare. Cod Verilog.

```
module sum (
input  [6:0] data , // 7 biți de date
output [2:0] sum    // suma bitilor data, in domeniul 0...7
);
// codul aici
endmodule // sum
```

Soluție

Sumatorul complet de un bit însumează 3 biți cu ponderi egale, 2 biți de date plus bitul de transport. Rezultatul, în domeniul $[0...3]$, este exprimat pe 2 biți, bitul de sumă, cu pondere 2^0 și bitul de transport, cu pondere 2^1 . Trebuie avut în vedere algoritmul de însumare (similar cu însumarea în baza 10) care impune adunarea doar a biților de același ordin (adică nu se adună biți de ponderi diferite).

Soluția propusă, implementată cu sumatoare complete de 1 bit și prezentată în figura 14.33, poate fi descrisă astfel:

- primul sumator adună 3 biți și produce un bit de ordin 2^0 și transportul de ordin 2^1 ,
- al 2-lea sumator, adună următorii 3 biți și produce un bit de ordin 2^0 și transportul de ordin 2^1 ,
- al 3-lea sumator, adună al 7-lea bit cu biții de sumă (ordin 2^0) determinați de primele două sumatoare și produce bitul 0 al rezultatului final și un alt 3-lea bit de transport, de ordin 2^1 ,
- al 4-lea sumator, adună biții de transport de ordin 2^1 generați de primele 3 sumatoare și produce bitul final de ordin 2^1 și, pe ieșirea de transport, bitul final de ordin 2^2 .

Rezultatul, în domeniul $[0...7]$, va fi reprezentat pe 3 biți.

Codul Verilog care modelează circuitul propus este prezentat în continuare:

```
module sum (
input  [6:0] data , // 7 biți de date
output [2:0] sum    // suma bitilor data, in domeniul 0...7
);
```

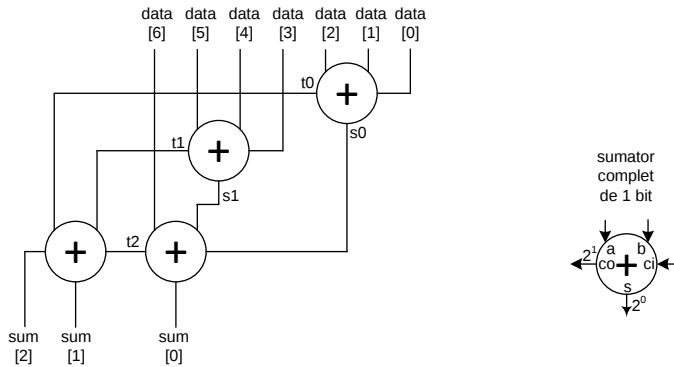


Figura 14.33 Soluția problemei 14.4.4-D. Detaliu cu simbolul bloc al sumatorului complet de 1 bit.

```

wire    s0, s1      ; // iesiri, ordinul 0
wire    t0, t1, t2  ; // transport, ordinul 1

assign {   t0,      s0 } = data[2] + data[1] + data[0]; // sum1
assign {   t1,      s1 } = data[5] + data[4] + data[3]; // sum2
assign {   t2, sum[0] } =      s1 +      s0 + data[6]; // sum3
assign {sum[2], sum[1] } =      t2 +      t1 +      t0; // sum4

endmodule // sum

```

14.5 Subiectul 5

14.5.1 A. Noțiuni generale

- **A-1** Implementați funcția având la dispoziție un multiplexor 4:1 și un inversor. Justificare.

$$F(A, B, C) = \sum (2, 4, 6, 7)$$

Soluție

Se realizează tabelul de adevăr al funcției $F(A, B, C)$ și se grupează liniile câte două.

A	B	C	F	F
0	0	0	0	0
0	0	1	0	0
0	1	0	1	$\overline{C}$
0	1	1	0	$\overline{C}$
0	0	0	1	$\overline{C}$
0	0	1	0	$\overline{C}$
0	1	0	1	1
0	1	1	1	1

Din câte două linii, se determină funcția reziduu pe baza variabilei cele mai puțin semnificative, C . Se obține circuitul cu MUX4:1 prezentat în figura 14.34, utilizând un inversor pentru obținerea funcției reziduu $\overline{C}$. Pe intrările de selecție se introduc cele mai semnificative variabile, A și B . Cele 4 funcții reziduu se introduc pe cele 4 intrări de date ale multiplexorului.

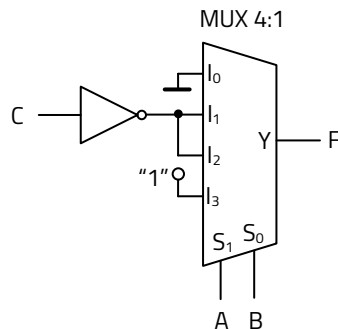


Figura 14.34 Soluția problemei 14.5.1-A-1.

- **A-2** Efectuați următoarea operație în baza 16. Ulterior, exprimați rezultatul în baza 2.

$$DEAD + B0DE = ?$$

Soluție

Se pun numerele hexazecimale unul sub altul și se adună cifră cu cifră din dreapta (lsb) în stânga (msb), luând în considerare transportul spre bitul superior.

- $D|_{16} + E|_{16} = 13|_{10} + 14|_{10} = 27|_{10} = 16|_{10} + 11|_{10} = 1B|_{16}$, rezultat B , transport 1.
- $A|_{16} + D|_{16} + 1 = 10|_{10} + 13|_{10} + 1 = 24|_{10} = 16|_{10} + 8|_{10} = 18|_{16}$, rezultat 8, transport 1.
- $E|_{16} + 0|_{16} + 1 = 14|_{10} + 0|_{10} + 1 = 15|_{10} = F|_{16}$, rezultat F , transport 0.
- $D|_{16} + B|_{16} + 1 = 13|_{10} + 11|_{10} + 0 = 24|_{10} = 16|_{10} + 8|_{10} = 18|_{16}$, rezultat 8, transport 1.

```

1011      <-transport
  DEAD  +
  B0DE
  -----
18F8B

```

Rezultă $DEAD|_{16} + B0DE|_{16} = 18F8B|_{16} = 1_1000_1111_1000_1011|_2$.

Observație: nu este nicio confuzie între cifra 0 (zero) și caracterul O . Fiind vorba de cifre în baza 16, este evident faptul că simbolurile sunt 0...9, A, B, C, D, E . Deci, litera O nu este un caracter hexazecimal.

- **A-3** Completați forma de undă a ieșirii unui registru secvențial sincron de 4 biți a cărui funcționare este descrisă de tabelul următor.

LD1	LD2	CD	SHR	ROR	$Q^+[3:0]$	Descriere
1	X	X	X	X	$DATA1[3:0]$	Încărcare din sursa 1
0	1	X	X	X	$DATA2[3:0]$	Încărcare din sursa 2
0	0	1	X	X	$Q - 1$	Decrement
0	0	0	1	X	$1'b0, Q[3:1]$	Deplasare dreapta
0	0	0	0	1	$Q[0], Q[3:1]$	Rotatie dreapta
0	0	0	0	0	$Q[3:0]$	Păstrează starea

Soluție

La fiecare front activ al semnalului de ceas, se va considera cea mai prioritară acțiune. Explicații în figura 14.36.

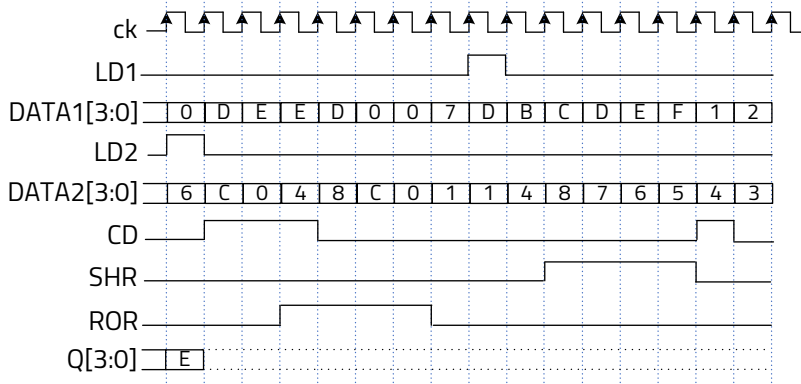


Figura 14.35 Forme de undă pentru problema 14.5.1.

- Intrarea $LD2$ este activă, se încarcă $DATA2$ și starea Q devine $6|_{16} = 0110|_2$
- Intrarea CD este activă, se decrementează starea, din $6|_{16}$ devine $5|_{16}$
- Intrarea CD este activă, se decrementează starea, din $5|_{16}$ devine $4|_{16}$
- Intrarea CD este activă, se decrementează starea, din $4|_{16}$ devine $3|_{16}$
- Intrarea ROR este activă, se rotește starea spre dreapta și din $3|_{16} = 0011|_2$ devine $1001|_2 = 9|_{16}$
- Intrarea ROR este activă, se rotește starea spre dreapta și din $9|_{16} = 1001|_2$ devine $1100|_2 = C|_{16}$
- Intrarea ROR este activă, se rotește starea spre dreapta și din $C|_{16} = 1100|_2$ devine $0110|_2 = 6|_{16}$
- Nicio intrare nu este activă, se păstrează starea 6
- Intrarea $LD1$ este activă, se încarcă $DATA1$ și starea Q devine $D|_{16} = 1101|_2$
- Nicio intrare nu este activă, se păstrează starea D
- Intrarea SHR este activă, se deplasează starea spre dreapta și din $D|_{16} = 1101|_2$ devine $0110|_2 = 6|_{16}$, msb $Q^+[3]$ devine 0
- Intrarea SHR este activă, se deplasează starea spre dreapta și din $6|_{16} = 0110|_2$ devine $0011|_2 = 3|_{16}$, msb $Q^+[3]$ devine 0
- Intrarea SHR este activă, se deplasează starea spre dreapta și din $3|_{16} = 0011|_2$ devine $0001|_2 = 1|_{16}$, msb $Q^+[3]$ devine 0

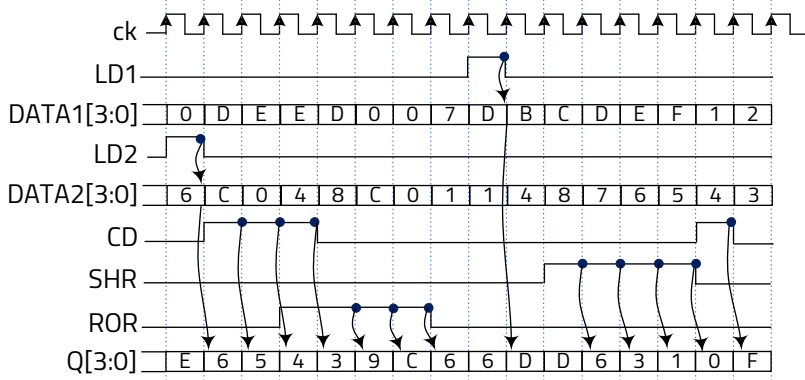


Figura 14.36 Soluția problemei 14.5.1-A-3.

- Intrarea *SHR* este activă, se deplasează starea spre dreapta și din $1|_{16} = 0001|_2$ devine $0000|_2 = 0|_{16}$, msb $Q^+[3]$ devine 0
 - Intrarea *CD* este activă, se decrementează starea, din $0|_{16} = 0000|_2$ devine $1111|_2 = F|_{16}$
 - Nicio intrare nu este activă, se păstrează starea *F*
 - Intrarea *LD2* este activă, se încarcă *DATA2* și starea *Q* devine $1|_{16} = 0001|_2$
 - Intrarea *CU* este activă, se incrementează starea, din $1|_{16}$ devine $2|_{16}$
 - Intrarea *CU* este activă, se incrementează starea, din $2|_{16}$ devine $3|_{16}$
 - Nicio intrare nu este activă, se păstrează starea 9 (se ignoră intrarea *SI*)
 - Nicio intrare nu este activă, se păstrează starea 9
 - Nicio intrare nu este activă, se păstrează starea 9
- **A-4** Prezentați rezultatul operației cu numere întregi (baza 10), cu operanzi codificați pe 16 biți în complement față de 2.

$$2020 + (-2025) = ?$$

Solutie

$$2020 + (-2025) = -5$$

Se convertesc în baza 2 numerele pozitive exprimate în baza 10 și se extinde reprezentarea la 16 biti.

$$2020|_{10} = 0000_0111_1110_0100|_2$$

$$2025|_{10} = 0000_0111_1110_1001|_2$$

$$5|_{10} = 0000_0000_0000_0101|_2$$

Se determină reprezentarea în complement față de 2 a numerelor negative.

Se poate utiliza algoritmul rapid:

- 1) se consideră biți reprezentării numărului pozitiv din dreapta în stânga,
- 2) se copiază toți biții până la primul 1 inclusiv, apoi se complementează toți biții.

