

ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ ԿՐԹՈՒԹՅԱՆ ԵՎ ԳԻՏՈՒԹՅԱՆ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ  
ՀԱՅԱՍՏԱՆԻ ՊԵՏԱԿԱՆ ԾԱՐՏԱՐԱԳԻՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ (ՊՈԼԻՏԵԽՆԻԿ)

*Կիրենդնեստիկայի ֆակուլտետ  
Էլեկտրոնային տեխնիկայի ամբիոն*

Վ.Մ. Մովսիսյան

## ԹՎԱՅԻՆ ՀԱՄԱԿԱՐԳԵՐԻ ՏՐԱՄԱԲԱՆԱԿԱՆ ՆԱԽԱԳԾՈՒՄ

Էլեկտրոնային ճարտարագիտություն մասնագիտության ուսանողների համար

Դասագիրք

ԵՐԵՎԱՆ  
ԾԱՐՏԱՐԱԳԵՏ  
2012

ՀՏԴ- 004.38 (075.8)  
ԳՄԴ- 32.973 ց73  
Մ 917

*Հրատարակվում է Հայաստանի պետական  
ճարտարագիտական համալսարանի  
30.01.2010թ. գիտական խորհրդի նիստում  
հաստատված 2010թ. հրատարակչական պլանի  
համաձայն*

Գրախոսներ՝

**Վազգեն Շավարշի Մելիքյան,**

Հայաստանի պետական ճարտարագիտական համալսարանի միկրոէլեկտրոնային շղթաներ և համակարգեր միջֆակուլտետային ամբիոնի վարիչ, Սինոփսիս Արմենիա ընկերության ուսումնական դեպատամենտի տնօրեն, Հայաստանի հանրապետության գիտության վաստակավոր գործիչ, տ.գ.դ., պրոֆեսոր

**Արթուր Յովհաննեսի Յովհաննիսյան,**

Երևանի Պետական Համալսարանի Ռադիոֆիզիկայի ֆակուլտետի ամբիոնի վարիչ, ֆ.մ.գ.դ., պրոֆեսոր

**Դավիթ Գեղամի Ասատրյան,**

Ռուս-հայկական (սլավոնական) պետական համալսարանի «Կիրառական մաթեմատիկա և ինֆորմատիկա» ֆակուլտետի «Մաթեմատիկական կիբեռնետիկա» ամբիոնի վարիչ, տ.գ.դ., պրոֆեսոր

Մովսիսյան Վ.Մ.

**Մ 917 Թվային համակարգերի տրամաբանական նախագծում:** Դասագիրք/ Վ.Մ. Մովսիսյան;  
ՀՊՃՀ.-Եր.: ճարտարագետ, 2012.- 468 էջ:

Ուսումնասիրվում են համակցական և հաջորդական գործողության թվային համակարգերի տրամաբանական նախագծման եղանակները: Ներկայացվում են թվային համակարգերի ավտոմատացված նախագծման եղանակները ժամանակակից ծրագրային գործիքներով:

Նախատեսված է «Էլեկտրոնային ճարտարագիտություն» մասնագիտության ուսանողների համար:  
Աղ. 125: Նկ. 335: Գրակ. 17:

ՀՏԴ- 004.38 (075.8)  
ԳՄԴ- 32.973 ց73

ISBN 978-9939-55-739-7

© ՃԱՐՏԱՐԱՐԳԵՏ 2012  
© Մովսիսյան Վ.Մ. 2012

## ԲՈՎԱՆԴԱԿՈՒԹՅՈՒՆ

Նախաբան .....	7
Ներածություն .....	9
<b>ԱՌԱՋԻՆ ԲԱԺԻՆ</b>	
<b>Համակցական թվային համակարգեր.....</b>	<b>15</b>
<b>Գլուխ 1. Տրամաբանական ֆունկցիաներ և թվային շղթաներ .....</b>	<b>15</b>
1.1. Տրամաբանական ֆունկցիաներ .....	15
1.1.1. Մեկ փոփոխականի ֆունկցիաներ .....	16
1.1.2. Երկու փոփոխականների տրամաբանական ֆունկցիաներ .....	18
1.1.3. Տրամաբանական ֆունկցիաների հատկությունները .....	21
1.1.4. Տրամաբանական արտահայտությունների ձևափոխություններ .....	26
1.2. Տրամաբանական ֆունկցիաների կատարյալ դիզյունկտիվ և կոնյունկտիվ նորմալ ձևեր .....	26
1.3. Տրամաբանական ֆունկցիաների լրիվ համակարգեր.....	30
1.4. Տրամաբանական ֆունկցիաների բանաձևերի նվազարկում .....	32
1.4.1. Տրամաբանական ֆունկցիաների բանաձևերի նվազարկման Քվայն-Մաք-Կլասկիի եղանակը .....	34
1.4.2. Տրամաբանական ֆունկցիաների բանաձևերի նվազարկման Կառնոյի քարտերի եղանակը .....	39
1.5. Խնդիրներ .....	46
<b>Գլուխ 2. Համակցական տրամաբանական շղթաներ .....</b>	<b>50</b>
2.1. Համակցական շղթաներ .....	50
2.2. Համակցական տրամաբանական շղթաների սինթեզ .....	51
2.3. Բազմաելք համակցական շղթաների նախագծում .....	59
2.4. Տրամաբանական շղթաների վերլուծություն .....	61
2.5. Հապաղումները և մրցումները համակցական շղթաներում .....	64
2.6. Համակցական թվային ֆունկցիաներ .....	70
2.6.1. Վերծանիչ (դեկոդեր) .....	71
2.6.2. Դեմուլտիպլեքսոր .....	75
2.6.3. Համակցական շղթաների իրագործումը դեմուլտիպլեքսորների և վերծանիչների միջոցով .....	77
2.6.4. Կոդավորող սարք (կոդեր) .....	79
2.6.5. Մուլտիպլեքսոր .....	82
2.6.6. Եռավիճակ ելքով բուֆեր և կոլեկտիվ օգտագործման կապի գիծ .....	87
2.6.7. Համակցական շղթաների նախագծում մուլտիպլեքսորների միջոցով.....	88
2.7. Համակցական սարքավորումների կառուցումը ծրագրավորվող ինտեգրալ սխեմաների վրա .....	96
2.7.1. Ընդհանուր տեղեկություններ ծրագրավորվող մեծ ինտեգրալ սխեմաների վերաբերյալ .....	96
2.7.2. Համակցական սարքավորումների սինթեզ ծրագրավորվող ԻՍ-երի վրա .....	98
2.8. Խնդիրներ .....	103
<b>Գլուխ 3. Տվյալների մշակման տրամաբանական սարքեր .....</b>	<b>108</b>
3.1. Թվերի դիրքային համակարգեր .....	108
3.1.2. Անցում մեկ թվային համակարգից մյուսին .....	110
3.1.3. Ութական և տասնվեցական թվեր .....	112
3.1.4. Երկուական-կոդավորված-տասական կոդ (ԵԿՏ) .....	112
3.1.5. Նշանով թվերի ներկայացումը թվային համակարգերում .....	113
3.1.6. Լրացուցիչ կոդեր .....	114

3.2. Նշանով թվերի գումարում .....	115
3.3. Երկուական գումարիչներ .....	117
3.3.1. Կիսագումարիչ .....	117
3.3.2. Լրիվ գումարիչ .....	118
3.4. Հանող տրամաբանական շղթա (հանիչ) .....	119
3.4.1. Կիսահանիչ .....	119
3.4.2. Լրիվ հանիչ .....	120
3.5. Երկուական զուգահեռ գումարիչ .....	121
3.6. Փոխանցման կանխագուշակամբ գումարիչ .....	123
3.7. Տասական գումարիչ .....	127
3.8. Երկուական բազմաթիվ թվերի բազմապատկիչներ .....	136
3.9. Երկուական գումարիչների կիրառությունը համակցական սխեմաներում .....	138
3.10. Շենային սխեմաների կառուցումը երկուական գումարիչներով .....	141
3.11. Երկուական թվերի համեմատման սարքեր՝ թվային կոմպարատորներ .....	144
3.12. Գրեյի, ջերմաչափի և “ո-ից մեկ” կոդեր .....	145
3.13. Աղմկապաշտպանված կոդավորում .....	153
3.14. Խնդիրներ .....	158

## **ԵՐԿՐՈՐԴ ԲԱԺԻՆ**

<b>Հաջորդական թվային համակարգեր .....</b>	<b>161</b>
---	------------

<b>Գլուխ 4. Հաջորդական տրամաբանական շղթաներ .....</b>	<b>161</b>
4.1. Վերջավոր ավտոմատ .....	161
4.2. Անցում Մուրի մոդելից Միլի մոդելի .....	170
4.3. Անցում Միլի մոդելից Մուրի մոդելի .....	171
4.4. Վիճակների նվազարկում .....	174
4.5. Ավտոմատի վիճակների նվազարկման իմպլիկացիաների քարտերի մեթոդը .....	178
4.6. Կառուցվածքային ավտոմատ. վիճակների, մուտքերի և ելքերի կոդավորում .....	185
4.7. Հաջորդական տրամաբանական շղթաներ .....	188
4.8. Տրիգերների հիմնական տիպերը .....	190
4.9. Տակտավորվող տրիգերներով թվային համակարգի աշխատանքի կայունությունը .....	193
4.10. Տակտավորվող հաջորդական շղթաների անխափան աշխատանքի պայմանները ....	197
4.11. Թվային հաջորդական շղթաների վերլուծություն .....	201
4.12. Տրիգերների բնութագրիչ աղյուսակներ .....	204
4.13. Հաջորդական թվային համակարգի սկզբնական վիճակի կարգման ասինքրոն մուտքեր .....	205
4.14. Թվային ավտոմատների (հաջորդական թվային շղթաների) նախագծում .....	206
4.15. Ավելցուկային վիճակներով հաջորդական շղթաների սինթեզ .....	212
4.16. Վիճակների նշանակում .....	214
4.17. Խնդիրներ .....	222

<b>Գլուխ 5. Տիպային հաջորդական թվային հանգույցներ .....</b>	<b>228</b>
5.1. Իմպուլսների թվային հաշվիչներ .....	228
5.2. Երկուական հաշվիչներ .....	230
5.3. Հաշվիչների սխեմաների ընդարձակում .....	233
5.4. Ոչ երկուական հաշվիչներ .....	235
5.5. Երկուական-տասական հաշվիչներ (M=10) .....	238
5.6. Դարձափոխելի հաշվիչներ .....	240
5.7. Հաճախականության թվային սինթեզատորներ .....	242
5.8. Ժամանակահատվածային սխեմաներ .....	249
5.9. Զուգահեռ ռեգիստրներ .....	251
5.10. Ընթերցվող-գրանցվող հիշասարքեր (ԸԳՀ) .....	253

5.11. Ռեգիստրների ֆայլ .....	255
5.11.1 Տեղաշարժող ռեգիստրներ .....	256
5.11.2. Տվյալների հաջորդական փոխանցում թվային համակարգերում .....	259
5.11.3. Համապիտանի ռեգիստրներ .....	261
5.12. Օղակաձև հաշվիչներ .....	264
5.13. Գծային հետադարձ կապերով տեղաշարժող ռեգիստր (ԳՀԿՏՌ) .....	267
5.13.1. Գծային հետադարձ կապերով տեղաշարժող ռեգիստրների (ԳՀԿՏՌ) կառուցվածքները .....	268
5.13.2. Առավելագույն երկարությամբ չկրկնվող հաջորդականություններ .....	274
5.14. Գծային կապերով տեղաշարժող ռեգիստրների կիրառություններ .....	274
5.14.1. Տվյալների հաջորդական փոխանցման ազդանշանային կոդեր .....	274
5.14.2. Թվային համակարգերի թեստավորման համակարգեր .....	279
5.14.3. Աղմկապաշտպանված ցիկլիկ կոդերի գեներացում և վերծանում .....	280
5.15. Խնդիրներ .....	283
<b>Գլուխ 6. Միկրոծրագրային ավտոմատներ .....</b>	<b>285</b>
6.1. Միկրոծրագրային կառավարման սկզբունքը .....	285
6.2. Ալգորիթմի գրաֆ-սխեմա .....	288
6.3. Միկրոծրագրային ավտոմատի սինթեզ .....	303
6.4. Միկրոծրագրային ավտոմատի իրականացում .....	308
6.5. Միկրոծրագրային ավտոմատների կառուցումը ծրագրավորվող միկրոսխեմաների վրա .....	313
6.6. Խնդիրներ .....	317
<b>ԵՐՐՈՐԴ ԲԱԺԻՆ</b>	
<b>Թվային համակարգերի ավտոմատացված նախագծում .....</b>	<b>318</b>
<b>Գլուխ 7. Թվային համակարգերի նկարագրություն Վերիլոգ լեզվով .....</b>	<b>318</b>
7.1. Ներածություն .....	318
7.2. Ավտոմատացված թվային նախագծման ընթացքը .....	319
7.2.1. Հիերարխիկ նախագծման և մոդելավորման սկզբունքները .....	321
7.2.2. Թվային նախագծման մեթոդաբանությունը .....	324
7.2.3. Վերիլոգ մոդուլներ .....	327
7.2.4. Նախագծի նմանակում և ստուգում .....	328
7.2.5. Վերիլոգ նախագծի օրինակ .....	332
7.2.6. Վարժություններ և խնդիրներ .....	332
7.3. Վերիլոգ լեզվի սինվոլները և տվյալների տիպերը .....	332
7.3.1. Վերիլոգ լեզվի բաղադրիչները .....	332
7.3.2. Տվյալների տիպերը .....	334
7.3.2.1 Ցանցեր .....	335
7.3.2.2. Ռեգիստրներ .....	336
7.3.2.3. Չանգվածներ .....	338
7.3.2.4. Պարամետրեր .....	339
7.3.2.5. Սինվոլների շղթաներ .....	341
7.3.3. Համակարգային առաջադրանքներ և կոմպիլատորի դիրեկտիվներ .....	341
7.4. Մոդուլներ և մատույցներ .....	345
7.4.1. Վերիլոգ մոդուլներ .....	345
7.4.2. Մոդուլների մատույցներ .....	347
7.4.3. Հիերարխիկ անվանումներ .....	350
7.4.4. Վարժություններ և խնդիրներ .....	351
7.5. Տրամաբանական տարրերի մակարդակով մոդելավորում .....	353
7.5.1. Վերիլոգ ներդրված պրիմիտիվներ .....	353