Rezultă:

$$-2025|_{10} = 1111_1000_0001_0111|_{C2}$$

$$-5|_{10} = 1111_1111_1111_1011|_{C2}$$

Se pun numerele exprimate în complement față de 2 unul sub altul și se adună în binar, cu considerarea transportului, de la lsb (dreapta) spre msb (stânga). Bitul de semn rezultă din operație iar eventualul transport (bitul al 17-lea) se ignoră.

	0 0000 0000 0000 100	<-transport
2020 +	0000_0111_1110_0100+	
-2025	1111_1000_0001_0111	
-----	-----	
- 5	1111_1111_1111_1011	

- **A-5** Utilizați operatori de reducere pentru a modela în Verilog următorii indicatori referitori la intrarea data.

```
//-- 120 -----
module opReducere (
input  [15:0] data      ,
output      data_div64  , // data este divizibila cu 64
output      data_zero   , // data este egala cu zero
output      data_gt63_div8 , // data este mai mare decat 63 si
                                // divizibila cu 8
output      data_4_63    , // data este intre 4 si 63
);
assign data_div64      = ..?.. ; // completati codul
assign data_zero       = ..?.. ;
assign data_gt63_div8  = ..?.. ;
assign data_4_63       = ..?.. ;
```

14.5.2 B. Circuite logice combinaționale

- **B-1** Se consideră o memorie (2K x 1B).

- a) Care este numărul de biți stocați în memorie?
- b) Se adaugă memorie o facilități de corecție de paritate (1 bit la fiecare byte). Care va fi numărul de biți stocați în memoria cu corecție de paritate?
- c) Cu cât crește procentual dimensiunea memoriei?

Soluție

- a) $2K \times 8b = 2 \times 2^{10} \times 8 = 2 \times 1024 \times 8 = 16383b$
- b) $2K \times 9b = 2 \times 2^{10} \times 9 = 2 \times 1024 \times 9 = 18432b$
- c) La fiecare byte, 8 biți, se adaugă 1 bit. Deci procentul de creștere este:

$$1/8 = 12.5\%$$

- **B-2** Minimizați funcția logică și implementați-o exclusiv cu porți NAND. Justificare.

$$F(A, B, C, D) = \sum (4, 5, 7, 8, 9, 11, 12, 15) + d(13)$$

Determinați și justificați valoarea $F(1, 1, 0, 1) = ?$

14.5.3 C. Circuite logice secvențiale

- **C-1** Modelați în Verilog circuitul secvențial sincron care funcționează conform tabelului prezentat la punctul 14.5.1-A-3.

- a) Simbol bloc.
- b) Cod Verilog cu comentarii.

Soluție

Simbolul bloc este prezentat în figura 14.37. Se observă existența porturilor de intrare pentru semnalul de ceas și cel de reset, semnale ne-menționate în tabelul de funcționare. Codul Verilog care modelează registrul este următorul:

```
module reg_seq (
input          clk      , // semnal de ceas
input          rst_n    , // semnal de reset activ low
input          LD1      , // incarcare DATA1
input          LD2      , // incarcare DATA2
input [3:0]    DATA1   , // data de intrare
input [3:0]    DATA2   , // data de intrare
input          CD       , // decrement (Count Down)
input          SHR      , // deplasare dreapta
input          ROR      , // rotatie dreapta
output reg [3:0] Q      // iesire registru
);
```

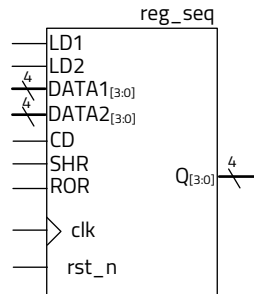


Figura 14.37 Simbolul bloc, problema 14.5.3-C-1.

```

always @(posedge clk or negedge rst_n)
if (~rst_n) Q <= 4'b0; else // reset
if (LD1) Q <= DATA1; else // incarca DATA1
if (LD2) Q <= DATA2; else // incarca DATA2
if (CD) Q <= Q-1; else // decrement
if (SHR) Q <= {1'b0, Q[3:1]}; else // deplasare dreapta
if (ROR) Q <= {Q[0], Q[3:1]}; // rotatie dreapta

endmodule // reg_seq

```

- **C-2** Comportamentul unui automat secvențial sincron este descris prin graful de tranziții prezentat în figura 14.38.

a) Implementați automatul cu un număr minim de bistabile D și porți logice NAND.

b) Completați formele de undă ale stării și ieșirii în cazul în care intrarea are tot timpul valoarea 1.

Soluție

Din graful de tranziții prezentat se deduc caracteristicile automatului:

Automat Moore ieșirea este menționată în simbolul de stare, deci nu depinde de intrare.

Automatul are 4 stări deoarece apar 4 simboluri de stare (cercuri), noduri ale grafului. Pentru codarea stărilor sunt necesari minimum 2 biți ($2^2 \geq 4$). Se vor coda stările cu un număr minim de biți, 2. Denumim biții stării prezente Q_1 și Q_0 , iar cei ai stării viitoare D_1 și D_0 .

Automatul are o singură intrare deoarece pe arce apare o singură valoare binară. Denumim intrarea X .

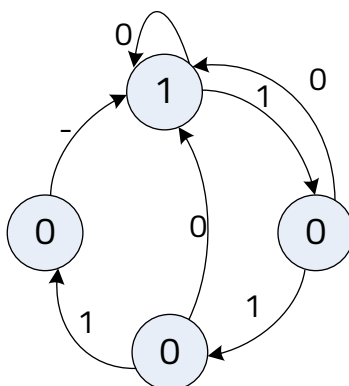


Figura 14.38 Graf, problema 14.5.3.

Automatul are o singură ieșire deoarece în nod este menționată o singură valoare binară. Denumim ieșirea Y .

Figura 14.39 prezintă graful actualizat cu codarea stărilor cu 2 biți. În aceeași figură este prezentat simbolul bloc al automatului proiectat.

Pe baza informațiilor prezentate pe graf, se completează tabelul de tranziții al automatului.

Intrare X	Stare prezentă $Q_1 Q_0$		Stare viitoare $D_1 D_0$		Ieșire Y
0	a	00	a	00	1
1	a	00	b	01	
0	b	01	a	00	0
1	b	01	c	10	
0	c	10	a	00	0
1	c	10	d	11	
-	d	11	a	00	0

Pe baza informațiilor prezente în tabelul de tranziții, se completează diagramele V-K ale ieșirilor: $D_1(X, Q_1, Q_0)$, $D_0(X, Q_1, Q_0)$ și $Y(Q_1, Q_0)$.

Diagramele V-K, prezentate în figura 14.40, se minimizează astfel:

$$D_1(X, Q_1, Q_0) = X \cdot \overline{Q_1} \cdot Q_0 + X \cdot Q_1 \cdot \overline{Q_0}$$

$$D_0(X, Q_1, Q_0) = X \cdot \overline{Q_0}$$

$$Y(Q_1, Q_0) = \overline{Q_1} \cdot \overline{Q_0}$$

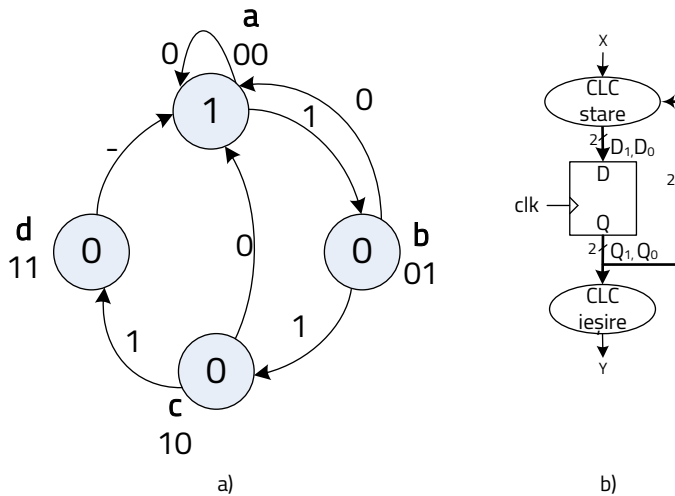


Figura 14.39 Graf de tranziții (a) și simbolul bloc (b) pentru problema 14.5.3-C-2.

Implementarea se face cu două bistabile D pentru memorarea stării și porți NAND pentru logică, conform circuitului prezentat în figura 14.41-a.

În cazul în care intrarea are tot timpul valoarea 1, se observă că automatul ciclează între stările *a*, *b*, *c*, *d*, prezentând pe ieșire secvența periodică 1000, așa ca în figura 14.41-b.

14.5.4 D. Proiectare sistem digital

Proiectați un circuit de înmulțire cu 9 utilizând un sumator. Numărul de intrare este întreg, pozitiv, reprezentat pe 16 biți. Desen. Justificare. Cod Verilog.

```
module mult9 (
input    [15:0]  data      ,
output   [?:0]   data_x9   // data x 9  (precizati numarul de biti)
);
// codul aici
endmodule // mult9
```

Soluție

Soluția se bazează pe descompunerea înmulțitorului (9) într-o sumă de

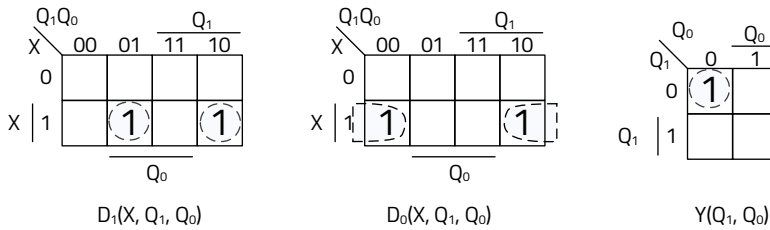


Figura 14.40 Diagrame V-K și minimizare pentru problema 14.5.3-C-2.

puteri ale lui 2 ($9 = 8 + 1$). Înmulțirea cu numere puteri ale lui 2 se realizează prin simpla deplasare a biților, fără circuite logice suplimentare:

$$A \times 9 = (A \times 8) + A = (A \ll 3) + A$$

Conceptual, circuitul proiectat este prezentat în figura 14.42.

Codul Verilog este prezentat în continuare. Se observă apariția simbolului de adunare $+$ și lipsa simbolului de înmulțire $*$ în codul comportamental.

```
module mult9 (
input  [15:0] data ,
output [19:0] data_x9 // data x 9 (16+4 biti)
);

assign data_x9 = {data, 3'b000} + data; // (data x 8) + data

endmodule // mult9
```

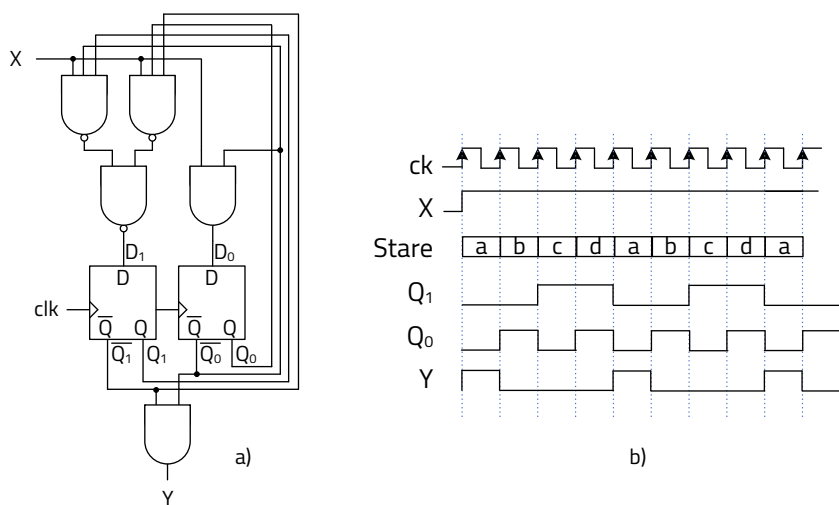


Figura 14.41 Rezolvare problema 14.5.3-C-2. a) Circuit b) Forme de undă.

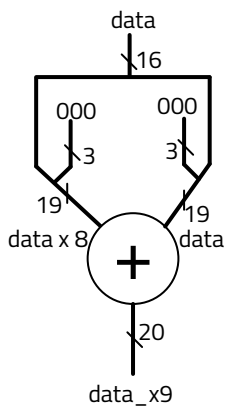


Figura 14.42 Circuit de înmulțire cu 9 realizat cu un sumator, problema 14.5.4-D.

Anexa 15

Subiecte pentru evaluat ingineri

15.1 Ingineri începători

15.1.1 Cunoștințe generale

1. Care este cel mai mare număr pozitiv, exprimat în complement față de 2 care se poate reprezenta 4 bytes?
2. Care este cel mai mare număr binar, exprimat pe 16 biți? Care este echivalentul său în baza 10? Dar în baza 16?
3. Scrieți numărul -123 reprezentat pe 8 biți, reprezentat în complement față de 2.
4. Cu cât crește dimensiunea memoriei la implementarea unui bit de paritate pentru fiecare byte, la o memorie de $2K \times 16$?
5. Ce capacitate de memorie (exprimată în biți) are o memorie RAM $1K \times 16$?
6. Care este lungimea maximă a ciclului unui circuit **LFSR** (*Linear Feedback Shift Register*, engl.) de 8 biți? Justificați răspunsul.

Soluție

Lungimea maximă a ciclului de stări pentru un circuit LFSR cu N biți este $2^N - 1$ stări. Acest număr se obține numai pentru anumite polinoame.

În cazul $N = 8$, lungimea maximă a ciclului are $2^8 - 1 = 255$ de stări.

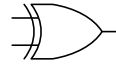
15.1.2 Definirea noțiunilor

1. Definiți noțiunea de „poartă logică XOR” (simbol, funcționare, aplicații, model Verilog).

Soluție

Poarta logică XOR (sau exclusiv) are tabelul de funcționare și simbolul prezentate în continuare.

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



Poarta XOR poate fi folosită ca:

- circuit sumator de 2 biți;
- circuit pentru determinarea inegalității dintre doi biți (număr impar de intrări egale cu 1);
- inversor comandat (o intrare de dată se propagă la ieșire negată sau ne-negată în funcție de starea celeilalte intrări de control).

Codul Verilog care modelează o poartă XOR cu două intrări:

```
module xor_gate(
  input  A, B, // intrari
  output Yxor // iesire
);

  assign Yxor = A ^ B; // XOR intre 2 intrari

endmodule // xor_gate
```

Codul Verilog care modelează o poartă XOR cu un număr parametrizabil de intrări:

```
module xor_gate #(
  parameter NO_INPUTS = 10
) (
  input  [NO_INPUTS-1:0] A, // vector intrari
  output Yxor // iesire
);

  assign Yxor = ^A; // XOR pe toti bitii (operator de reducere)

endmodule // xor_gate
```

De remarcat că, o poartă logică este mult prea simplă pentru a se încapsula singură într-un modul.

Poarta logică se descrie printr-o linie într-un modul care descrie un sistem mai mare:

```
assign Yxor = A ^ B; // XOR între 2 intrari
```

```
assign Yxor = ^A; // XOR pe toti bitii (operator de reducere)
```

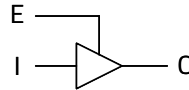
2. Definiți noțiunea de „poartă logică în 3 stări” (simbol, funcționare, aplicații, model Verilog).

Soluție

Poarta logică în 3 stări prezintă, pe lângă cele două stări binare (0 și 1), o stare caracterizată printr-o impedanță de ieșire mare, ideal infinită. Ieșirea porții logice în stare de înaltă impedanță, simbolizată HiZ („High Z”, engl.) poate fi conectată la ieșirea altei porți logice, fără a afecta funcționalitatea acesteia.

Poarta în 3 stări are două intrări: o *intrare de date* și o *intrare de validare*. Tabelul de funcționare și simbolul unei porți în 3 stări sunt prezentate în continuare.

E	I	O
0	–	HiZ
1	0	0
1	1	1



Codul Verilog care modelează o poartă în 3 stări, cu operator ternar:

```
module gate_3state (
input    I,      // intrare
input    E,      // validare
output   O       // iesire
);

assign O = E ? I : 1'bz;

endmodule // gate_3state
```

De remarcat că poartă în 3 stări se poate modela cu o specificație Verilog dedicată modelării la nivel de poartă logică astfel:

```
bufif1(O, I, E); // iesire, intrare, control
```

3. Definiți noțiunea de „multiplexor $2^N : 1$ ” (număr de intrări, număr de ieșiri, simbol, funcție de transfer, funcționare, aplicații, model Verilog).

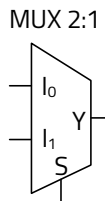
Soluție

Multiplexorul $2^N : 1$ este un circuit logic combinațional cu o intrare de selecție de N biți, 2^N intrări de date și o ieșire. Pe baza intrării de selecție se decide care din intrările de date se transmite la ieșire.

Ecuția logică a ieșirii multiplexorului 2:1 este:

$$Y = S \cdot I_1 + \bar{S} \cdot I_0$$

Simbolul multiplexorului 2:1 este:



Codul Verilog care modelează un multiplexor 2:1 cu operatorul ternar este:

```
assign Y = S ? I1 : I0;
```

Un multiplexor cu mai multe intrări poate fi modelat cu o specificație case.

```
wire [1:0] S           ; // selectie, 2 biti
wire      I0, I1, I2, I3; // 4 intrari de date

always @(*)           // model combinational
  case (S)
    2'b00 : Y = I0;
    2'b01 : Y = I1;
    2'b10 : Y = I2;
    default: Y = I3;
  endcase
```

4. Definiți noțiunea de „**demultiplexor $1 : 2^N$** ” (număr de intrări, număr de ieșiri, simbol, funcție de transfer, funcționare, aplicații, model Verilog).

Soluție

Demultiplexorul este un circuit logic combinațional cu o intrare de selecție de N biți, o intrare de date și 2^N ieșiri. Pe baza intrării de selecție se decide pe care din ieșirile de date se transmite intrarea. Ieșirile neselectate au valoare 0, indiferent de valoarea intrării de date.

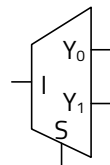
Ecuatiile logice ale ieșirilor demultiplexorului 1:2 sunt:

$$Y_1 = S \cdot I$$

$$Y_0 = \bar{S} \cdot I$$

Simbolul demultiplexorului 1:2 este:

DMUX 1:2



5. Definiți noțiunea de „*decodificator de N biți*” (număr de intrări, număr de ieșiri, simbol, funcție de transfer, funcționare, aplicații, model Verilog).
6. Definiți noțiunea de „*ROM $2^N \times M$* ” (număr de intrări, număr de ieșiri, simbol, funcție de transfer, funcționare, aplicații).
7. Definiți noțiunea de „*sumator complet de 1 bit*” (număr de intrări, număr de ieșiri, simbol, funcție de transfer, funcționare, aplicații, model Verilog).

Soluție

Sumatorul complet de 1 bit adună 3 biți (2 biți de date și transportul de intrare). Rezultatul în domeniul 0...3 este reprezentat pe 2 biți, unul de sumă și, cel mai semnificativ bit, cel de transport.

Codul Verilog care modelează un sumatorul complet de un bit este:

```
module full_adder(
input    A, B, CI,    // intrari de date si transport de intrare
output   S, CO        // suma si transport de iesire
);
```

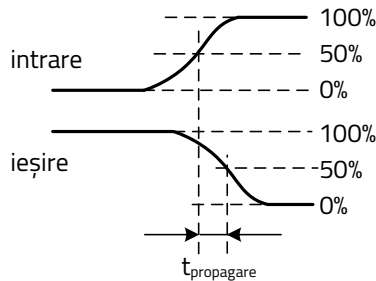
```
// rezultat pe 2 biti cu MSB=transport
assign {CO, S} = A + B + CI; // suma a 3 biti

endmodule // full_adder
```

8. Definiți noțiunea de „*țimp de propagare al unei porți logice*”.

Soluție

Țimpul de propagare al unei porți logice este intervalul de timp necesar propagării unui semnal între intrarea și ieșirea unei porți logice. Timpul de propagare se măsoară la nivelul de 50% al semnalelor de intrare și de ieșire.



9. Definiți noțiunea de „*fan-out*”.

Soluție

Fan-out-ul unei porți logice reprezintă numărul de porți logice de același tip care pot fi comandate de la ieșirea unei porți logice. Se definește ca raportul între curentul de ieșire și curentul de intrare al porții logice.

Fan-out în starea H (High = 1 logic): $FO_H = I_{OH} - I_{IH}$

Fan-out în starea L (Low = 0 logic): $FO_L = I_{OL} - I_{IL}$

Fan-out: $FO = \min(FO_H, FO_L)$

10. Definiți noțiunea de „*margine de zgomot*”.

11. Definiți proprietatea fundamentală a „*codului Gray*”.

Soluție

Codul Gray are proprietatea că două coduri alăturate diferă printr-un singur bit.

12. Definiți noțiunea de „latch D”.

Soluție

Latch-ul D este un circuit care memorează starea, activ pe palierul semnalului de ceas. Dacă $CK = 1$, latch-ul este transparent și preia starea intrării D. Dacă $CK = 0$, latch-ul memorează starea.

Tabelul de determinare a stării viitoare a latch-ului D este:

CK	D	Q^+	Acțiune
0	-	Q	păstrează starea
1	0	0	copiază intrarea
1	1	1	copiază intrarea

13. Definiți noțiunea de „bistabil D” (intrări, ieșiri, simbol, funcție de transfer, funcționare, aplicații, model Verilog).

Soluție

Bistabilul D (Delay, engl.) este un circuit logic secvențial care prezintă două stări stabile. Comutarea bistabilului se face doar în momentele fronturilor active ale semnalului de ceas. La fiecare front activ al semnalului de ceas, bistabilul D eșantionează intrarea și își păstrează starea până la următorul front activ al semnalului de ceas.

Tabelul de determinare a stării viitoare a bistabilului D și simbolul acestuia sunt:

CK	D	Q^+	Acțiune
↑	0	0	copiază intrarea
↑	1	1	copiază intrarea
-	-	Q	păstrează starea



Modelul Verilog al bistabilului D se descrie cu o specificație secvențială `always` pe baza tabelului de funcționare.

```

module DFF (
    input      ck,    // intrare de ceas
    input      D ,    // intrare de date
    output reg  Q     // iesire/stare bistabil
);

always @(posedge ck)
    Q <= D;

endmodule // DFF

```

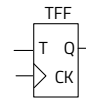
14. Definiți noțiunea de „bistabil T” (intrări, ieșiri, simbol, funcție de transfer, funcționare, aplicații, model Verilog).

Soluție

Bistabilul T (Toggle, engl.) este un circuit logic secvențial care memorează starea, activ pe frontul semnalului de ceas. În momentul frontului activ al semnalului de ceas, dacă $T = 1$, bistabilul își complementează starea, iar dacă $T = 0$, bistabilul își păstrează starea. În rest, pe o perioadă de semnal de ceas, până la următorul front activ al acestuia, bistabilul memorează starea.

Tabelul de determinare a stării viitoare a bistabilului T și simbolul acestuia sunt:

CK	T	Q^+	Acțiune
↑	0	$\bar{Q}$	păstrează starea
↑	1	$\bar{Q}$	complementează starea
-	-	Q	păstrează starea



Modelul Verilog al bistabilului T se descrie cu o specificație secvențială `always` pe baza tabelului de funcționare.

```
module TFF (
input      ck      , // intrare de ceas
input      rst_n   , // reset asincron activ low
input      T       , // intrare de date (toggle)
output reg  Q      // iesire/stare bistabil
);

always @(posedge ck or negedge rst_n)
if (~rst_n) Q <= 1'b0; else
if (T)      Q <= ~Q;
```

Se remarcă faptul că bistabilul T are nevoie de un semnal suplimentar de reset, care să inițializeze bistabilul.

15. Definiți noțiunea de „**bistabil JK**”. Intrări, ieșiri, simbol, funcție de transfer, funcționare, aplicații.
16. Definiți noțiunea de „**timp de setup**”.
17. Definiți noțiunea de „**LFSR - Linear Feedback Shift Register**”.
18. Definiți noțiunea de „**numărător presetabil de N biți**”. Intrări, ieșiri, simbol, funcție de transfer, funcționare, aplicații, model Verilog.

Soluție

Numărătorul presetabil este un circuit secvențial capabil atât să numere cât și să se inițializeze într-o anumită stare. Facilitățile numărătoarelor se proiectează după necesități, nu sunt „standardizate”. De exemplu, un numărător presetabil cu numărare în sens crescător

are funcționarea descrisă prin tabelul următor. Desigur, comutarea are loc exclusiv în momentele determinate de frontul pozitiv al semnalului de ceas.

ld	cu	q^+	Acțiune
1	0	$data$	încarcă data (presetare)
0	1	$q + 1$	incrementează starea
0	0	q	păstrează starea

Modelul Verilog al numărătorului presetabil descris este următorul. Se consideră numărătorul pe 4 biți.

```
module num_preset (
input          ck      , // intrare de ceas
input          rst_n, // reset asincron activ low
input          ld      , // load, control incarcare
input [3:0]    data   , // date incarcate
input          cu      , // count up
output reg [3:0] q     // iesire/stare numarator
);

always @(posedge ck or negedge rst_n)
if (~rst_n) q <= 4'b0; else
if (ld)     q <= data; else
if (cu)     q <= q + 1;

endmodule // num_preset
```

Se observă că modelarea păstrării stării se face prin lipsa clauzei `else` a ultimei specificații `if`. Adică, dacă nu este activă nici intrarea `ld`, nici intrarea `cu`, numărătorul nu își modifică starea, deci și-o păstrează.