7.5.2. Տրամաբանական տարրերի մակարդակով մոդելավորման օրինակներ .....	356
7.5.3. Տարրերի հապաղումները .....	360
7.5.4. Հապաղումներով նմանակման օրինակ .....	362
7.5.6. Խնդիրներ և վարժություններ .....	363
7.6. Տվյալների հոսքի մոդելավորում .....	363
7.6.1. Շարունակական վերագրում .....	363
7.6.2. Արտահայտություններ, օպերատորներ և օպերանդներ .....	366
7.6.3. Թվաբանական, տրամաբանական, համեմատության և հավասարության օպերատորներ .....	368
7.6.4. Բիթառօրթ զործողությունների, սեղմման, տեղաշարժի, միակցման և պայմանական օպերատորներ .....	369
7.6.5. Տվյալների հոսքերի մակարդակով մոդելավորման օրինակներ .....	372
7.6.6. Խնդիրներ .....	375
7.7. Վարքագծային մոդելավորում .....	375
7.7.1. Կառուցվածքային ընթացակարգեր (պրոցեդուրներ) .....	375
7.7.2. Ընթացակարգային վերագրումներ .....	377
7.7.3. Նմանակման ժամանակի կառավարում .....	381
7.7.3.1. Հապաղումների վրա հիմնված ժամանակի կառավարում .....	381
7.7.3.2. Պատահարների վրա հիմնված ժամանակի կառավարում .....	382
7.7.4. Ծրագրի ճյուղավորումների հրամաններ .....	385
7.7.5. Ծրագրի օղակներ .....	389
7.7.6. Հաջորդական և զուգահեռ բլոկներ .....	393
7.7.7. Անվանումով բլոկներ .....	394
7.7.8. Պարամետրացված մոդելների գեներացում՝ generate կառուցվածք .....	394
7.7.9. Վարքագծային մակարդակով մոդելավորման օրինակներ .....	400
7.7.10. Վերջավոր ավտոմատների Վերիլոգ կոդավորում .....	405
7.7.11. Վարժություններ և խնդիրներ .....	413
7.8. Ֆունկցիաներ և առաջադրանքներ .....	414
7.8.1. Վերլոգ առաջադրանքներ .....	414
7.8.2. Վերլոգ ֆունկցիաներ .....	418
7.8.3. Վարժություններ և խնդիրներ .....	423
<b>Գլուխ 8. Թվային համակարգերի ավտոմատացված սինթեզ .....</b>	<b>424</b>
8.1. Սինթեզի ընթացակարգը .....	424
8.2. ՌՓՄ նկարագրության թարգմանությունը կառուցվածքային նկարագրության .....	428
8.3. Սինթեզվող Վերիլոգ կոդին ներկայացվող սահմանափակումները .....	432
8.4. ՌՓՄ-ից տրամաբանական տարրերի կապերի ցուցակի անցման օրինակ .....	436
8.5. Թվային համակարգերի անսխալ սինթրոնացման պայմանների ստուգում .....	441
8.5.1. Թվային համակարգի ժամանակային սահմանափակումները իդեալական տակտային իմպուլսների դեպքում .....	441
8.5.2. Ազդանշանների կոնվերտացված մշակում .....	445
8.6. Խնդիրներ .....	452
<b>Խնդիրների լուծումներ և պատասխաններ .....</b>	<b>454</b>
<b>Օգտագործված գրականության ցուցակ .....</b>	<b>467</b>

## Նախաբան

Ներկայացվող դասագիրքն ուղղված է ժամանակակից միկրոէլեկտրոնային տեխնոլոգիաների վրա հիմնված թվային համակարգերի նախագծման մեջ մասնագիտացող ուսանողներին և ճարտարագետներին: Գրքի հիմնական նպատակն է սովորեցնել մտածել իբրև թվային էլեկտրոնային համակարգերի նախագծող, տալ սկզբունքներ և մոդելներ, որոնցով կարելի է վերլուծել գոյություն ունեցող համակարգերը և ստեղծել նորերը:

Դասագիրքը բաղկացած է 8 գլխից:

Առաջին գլուխն ընդգրկում է ներածություն տրամաբանական հանրաշխի և տրամաբանական տարրերի վերաբերյալ, որոնք օգտագործվում են թվային համակարգերի վարքագծային նկարագրության և սինթեզի համար: Ներկայացվում են տրամաբանական շղթաների աղյուսակային նկարագրությունից տրամաբանական արտահայտությունների անցնելու մաթեմատիկական եղանակները: Քննարկվում են տրամաբանական արտահայտությունների նվազարկման եղանակները, որոնք հնարավորություն են տալիս կառուցել պարզ տրամաբանական շղթաներ: Նվազարկման եղանակները պարզաբանվում են բազմաթիվ օրինակներով:

Համակցական սարքավորումների նախագծման ընթացակարգերը և կառուցման եղանակները ներկայացված են գլուխ 2-ում: Տրամաբանական փականների միջոցով համակցական թվային շղթաների սինթեզի մանրամասները բացահայտվում են նախագծման օրինակներով: Քննարկվում են համակցական թվային ֆունկցիաների՝ կոդավորիչների, վերծանիչների, մուլտիպլեքսորների, դեմուլտիպլեքսորների միջոցով համակցական շղթաների կառուցման եղանակները: Ներկայացված են նաև ծրագրավորվող ԻՍ-երի միջոցով համակցական շղթաների իրականացման եղանակները:

Երրորդ գլխում ուսումնասիրվում են տվյալների մշակման թվային շղթաները: Ներկայացված են նաև հաշվարկման երկուական թվային համակարգերը, որոնցով թվային համակարգերում ներկայացվում է մշակվող տեղեկատվությունը: Քննարկվում են տվյալների մշակման հանգույցների կառուցման եղանակները:

Չորրորդ գլխում ներկայացված են հաջորդական գործողության թվային շղթաների նախագծման սկզբունքները: Դիտարկվում է հաջորդական շղթաների վերացական նկարագրությունը վերջավոր ավտոմատների մոդելներով: Վարքագծային նկարագրության օրինակներում հիմնականում օգտագործվում են անցումների աղյուսակները և գրաֆները: Ներկայացված են վերջավոր ավտոմատի օպտիմալացման խնդիրները՝ վիճակների նվազարկումը և վիճակների կոդավորումը: Քննարկվում են տրիգերների միջոցով տակտավորվող հաջորդական շղթաների իրականացման խնդիրները: Ներկայացված է վերջավոր ավտոմատների կանոնական սինթեզի ընթացակարգը՝ հիմնված տրիգերների մուտքային ֆունկցիաների աղյուսակային որոշման վրա: Սինթեզի ընթացակարգը մանրամասնված է նախագծման օրինակներով:

Հինգերորդ գլխում ներկայացված են տիպային հաջորդական թվային ֆունկցիաները՝ հաշվիչներ, զուգահեռ և տեղաշարժող ռեգիստրներ: Քննարկված են դրանց կի-

րառությունները՝ թվային համակարգերի իրականացման համար

Վեցերորդ գլխում ներկայացված են միկրոծրագրային ավտոմատները՝ ալգորիթմային վերջավոր ավտոմատները: Ներկայացված են ալգորիթմի գրաֆ-սխեմայի օգնությամբ միկրոծրագրի նկարագրության եղանակը և ալգորիթմի գրաֆ-սխեմայի միջոցով միկրոծրագրային ավտոմատի սինթեզի ընթացակարգը: Բերված են նախագծման մի շարք օրինակներ:

Յոթերորդ գլխում ներկայացված է թվային սխեմաների նկարագրության Վերիլոգ (Verilog) լեզուն, որն օգտագործվում է թվային համակարգերի վարքագծային նկարագրության ու նմանակման, ինչպես նաև ավտոմատացված նախագծման համակարգերում նախագծվող սարքի տրամաբանության ներածման համար: Ներկայացված են հիերարխիկ մոդելավորման սկզբունքները և Վերիլոգ լեզվի հիմնական կառուցվածքները՝ սխեմաների նկարագրության և նմանակման համար:

Ութերորդ գլխում քննարկվում են թվային համակարգերի ավտոմատացված սինթեզի ծրագրային գործիքները և սկզբունքները, երբ նախագծման համար մուտք է հանդիսանում ռեգիստրային փոխանցումների մակարդակի (ՌՓՄ) Վերիլոգ կոդը: Ներկայացված են ավտոմատացված նախագծման գործընթացը, ինչպես նաև թվային համակարգի նախագծի հիերարխիկ կառուցվածքը և նախագծման արդյունքների ստուգման եղանակները:



## Ներածություն

Կառավարման և տեղեկատվական համակարգերում ազդանշանների մշակումը կարելի է իրագործել անալոգային և թվային էլեկտրոնային շղթաներով: Միկրոէլեկտրոնային տեխնոլոգիաների ժամանակակից ձեռքբերումները հնարավորություն են ընձեռում մեկ ինտեգրալ սխեմայում (ԻՍ) միավորել հարյուր միլիոնավոր տրանզիստորներ, որոնք կազմում են թվային համակարգեր: Թվային տեխնիկայի լայն տարածումը պայմանավորված է թվային շղթաների փոքր չափերով, ցածր էներգատարողությամբ, բարձր արագագործությամբ, ցածր ինքնարժեքով, բարձր հուսալիությամբ և բազմապիսի ֆունկցիոնալ հնարավորություններով:

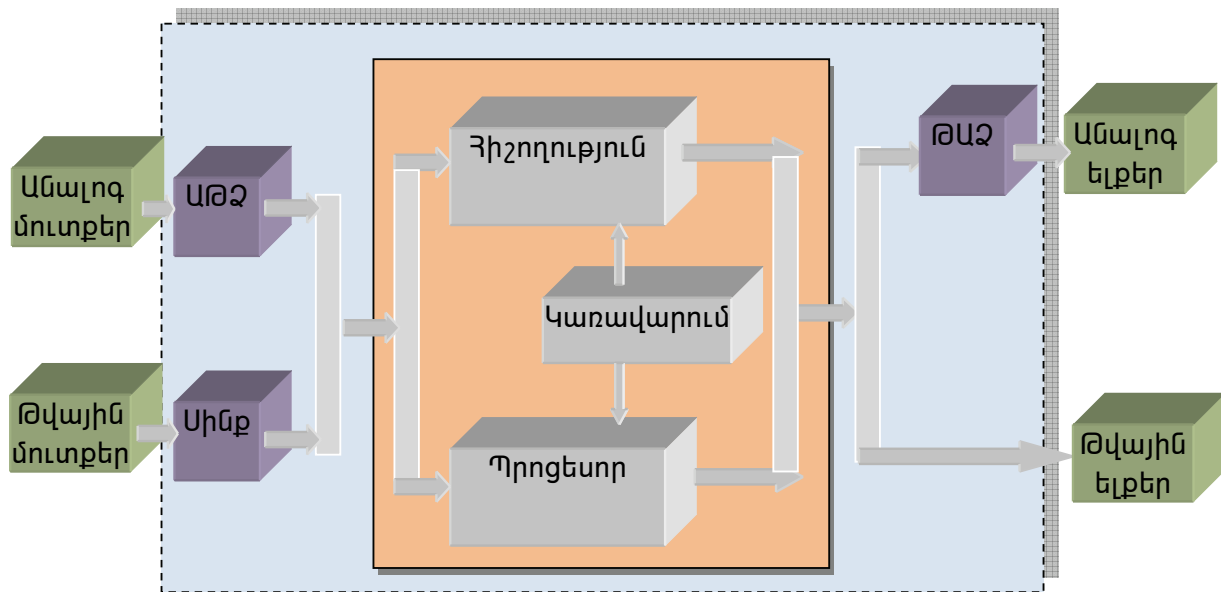
Թվային շղթաները կառուցվում են տրանզիստորների միջոցով, որոնք օգտագործվում են որպես երկու վիճակով էլեկտրոնային բանալիներ: Դրա շնորհիվ թվային համակարգերում տեղեկատվությունը երկուսական է: Տեղեկատվական տեխնոլոգիաներում թվային և երկուսական տերմինները համարժեք են: Տեղեկատվությունը ներկայացվում է երկուսական թվերի միջոցով, քանի որ ինչպես թվային շղթայի վիճակը, այնպես էլ երկուսական միջերն ունեն միայն երկու վիճակ, որոնք հաճախ նշվում են 0 և 1 սիմվոլներով: Ազդանշանների միայն երկու վիճակի օգտագործումը պարզեցնում է շղթաները և բարձրացնում տեղեկատվության աղմկակայունությունը:

Շատ համակարգեր, ինչպես օրինակ MP3 նվագարկիչը, գործ ունեն անալոգային ազդանշանների հետ, բայց տվյալները պահպանվում և մշակվում են թվային տեսքով: Նման համակարգերի անբաժան մաս է կազմում անալոգ-թիվ ձևափոխիչը (ԱԹՁ), որը անալոգային ազդանշանը ձևափոխում է համարժեք թվայինի՝ հետագա մշակման և պահպանման համար: Երբ այդ թվային տեղեկատվությունը պետք է վերարտադրել, այն թիվ-անալոգ ձևափոխիչի (ԹԱՁ) միջոցով ձևափոխվում է անալոգայինի:

Տվյալների մշակման համակարգի ընդհանուր տեսքը ցույց է տրված նկ. 1-ում: Տվյալների թվային մշակման հանգույցը բաղկացած է հիշողությունից, տվյալների հետ թվաբանական և տրամաբանական գործողություններ իրականացնող թվային պրոցեսորից և կառավարման հանգույցից: Մուտքային տվյալները կարող են լինել անալոգային, որոնք գալիս են անալոգային ազդանշանների առաջնային կերպափոխիչներից, ինչպես նաև թվային, որոնք գալիս են երկդիրք փոխանջատիչներից, վերջույթային սարքերից կամ տվյալների դոդերից: Արտաքինից տրվող թվային ազդանշանները պետք է սինքրոնացվեն համակարգի հետ: Անալոգային տվյալները, նախքան թվային մշակման ենթարկվելը, թվայնացվում են (ԱԹՁ-ի միջոցով: Համակարգի ելքերը նույնպես կարող են լինել անալոգային և թվային: Անալոգային ելքերը ձևավորվում են թվային տվյալներից՝ ԹԱՁ ձևափոխիչի միջոցով: Ընդ որում, համակարգի բոլոր հանգույցները հնարավոր է տեղադրել մեկ ԻՍ-ում:

Թվային համակարգերը իրականացնում են տեղեկատվության մշակման գործողություններ, որոնք նկարագրվում են տրամաբանական ֆունկցիաներով: Թվային համակարգերի նախագծման սկզբունքները հիմնված են թվային շղթաների տրամաբանական սինթեզի և նվազարկման մեթոդների վրա: Նախագծման հիմնական չափանիշներն են՝ արագագործությունը, որը գնահատվում է տակտավորման առավելագույն

հաճախականությամբ, օգտագործվող տարրերի թիվը կամ զբաղեցրած մակերեսը և ծախսվող հզորությունը:



Նկ. 1. Տվյալների մշակման միկրոէլեկտրոնային համակարգ

Ոչ բարդ թվային համակարգերի նախագծումը կարելի է իրականացնել ձեռքի հաշվարկներով՝ հետևելով սինթեզի ալգորիթմներին: Այդ համակարգերի իրականացման համար օգտագործվում են ցածր ինտեգրացման աստիճանի ինտեգրալ սխեմաներ, որոնք պարունակում են տրամաբանական տարրեր (ԵՎ, ԿԱՄ, ԵՎ-ՈՉ, ԿԱՄ-ՈՉ և այլն) և տրիգերներ: Ինտեգրալ սխեմաների տեխնոլոգիաների զարգացման շնորհիվ միջին ինտեգրացման աստիճանի ինտեգրալ սխեմաները իրականացնում են ավարտուն թվային ֆունկցիաներ ինչպիսիք են՝ ռեգիստրները, հաշվիչները, մուլտիպլեյսորները և այլն: Այդ ֆունկցիաները թույլ են տալիս զգալի կրճատել օգտագործվող ԻՍ-երի թիվը և պարզեցնել տպասալի վրա ազդանշանների ծրուծը: Միաժամանակ դրանք թույլ են տալիս ընդարձակել կառուցվող համակարգերի ֆունկցիոնալ հնարավորությունները:

Ժամանակակից թվային համակարգերը շատ բարդ են, որոնք ընդգրկում են մեծ թվով ենթահամակարգեր և բլոկներ: Այդպիսի համակարգերն իրագործվում են շատ բարձր ինտեգրացման աստիճանի (ՇԲԻԱ) ԻՍ-երի միջոցով: ՇԲԻԱ ԻՍ-ը պարունակում է հարյուր միլիոնավոր, նույնիսկ միլիարդավոր, տրանզիստորներ, և այդ թիվն աճում է Մուրի օրենքով: Տրամաբանական նախագծման մակարդակով համակարգի կառուցման առաջնային բլոկները տրամաբանական տարրերն են, որոնք ընդգրկում են մի քանի տրանզիստորներ:

ԻՍ-ում բաղադրիչների թվի և իրագործվող ալգորիթմների բարդության աստիճանի աճը հարկադրում է մշակել նախագծման նոր մեթոդներ և մոտեցումներ: Հիմնականում դա համակարգիչների վրա հիմնված ավտոմատացված նախագծում է, որն օգտագործում է էլեկտրոնային նախագծման ավտոմատացված (ԷՆԱ) ծրագրային

գործիքներ: Թվային նախագծի նկարագրության համար օգտագործվում են սխեմաների նկարագրության լեզուներ (ՄՆԼ), ինչպիսիք են Վերիլոգը (Verilog) և VHDL-ը: Դրանք բարձր մակարդակի ծրագրավորման լեզուներ են, որոնք հնարավորություն են տալիս նախագծողին նկարագրել նախագծվող համակարգը և նմանակել նրա աշխատանքը, ստուգել ալգորիթմների ճշտությունը և արդյունավետությունը, համոզվել որ թերություններ և սխալներ չկան, նախքան բուն նախագծմանն անցնելը:

Կոնկրետ կիրառություններում նախագծման որևէ մեթոդաբանության ընտրումը կախված է այն ԻՍ-ի տրամաբանական կառուցվածքից, որի միջոցով պետք է իրականացվի նպատակային թվային համակարգը:

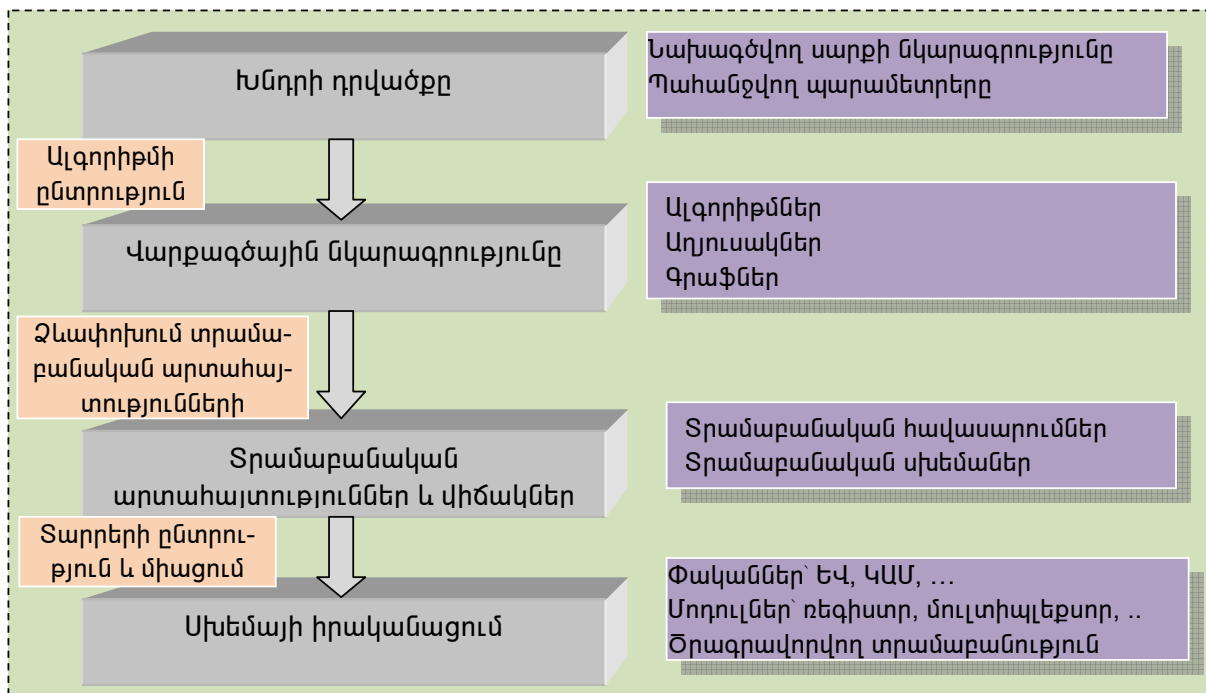
Որևէ անհատական կիրառության համար նախագծված և կառուցված ԻՍ-ը կոչվում է կիրառությանը յուրահատուկ ԻՍ՝ ԿՅԻՍ (Application Specific IC, ASIC): ԿՅԻՍ-ները ունեն առավելագույն արագագործություն, նվազագույն մակերես և ծախսվող հզորություն: Միաժամանակ դրանք պահանջում են նախագծման և արտադրության երկար ժամանակ և ունեն համեմատաբար բարձր ինքնարժեք: Այդ ԻՍ-երը հիմնականում կառուցվում են նախապես նախագծված թվային տարրերի բազմության միջոցով: Տարրերի այդ բազմությունը կոչվում է ստանդարտ թվային գրադարան:

Ծրագրավորվող ԻՍ-երի կիրառությամբ նախագծման դեպքում տվյալ կիրառության համար ուղղակի ծրագրավորվում են արդեն պատրաստի ԻՍ-ի տարրերի միջև կապերը, որպեսզի ստացվի պահանջվող ֆունկցիոնալությունը: Ծրագրավորվող թվային ԻՍ-երը կոչվում են ծրագրավորվող տրամաբանական սարքեր՝ ԾՏՍ (Programmable logic device, PLD): ԾՏՍ-երն ունեն ստանդարտ կառուցվածք և արտադրվում են մեծ քանակով: Նախագծման այս եղանակի հիմնական առավելությունը նախագծման և ֆիզիկական իրականացման կարճ ժամանակն է: Ծրագրավորվող ԻՍ-երից պարզագույնը միայն ընթերցվող հիշողությունն է (ՄԸՀ): ՄԸՀ-երի ծրագրավորման տեխնոլոգիան կարելի է կիրառել նաև ավելի ճկուն տրամաբանական կառուցվածքների՝ ծրագրավորվող տրամաբանական տարրերի զանգվածների՝ ԾՏՏԶ (Programmable Logic Array, PLA) համար: ԾՏՏԶ ԻՍ-երը կարող են պարունակել նաև հիշողության ռեգիստրներ, ինչը հնարավորություն է տալիս մեկ ԻՍ-ի միջոցով ամբողջությամբ իրականացնել նպատակային թվային համակարգը:

ԻՍ-ի բարդության և կիրառության համար ընձեռած հնարավորություններով ԾՏՏԶ-ից մեկ քայլ բարձր է կիրառությունում ծրագրավորվող փականների զանգվածը՝ ԿԾՓԶ (Field-Programmable Gate Array, FPGA): Սկզբունքորեն քիչ տարբերություն կա ԿԾՓԶ-ի և ԾՏՏԶ-ի միջև՝ սովորաբար ԿԾՓԶ-ն ավելի մեծ է և ավելի բարդ: ԿԾՓԶ-ն ավելի նոր տիպի ԻՍ է, նրա միջուկը բաղկացած է ծրագրավորվող տրամաբանական բլոկներից, որոնց միջև կապերը նույնպես ծրագրավորվող են. այն կարող է պարունակել նաև մեկ կամ ավելի հզոր միկրոպրոցեսորներ: ԿԾՓԶ-ով բավականին բարդ թվային համակարգի իրագործումը կարող է պահանջել ընդամենը մի քանի ժամ: ԿԾՓԶ ԻՍ-երն ունեն բարձր ինքնարժեք և հիմնականում օգտագործվում են բարդ համակարգերի նախատիպերի մշակման և թեստավորման համար:

Ոչ բարդ թվային համակարգի նախագծումն ու կառուցումը կարելի իրականաց-

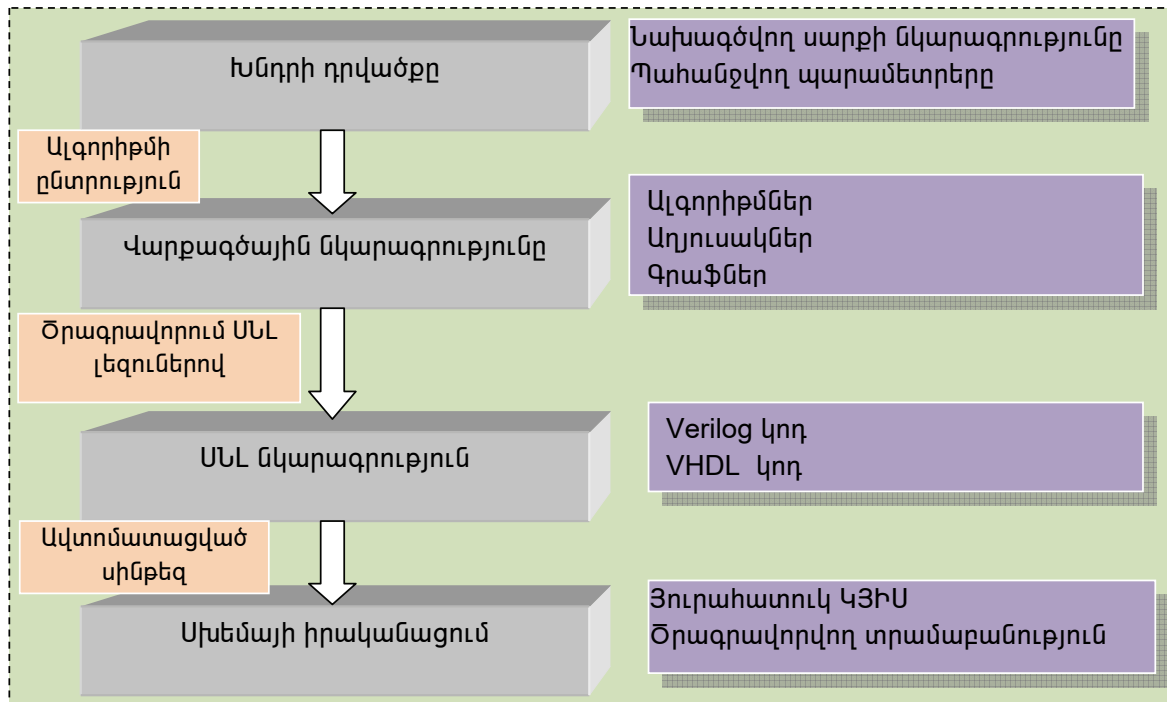
Նել Նկ. 2-ում ցույց տրված եղանակով:



Նկ. 2. Թվային համակարգի տրամաբանական նախագծման հաջորդականությունը

Նախագծման առաջին փուլում պետք է հստակ շարադրել նախագծման խնդիրը՝ կոնկրետացնելով այն պահանջվող հատկություններով և քանակական պարամետրերով: Երկրորդ փուլում պետք է խնդրի դրվածքից անցնել նախագծվող սարքի աշխատանքի վարքագծային նկարագրության ճարտարագիտական եղանակներին՝ օգտագործելով ալգորիթմներ, աղյուսակներ, գրաֆներ, ժամանակային դիագրամներ: Հաջորդ փուլում համակարգի աշխատանքը պետք է ներկայացնել տրամաբանական արտահայտություններով: Տրամաբանական նախագծման վերջին փուլում անհրաժեշտ է տրամաբանական արտահայտություններից անցնել տրամաբանական սխեմաների՝ ընտրելով համապատասխան տարրեր (տրամաբանական փականներ, տրիգերներ, ռեգիստրներ և այլն) և իրականացնելով դրանց միջմիացումները, կամ ընտրել համապատասխան ծրագրավորվող ԻՍ և տալ ԻՍ-ի ծրագրավորման տվյալները:

Բարդ թվային համակարգերի ձեռքով նախագծումը չափազանց բարդ է, եթե ոչ անհնարին: Նախագծումն իրականացվում է ԷՆԱ ծրագրային գործիքներով: Ավտոմատացված տրամաբանական նախագծման հաջորդականությունը ցույց է տրված Նկ. 3-ում: Վարքագծային նկարագրությունից նախագծվող սխեմային անցնելու համար անհրաժեշտ է նախ այդ նկարագրությունը ներկայացնել որևէ ՄՆԼ-ով, այնուհետև կատարել սարքի սինթեզ համապատասխան ծրագրային գործիքով:



Նկ. 3. Թվային համակարգի ավտոմատացված տրամաբանական նախագծման հաջորդականությունը



# ԱՌԱՋԻՆ ԲԱԺԻՆ

## Համակցական թվային համակարգեր

### Գլուխ 1. Տրամաբանական ֆունկցիաներ և թվային շղթաներ

#### 1.1. Տրամաբանական ֆունկցիաներ

Թվային շղթաների ֆունկցիոնալ նկարագրության համար օգտագործվող ամենատարածված մաթեմատիկական ապարատը տրամաբանական կամ բուլյան հանրահաշիվն է:

Տրամաբանական կոչվում է այն փոփոխականը, որը կարող է ընդունել միայն երկու արժեք՝ “այո” և “ոչ”: Թվային համակարգերում տրամաբանական արժեքների համար օգտագործվում են հետևյալ բազմությունները՝ {միացված, անջատված}; {1, 0}; {բարձր(H); ցածր(L)}: Ավելի հաճախ կօգտագործենք տրամաբանական փոփոխականի  $x \in \{0; 1\}$  սահմանումը:

Տրամաբանական փոփոխականների  $y=f(x_1, x_2, \dots, x_n)$  ֆունկցիան, ինչպես նաև դրա փոփոխականները, ընդունում է միայն երկու արժեք՝  $y, x_1, x_2, \dots, x_n \in \{0; 1\}$ :

Տրամաբանական ֆունկցիան կարող է ներկայացվել երեք եղանակով՝ աղյուսակով (այս դեպքում այն կոչվում է իսկության աղյուսակ), գրաֆիկով, բանաձևով:

Թվային շղթաները նկարագրող տրամաբանական ֆունկցիաները, սովորաբար, կախված են մեծ թվով փոփոխականներից: Այդ պատճառով դրանց գրաֆիկական ներկայացումը հարմար չէ:

Տրամաբանական ֆունկցիայի արգումենտների արժեքների համակցությունները կոչվում են հավաքածուներ: Ցանկացած տրամաբանական ֆունկցիա կարող է ներկայացվել իսկության աղյուսակով: Իսկության աղյուսակում փոփոխականների յուրաքանչյուր հավաքածուի համար նշվում է ֆունկցիայի համապատասխան արժեքը: Իսկության աղյուսակի տողերի թիվը հավասար է փոփոխականների բոլոր հնարավոր հավաքածուների թվին:  $n$  փոփոխականների դեպքում հավաքածուների լրիվ թիվը կազմում է  $2^n$ : Ընդունված է հավաքածուները համարակալել ըստ դրանցով կազմված երկուական թվերի արժեքների՝ 0-ից (00...00) մինչև  $2^n-1$  (11...11): Աղյուսակ 1.1-ում ցույց է տրված իսկության աղյուսակի ընդհանուր տեսքը  $n$  փոփոխականների դեպքում: Աղյուսակի յուրաքանչյուր տողի համար գոյություն ունի ֆունկցիայի մեկ արժեք, որը կարող է լինել 0 կամ 1:

Տրամաբանական ֆունկցիան կոչվում է լրիվ որոշված, եթե տրված են նրա արժեքները փոփոխականների բոլոր հավաքածուների համար, օրինակ  $y_1$  ֆունկցիան աղյուսակ 1.1-ում: Եթե ֆունկցիայի արժեքը որոշված չէ փոփոխականների թեկուզ մեկ հավաքածուի համար, ապա ֆունկցիան կոչվում է թերի կամ ոչ լրիվ որոշված: Աղյուսակ 1.1-ում  $y_2$ -ը թերի որոշված ֆունկցիա է՝  $y_2$ -ի արժեքը որոշված չէ  $2^n-2$  (11...10) հավաքածուի համար: Ֆունկցիայի անորոշ արժեքը նշվում է  $x$ -ով, իսկ դրան համապատասխանող հավաքածուն կոչվում է արգելված:

Աղյուսակ 1.1

i	$x_1 x_2 \dots x_{n-1} x_n$	$y_1$	$y_2$
0	0 0 ... 0 0	1	0
1	0 0 ... 0 1	0	1
2	0 0 ... 1 0	0	0
.			
.			
$2^n-2$	1 1 ... 1 0	1	x
$2^n-1$	1 1 ... 1 1	1	1

**Օրինակ 1.1.** Թվային սարքը պետք է գրանցի քվերկությանը ընդունվող վճռի կայացումը, երբ քվեարկում են երեք մասնակից: Վճիռը կայացված է համարվում, եթե երեք քվեարկողից առնվազն երկուսը միաժամանակ սեղմում են կոճակները: Քվեարկողների կոճակների վիճակները A, B, C տրամաբանական փոփոխականները են: Տրամաբանական 1 արժեքը համապատասխանում է սեղմված վիճակին, 0-ն՝ չսեղմված վիճակին: Վճռի կայացումը տրվում է Y տրամաբանական ֆունկցիայով: Վճիռը կայացված է, երբ  $Y=1$ : Այս ֆունկցիայի իսկության աղյուսակը ցույց է տրված աղյուսակ 1.2-ում:

Աղյուսակ 1.2

i	A	B	C	Y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

### 1.1.1. Մեկ փոփոխականի ֆունկցիաներ

n տրամաբանական փոփոխականների դեպքում հավաքածուների լրիվ թիվը  $2^n$  է: Նույն  $x_1, x_2, \dots, x_n$  փոփոխականներից որոշված տարբեր տրամաբանական ֆունկցիաներ տարբերվում են միայն աղյուսակ 1.1-ի աջ սյունակով: Յուրաքանչյուր հավաքածուի վրա ֆունկցիան կարող է ընդունել 0 կամ 1 արժեք, անկախ մյուս հավաքածուների վրա ընդունած արժեքներից: Հետևաբար, միմյանցից տարբերվող n փոփոխականի ֆունկցիաների ընդհանուր թիվը կլինի՝  $N = 2^{2^n}$ .

Երբ  $n=1$ ,  $N=4$ . Մեկ փոփոխականի բոլոր չորս ֆունկցիաները ներկայացված են աղյուսակ 1.3-ում:



Աղյուսակ 1.3

x	y <sub>1</sub>	y <sub>2</sub>	y <sub>3</sub>	y <sub>4</sub>
0	0	1	0	1
1	0	1	1	0

Այդ ֆունկցիաներն ունեն ստորև ցույց տրված անվանումներ, նշանակումները և դրանց համապատասխանող թվային շղթաներով իրագործվող գործողությունները.

$y_1=0$ , հաստատուն 0 (միշտ տրամաբանական ցածր մակարդակ, L),

$y_2=1$ , հաստատուն 1 (միշտ տրամաբանական բարձր մակարդակ, H),

$y_3=x$ , կրկնություն, բուֆերացում,

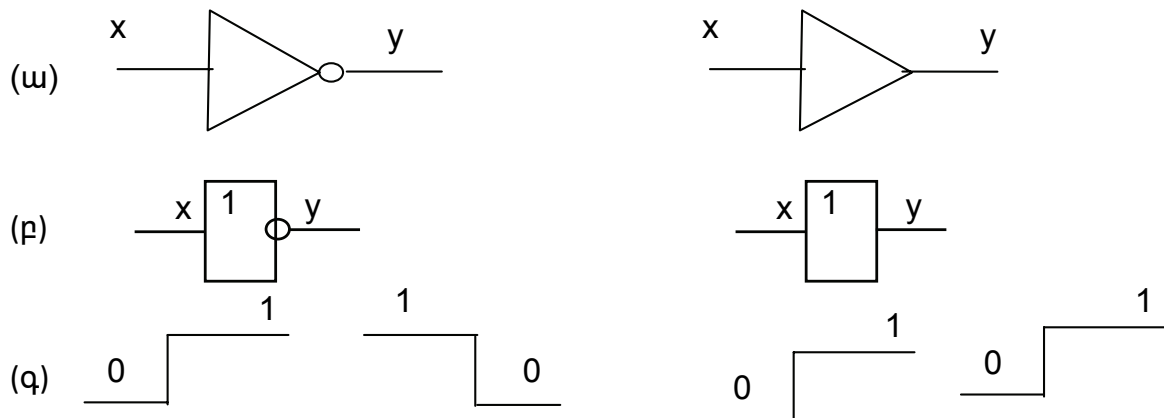
$y_4=\bar{x}$  ինվերսում (ժխտում կամ բացասում), ՈՉ տրամաբանական գործողություն:

«Բացասում» տրամաբանական գործողությունը բանաձևերում նշվում է նաև  $!x$ ,  $\sim x$ , նշաններով:

Թվային շղթաներում լարման մակարդակները՝  $V_H$  (լարման բարձր մակարդակ) և  $V_L$  (լարման ցածր մակարդակ), որոնք իրականացնում են 0 և 1 տրամաբանական մակարդակները, որոշվում են սնման աղբյուրի բևեռների լարումներով: Սնման աղբյուրի դրական բևեռը համապատասխանում է տրամաբանական 1-ին, իսկ բացասական բևեռը՝ տրամաբանական 0-ին:  $y_1$  և  $y_2$  ֆունկցիաներն իրականացնող էլեկտրական շղթաները ցույց են տրված նկ. 1.1-ում: Էլեկտրոնային շղթաները, որոնցով իրականացվում են հիմնական տրամաբանական ֆունկցիաները՝ կախված տրամաբանական մակարդակները ներկայացնող ազդանշաններից, կոչվում են տրամաբանական տարրեր կամ փականներ: Պարզագույն տրամաբանական տարրը բուֆերն է, որն իրականացնում է «կրկնություն» տրամաբանական ֆունկցիան՝  $y_3 = x$ : Շրջիչ կամ ՈՉ տրամաբանական տարրն իրականացնում է բացասում տրամաբանական ֆունկցիան՝  $y_4 = \bar{x}$ . Բուֆերի և շրջիչի գրաֆիկական սիմվոլները ցույց են տրված նկ. 1.2-ում:



Նկ. 1.1. Տրամաբանական մակարդակների համապատասխանող լարման մակարդակներ



Նկ. 1.2. Բուժեր և շրջիչ տրամաբանական տարրեր՝ (ա) գրականությունում հաճախ օգտագործվող և սխեմաների խմբագրման համակարգչային ծրագրերում օգտագործվող գրաֆիկական սիմվոլներ, (բ) եվրոպական էլեկտրատեխնիկական հանձնաժողովի ստանդարտով որոշված գրաֆիկական սիմվոլներ, (գ) մուտքի և ելքի տրամաբանական մակարդակները

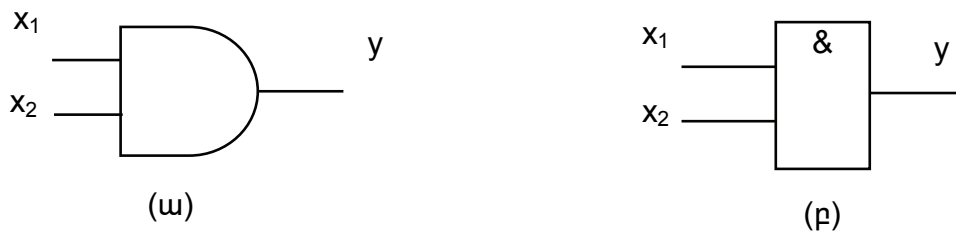
### 1.1.2. Երկու փոփոխականների տրամաբանական ֆունկցիաներ

Երկու փոփոխականների դեպքում՝  $n=2$ , գոյություն ունեն 16 միմյանցից տարբեր ֆունկցիաներ: Թվային տեխնիկայում առավել մեծ կիրառություն ունեցող երկու փոփոխականի տրամաբանական ֆունկցիաները ներկայացված են աղյուսակ 1.4-ում:

Աղյուսակ 1.4

Հավաքածուի թիվը	Փոփոխականներ		Ֆունկցիաներ					
	$x_1$	$x_2$	$y_5$	$y_6$	$y_7$	$y_8$	$y_9$	$y_{10}$
0	0	0	0	0	1	1	0	1
1	0	1	0	1	1	0	1	0
2	1	0	0	1	1	0	1	0
3	1	1	1	1	0	0	0	1

$y_5 = x_1 \& x_2 = x_1 x_2$  ֆունկցիան կոչվում է կոնյունկցիա, տրամաբանական բազմապատկում, ԵՎ (AND,  $\cap$ ) ֆունկցիա: Կոնյունկցիա տրամաբանական գործողությունը բանաձևերում նշվում է  $\&$ ,  $\cdot$ ,  $\wedge$  նշաններով, կամ ընդհանրապես ոչ մի նշան չի դրվում. ուղղակի փոփոխականները գրվում են իրար կողքի, ինչպես օրինակ՝  $x_1 x_2$ : Կոնյունկցիա ֆունկցիան իրականացնող տրամաբանական տարրը կոչվում է ԵՎ (AND,  $\cap$ ) տարր: Նկ. 1.3-ում ցույց են տրված երկու մուտքերով ԵՎ տարրի՝ 2ԵՎ (AND2,  $2\cap$ ) տարրի գրաֆիկական սիմվոլները:



Նկ. 1.3. Երկու մուտքերով ԵՎ տարր (2ԵՎ, AND2, 2И)՝ (ա) գրականությունում հաճախ օգտագործվող և սխեմաների խմբագրման համակարգչային ծրագրերում օգտագործվող գրաֆիկական սիմվոլը, (բ) Եվրոպական էլեկտրատեխնիկական հանձնաժողովի ստանդարտով որոշված գրաֆիկական սիմվոլը

Աղյուսակ 1.4-ից հետևում է, որ  $y_5=1$  միայն այն դեպքում, երբ երկու փոփոխականները միաժամանակ հավասար են 1-ի՝  $x_1=x_2=1$ :

$y_6=x_1+x_2=x_1 \vee x_2=x_1 | x_2$  ֆունկցիան կոչվում է դիզյունկցիա, տրամաբանական գումարում, ԿԱՍ (OR, ИЛИ) ֆունկցիա: Դիզյունկցիա տրամաբանական գործողությունը բանաձևերում նշվում է +, V, | նշաններով: Կոնյունկցիա ֆունկցիան իրականացնող տրամաբանական տարրը կոչվում է ԿԱՍ (OR, ИЛИ) տարր: Նկ. 1.4-ում ցույց են տրված երկու մուտքերով ԵՎ տարրի՝ 2ԿԱՍ (OR2, 2ИЛИ) տարրի գրաֆիկական սիմվոլները:

Աղյուսակ 1.4-ից հետևում է, որ  $y_6=0$  միայն այն դեպքում, երբ երկու փոփոխականները միաժամանակ հավասար են 0-ի՝  $x_1=x_2=0$ :  $y_6=1$ , եթե  $x_1$  կամ  $x_2$ -ը հավասար է 1-ի:



Նկ. 1.4. Երկու մուտքերով ԿԱՍ տարր (2ԿԱՍ, OR2, 2ИЛИ)՝ (ա) գրականությունում հաճախ օգտագործվող և սխեմաների խմբագրման համակարգչային ծրագրերում օգտագործվող գրաֆիկական սիմվոլը, (բ) Եվրոպական էլեկտրատեխնիկական հանձնաժողովի ստանդարտով որոշված գրաֆիկական սիմվոլը

$y_7=x_1/x_2=\overline{x_1 \& x_2}$  կոչվում է Շեֆերի ֆունկցիա կամ ԵՎ-ՈՉ (NAND, И-НЕ) ֆունկցիա: Այս ֆունկցիան իրականացնող տրամաբանական տարրը կոչվում է ԵՎ-ՈՉ (NAND, И-НЕ) տարր: Նկ. 1.5-ում ցույց են տրված երկու մուտքերով ԵՎ-ՈՉ տարրի՝ 2ԵՎ-ՈՉ (NAND2, 2И-НЕ) տարրի գրաֆիկական սիմվոլները:



Նկ. 1.5. Երկու մուտքերով ԵՎ-ՈՉ (2ԵՎ, AND2, 2И) տարրի գրաֆիկական սիմվոլները

Աղյուսակ 1.4-ից հետևում է, որ  $y_7=0$  միայն այն դեպքում, երբ երկու փոփոխականները միաժամանակ հավասար են 1-ի՝  $x_1=x_2=1$ :  $y_7=1$ , եթե  $x_1$  կամ  $x_2$ -ը հավասար է 0-ի:  $y_8 = x_1 \uparrow x_2 = \overline{x_1 + x_2}$  կոչվում է Վեբի ֆունկցիա կամ ԿԱՍ-ՈՉ (NOR, ИЛИ-НЕ) ֆունկցիա: Այս ֆունկցիան իրականացնող տրամաբանական տարրը կոչվում է ԿԱՍ-ՈՉ (NOR, ИЛИ-НЕ) տարր: Նկ. 1.6-ում ցույց են տրված երկու մուտքերով ԿԱՍ -ՈՉ տարրի՝ 2ԿԱՍ-ՈՉ (NOR2, 2ИЛИ-НЕ) տարրի գրաֆիկական սիմվոլները:



Նկ. 1.6. Երկու մուտքերով ԿԱՍ-ՈՉ (2ԿԱՍ-ՈՉ, NOR2, 2ИЛИ-НЕ) տարրի գրաֆիկական սիմվոլները

Ըստ աղյուսակ 1.4-ի  $y_8=1$  միայն այն դեպքում, երբ երկու փոփոխականները միաժամանակ հավասար են 0-ի՝  $x_1=x_2=0$ :  $y_8=0$ , եթե  $x_1$  կամ  $x_2$ -ը հավասար է 1-ի:  $y_9 = x_1 \oplus x_2$  ֆունկցիան կոչվում է անհավասարագործություն, մոդուլ 2-ով գումարում, Բացառող-ԿԱՍ (XOR, Исключающее-ИЛИ) ֆունկցիա: Մոդուլ 2-ով գումարում տրամաբանական գործողությունը բանաձևերում նշվում է  $\oplus$ ,  $\wedge$  նշաններով: Մոդուլ 2-ով գումարում ֆունկցիան իրականացնող տրամաբանական տարրը կոչվում է Բացառող-ԿԱՍ (XOR, Исключающее-ИЛИ) տարր: Նկ. 1.7-ում ցույց են տրված երկու մուտքերով Բացառող-ԿԱՍ (2Բացառող-ԿԱՍ, XOR2, 2Исключающее-ИЛИ) տարրի գրաֆիկական սիմվոլները:



Նկ. 1.7. Երկու մուտքերով Բացառող-ԿԱՍ (2Բացառող-ԿԱՍ, XOR2, 2Исключающее-ИЛИ) տարրի գրաֆիկական սիմվոլները

Աղյուսակ 1.4-ից հետևում է, որ  $y_9=1$  այն դեպքում, երբ  $x_1 \neq x_2$ :  $y_{10} = x_1 \sim x_2 = \overline{x_1 \oplus x_2}$  կոչվում է հավասարագործության ֆունկցիա, այն մոդուլ 2-ով գումարման (անհավասարագործության) ժխտումն է: Նկ. 1.8-ում ցույց են տրված երկու մուտքերով հավասարագործություն (XNOR2) տարրի գրաֆիկական սիմվոլները:



Նկ. 1.8. Երկու մուտքերով հավասարագործություն (XNOR2) տարրի գրաֆիկական սիմվոլները

Աղյուսակ 1.4-ից հետևում է, որ  $y_{10}=1$  այն դեպքում, երբ  $x_1=x_2$ :

### 1.1.3. Տրամաբանական ֆունկցիաների հատկությունները

Ցանկացած թվային շղթա նկարագրվում է տրամաբանական ֆունկցիաներով և համապատասխան բանաձևերով: Դիտարկենք մի քանի կարևոր նույնություններ, որոնք բնութագրում են թվային համակարգերում լայն կիրառություն գտած ֆունկցիաների հատկությունները: Այդ նույնությունները թույլ են տալիս ձևափոխել և պարզեցնել տրամաբանական ֆունկցիաների բանաձևերը և դրանցով նկարագրվող թվային շղթաները:

(1) **Զուգորդական** (ասոցիատիվ) հատկություն: Այս հատկությամբ օժտված են  $\&$ ,  $+$ ,  $\oplus$  տրամաբանական գործողությունները.

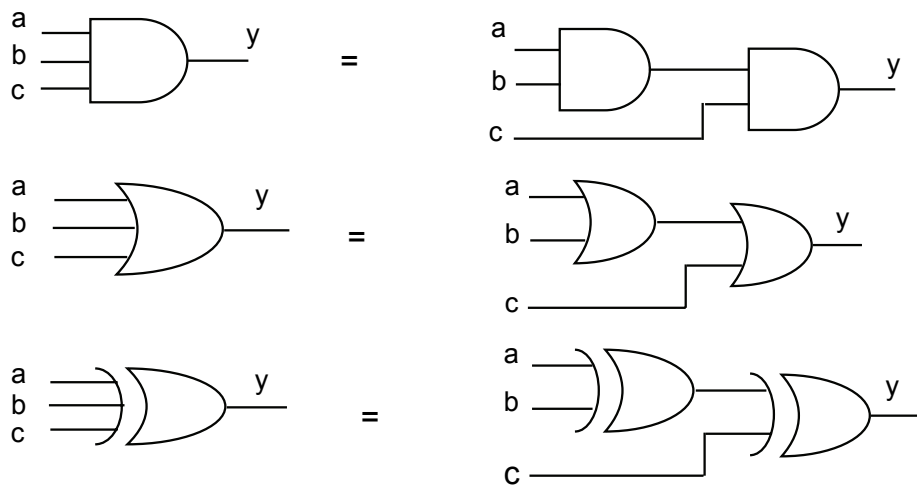
$$y = a \& (b \& c) = (a \& b) \& c = a \& b \& c \quad (1.1)$$

$$y = a + (b + c) = (a + b) + c = a + b + c \quad (1.2)$$

$$y = a \oplus (b \oplus c) = (a \oplus b) \oplus c = a \oplus b \oplus c \quad (1.3)$$

Այս հատկությունից հետևում է, որ մի քանի փոփոխականների միջև միևնույն գործողությունը կատարելիս փոփոխականների ընտրության կարգը կարևոր չէ:

Նկ. 1.9-ում ցույց տրված տրամաբանական շղթաները համապատասխանում են (1.1) - (1.3) բանաձևերին:



Նկ. 1.9. (1.1) - (1.3) բանաձևերով տրված ձևափոխություններին համապատասխանող տրամաբանական շղթաները

(2) **Փոխադասելի** (կոմուտատիվ) հատկություն: Այս հատկությամբ օժտված են  $\&$ ,  $+$ ,  $\oplus$  տրամաբանական գործողությունները.

$$y = a \& b = b \& a, \quad (1.4)$$

$$y = a + b = b + a, \quad (1.5)$$

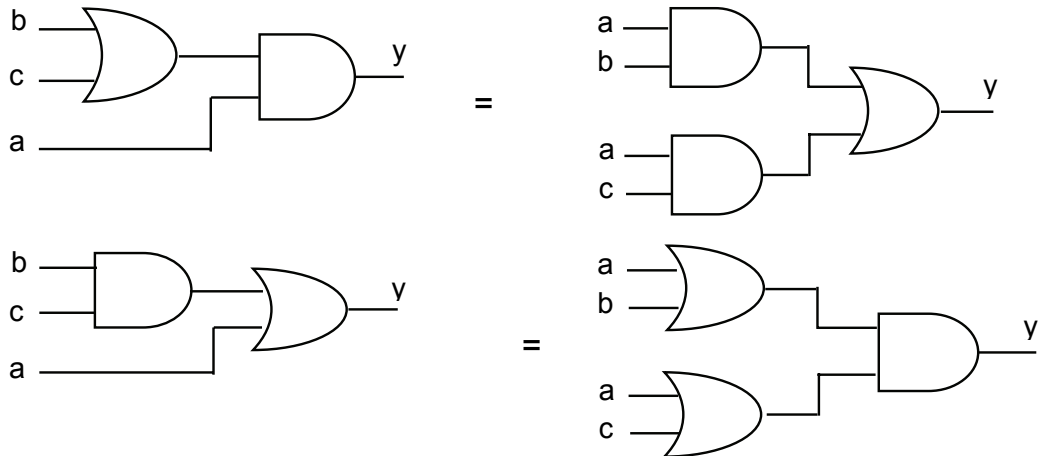
$$y = a \oplus b = b \oplus a: \quad (1.6)$$

(3) Կոմյունկցիա և դիզյունկցիա ֆունկցիաները օժտված են **բաշխական** (դիստրիբուտիվ) հատկությամբ.

$$y = a \& (b + c) = a \& b + a \& c, \quad (1.7)$$

$$y = a + (b \& c) = (a + b) \& (a + c): \quad (1.8)$$

Նկ. 1.10-ում ցույց տրված տրամաբանական շղթաները համապատասխանում են (1.7) - (1.8) բանաձևերին:



Նկ. 1.10. (1.7) - (1.8) բանաձևերով տրված ձևափոխություններին համապատասխանող տրամաբանական շղթաները

(4) Կրկնակի բացասման հատկությունը.

$$\bar{\bar{x}} = x \quad (1.9)$$

(5) ԵՎ, ԿԱՄ և մոդուլ 2-ով գումարման գործողությունները օժտված են հետևյալ հատկություններով, որոնք հետևում են աղյուսակ 1.4-ից:

$$a \& 0 = 0 \quad a + 0 = a$$

$$a \& 1 = a \quad a + 1 = 1$$

$$a \& a = a \quad a + a = a$$

$$a \& \bar{a} = 0 \quad a + \bar{a} = 1$$

$$a \oplus 0 = a \quad a \oplus 1 = \bar{a}$$

$$a \oplus a = 0 \quad a \oplus \bar{a} = 1$$

$$\bar{a} \oplus b = a \oplus \bar{b} = \overline{a \oplus b} = a \oplus b \oplus 1$$

(6) **Դե Մորգանի բանաձևերը.** Բացասում, կոնյունկցիա և դիզյունկցիա գործողությունների միջև գոյություն ունեն հետևյալ առնչությունները.

$$a \& b = \overline{\bar{a} + \bar{b}}, \quad (1.10)$$

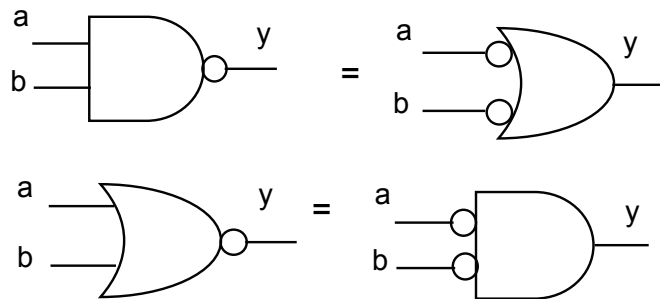
$$a + b = \overline{\bar{a} \& \bar{b}}: \quad (1.11)$$

Ժխտելով (1.10), (1.11) բանաձևերի աջ և ձախ մասերը՝ կարելի է ստանալ դե Մորգանի ձևափոխված բանաձևերը.

$$y = \overline{a \& b} = \bar{a} + \bar{b}, \quad (1.12)$$

$$y = \overline{a + b} = \bar{a} \& \bar{b}: \quad (1.13)$$

(1.12) և (1.13) բանաձևերով տրվող ձևափոխությունները ցույց են տրված նկ. 1.11-ում բերված տրամաբանական շղթաներով:



Նկ. 1.11. (1.12) և (1.13) բանաձևերով տրվող ձևափոխություններին համապատասխանող տրամաբանական շղթաները

(7) **Գործողությունների կատարման կարգը.** տրամաբանական արտահայտությունների արժեքների անսխալ հաշվման համար պետք է պահպանել գործողությունների կատարման որոշակի հաջորդականություն: Առաջին հերթին պետք է կատարել փակագծերի ներսում գրված գործողությունները, այնուհետև՝ բացասման գործողությունը, կոնյունկցիան, դիզյունկցիան: Բանաձևերի գրառան պարզեցման համար շատ դեպքերում կարելի է փակագծերը բաց թողնել՝ պահպանելով գործողությունների նշված առաջնահերթությունները: Օրինակ, ժխտման նշանի տակ գտնվող արտահայտությունը պարտադիր չէ փակագծերի մեջ վերցնել՝

$$\overline{a + b \& c} = \overline{(a + b) \& c}:$$

Կամ, օգտվելով այն բանից, որ կոնյունկցիան ուժեղ է դիզյունկցիայից, փակագծերը կարելի է բաց թողնել բոլոր այն դեպքերում, երբ դա չի բերում երկիմաստության: Օրինակ՝

$$(a \& b) + c = a \& b + c:$$

Բանաձևերի պարզեցման համար կօգտագործենք նաև հետևյալ կրճատ գրառման ձևերը.

$$\sum_{i=1}^n x_i = x_1 + x_2 + \dots + x_n, \quad (1.14)$$

$$\&_{i=1}^n x_i = x_1 \& x_2 \& \dots \& x_n: \quad (1.15)$$

(8) **Դե Մորգանի բանաձևերը  $n$  փոփոխականների համար կունենան հետևյալ տեսքը.**

$$\overline{\&_{i=1}^n x_i} = \bar{x}_1 + \bar{x}_2 + \dots + \bar{x}_n = \sum_{i=1}^n \bar{x}_i, \quad (1.16)$$

$$\overline{\sum_{i=1}^n x_i} = \overline{x_1} \& \overline{x_2} \& \dots \& \overline{x_n} = \&_{i=1}^n \overline{x_i} : \quad (1.17)$$

(9) Տրամաբանական բազմապատկման և գումարման հատկություններից բխում են հետևյալ հատկությունները, որոնք թույլ են տալիս արագ հաշվել բանաձևերի արժեքները.