19. Definiți noțiunea de „**organigramă**”.

20. Definiți noțiunea de „**graf de tranziții**”.

21. Definiți noțiunea de „**memorie LIFO**”.

Soluție

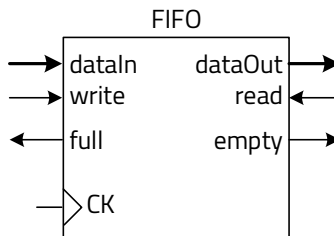
Memoria LIFO („*Last In First Out*”, engl.) este o formă de organizare a memoriei fără adrese, accesul la date făcându-se după regula „*ultima dată scrisă este prima dată citită*”. Memoriile LIFO sunt utilizate pentru implementarea unor stive de date. Organizarea memoriei LIFO constă într-o memorie RAM și un pointer pentru vârful stivei (*Stack Pointer*, *SP*, engl.) implementat cu un numărător cu posibilitate de numărare atât în sens crescător cât și descrescător.

22. Definiți noțiunea de „memorie FIFO”.

Soluție

Memoria FIFO („First In First Out”, engl.) este o formă de organizare a memoriei fără adrese, accesul la date făcându-se după regula „*prima dată scrisă este prima dată citită*”. Memoriile FIFO sunt utilizate pentru implementarea unor cozi de date. Organizarea memoriei FIFO constă într-o memorie RAM și pointere pentru adrese de scriere și citire implementate cu numărătoare care se incrementează la fiecare acțiune a memoriei (scriere sau citire).

Simbolul bloc al unei memorii FIFO este:



Se remarcă prezența unor ieșiri care indică starea memoriei:

full = memorie plină, nu se mai acceptă scrieri;

empty = memorie goală, nu se mai acceptă citiri.

23. Definiți noțiunea de „automat Mealy”.

Soluție

Automatul Mealy este un automat la care ieșirea depinde atât de starea curentă cât și de intrări.

24. Definiți noțiunea de „automat Moore”.

Soluție

Automatul Moore este un automat la care ieșirea depinde exclusiv de starea curentă, nu și de intrări.

25. Definiți noțiunea de „automat cu stări codate one-hot”.

Soluție

Codarea „one-hot” presupune asocierea fiecărei stări cu un bistabil. Starea curentă este determinată de bistabilul a cărui valoare este 1, în timp ce toate celelalte bistabile au valoarea 0. Deși codarea utilizează un număr mai mare de bistabile (mai mare decât numărul

minim necesar), codarea „one-hot” prezintă avantajul realizării unor implementări care operează la frecvențe mai mari, deoarece complexitatea circuitului logic combinațional de stare este mai mică.

26. Definiți noțiunea de „*structură pipeline*”.

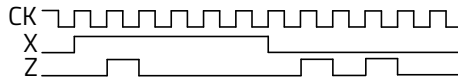
Soluție

Structură pipeline este o modalitate de organizare a unui sistem digital similar cu o „linie de asamblare”. Structura pipeline este formată din succesiuni de circuite combinaționale și registre. Scopul acestui tip de organizare constă în creșterea frecvenței de operare a sistemului prin micșorarea numărului de nivele logice dintre două registre succesive.

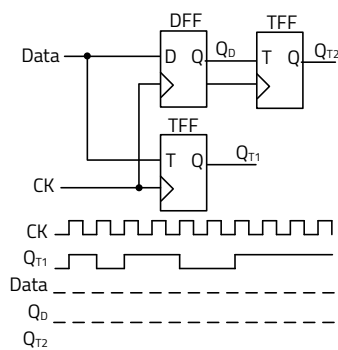
15.1.3 Circuite cu bistabile

1. Proiectați un sistem secvențial sincron cu o intrare X și o ieșire Z . La fiecare front pozitiv al semnalului X apare un puls pe ieșirea Z . La fiecare front negativ al semnalului X apare o secvență de două pulsuri pe ieșirea Z .

Formele de undă și temporizarea sunt prezentate în figură.

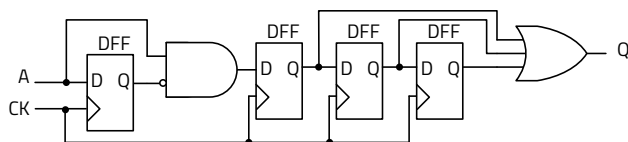


2. Se consideră circuitul din figură. La o ieșire se observă forma de undă prezentată. Să se determine formele de undă ale semnalelor de la ieșirile celorlalte bistabile.



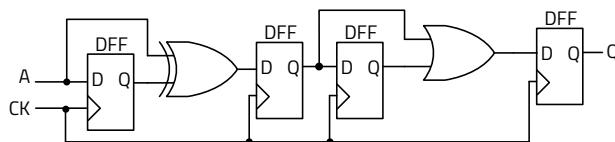
3. Explicați comportamentul circuitului prezentat în figură. Forme de undă în nodurile circuitului. Descrierea funcționării circuitului.

Aplicații.

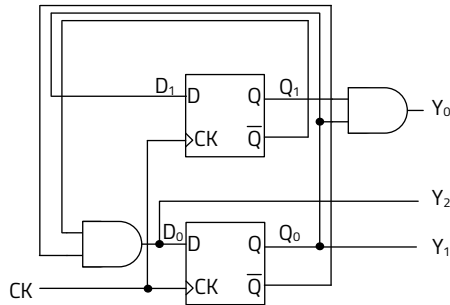


4. Explicați comportamentul circuitului prezentat în figură. Forme de undă în nodurile circuitului. Descrierea funcționării circuitului.

Aplicații.



5. Analizați circuitul prezentat:



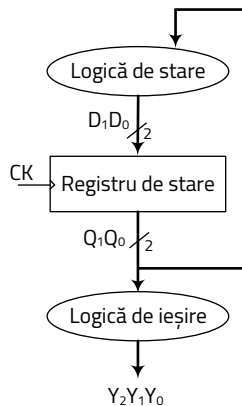
- Schema bloc a automatului, cu precizarea numărului de intrări, numărului de ieșiri, numărului de stări și al tipului automatului.
- Organigrama automatului.
- Forme de undă generate de automat.

Soluție

Circuitul prezentat este un circuit logic secvențial ce conține două bistabile D. Deci, numărul maxim de stări ale circuitului este 4.

Circuitul nu prezintă intrări de date și are 3 ieșiri Y_2, Y_1, Y_0 .

Este un automat de tip Moore (nu are intrări, ieșirea depinde exclusiv de starea prezentă). Schema bloc a automatului este:



Starea prezentă este determinată de ieșirile bistabilelor, Q_1, Q_0 .

Starea viitoare este determinată de intrările bistabilelor, D_1, D_0 .

Ecuatiile stărilor viitoare sunt deduse din circuit:

$$D_1 = Q_0$$

$$D_0 = \overline{Q_1} \cdot \overline{Q_0}$$

Ecuatiile ieșirilor sunt deduse din circuit:

$$Y_2 = \overline{Q_1} \cdot \overline{Q_0}$$

$$Y_1 = Q_0$$

$$Y_0 = Q_1 \cdot Q_0$$

Cu ajutorul ecuațiilor, se poate deduce tabelul de tranziții. Tabelul se completează astfel:

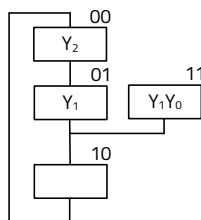
- Se completează coloana stărilor prezente cu codurile tuturor celor 4 stări posibile.
- Pe baza ecuațiilor $D_{1,0}(Q_{1,0})$ se completează coloanele stărilor viitoare.
- Pe baza ecuațiilor $Y_{2,1,0}(Q_{1,0})$ se completează coloanele ieșirilor.

Stare prezentă		Stare viitoare		Ieșiri		
Q_1	Q_0	D_1	D_0	Y_2	Y_1	Y_0
0	0	0	1	1	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	1	1	0	0	1	1

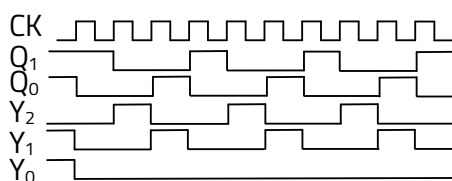
Organigrama automatului se poate deduce din tabelul de tranziții. Organigrama prezintă 4 stări. Trecerea dintr-o stare în alta se face necondiționat, conform tabelului de tranziții. Conform tabelului de tranziții, din starea 00 se trece necondiționat în starea 01, din 01 în 10, din 10 în 00 și din 11 în 10.

Ieșirile sunt active în stările corespunzătoare prezente în tabel. Y_2 este activă în starea 00, Y_1 este activă în stările 01 și 11 iar Y_0 este activă în starea 11.

Organigrama automatului este:



Automatul prezintă 3 stări între care evoluează ciclic, 00, 01, 10. În cazul că automatul se inițializează în starea 11, urmează starea 10 și ciclul se reia. Forme de undă generate de automat pot fi descrise complet dacă se pornește din starea 11. Urmează stările 10, 00, 01, 10, 00,



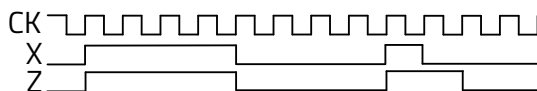
15.1.4 Proiectare circuite secvențiale pe baza specificațiilor

1. Proiectați un sistem secvențial sincron cu o intrare X și o ieșire Z . Între două pulsuri pe intrare există o pauză de minimum 3 perioade de ceas.

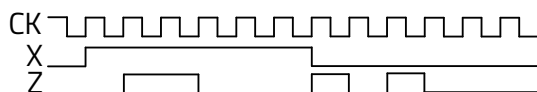
Ieșirea Z are întotdeauna lățimea un număr par de perioade de ceas. Dacă lățimea pulsului X este un număr par de perioade de ceas, atunci Z este la fel ca X .

Dacă lățimea pulsului X este un număr impar de perioade de ceas, atunci pulsul pe ieșirea Z este cu o perioadă mai lat decât cel de pe intrarea X .

Formele de undă și temporizarea sunt prezentate în figură.

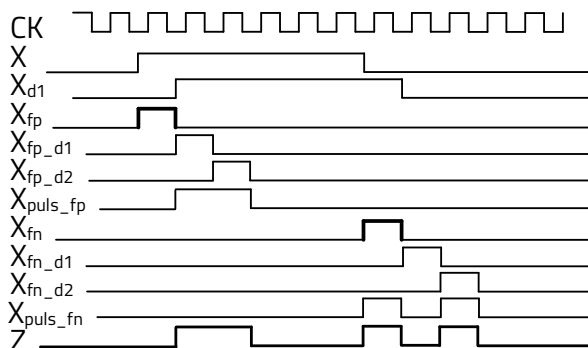


2. Proiectați un sistem secvențial sincron cu o intrare X și o ieșire Z . La fiecare front pozitiv al semnalului X apare un puls pe ieșirea Z . La fiecare front negativ al semnalului X apare o secvență de două pulsuri pe ieșirea Z . Formele de undă și temporizarea sunt prezentate în figură.



Soluție

Rezolvarea problemei constă în detectarea celor două fronturi (pozitiv și negativ) ale semnalului X și prelucrarea pulsurilor pentru a genera secvența cu forma și temporizarea dorite.



X_{d1} este X întârziat cu o perioadă, printr-un bistabil.

X_{fp} este un puls care marchează frontul pozitiv al semnalului X :
 $X_{fp} = X \cdot \overline{X_{d1}}$.

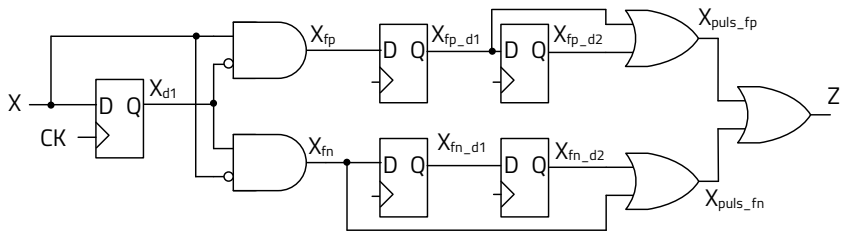
X_{fp_d1} și X_{fp_d2} sunt versiuni întârziate cu 1 și 2 perioade față de X_{fp} , utilizate pentru a forma secvența necesară a fi generată la frontul pozitiv: $X_{puls_fp} = X_{fp_d1} + X_{fp_d2}$.

X_{fn} este un puls care marchează frontul negativ al semnalului X :
 $X_{fn} = \overline{X} \cdot X_{d1}$.

X_{fn_d1} și X_{fn_d2} sunt versiuni întârziate cu 1 și 2 perioade față de X_{fn} , utilizate pentru a forma secvența necesară a fi generată la frontul negativ: $X_{puls_fn} = X_{fn} + X_{fn_d2}$.

În final $Z = X_{puls_fp} + X_{puls_fn}$.

Circuitul rezultat este:



Modelul Verilog al circuitului proiectat este:

```
module pulse1(
  input  ck ,    // intrare de ceas
  input  X  ,    // intrare de date
  output Z      // iesire, puls
);
  reg      Xd1          ;
  wire     Xfp          , Xfn          ;
  reg      Xfp_d1       , Xfn_d1       ;
  reg      Xfp_d2       , Xfn_d2       ;
  wire     Xpuls_fp     , Xpuls_fn     ;

  // bistabile
  always @(posedge ck) begin
    Xd1      <= X          ;
    Xfp_d1   <= Xfp        ;
    Xfn_d1   <= Xfn        ;
```

```

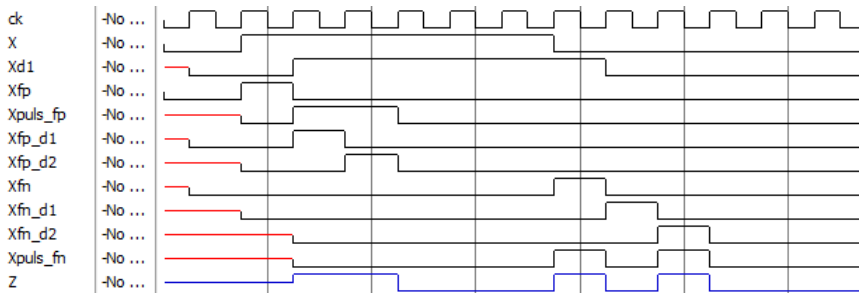
Xfp_d2 <= Xfp_d1 ;
Xfn_d2 <= Xfn_d1 ;
end

// porti logice
assign Xfp      = X & (~Xd1)      ;
assign Xfn      = (~X) & Xd1      ;
assign Xpuls_fp = Xfp_d1 | Xfp_d2 ;
assign Xpuls_fn = Xfn | Xfn_d2    ;
assign Z        = Xpuls_fp | Xpuls_fn ;

endmodule // pulse1

```

Formele de undă rezultate din simulare sunt identice cu cele proiectate.

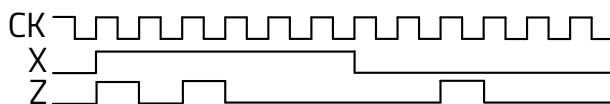


3. Proiectați un sistem secvențial sincron cu o intrare X și o ieșire Z .

La fiecare front pozitiv al semnalului X apar două pulsuri pe ieșirea Z .

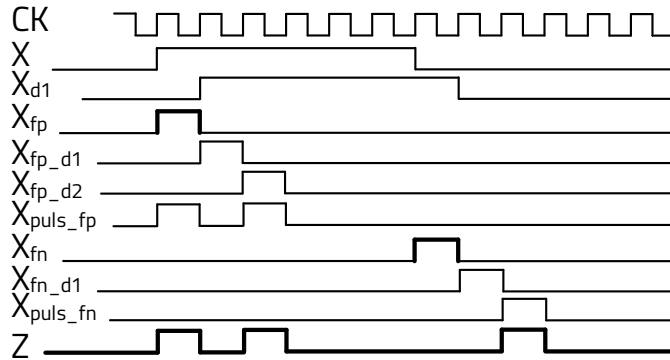
La fiecare front negativ al semnalului X apare un puls pe ieșirea Z .

Formele de undă și temporizarea sunt prezentate în figură.



Soluție

Rezolvarea problemei constă în detectarea celor două fronturi (pozitiv și negativ) ale semnalului X și prelucrarea pulsurilor pentru a genera secvența cu forma și temporizarea dorite.



X_{d1} este X întârziat cu o perioadă, printr-un bistabil.

X_{fp} este un puls care marchează frontul pozitiv al semnalului X :
 $X_{fp} = X \cdot \overline{X_{d1}}$.

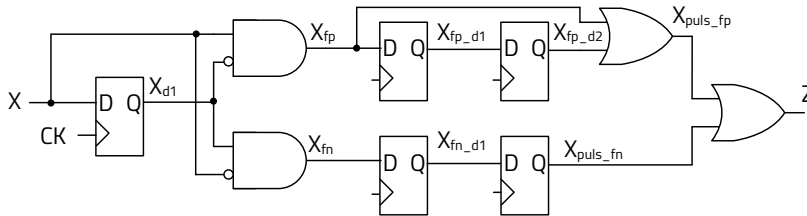
X_{fp_d1} și X_{fp_d2} sunt versiuni întârziate cu 1 și 2 perioade față de X_{fp} , utilizate pentru a forma secvența necesară a fi generată la frontul pozitiv: $X_{puls_fp} = X_{fp} + X_{fp_d2}$.

X_{fn} este un puls care marchează frontul negativ al semnalului X :
 $X_{fn} = \overline{X} \cdot X_{d1}$.

X_{fn_d1} și X_{puls_fn} sunt versiuni întârziate cu 1 și 2 perioade față de X_{fn} .

În final $Z = X_{puls_fp} + X_{puls_fn}$.

Circuitul rezultat este:

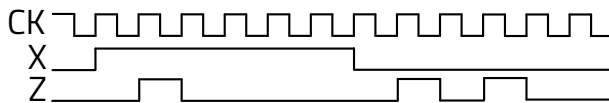


4. Proiectați un sistem secvențial sincron cu o intrare X .

La fiecare front pozitiv al semnalului de intrare X la ieșirea Z apare un puls cu lățime de două perioade de ceas.

La fiecare front negativ al semnalului de intrare X la ieșirea Z apar două pulsuri consecutive.

Formele de undă și temporizarea sunt prezentate în figură.



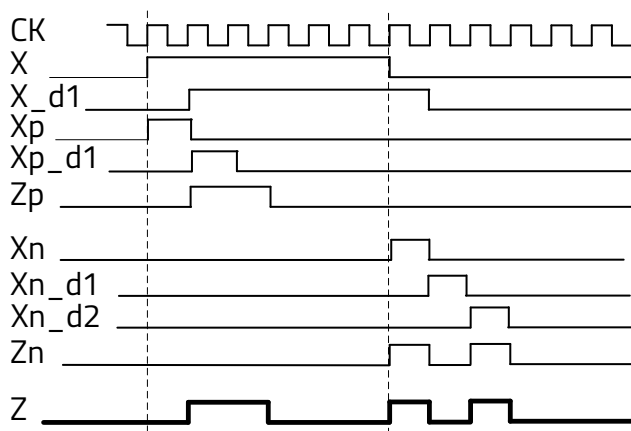
Soluție

Evenimentele care determină apariția pulsurilor la ieșirea Z sunt determinate de fronturile semnalului de intrare X . X_{d1} este versiunea întârziată cu un tact a intrării (semnal obținut la ieșirea unui bistabil D având la intrare X). Semnalul X_p semnalează apariția unui front pozitiv pe semnalul X prin funcția ȘI între X și $\overline{X_{d1}}$. X_{p_d1} este X_p întârziat cu un tact. Z_p , pulsul generat pe ieșire la frontul pozitiv al intrării, se obține din funcția OR între X_p și X_{p_d1} , întârziată printr-un bistabil (pentru respectarea temporizării impuse).

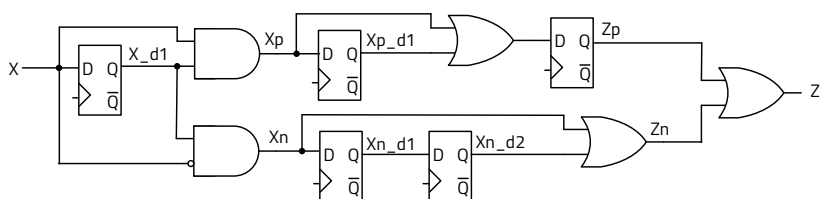
Semnalul X_n semnalează apariția unui front negativ pe semnalul X prin funcția ȘI între $\overline{X}$ și X_{d1} . X_{n_d1} și X_{n_d2} sunt X_n întârziat cu unul, respectiv două tacte. Z_n , pulsul generat pe ieșire la frontul negativ al intrării, se obține din funcția OR între X_n și X_{n_d2} , conform formei de undă și a temporizării impuse.

În final, ieșirea Z se obține prin funcția OR între Z_p și Z_n .

Formele de undă ce caracterizează circuitul proiectat sunt:

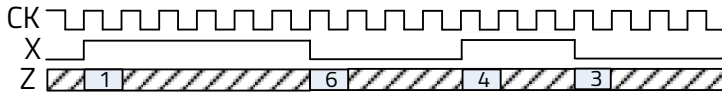


Circuitul cu bistabile care implementează funcționalitatea impusă este prezentat în continuare.



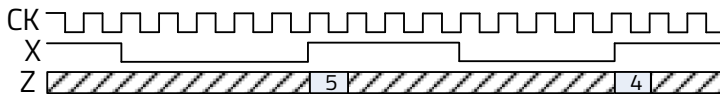
- Proiectați un sistem secvențial sincron cu o intrare X care este constantă un număr de maximum 7 perioade de ceas. După fiecare puls ($X = 1$) al intrării, pe ieșirea Z (3 biți) se prezintă lățimea pulsului (exprimată în perioade de ceas).

Formele de undă și temporizarea sunt prezentate în figură.

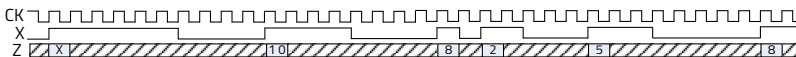


6. Proiectați un sistem secvențial sincron cu o intrare X . Pauza maximă între două pulsuri consecutive pe X este de 7 perioade de ceas. După fiecare pauză pe intrarea X , ieșirea Z (3 biți) prezintă durata pauzei (exprimată în perioade de ceas).

Formele de undă și temporizarea sunt prezentate în figură.



7. Proiectați un sistem secvențial sincron cu o intrare X care prezintă maximum 15 perioade de ceas între două fronturi crescătoare consecutive. După fiecare front crescător al intrării X , pe ieșirea Z (4 biți) se prezintă durata între ultimele două fronturi crescătoare ale intrării X (exprimată în perioade de ceas). Formele de undă și temporizarea sunt prezentate în figură.

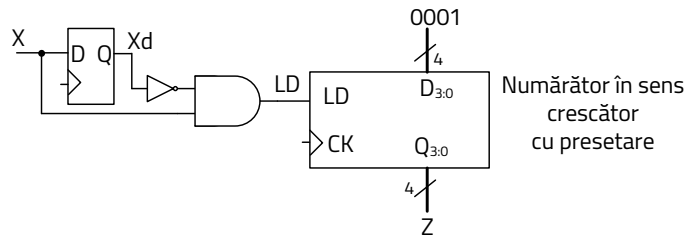
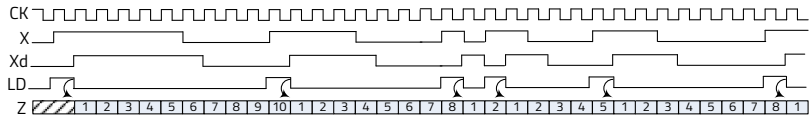


Soluție

Circuitul trebuie să numere în sens crescător. Evenimentul care determină re-inițializarea numărătorului în starea 1 este determinat de apariția frontului crescător al semnalului de intrare X . Cu un circuit format dintr-un bistabil și porți logice se detectează momentul apariției frontului crescător al intrării X ($X = 1$ și $X_d = 0$, unde s-a notat cu X_d valoarea X întârziată cu un tact). Acest puls se aplică

pe intrarea de inițializare sincronă a numărătorului LD și determină încărcarea numărătorului cu 1.

La ieșirea Z se va prezenta ieșirea numărătorului. Formele de undă rezultate și circuitul proiectat sunt prezentate în continuare.



Modelul Verilog al circuitului proiectat este:

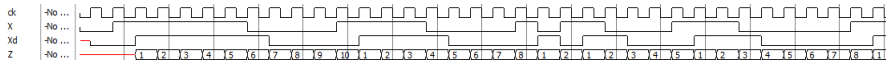
```
module count_period(
input          ck      , // intrare de ceas
input          X       , // intrare de date
output reg [3:0] Z     // durata
);
reg           Xd ;

always @(posedge ck)
    Xd <= X ;

always @(posedge ck)
    if (X & (~Xd)) Z <= 4'b0001; else
        Z <= Z + 1 ;

endmodule // count_period
```

Formele de undă rezultate din simularea modelului sunt similare cu cele proiectate.