- եթե տրամաբանական արտադրյալում որևէ արտադրիչ հավասար է 0-ի, ապա ամբողջ արտադրյալը նույնպես 0 է:
- եթե տրամաբանական արտադրյալում կան արտադրիչներ հավասար հաստատուն 1-ի, ապա դրանք կարելի է չգրել:
- եթե տրամաբանական գումարում որևէ գումարելի հավասար է 1-ի, ապա ամբողջ գումարը հավասար է 1-ի:
- եթե տրամաբանական գումարում կան հաստատուն 0-ի հավասար գումարելիներ, ապա դրանք կարելի է չգրել:

(10) **Արտադրյալի կլանումը փոփոխականի կողմից**

$$y = x_1 + x_1 x_2 = x_1 (1 + x_2) = x_1 \cdot 1 = x_1 \quad (1.18)$$

Այստեղ  $x_1 x_2$  արտադրյալը կլանվեց  $x_1$  փոփոխականի կողմից:

(11) **Արտադրյալների սոսնձում.** եթե երկու արտադրյալ բաղկացած են միևնույն փոփոխականներից և տարբերվում են միայն մեկ փոփոխականի մասնակցության ձևով (ուղիղ կամ ժխտված), ապա այդ երկու արտադրյալները սոսնձվում են՝ տարբեր ձևով մասնակցող փոփոխականը կրճատվում է: Սոսնձումը ցույց տանք օրինակներով.

$$y = x_1 x_2 + x_1 \overline{x_2} = x_1 (x_2 + \overline{x_2}) = x_1, \quad (1.19)$$

$$y = x_1 x_2 \dots x_{n-1} x_n + x_1 x_2 \dots x_{n-1} \overline{x_n} = x_1 x_2 \dots x_{n-1} (x_n + \overline{x_n}) = x_1 x_2 \dots x_{n-1} : \quad (1.20)$$

(12) **Երկակիության (դուալության) սկզբունքը.** ցանկացած տրամաբանական ձևափոխություններ, որոնք կիրառելի են  $F$  ֆունկցիայի բանաձևի նկատմամբ, կիրառելի են նաև երկակի ( $F^0$ ) ֆունկցիայի նկատմամբ:  $F$  ֆունկցիային երկակի  $F^0$  ֆունկցիայի բանաձևն ստացվում է  $F$ -ի բանաձևից՝ ԿԱՄ գործողությունները պետք է փոխարինվեն ԵՎ-ով, ԵՎ գործողությունները՝ ԿԱՄ-ով, 1-ը՝ 0-ով, 0-ն՝ 1-ով.

$$F^0(x_1, x_2, \dots, x_n, '&', '+') = F(x_1, x_2, \dots, x_n, '+', '&'): \quad (1.21)$$

**Օրինակ 1. 2.** Որոշել հետևյալ ֆունկցիայի երկակի ֆունկցիայի բանաձևը.

$$F = \bar{x}y\bar{z} + \bar{x}\bar{y}z: \quad (1.22)$$

$F$ -ի երկակի ֆունկցիայի բանաձևը կլինի.

$$F^0 = (\bar{x} + y + \bar{z})(\bar{x} + \bar{y} + z): \quad (1.23)$$

(13) **Դե Մորգանի ընդհանրացված թեորեմը.** տրված ֆունկցիայի ժխտված տեսքը կարելի ստանալ՝ ԿԱՄ գործողությունները փոխարինելով ԵՎ-ով, ԵՎ գործողությունները՝ ԿԱՄ-ով և ժխտելով բոլոր փոփոխականները.

$$\bar{F}(x_1, x_2, \dots, x_n, '&', '+') = F(\overline{x_1}, \overline{x_2}, \dots, \overline{x_n}, '+', '&'): \quad (1.24)$$



**Օրինակ 1.3.** Որոշել հետևյալ ֆունկցիայի ժխտված տեսքը.

$$F = \bar{x}y\bar{z} + \bar{x}\bar{y}z: \quad (1.25)$$

Կիրառելով դե Մորգանի թեորեմն այնքան ազամ, ինչքան անհրաժեշտ է, կստանանք.

$$\bar{F} = \overline{\bar{x}y\bar{z} + \bar{x}\bar{y}z} = \overline{\bar{x}y\bar{z}} \& \overline{\bar{x}\bar{y}z} = (x + \bar{y} + z)(x + y + \bar{z}): \quad (1.26)$$

Ժխտված ֆունկցիայի բանաձևը կարելի է ստանալ հետևյալ պարզ եղանակով՝ ստանալ տրված ֆունկցիայի երկակի ֆունկցիայի բանաձևը և ժխտել փոփոխականները: Այս եղանակը հետևում է դե Մորգանի ընդհանրացված թեորեմից և երկակի ֆունկցիայի սահմանումից: Եթե երկակի ֆունկցիայի (1.23) բանաձևում ժխտվեն բոլոր փոփոխականները, կստացվի ժխտված ֆունկցիայի (1.26) բանաձևը:

**(14) Ֆունկցիայի վերլուծությունը ըստ փոփոխականների** (Շենոնի թեորեմը): Ֆունկցիայի վերլուծությունը ըստ որևէ փոփոխականի տրվում է հետևյալ նույնությամբ.

$$F(x_1, x_2, \dots, x_n) = \bar{x}_1 F(0, x_2, \dots, x_n) + x_1 F(1, x_2, \dots, x_n): \quad (1.27)$$

Այստեղ  $F$  ֆունկցիան վերլուծված է ըստ  $x_1$ -ի: (1.27) նույնությունը կարել է ապացուցել՝  $x_1$ -ին տալով նախ 0, այնուհետև 1 արժեք:  $F(0, x_2, \dots, x_n)$  և  $F(1, x_2, \dots, x_n)$  ֆունկցիաները կոչվում են վերլուծության բաղադրիչներ: Ֆունկցիայի վերլուծության գործնական կիրառություններից մեկը հիմնված է այն բանի վրա, որ  $F(0, x_2, \dots, x_n)$  և  $F(1, x_2, \dots, x_n)$  ֆունկցիաները պարունակում են մեկ փոփոխական պակաս, քան  $F(x_1, x_2, \dots, x_n)$ -ը:  $F(0, x_2, \dots, x_n)$  և  $F(1, x_2, \dots, x_n)$ -ն իրականացնելու համար անհրաժեշտ կլինի  $n$ -ի փոխարեն ունենալ միայն  $(n-1)$  մուտքերով տարրեր:

Նույն եղանակով կարելի է վերլուծել  $F(0, x_2, \dots, x_n)$  և  $F(1, x_2, \dots, x_n)$ -ը ըստ  $x_2$ -ի.

$$F(x_1, x_2, \dots, x_n) = \bar{x}_1 F(0, x_2, \dots, x_n) + x_1 F(1, x_2, \dots, x_n) = \bar{x}_1 \cdot \bar{x}_2 F(0, 0, x_3, \dots, x_n) + \bar{x}_1 x_2 F(0, 1, x_3, \dots, x_n) + x_1 \bar{x}_2 F(1, 0, x_3, \dots, x_n) + x_1 x_2 F(1, 1, x_3, \dots, x_n): \quad (1.28)$$

Շարունակելով ֆունկցիայի վերլուծությունը ըստ մնացած բոլոր փոփոխականների, կստանանք.

$$F(x_1, x_2, \dots, x_n) = \bar{x}_1 \cdot \bar{x}_2 \cdots \bar{x}_n F(0, 0, \dots, 0) + \bar{x}_1 \cdot \bar{x}_2 \cdots x_n F(0, 0, \dots, 1) + \cdots + x_1 \cdot x_2 \cdots \bar{x}_n F(1, 1, \dots, 0) + x_1 \cdot x_2 \cdots x_n F(1, 1, \dots, 1): \quad (1.29)$$

Այս բանաձևում  $F(0, 0, \dots, 0)$ , ...,  $F(1, 1, \dots, 1)$  վերլուծության բաղադրիչները ֆունկցիայի արժեքներն են մուտքային փոփոխականների համապատասխան հավաքածուների վրա, որոնք ունեն 0 կամ 1 արժեք: (1.29) բանաձևը ցույց է տալիս, որ տրամաբանական ֆունկցիայի բանաձևը կարելի է արտահայտել փոփոխականների այն արտադրյալների գումարի տեսքով, որոնց համապատասխանող հավաքածուների վրա ֆունկցիան ունի 1 արժեք: Եթե որևէ հավաքածուի վրա ֆունկցիան ընդունում է 0 արժեք, (1.29) բանաձևում համապատասխան բաղադրիչը հավասար է 0-ի, և այդ արտադրյալը բացակայում է ֆունկցիայի բանաձևից:

#### 1.1.4. Տրամաբանական արտահայտությունների ձևափոխություններ

Թվային համակարգի սխեմատիկական իրականացման բարդությունը կախված է այն նկարագրող տրամաբանական ֆունկցիաների բանաձևերում մասնակցող փոփոխականները ներկայացնող տառերի (լիտերալների) թվից. յուրաքանչյուր տառ համապատասխանում է մեկ ազդանշանային մուտքի: Տրամաբանական արտահայտության մեջ տառերի թիվը կարելի է նվազարկել հանրահաշվական ձևափոխություններով, որոնք հիմնված են տրամաբանական ֆունկցիաների հատկությունների վրա: Սակայն չկան հատուկ կանոններ, որոնց միջոցով կարելի կլինի ստանալ երաշխավորված նվազագույն բանաձևը: Յուրաքանչյուր դեպքում պետք է փորձել ձևափոխությունների տարբեր մոտեցումներ և գնահատել դրանց արդյունավետությունը: Հետևյալ օրինակները ցուցադրում են բանաձևերի նվազարկման ձևափոխությունները:

**Օրինակ 1.4.** Ձևափոխել հետևյալ տրամաբանական արտահայտությունները մինչև նվազագույն թվով տառեր պարունակող բանաձևերի ստանալը.

$$(ա) \quad x + \bar{x}y = (x + \bar{x})(x + y) = 1 \cdot (x + y) = x + y ,$$

որտեղ օգտագործվեց ԿԱՄ գործողության նկատմամբ բաշխական հատկությունը:

$$(բ) \quad x(\bar{x} + y) = x\bar{x} + xy = 0 + xy = xy ,$$

որտեղ օգտագործվեց ԵՎ գործողության նկատմամբ բաշխական հատկությունը:

$$(գ) \quad \bar{x}\bar{y}z + \bar{x}yz + x\bar{y} = \bar{x}z(\bar{y} + y) + x\bar{y} = \bar{x}z + x\bar{y};$$

$$(դ) \quad xy + \bar{x}z + yz = xy + \bar{x}z + yz(x + \bar{x}) = xy + \bar{x}z + xyz + \bar{x}yz = \\ = xy(1 + z) + \bar{x}z(1 + y) = xy + \bar{x}z;$$

Այս օրինակում ձևափոխությունների միջանկյալ փուլում ավելացվեցին տառեր, որոնք հնարավորություն տվեցին ստանալ նվազագույն թվով տառեր պարունակող վերջնական բանաձև: Բերված օրինակը կոչվում է նաև ընդհանրացված սոսնձման հատկություն:

$$(ե) \quad (x + y)(\bar{x} + z)(y + z) = (x + y)(\bar{x} + z) :$$

Սա ստացվեց օրինակ (դ)-ից՝ կիրառելով երկակիության սկզբունքը:

#### 1.2. Տրամաբանական ֆունկցիաների կատարյալ դիզյունկտիվ և կոնյունկտիվ նորմալ ձևեր

Թվային շղթայի աշխատանքը հարմար է նկարագրել իսկության աղյուսակներով, իսկ սխեմայի կառուցման համար՝ օգտվել բանաձևերից: Յուրաքանչյուր տրամաբանական փոփոխական բանաձևում կարող է մասնակցել ուղիղ ձևով ( $x$ ) կամ ժխտված ձևով ( $\bar{x}$ ):

Գրառումների ընդհանրության համար կօգտվենք հետևյալ նշանակումից՝

$$x^\sigma = \begin{cases} \bar{x}, & \text{երբ } \sigma = 0 \\ x, & \text{երբ } \sigma = 1 \end{cases} : \quad (1.30)$$

Տարրական կոնյունկցիա է կոչվում  $K_r = x_1^{\sigma_1} \& x_2^{\sigma_2} \& \dots \& x_r^{\sigma_r}$  տեսքի արտադրյալը, որտեղ բոլոր փոփոխականները միմյանցից տարբեր են:  $r$ -ը կոչվում է կոնյունկցիայի ռանգ:  $n$  փոփոխականների դեպքում, երբ  $r=n$ , այսինքն, երբ արտադրյալում մասնակցում են բոլոր  $n$  փոփոխականները, արտադրյալը կոչվում է միներմ: Օրինակ, երկու փոփոխականների դեպքում կունենանք չորս միներմներ՝  $\overline{x}\overline{y}$ ,  $\overline{x}y$ ,  $x\overline{y}$ ,  $xy$ : Որևէ միներմ 1 արժեք է ընդունում փոփոխականների միայն մեկ հավաքածուի վրա, մնացած բոլոր հավաքածուների վրա այդ միներմը ընդունում է 0 արժեք: Օրինակ,  $x\overline{y}$  միներմը 1 արժեք է ընդունում միայն 10 հավաքածուի վրա, մնացած 00, 01 և 11 հավաքածուների վրա այն ընդունում է 0 արժեք:

Նույն ձևով,  $n$  փոփոխականների դեպքում կան  $2^n$  միներմներ, որոնցից յուրաքանչ-յուրը համապատասխանում է մեկ հավաքածուի: Երեք փոփոխականի հավաքածուները և դրանց համապատասխանող միներմները բերված են աղյուսակ 1.5-ում: Եթե հավաքածուում  $x=0$ , ապա միներմում  $x$ -ը մասնակցում է ժխտված, իսկ հավաքածուում  $x=1$  դեպքում միներմում  $x$ -ը մասնակցում է ուղիղ ձևով: Աղյուսակում ցույց է տրված նաև միներմի կրճատ գրառումը՝  $m_j$ , որտեղ  $j$ -ն տվյալ միներմին համապատասխանող երկուական հավաքածուի տասական արժեքն է:

Աղյուսակ 1.5

xyz	Միներմներ		Մաքսթերմներ	
000	$\overline{x}\overline{y}\overline{z}$	$m_0$	$x + y + z$	$M_0$
001	$\overline{x}\overline{y}z$	$m_1$	$x + y + \overline{z}$	$M_1$
010	$\overline{x}y\overline{z}$	$m_2$	$x + \overline{y} + z$	$M_2$
011	$\overline{x}yz$	$m_3$	$x + \overline{y} + \overline{z}$	$M_3$
100	$x\overline{y}\overline{z}$	$m_4$	$\overline{x} + y + z$	$M_4$
101	$x\overline{y}z$	$m_5$	$\overline{x} + y + \overline{z}$	$M_5$
110	$x\overline{y}\overline{z}$	$m_6$	$\overline{x} + \overline{y} + z$	$M_6$
111	$xyz$	$m_7$	$\overline{x} + \overline{y} + \overline{z}$	$M_7$

Երբ արտադրյալում մասնակցում են ոչ բոլոր փոփոխականները՝  $r < n$ , ապա այդպիսի արտադրյալը կոչվում է իմպլիկանտ: Իմպլիկանտը 1 արժեք է ընդունում  $2^{n-r}$  հավաքածուների վրա: Օրինակ, երեք փոփոխականների  $(x, y, z)$  դեպքում  $x\overline{y}$  միներմն ընդունում 1 արժեք  $2^{n-r}=2^{3-2}=2$  հավաքածուների վրա՝ 010, 011:

Տարրական դիզյունկցիա է կոչվում  $D_r = x_1^{\sigma_1} + x_2^{\sigma_2} + \dots + x_r^{\sigma_r}$  տեսքի գումարը, որտեղ բոլոր փոփոխականները միմյանցից տարբեր են:  $r$ -ը կոչվում է դիզյունկցիայի երկարություն:  $n$  փոփոխականների դեպքում, երբ  $r=n$ , այսինքն, երբ գումարում մասնակցում են բոլոր  $n$  փոփոխականները, գումարը կոչվում է մաքսթերմ: Օրինակ, երկու փոփոխականների դեպքում կունենանք չորս մաքսթերմ՝  $\overline{x} + \overline{y}$ ,  $\overline{x} + y$ ,  $x + \overline{y}$ ,  $x + y$ : Որևէ մաքսթերմ ընդունում է 0 արժեք փոփոխականների միայն մեկ հավաքածուի վրա, մնացած բոլոր հավաքածուների վրա այդ մաքսթերմն ընդունում է 1 արժեք: Օրինակ,  $x + \overline{y}$  միներմը 0 արժեք է ընդունում միայն 01 հավաքածուի վրա, մնացած 00, 10 և

11 հավաքածուների վրա այն ընդունում է 1 արժեք: Նույն ձևով,  $n$  փոփոխականների դեպքում կան  $2^n$  մաքսերմներ, որոնցից յուրաքանչյուրը համապատասխանում է մեկ հավաքածուի: Երեք փոփոխականի բոլոր մաքսերմները բերված են աղյուսակ 1.5-ում: Աղյուսակում ցույց է տրված նաև մաքսերմի կրճատ գրառումը՝  $M_j$ , որտեղ  $j$ -ն տվյալ միներմին համապատասխանող երկուական հավաքածուի տասական արժեքն է:

Դիզյունկտիվ նորմալ ձև (ԴՆՁ) է կոչվում տարրական կոնյունկցիաների դիզյունկցիան.

$$D = K_1 + K_2 + \dots + K_m : \quad (1.30)$$

ԴՆՁ-ն կոչվում է նաև արտադրյալների գումար (ԱԳ): Եթե (1.30)-ում բոլոր տարրական կոնյունկցիաները միներմներ են, ԴՆՁ կոչվում է կատարյալ ԴՆՁ (ԿԴՆՁ):

Կոնյունկտիվ նորմալ ձև (ԿՆՁ) է կոչվում տարրական դիզյունկցիաների կոնյունկցիան.

$$K = D_1 \& D_2 \& \dots \& D_s : \quad (1.31)$$

ԿՆՁ-ն կոչվում է նաև գումարների արտադրյալ (ԳԱ): Եթե (1.31)-ում բոլոր տարրական դիզյունկցիաները մաքսերմներ են, ԿՆՁ կոչվում է կատարյալ ԿՆՁ (ԿԿՆՁ):

Հետևյալ երկու թեորեմները որոշում են տրամաբանական ֆունկցիայի բանաձևերը դիզյունկտիվ և կոնյունկտիվ նորմալ ձևերով:

Թեորեմ. Ցանկացած տրամաբանական ֆունկցիա կարելի է ներկայացնել կատարյալ դիզյունկտիվ նորմալ ձևով (ԿԴՆՁ)՝

$$F(x_1, x_2, \dots, x_n) = \sum_{F(\sigma_1, \sigma_2, \dots, \sigma_n)=1} x_1^{\sigma_1} \& x_2^{\sigma_2} \& \dots \& x_n^{\sigma_n} , \quad (1.32)$$

որտեղ տրամաբանական գումարը վերցվում է այն  $x_1^{\sigma_1} \& x_2^{\sigma_2} \& \dots \& x_n^{\sigma_n}$  միներմներից, որոնց համապատասխանող  $(\sigma_1, \sigma_2, \dots, \sigma_n)$  հավաքածուների վրա ֆունկցիան ընդունում է 1 արժեք:

Այս թեորեմի ապացույցը ուղղակիորեն բխում է տրամաբանական ֆունկցիայի վերլուծության (1.29) բանաձևից:

Ցանկացած տրամաբանական ֆունկցիայի ԿԴՆՁ-ով արտահայտված բանաձևը կարելի է ստանալ իսկության աղյուսակից՝ կազմելով միներմներ բոլոր այն հավաքածուների համար, որոնց վրա ֆունկցիան ընդունում է 1 արժեք, և միացնելով այդ միներմները ԿԱՄ գործողությամբ:

**Օրինակ 1.5.** Ստանալ աղյուսակ 1.6-ով տրված  $F$  ֆունկցիայի բանաձևը ԿԴՆՁ-ով:

$F$  ֆունկցիան 000, 001, 100, 101 հավաքածուների վրա 1 է: Հետևաբար, ֆունկցիայի բանաձևը որոշվում է այդ հավաքածուներին համապատասխանող  $\bar{x}\bar{y}\bar{z}$ ,  $\bar{x}\bar{y}z$ ,  $x\bar{y}\bar{z}$ ,  $xy\bar{z}$  միներմներով.

$$F = \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + x\bar{y}\bar{z} + xy\bar{z} = m_0 + m_1 + m_4 + m_5 : \quad (1.33)$$

#	x	y	z	F
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Թեորեմ. Ցանկացած տրամաբանական ֆունկցիա կարելի է ներկայացնել կատարյալ կոնյունկտիվ նորմալ ձևով (ԿԿՆՁ).

$$F(x_1, x_2, \dots, x_n) = \&_{F(\sigma_1, \sigma_2, \dots, \sigma_n)=0} (x_1^{\sigma_1} + x_2^{\sigma_2} + \dots + x_n^{\sigma_n}), \quad (1.34)$$

որտեղ կոնյունկցիան վերցվում է փոփոխականների այն  $(\sigma_1, \sigma_2, \dots, \sigma_n)$  հավաքածուների համար, որոնց վրա ֆունկցիան ընդունում է 0 արժեք:

Երբ  $\sigma = 0, x^{\sigma} = x^{\bar{0}} = x^1 = x$ , իսկ երբ  $\sigma = 1, x^{\sigma} = x^{\bar{1}} = x^0 = \bar{x}$ :

**Օրինակ 1.6.** Ստանալ աղյուսակ 1.6-ով տրված F ֆունկցիայի բանաձևը ԿԿՆՁ-ով:

F ֆունկցիան 010, 011, 110, 111 հավաքածուների վրա հավասար է 0: Հետևաբար, ֆունկցիայի բանաձևը որոշվում է այդ հավաքածուներին համապատասխանող  $x + \bar{y} + z, x + \bar{y} + \bar{z}, \bar{x} + \bar{y} + z, \bar{x} + \bar{y} + \bar{z}$  մաքսթերմների արտադրյալով.

$$F = (x + \bar{y} + z)(x + \bar{y} + \bar{z})(\bar{x} + \bar{y} + z)(\bar{x} + \bar{y} + \bar{z}) = M_2 \& M_3 \& M_6 \& M_7 : \quad (1.35)$$

Ապացուցենք, որ F ֆունկցիան, իրոք, նկարագրվում է (1.35) բանաձևով: Նախ կառուցենք  $\bar{F}$  ֆունկցիայի ԿԿՆՁ-ն: Իսկության աղյուսակից պետք է ընտրել այն հավաքածուները, որոնց վրա F ֆունկցիան ընդունում է 0 արժեք և կազմել դրանց համապատասխանող միներմների գումարը.

$$\bar{F} = \bar{x}y\bar{z} + \bar{x}yz + xy\bar{z} + xyz : \quad (1.36)$$

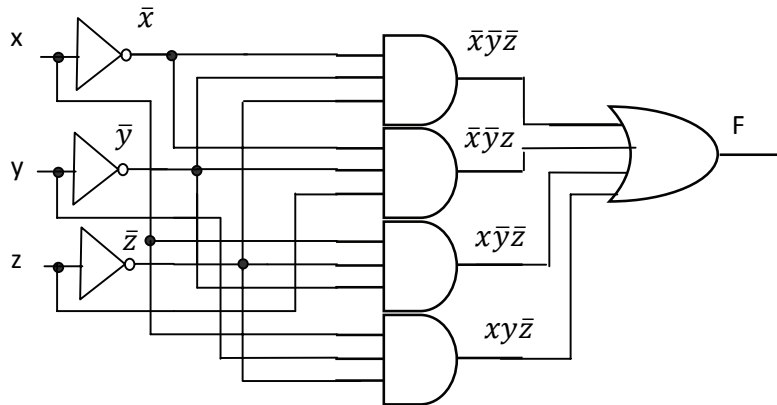
Այնուհետև, ժխտելով (1.36) բանաձևի երկու մասերը, կստանանք F ֆունկցիայի բանաձևը.

$$F = \overline{\bar{x}y\bar{z} + \bar{x}yz + xy\bar{z} + xyz} = \overline{\bar{x}y\bar{z}} \& \overline{\bar{x}yz} \& \overline{xy\bar{z}} \& \overline{xyz} = (x + \bar{y} + z)(x + \bar{y} + \bar{z})(\bar{x} + \bar{y} + z)(\bar{x} + \bar{y} + \bar{z}) : \quad (1.37)$$

Կարելի է նկատել, որ (1.35) և (1.37)-ը համընկնում են:

Բուլյան ֆունկցիան հանրահաշվական արտահայտությունից կարելի է ձևափոխել տրամաբանական սխեմայի՝ իրականացված ԵՎ, ԿԱՍ և ՈՉ տրամաբանական տարրերով: (1.33) արտահայտությամբ նկարագրվող ֆունկցիայի իրականացումը ցույց է տրված նկ.1.12–ում: Այս տրամաբանական սխեման պարունակում է ՈՉ տարր՝ յուրա-

քանջյուր փոփոխականի համար, որը ֆունկցիայում մասնակցում է ժխտված ձևով: Այն դեպքում, երբ փոփոխականի ժխտված ձևն առկա է իբրև մուտքային փոփոխական, շրջիչի կիրառությունն անհրաժեշտ չէ:



Նկ. 1.12.  $F = \bar{x}\bar{y}\bar{z} + \bar{x}yz + x\bar{y}\bar{z} + xyz$  բանաձևով ֆունկցիային համապատասխանող տրամաբանական սխեման

### 1.3.Տրամաբանական ֆունկցիաների լրիվ համակարգեր

Ֆունկցիաների  $\{f_1, f_2, \dots, f_s\}$  համակարգը կոչվում է ֆունկցիոնալ լրիվ կամ բազիս, եթե ցանկացած տրամաբանական ֆունկցիա կարող է ներկայացվել այդ համակարգի ֆունկցիաներից կազմված բանաձևով:

Տրամաբանական ֆունկցիաների ԿՂՆԶ և ԿԿՆԶ բանաձևերից բխում է, որ ցանկացած տրամաբանական ֆունկցիայի բանաձև կարելի է կառուցել՝ օգտագործելով միայն ԵՎ, ԿԱՄ, ՈՉ ֆունկցիաներ: Հետևաբար, տրամաբանական ֆունկցիաների

$$U = \{a \& b, a + b, \bar{a}\} \quad (1.38)$$

համակարգը լրիվ է՝ բազիս է:

Ակնհայտ է, որ ոչ բոլոր համակարգերն են ֆունկցիոնալ լրիվ: Օրինակ,  $\{0, 1\}$ ,  $\{a \& b\}$ ,  $\{a + b\}$ ,  $\{a \& b, a + b\}$  համակարգերից ոչ մեկը լրիվ չէ:

Բերենք թեորեմ (առանց ապացույցի), որը թույլ է տալիս մեկ համակարգի լրիվության ստուգման խնդիրը բերել որևէ լրիվ համակարգի հետ համեմատման խնդրի:

Թեորեմ. Ենթադրենք տրված են հետևյալ երկու համակարգերը՝

$$U_f = \{f_1, f_2, \dots, f_s\},$$

$$U_g = \{g_1, g_2, \dots, g_s\}$$

և հայտնի է, որ  $U_f$ -ը համակարգը լրիվ է, և նրա յուրաքանչյուր ֆունկցիա կարելի է արտահայտել  $U_g$  համակարգի ֆունկցիաներով: Այդ դեպքում  $U_g$  համակարգը նույնպես ֆունկցիոնալ լրիվ է:

Այս թեորեմի հիման վրա կարելի է ընդարձակել ֆունկցիոնալ լրիվ համակարգերի ցուցակը:

**Օրինակ 1.7.** Հիմնվելով  $U = \{ a \& b, a+b, \bar{a} \}$  համակարգի լրիվության վրա՝ ցույց տանք, որ

$$U1 = \{ a \& b, \bar{a} \} \quad (1.39)$$

համակարգը նույնպես լրիվ է: Դրա համար բավարար է ցույց տալ, որ  $a+b$  ֆունկցիան արտահայտվում է  $a \& b$ ,  $\bar{a}$  ֆունկցիաների միջոցով: Դա ցույց է տրված (1.11) բանաձևով տրվող դե Մորգանի թեորեմով:

Նույն եղանակով կարելի է ցույց տալ, որ

$$U2 = \{ a+b, \bar{a} \} \quad (1.40)$$

համակարգը նույնպես ֆունկցիոնալ լրիվ է:

Կարելի է ցույց տալ, որ հետևյալ համակարգերը նույնպես ֆունկցիոնալ լրիվ են.

$$U3 = \{ \overline{a \& b} \}, \quad (1.41)$$

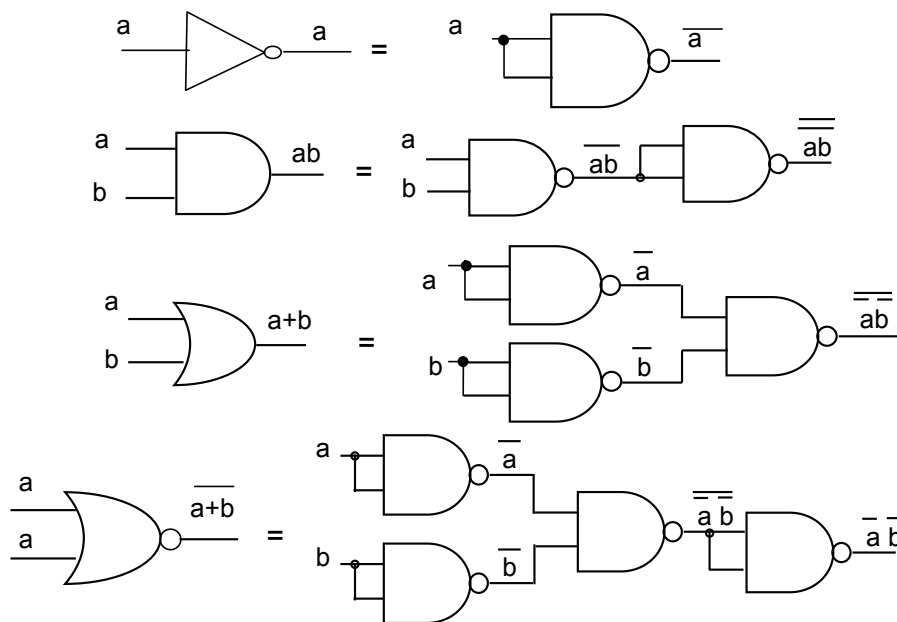
$$U4 = \{ \overline{a+b} \}: \quad (1.42)$$