15.1.5 Proiectare circuite secvențiale după model

1. Semnalizare încălzire scaune auto.



Sistemul are un singur buton a cărui apăsare succesivă comută o secvență de patru stări de aprindere a LED-urilor: 000, 111, 110, 100.

S-a notat starea ieșirii, LED-ul:

0 - stins

1 - aprins

2. Semnalizator auto secvențial.



Proiectați un sistem digital cu un buton de intrare și 8 ieșiri. Dacă butonul este apăsat (comanda semnalizării) cele 8 ieșiri prezintă o secvență de 9 stări care simulează creșterea unei linii.

```
00000000
00000001
00000011
00000111
00001111
00011111
00111111
00111111
01111111
11111111
```

S-a notat starea ieșirii, LED-ul:

0 - stins

1 - aprins

3. Semnalizare biciclete.



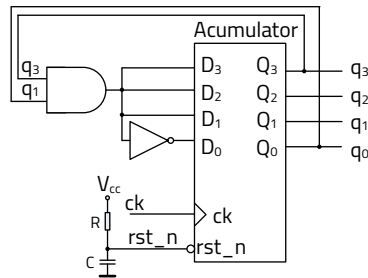
Sistemul are 4 faze comutate prin relaxarea singurului buton:

- LED aprins,
- LED cu lumină intermitentă, 5 aprinderi/secundă,
- LED cu lumină intermitentă rapidă, serii scurte de 5 aprinderi urmate de o pauză,
- LED stins.

15.2 Ingineri avansați

15.2.1 Analiza circuitelor digitale

1. Se consideră circuitul:



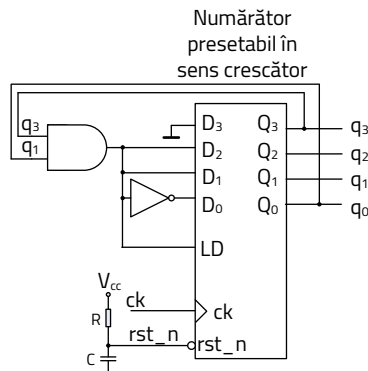
La fiecare front crescător de ceas, registrul acumulator pe 4 biți acumulează valoarea de pe intrare. Adică, dacă starea curentă este Q și pe intrare valoarea este D atunci starea viitoare va fi $Q^+ = Q + D$ (suma dintre valoarea curentă și valoarea de pe intrare).

- Explicați funcționarea circuitului.
- Reprezentați grafic comportamentul circuitului.
- Ce rol are condensatorul prezent în schemă?
- Model Verilog pentru tot circuitul.

Funcționarea acumulatorului este descrisă de codul Verilog:

```
input      [3:0] D ;    // intrarea de date (4 biti)
output reg [3:0] Q ;    // iesirea acumulatorului (4 biti)
always @(posedge ck or negedge rst_n)
if (~rst_n) Q <= 4'b0000; else // initializare (reset activ "0")
    Q <= Q + D;               // acumularea intrarii
```

2. Se consideră circuitul:



Numărătorul presetabil comută sincron cu frontul pozitiv al semnalului de ceas.

Dacă $LD = 1$, atunci numărătorul se încarcă cu valoarea aflată pe intrarea D .

Dacă $LD = 0$, atunci numărătorul se incrementează, starea viitoare fiind $Q^+ = Q + 1$.

- Explicați funcționarea circuitului.
- Reprezentați grafic comportamentul circuitului.
- Ce rol are condensatorul prezent în schemă?
- Model Verilog pentru tot circuitul.

Funcționarea numărătorul presetabil este descrisă în Verilog astfel:

```
module cu_preset (
input          ck      , // intrare de ceas
input          rst_n,   // reset asincron activ low
input          LD      , // intrare incarcare
input [3:0]    D       , // intrarea de date presetate (4 biti)
output reg [3:0] Q      // iesirea numaratorului (4 biti)
);

always @(posedge ck or negedge rst_n)
if (~rst_n) Q <= 4'b0000; else // initializare (reset activ "0")
if (LD)    Q <= D; else      // se incarca noua stare
Q <= Q + 1;                // se incrementeaza starea

endmodule // cu_preset
```

Soluție

Circuitul prezintă un numărător de 4 biți, în sens crescător, presetabil.

Din circuit, se deduce ecuația intrării de încărcare a numărătorului:

$$LD = q_3 \cdot q_1$$

Deci, numărătorul se încarcă dacă starea prezentă satisface condiția $LD = q_3 \cdot q_1 = 1$ adică $q_3 = 1$ și $q_1 = 1$, adică starea respectă pattern-ul $q_3q_2q_1q_0 = 1X1X$.

Patternul este respectat de stările:

$$10|_{10} = 1010|_2, 11|_{10} = 1011|_2, 14|_{10} = 1110|_2, 15|_{10} = 1011|_2.$$

Din circuit se deduc ecuațiile intrărilor de date:

$$D_3 = 0$$

$$D_2 = q_3 \cdot q_1$$

$$D_1 = q_3 \cdot q_1$$

$$D_0 = \overline{q_3 \cdot q_1}$$

Stare prez. $q[3:0]$		LD	Intrări $D[3:0]$		Acțiune	Stare viit. $q^+[3:0]$	
0	0000	0	1	0001	Increment	1	0001
1	0001	0	1	0001	Increment	2	0010
2	0010	0	1	0001	Increment	3	0011
3	0011	0	1	0001	Increment	4	0100
4	0100	0	1	0001	Increment	5	0101
5	0101	0	1	0001	Increment	6	0110
6	0110	0	1	0001	Increment	7	0111
7	0111	0	1	0001	Increment	8	1000
8	1000	0	1	0001	Increment	9	1001
9	1001	0	1	0001	Increment	10	1010
10	1010	1	6	0110	Încărcare	6	0110
11	1011	1	6	0110	Încărcare	6	0110
12	1100	0	1	0001	Increment	13	1101
13	1101	0	1	0001	Increment	14	1110
14	1110	1	6	0110	Încărcare	6	0110
15	1111	1	6	0110	Încărcare	6	0110

Se observă secvența repetitivă de stări 6, 7, 8, 9, 10, 6, ...

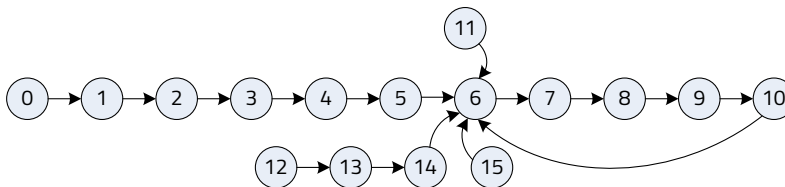
Se analizează și comportamentul din stările neprezente în ciclu: 11, 12, 13, 14, 15.

Numărătorul parcurge în sens crescător stările:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

La apariția stării $q[3:0] = 1010|_2 = 10|_{10}$ se activează $LD = 1$ și intrările devin $D[3:0] = 0110|_2 = 6|_{10}$. Din stările 10, 11, 14, 15 se produce o încărcare și starea devine 6. Din stările 12, 13 circuitul numără în sens crescător.

Rezultă graful de tranziții:



Condensatorul din schemă, împreună cu rezistența, formează un circuit

de generare a unui puls de reset asincron, activ 0, la punerea sub tensiune a circuitului. Din punct de vedere digital, semnalul rst_n va avea inițial valoarea logică 0 (fiind activ) iar după un timp determinat de încărcarea condensatorului va ajunge la valoarea logică 1, deci se va dezactiva. Valorile $R \times C$ determină constanta de timp a circuitului de integrare și lățimea pulsului de reset asincron.

Modelul Verilog al automatului implementat cu o instanțiere de numărător presetabil este următorul:

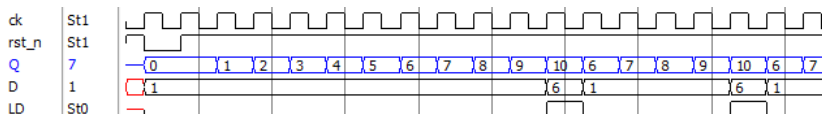
```
module counter_preset (
input          ck      , // intrare de ceas
input          rst_n,   // reset asincron activ low
output reg [3:0] Q      // iesirea numaratorului (4 biti)
);
wire [3:0] D ;
wire      LD ;

assign LD = Q[1] & Q[3];
assign D  = {1'b0, LD, LD, ~LD};

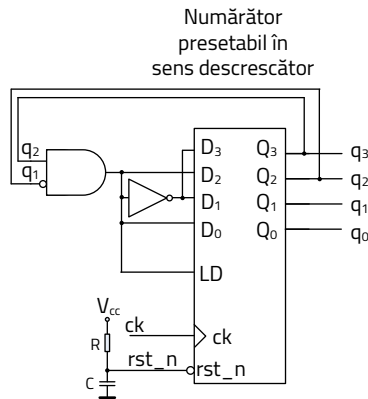
cu_preset i_cu_preset (
.ck      (ck      ),
.rst_n   (rst_n),
.D       (D       ),
.LD      (LD      ),
.Q       (Q       )
);

endmodule // counter_preset
```

Se observă pe formele de undă comportamentul proiectat, cu repetarea secvenței de stări 6, 7, 8, 9, 10, 6, ...



3. Se consideră circuitul:



Numărătorul presetabil comută sincron cu frontul pozitiv al semnalului de ceas.

Dacă $LD = 1$, atunci numărătorul se încarcă cu valoarea aflată pe intrarea D .

Dacă $LD = 0$, atunci numărătorul se decrementează, starea viitoare fiind $Q^+ = Q - 1$.

- Explicați funcționarea circuitului.
- Reprezentați grafic comportamentul circuitului.
- Ce rol are condensatorul prezent în schemă?
- Model Verilog pentru tot circuitul.

Funcționarea numărătorul presetabil este descrisă în Verilog astfel:

```
input          LD; // incarcare date
input  [3:0] D ; // intrarea de date presetate (4 biti)
output reg [3:0] Q ; // iesirea numaratorului (4 biti)
always @(posedge ck or negedge rst_n)
if (~rst_n) Q <= 4'b0000; else // initializare (reset activ "0")
if (LD)    Q <= D; else      // se incarca noua stare
          Q <= Q - 1;        // se decrementeaza starea
```

Soluție

Circuitul prezintă un numărător de 4 biți, în sens descrescător, presetabil. Intrarea de încărcare are ecuația:

$$LD = q_3 \cdot \overline{q_2}, \text{ activă când } q[3:0] = 10XX \text{ (adică } 8, 9, 10, 11|_{10}).$$

Din circuit, se deduc ecuațiile intrărilor de presetare ale numărătorului:

$$D_3 = \overline{q_3 \cdot \overline{q_2}}$$

$$D_2 = q_3 \cdot \overline{q_2}$$

$$D_1 = q_3 \cdot \overline{q_2}$$

$$D_0 = q_3 \cdot \overline{q_2}$$

Numărătorul va parcurge în sens descrescător stările:

7, 6, 5, 4, 3, 2, 1, 0, 15, 14, 13, 12, 11.

La apariția stării $q[3:0] = 1011|_2 = 11|_{10}$ (prima din setul de stări care respectă patternu-ul care determină presetarea numărătorului), se activează $LD = 1$ și intrările devin $D[3:0] = 0101|_2 = 5|_{10}$.

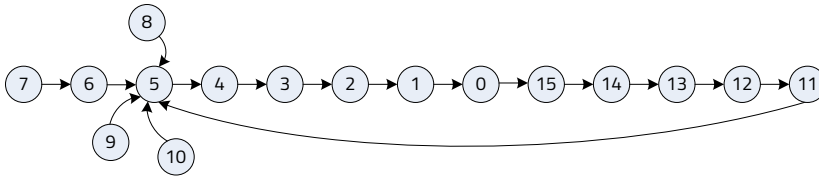
Datele pot fi centralizate într-un tabel:

Stare prez. q[3:0]		LD	Intrări D[3:0]		Acțiune	Stare viit. q ⁺ [3:0]	
0	0000	0	10	1010	Decrement	15	1111
15	1111	0	10	1010	Decrement	14	1110
14	1110	0	10	1010	Decrement	13	1101
13	1101	0	10	1010	Decrement	12	1100
12	1100	0	10	1010	Decrement	11	1011
11	1011	1	5	0101	Încărcare	5	0101
5	0101	0	10	1010	Decrement	4	0100
4	0100	0	10	1010	Decrement	3	0011
3	0011	0	10	1010	Decrement	2	0010
2	0010	0	10	1010	Decrement	1	0001
1	0001	0	10	1010	Decrement	0	0000
6	0110	0	10	1010	Decrement	5	0101
7	0111	0	10	1010	Decrement	6	0110
8	1000	1	5	0101	Încărcare	5	0101
9	1001	1	5	0101	Încărcare	5	0101
10	1010	1	5	0101	Încărcare	5	0101

Se observă secvența repetitivă de stări 0, 15, 14, 13, 12, 11, 5, 4, 3, 2, 1, 0, ...

Se analizează și comportamentul circuitului din stările neprezente în ciclu 6, 7, 8, 9, 10.

Rezultă graful de tranziții:



Condensatorul din schemă, împreună cu rezistența, formează un circuit de generare a unui puls de reset asincron, activ 0, la punerea sub tensiune a circuitului. Din punct de vedere digital, semnalul rst_n va avea inițial valoarea logică 0 (fiind activ) iar după un timp determinat de încărcarea condensatorului va ajunge la valoarea logică 1, deci se va dezactiva. Valorile $R \times C$ determină constanta de timp a circuitului de integrare și lățimea pulsului de reset asincron.

15.2.2 Proiectare sisteme digitale de control

1. Un **aparat de radio** are un buton prin apăsare pentru selecția digitală a frecvenței.



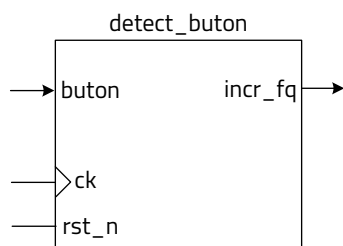
La o apăsare, frecvența se incrementează cu o unitate. Dacă butonul se ține apăsat mai mult de 2 secunde, frecvența se incrementează automat cu 10 unități pe secundă (*auto-repeat*).

Să se proiecteze un sistem secvențial sincron cu frecvența de 100 MHz care să genereze *pulsurile de incrementare* pe baza semnalului provenit de la *buton*.

Soluție

Modulul proiectat este un sistem sincron cu o intrare denumită *buton* și o ieșire denumită *incr_freq*.

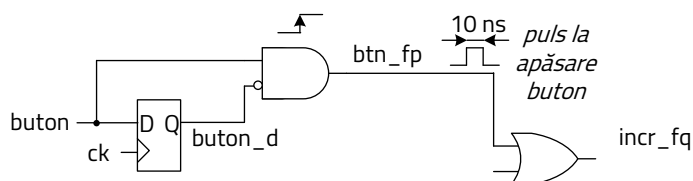
Perioada semnalului de ceas este $T_{CK} = 1/f_{CK} = 1/100\text{MHz} = 10\text{ns}$.



Ieșirea *incr_fq* se activează din două motive (SAU logic):

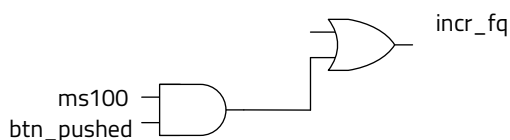
- la detectarea apăsării butonului (front pozitiv la intrării *buton*) și
- periodic, la fiecare 100 ms, dacă butonul este ținut apăsat (*buton* = 1) mai mult de 2 secunde.

Prima condiție se monitorizează cu un circuit detector de front pozitiv:



A doua condiție se obține prin funcția AND între:

- un semnal activ dacă butonul este ținut apăsat mai mult de 2 secunde (denumit *btn_pushed*) și
- un semnal care prezintă un puls cu lățime de o perioadă de ceas la fiecare 100 ms (denumit *ms100*).



Semnalul *btn_pushed* se obține la ieșirea unui bistabil RS care se setează dacă *buton* = 1 mai mult de 2 secunde și se resetează la revenirea butonului (*buton* comută din 1 în 0, *btn_fn* = 1). Cât timp *btn_pushed* = 1 la ieșire se propagă un set de pulsuri de la un numărător cu perioadă de 100 ms.

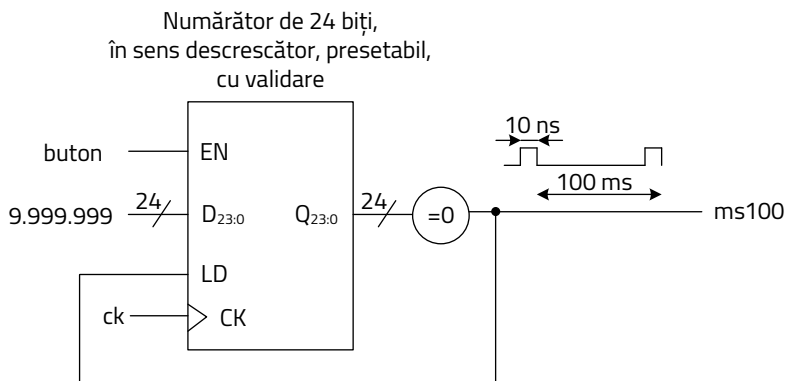
Numărătorul cu perioadă de 100 ms contorizează un număr de perioade egal cu raportul între intervalul de timp și perioada de ceas:

$$100ms/10ns = 10^7 = 10.000.000$$

Numărul de biți necesari este 24 deoarece $2^{24} = 16M > 10.000.000$.

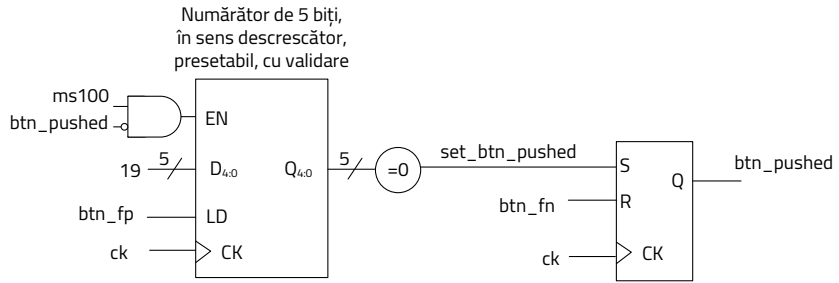
Numărătorul trebuie să aibă următoarele caracteristici:

- facilitate de presetare la valoarea 9.999.999 (numărarea se face între 9.999.999 și 0);
- semnalare printr-un puls a ajungerii la zero (acest puls va preseta număratorul, astfel încât numărătoarea să fie reluată);
- facilitate de validare doar dacă butonul este apăsat (*buton* = 1).

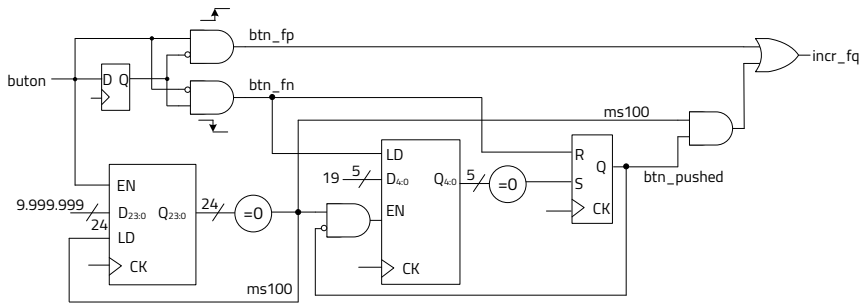


Durata de 2 secunde se măsoară (cu un alt numărător) ca fiind 20 de pulsuri ale semnalului *ms100*. Numărătorul are 5 biți ($2^5 > 20$) și următoarele caracteristici:

- numără în sens descrescător la fiecare puls cu perioada de 100 ms (*ms100* = 1);
- se inițializează cu 19 la apăsarea pe buton (numărarea va fi între 19 și 0);
- dacă ajunge la zero, activează un indicator (pe baza căruia se setează *btn_pushed* = 1, semnificând apăsarea pe buton menținută mai mult de 2 secunde).



Circuitul care implementează modulul cerut este:



Codul care modelează circuitul cerut este prezentat în continuare.

```
module detect_buton #(
    parameter NO_PERIODS = 'd10_000_000 ,
    parameter NO_SEC      = 'd2
) (
    input  ck      , // intrare de ceas
    input  rst_n   , // reset asincron activ low
    input  buton   , // intrare de la buton
    output incr_fq // puls incrementare frecventa
) ;

    reg      buton_d      ; // intrare buton intarziata
    wire     bnt_fp       ;
    wire     bnt_fn       ;
    reg [24-1:0] num_100ms ;
    reg [ 5-1:0] num_2sec  ;
    wire     ms100        ;
```

```

wire          set_btn_pushed ;
reg           btn_pushed      ;

always @(posedge ck)
    buton_d <= buton ;

assign bnt_fp = buton & (~buton_d); // front pozitiv
assign bnt_fn = (~buton) & buton_d ; // front negativ

always @(posedge ck or negedge rst_n)
if (~rst_n) num_100ms <= 'b0 ; else
if (ms100) num_100ms <= NO_PERIODS-1 ; else
if (buton) num_100ms <= num_100ms - 1;

assign ms100 = ~|num_100ms ; // = 0

always @(posedge ck or negedge rst_n)
if (~rst_n) num_2sec <= 'b0 ; else
if (bnt_fp) num_2sec <= NO_SEC*10-1 ; else
if (ms100 & ~btn_pushed) num_2sec <= num_2sec - 1;

assign set_btn_pushed = buton & (~|num_2sec); // = 0

always @(posedge ck or negedge rst_n)
if (~rst_n) btn_pushed <= 1'b0; else
if (set_btn_pushed) btn_pushed <= 1'b1; else
if (bnt_fn) btn_pushed <= 1'b0;

assign incr_fq = bnt_fp | (ms100 & btn_pushed);

endmodule // detect_buton

```

Formele de undă obținute din simulare certifică funcționarea corectă a circuitului. Se observă un singur puls *incr_fq* la o apăsare scurtă pe buton și un puls urmat de un tren de impulsuri în cazul unei apăsări lungi pe buton.

- Manualul de utilizare a sursei de tensiune neîntreruptibile Smart-UPS APC prezintă panoul frontal și modalitatea de pornire și testare prin acționarea aceluiași buton.

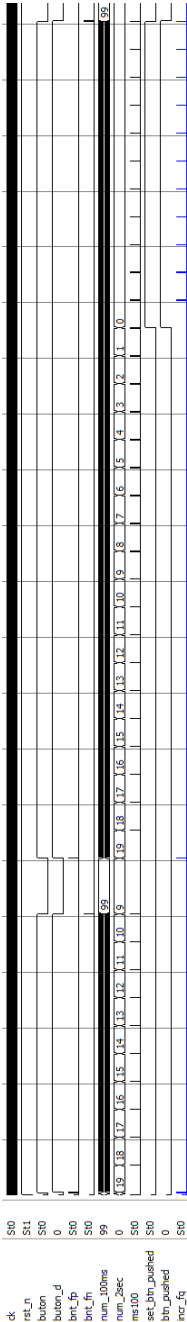
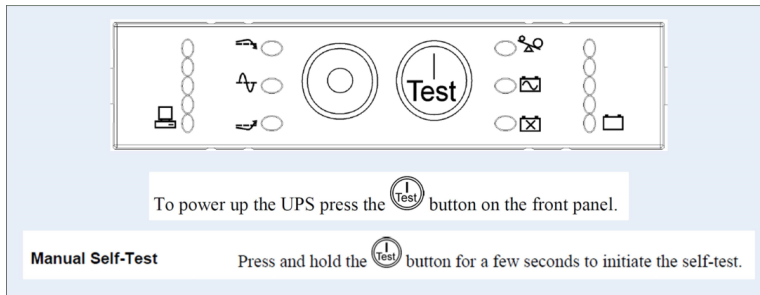


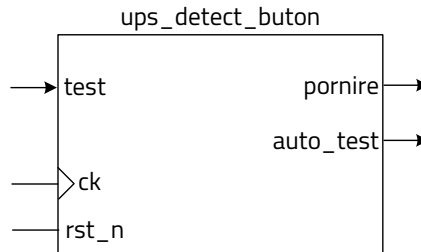
Figura 15.1 Forme de undă obținute la simularea butonului de setare a frecvenței la radio. Puls *incr_fn* la apăsare scurtă și tren de pulsuri la apăsare lungă.



Proiectati un sistem digital secvențial sincron, având frecvența de ceas de 100 MHz, care preia intrarea de la butonul *test* și generează două impulsuri (cu lățime o perioadă de ceas) pentru începerea celor două funcții: *pornire* și *auto_test*.

Soluție

Modulul proiectat este un sistem sincron cu o intrare denumită *test* și două ieșiri denumite *pornire* și *auto_test*. Perioada semnalului de ceas este $T_{CK} = 1/f_{CK} = 1/100MHz = 10ns$.



Cele două ieșiri prezintă câte un puls cu lățimea egală cu o perioadă a semnalului de ceas în următoarele condiții:

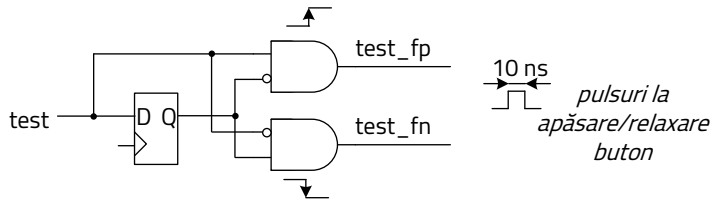
- *pornire*, în momentul relaxării butonului (*test* comută din 1 în 0) dacă intervalul de timp de la apăsarea butonului a fost mai mic decât „câteva secunde”;
- *auto_test*, în momentul relaxării butonului (*test* comută din 1 în 0) dacă intervalul de timp de la apăsarea butonului a fost mai mare decât „câteva secunde”.

Se consideră intervalul de timp a fi egal cu 2 secunde.

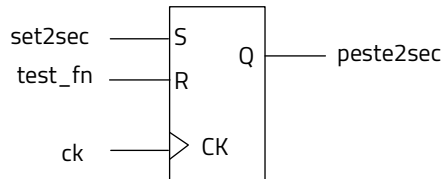
Cele 2 secunde trebuie măsurate din momentul apăsării butonului.

După scurgerea intervalului impus de 2 secunde, se setează un indicator *peste2sec*. Dacă depistarea relaxării butonului se face când *peste2sec* = 0, atunci se activează *pornire*. Altfel, dacă depistarea relaxării butonului se face când *peste2sec* = 1, atunci se activează *auto_test*. Este necesar un circuit demultiplexor 1:2, care să aibă conectat pe selecție semnalul *peste2sec* și la intrarea de date pulsul care marchează frontul negativ al intrării *test*.

Circuitele care generează pulsuri la fronturile semnalului de intrare *test*:



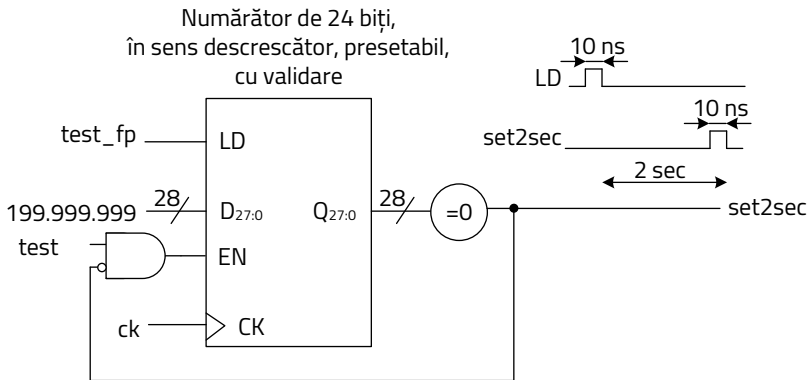
Semnalul *peste2sec* se obține la ieșirea unui bistabil RS care se setează dacă *test* = 1 mai mult de 2 secunde și se resetează la revenirea butonului (*test* comută din 1 în 0, *test_fn* = 1).



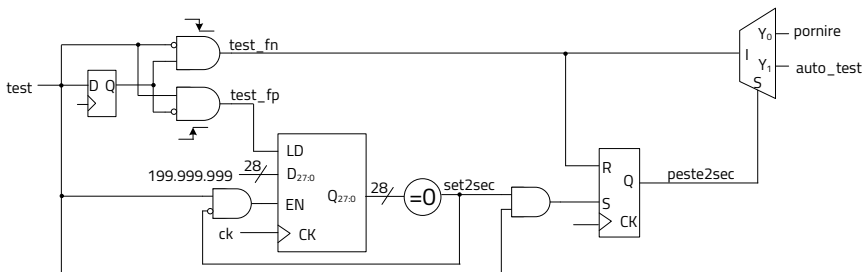
Numărătorul pentru măsurarea intervalului de 2 secunde contorizează un număr de perioade egal cu raportul între intervalul de timp și perioada de ceas: $2sec/10ns = 2 \cdot 10^8 = 200.000.000$. Numărul de biți necesari este 28 deoarece $2^{28} = 256M > 200.000.000$.

Numărătorul trebuie să aibă următoarele caracteristici:

- facilitare de presetare la valoarea 199.999.999 (numărarea se face între 199.999.999 și 0);
- semnalare printr-un puls a ajungerii la zero (acest puls va opri numărătorul);
- facilitare de validare doar dacă butonul este apăsat (*test* = 1).



Circuitul care implementează modulul cerut este:



Codul care modelează circuitul cerut este prezentat în continuare.

```

module ups_detect #(
parameter NO_PERIODS = 'd200_000_000
) (
input    ck                , // intrare de ceas
input    rst_n             , // reset asincron activ low
input    test              , // intrare de la test
output    pornire          , // puls pornire
output    auto_test        , // puls auto-testare
);

reg        test_d          ; // intrare test intarziata
wire       test_fn         ;
wire       test_fp         ;
reg [28-1:0] num_2sec      ;

```

```

wire          set2sec    ;
reg           peste2sec  ;

always @(posedge ck)
    test_d <= test ;

assign test_fp = test & (~test_d); // front pozitiv
assign test_fn = (~test) & test_d ; // front negativ

always @(posedge ck or negedge rst_n)
if (~rst_n) peste2sec <= 1'b0; else
if (set2sec) peste2sec <= 1'b1; else
if (test_fn) peste2sec <= 1'b0;

always @(posedge ck or negedge rst_n)
if (~rst_n) num_2sec <= NO_PERIODS-1 ; else
if (test_fp) num_2sec <= NO_PERIODS-1 ; else
if (test & ~set2sec) num_2sec <= num_2sec - 1 ;

assign set2sec = test & (~|num_2sec) & (~peste2sec); // = 0

assign pornire    = (~peste2sec) & test_fn;
assign auto_test  = peste2sec & test_fn;

endmodule // ups_detect

```

Formele de undă obținute din simulare certifică funcționarea corectă a circuitului. Se observă apariția unui puls *pornire* la o apăsare scurtă pe buton și a unui puls *auto_test* în cazul unei apăsări lungi pe butonul *test*.

- Manualul de utilizare a **automatului de cafea Tassimo** specifică următoarele două acțiuni comandate de la același buton:

- *automat* (inițierea procedurii pentru a face o cafea) și
- *decalcfiere* (inițierea procedurii de curățare periodică).

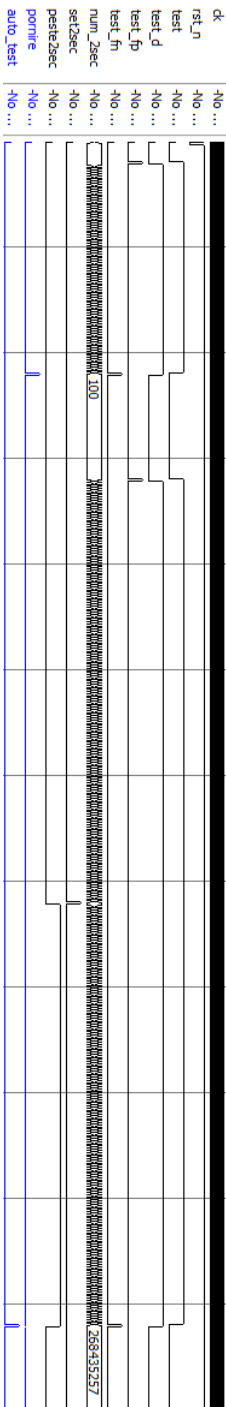


Figura 15.2 Forme de undă obținute la simularea butonului sursei de tensiune neîntreruptibile APC Smart-UPS. Puls *porfire* la apăsare scurtă și puls *auto_test* la apăsare lungă.

1

1. Automat:
Porniți procesul de infuzare prin apăsarea butonului de pornire/oprire.

Efectuarea programului de decalcifiere

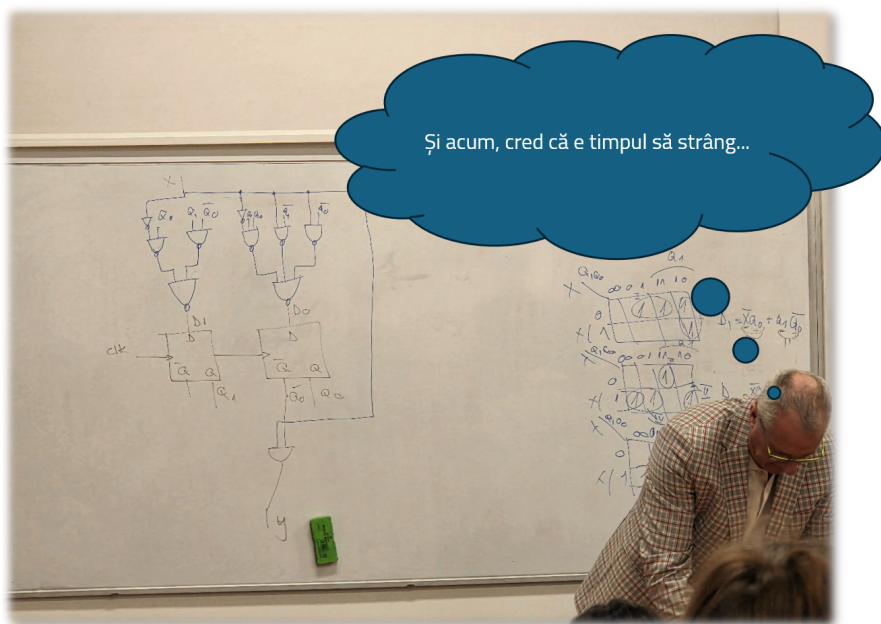
► **Țineți apăsat butonul de Pornire/ Opre** timp de cel puțin 5 secunde pentru a porni programul de decalcifiere.

Proiectați un sistem digital secvențial sincron, având frecvența de ceas de 100 MHz, care preia intrarea de la *butonul pornire/oprire* și generează *două impulsuri* (cu lățime o perioadă de ceas) pentru începerea celor două funcții: *automat* și *decalcifiere*.

Grăirea de încheiere

Într-o perioadă în care performanța profesorilor este măsurată
exclusiv prin numărul de articole publicate,
dăruiesc generațiilor viitoare o

„Carte de învățătură”.



ELECTRONICĂ DIGITALĂ

CARTE DE ÎNVĂȚĂTURĂ

TOMUL PENTRU CHIBZUIT ȘI BUCHISIT
DAN NICULA

Text generat cu ChatGPT pe baza unei structuri impuse și a unei indicații de stil.

📖 Această carte face parte din preacinstita serie „Carte de învățatură de Electronică Digitală”, un fel de almanah pentru sufletele curioase 🤔 și logica răzvrătită 🤖. Scrisă cu sudoare de inginer 💦 și grijă de dascăl 🍎, ediția 2025 aduce cu sine o așezare mai cu cap 🧠 a cunoștințelor, pe măsura vremurilor ⌚ și a sesiunilor care vin pe neașteptate 📅.

📖 Ea slujește ca sprijin pentru curs, seminar, laborator, ori pentru studiul de la ceas de noapte 🌙, când colegii dorm 😴, dar logica digitală nu iartă 🚫🔧.

Se adresează cu drag celor ce urmează:

- Electronică Aplicată 📖 – că tot ce merge, merge prin ea ⚡
- Telecomunicații 🌐 – unde semnalele călătoresc mai repede decât bursa 📈
- Calculatoare 🖥️ – unde clickul e lege și bug-ul e blestem 🐛
- Automatică 🤖 – unde totul se mișcă singur, dacă ai scris codul bine ✅
- Tehnologia Informației 📁 – unde știi tot, dar nu găsești nimic în folderul „final_final_vera.docx” 🤔

📖 Întregul demers e împărțit în patru tomuri, fiecare cu rostul și răbdarea lui:

1. Tomul pentru zăbovit și gândit 🧠 – cu teorii, explicații și Verilog cât să-ți prindă bine la examen, dar și la viață.
2. Tomul pentru chibzuit și buchisit 📖 – unde se învârtesc probleme și idei de proiect, cu calcule mai grele decât un server pe genunchi 🖥️.
3. Tomul pentru meșterit și muncit 🔧 – adică laboratorul, unde mâinile vorbesc în semnale 📡, și uneori scurtcircuitază ⚡.
4. Tomul pentru grabă mare 🏃 – chestionare, recapitulări și bătăi ușoare la poarta memoriei 🧠.

📖 Așadar, deschide cartea, caută semnalul bun 📶⚡ și nu te teme dacă la început nu merge tot cum vrei.

👉 Căci, așa cum bine știm cu toții... viața începe după reset. 🔄💡

Dan NICULA este profesor la Universitatea Transilvania din Brașov.
www.dannicula.ro



Editura
Universității
Transilvania
din Brașov

ISBN 978-606-19-1794-5