Միայն մեկ ֆունկցիայից բաղկացած ֆունկցիոնալ լրիվ համակարգը կոչվում է մոնոֆունկցիոնալ բազիս:

**ԵՎ-ՈՉ մոնոֆունկցիոնալ բազիս.**  $U3$ -ի և  $U1$ -ի համեմատումից հետևում է, որ  $U3$ -ի բազիս լինելը ապացուցելու համար, բավարար է ցույց տալ, որ ԵՎ և ՈՉ ֆունկցիաներն արտահայտվում են ԵՎ-ՈՉ ֆունկցիայով՝

$$a \& b = \overline{\overline{a \& b}} = \overline{\overline{a} \& \overline{b}}: \quad (1.43)$$

Նկ. 1.13-ում ցույց են տրված ԵՎ և ՈՉ ֆունկցիաներն ԵՎ-ՈՉ ֆունկցիայով արտահայտելուն համապատասխանող շղթաների համարժեք ձևափոխությունները:



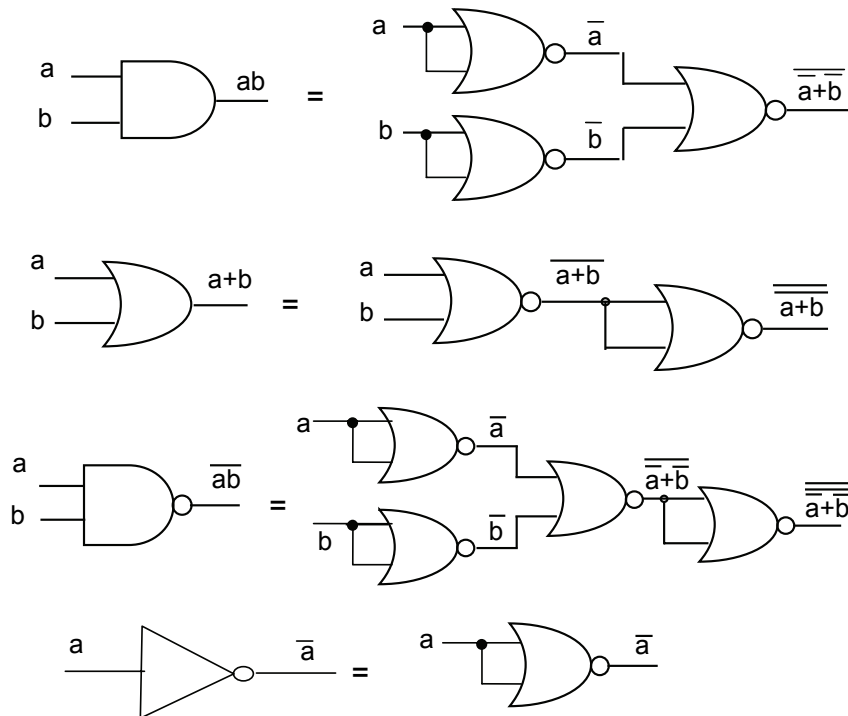
Նկ. 1.13. Երկմուտք ՈՉ, ԵՎ, ԿԱՄ, ԿԱՄ-ՈՉ տարրերի փոխարինումը ԵՎ-ՈՉ տարրերով

Օգտվելով նկ. 1.13-ում ցույց տրված շղթաների համարժեք ձևափոխություններից՝ կարելի է ցանկացած տրամաբանական շղթա, որը կառուցված է ԵՎ, ԿԱՄ, ԿԱՄ-ՈՉ, ԵՎ-ՈՉ, ՈՉ տարրերի միջոցով, ձևափոխել այնպես, որ այն բաղկացած լինի միայն ԵՎ-ՈՉ տարրերից:

**ԿԱՄ-ՈՉ մոնոֆունկցիոնալ բազիս.** Ս4-ի և Ս2-ի համեմատումից պարզվում, է որ Ս4-ի բազիս լինելը ապացուցելու համար, բավարար է ցույց տալ, որ ԿԱՄ և ՈՉ ֆունկցիաներն արտահայտվում են ԿԱՄ-ՈՉ ֆունկցիայով՝

$$a + b = \overline{\overline{a + b}} = \overline{\overline{a} \cdot \overline{b}} : \quad (1.44)$$

Օգտվելով նկ. 1.14-ում ցույց տրված շղթաների համարժեք ձևափոխություններից՝ կարելի է ցանկացած տրամաբանական շղթա, որը կառուցված է ԵՎ, ԿԱՄ, ԿԱՄ-ՈՉ, ԵՎ-ՈՉ, ՈՉ տարրերի միջոցով, ձևափոխել այնպես, որ այն բաղկացած լինի միայն ԿԱՄ-ՈՉ տարրերից:



Նկ. 1.14. Երկմուտք ՈՉ, ԵՎ, ԿԱՄ, ԿԱՄ-ՈՉ տարրերի փոխարինումը 2ԿԱՄ-ՈՉ տարրերով

#### 1.4. Տրամաբանական ֆունկցիաների բանաձևերի նվազարկում

Տրամաբանական ֆունկցիաների բանաձևերից համապատասխան թվային շղթային անցնելու համար տրամաբանական գործողությունները պետք է իրականացվեն տրամաբանական տարրերով: Կառուցվող շղթայի բարդությունը ուղղակիորեն կախված է բանաձևի բարդությունից: Հետևաբար, թվային շղթայի պարզեցման համար պետք է պարզեցնել համապատասխան տրամաբանական ֆունկցիաների բանաձևերը՝ նվազարկել բանաձևում թերմերի թիվը և յուրաքանչյուր թերմում՝ սիմվոլների՝ տառերի



թիվը: Այդ երկուսի համատեղ նվազարկումը միշտ չէ, որ հնարավոր է: Սովորաբար խնդրի լուծվումը սահմանափակվում է տառերի թվի նվազարկմամբ:

Առավելապես լավ մշակված են դիզյունկտիվ նորմալ ձևով տրված բանաձևերի նվազարկման եղանակները: Դրանք փնտրում են դիզյունկտիվ ձևով ներկայացվող այնպիսի բանաձևեր, որոնք պարունակում են նվազագույն թվով տառեր՝ տրամաբանական փոփոխականներ: Այդպիսի ԴՆՁ-երը կոչվում են նվազագույն ԴՆՁ-եր: Նվազարկման այս եղանակները կոչվում են կանոնական:

Նվազարկման կանոնական խնդրի լուծման համար ելակետային է հանդիսանում ֆունկցիայի ԿԴՆՁ-ն կամ իսկության աղյուսակը:

Նվազարկման համակարգված եղանակները նկարագրվում են խիստ ալ գործիքներով և իրականացվում են ավտոմատացված էլեկտրոնային նախագծման ծրագրային գործիքներով:

Նախքան նվազարկման կոնկրետ ալգորիթմների նկարագրությանն անցնելը բերենք մի քանի անհրաժեշտ սահմանում:

Հարակից կոչվում են այն միներմները, որոնք տարբերվում են միայն մեկ փոփոխականի մասնակցության ձևով: Հարակից միներմներից մեկում այդ փոփոխականը մասնակցում է ուղիղ, իսկ մյուսում՝ ժխտված ձևով: Օրինակ,  $w\bar{x}y\bar{z}$  և  $wxyz$  միներմները հարակից են, քանի որ տարբերվում են միայն  $x$  փոփոխականի մասնակցության ձևով:

Երկու հարակից միներմներ կարող են սոսնձվել ըստ տարբերվող փոփոխականի: Արդյունքում այդ միներմները կփոխարինվեն մեկ տարրական կոնյունկցիայով, որում փոփոխականների թիվը մեկով պակաս է սկզբնական միներմների համեմատ: Օրինակ՝

$$w\bar{x}y\bar{z} + wxyz = wy\bar{z} : \quad (1.45)$$

Երկու հարակից միներմի սոսնձումից ստացված արտադրյալը կոչվում է իմպլիկանտ:

Երկու միներմի սոսնձումից ստացված իմպլիկանտը համարժեք է այդ միներմների, քանի որ իմպլիկանտը և միներմները “1” արժեք են ընդունում փոփոխականների միևնույն հավաքածուների վրա: Օրինակ,  $w\bar{x}y\bar{z}$  միներմը “1” արժեք է ընդունում 1010 հավաքածուի վրա,  $wxyz$ -ը՝ 1110-ի վրա, իսկ  $wy\bar{z}$  իմպլիկանտը՝ 1010 և 1110 հավաքածուների վրա: Այդ պատճառով ասում են, որ իմպլիկանտը ծածկում է այն միներմները, որոնց սոսնձումից այն առաջացել է:

Նույն փոփոխականներից կազմված իմպլիկանտները նույնպես կարող են հարակից լինել և սոսնձվել: Օրինակ, հետևյալ չորս միներմի՝  $w\bar{x}y\bar{z}$ ,  $wxyz$ ,  $w\bar{x}y\bar{z}$ ,  $wx\bar{y}\bar{z}$  զույգ առ զույգ սոսնձումներից կստացվեն  $wy\bar{z}$  և  $w\bar{y}\bar{z}$  իմպլիկանտները, որոնք կարող են սոսնձվել ըստ  $y$ -ի՝

$$(w\bar{x}y\bar{z} + wxyz) + (w\bar{x}y\bar{z} + wx\bar{y}\bar{z}) = wy\bar{z} + w\bar{y}\bar{z} = w\bar{z} : \quad (1.46)$$

Այն իմպլիկանտը, որը չի սոսնձվում ոչ մի ուրիշ իմպլիկանտի հետ, կոչվում է պարզ իմպլիկանտ:

Կան տրամաբանական ֆունկցիաների կանոնական նվազարկման բազմաթիվ մեթոդներ: Հիմնական մեթոդը, որն ընկած է ավտոտացված էլեկտրոնային նախագծման ծրագրերի հիմքում, Քվայն-Մաք-Կլասկիի մեթոդն է:

#### 1.4.1 Տրամաբանական ֆունկցիաների բանաձևերի նվազարկման Քվայն-Մաք-Կլասկիի եղանակ

Այս եղանակը ենթադրում է, որ ֆունկցիան տրված է ԿԴՆԶ-ով՝ իսկության աղյուսակով: Նվազարկման ալգորիթմն իրականացվում է երկու փուլով.

1. Իրականացվում են բոլոր հնարավոր սոսնձումները մինթերմների և իմպլիկանտների միջև: Արդյունքում ստացվում է ֆունկցիայի բոլոր պարզ իմպլիկանտների բազմությունը:
2. Այդ բազմությունից ընտրվում է այնպիսի ենթաբազմություն, որը կպարունակի այն իմպլիկանտները, որոնք ծածկում են ֆունկցիայի ԿԴՆԶ-ի բոլոր մինթերմները և կազմում են նվազագույն թվով տառեր պարունակող բանաձև:

Աղյուսակային եղանակով պարզեցման նախնական տվյալը (ելակետը) իսկության աղյուսակն է կամ ֆունկցիային բնորոշող մինթերմների ցուցակը: Պարզ իմպլիկանտների ցուցակը որոշվում է մինթերմների սոսնձման գործընթացի միջոցով: Այս գործընթացի արդյունքում յուրաքանչյուր մինթերմ համեմատվում է ցանկացած մյուս մինթերմների հետ: Եթե երկու մինթերմ տարբերվում են միայն մեկ փոփոխականի մասնակցության ձևով (ուղիղ կամ շրջված), ապա այդ փոփոխականը կրճատվում է և ստացվում է իմպլիկանտ, որը պարունակում է մեկով պակաս փոփոխականներ: Այս գործողությունը կրկնվում է յուրաքանչյուր մինթերմի համար մինչև բոլոր հնարավոր տարբերակների դիտարկման ավարտը: Պարզեցման այս գործընթացի ցիկլը կրկնվում է ստացված նոր բաղադրիչների համար ևս, և այդպես շարունակ, մինչև վերջում ստացվեն իմպլիկանտներ, որոնցում անհնար է այլևս փոփոխականների հետագա կրճատումը: Ձևավորված իմպլիկանտները և բոլոր այն մինթերմները, որոնք չեն պարզեցվել գործընթացի արդյունքում, կազմում են պարզ իմպլիկանտների ցուցակ:

Նվազարկման աղյուսակային եղանակը ներկայացվում է աղյուսակ 1.7-ով տրված ֆունկցիայի օրինակով: Տրված ֆունկցիան բնորոշող մինթերմներն են.

$$f = \Sigma(0, 1, 2, 8, 9, 12, 13) \quad (1.47)$$

Աղյուսակ 1.7

#	wxyz	f	#	wxyz	f
0	0000	1	8	1000	1
1	0001	1	9	1001	1
2	0010	1	10	1010	0
3	0011	0	11	1011	0
4	0100	0	12	1100	1
5	0101	0	13	1101	1
6	0110	0	14	1110	0
7	0111	0	15	1111	0

Քայլ 1. Խմբավորել երկուական թվերի տեսքով ներկայացված մինթերմները՝ ելնելով 1-երի քանակից, ինչպես ներկայացված է աղյուսակ 1.8 –ի (ա) սյունակում: Այս օրինակում մինթերմները խմբավորվել են հինգ խմբի՝ առաջին խմբում չկա ոչ մի '1', երկրորդ խմբում յուրաքանչյուր մինթերմ պարունակում է միայն մեկ '1', երրորդ, չորրորդ և հինգերորդ խմբերը բաղկացած են համապատասխանաբար երկու, երեք և չորս '1' պարունակող մինթերմներից: Այս մինթերմների տասական համարժեքները ևս ներկայացված են նույնականացման համար:

Քայլ 2. Ցանկացած երկու մինթերմ, որոնք իրարից տարբերվում են միայն մեկ փոփոխականով, կարող են միավորվել (սոսնձվել), իսկ չհամընկնող փոփոխականը՝ կրճատվել: Խմբերից մեկի մինթերմները համեմատվում են միայն հաջորդ (մեկ համարով բարձր) խմբի հետ, քանի որ երկու մինթերմները, որոնք տարբերվում են մեկից ավել բիթերով, չեն կարող սոսնձվել: Առաջին խմբի մինթերմը համեմատվում է երկրորդ խմբի յուրաքանչյուր մինթերմի հետ: Եթե երկու մինթերմ ունեն նույն բիթերը բոլոր կարգերում, բացի մեկից, ապա 'V' նշանն է դրվում երկու մինթերմի աջ կողմում՝ ցույց տալու համար, որ դրանք օգտագործվել են: Միավորման արդյունքը և միավորված մինթերմների տասական համարժեքները ներկայացված են աղյուսակ 1.8-ի (բ) սյունակում: Փոփոխականը, որը կրճատվել է միավորման արդյունքում, փոխարինվում է զծիկով: Օրինակ,  $m_0$  (0000) և  $m_1$  (0001) մինթերմների սոսնձումից կստացվի (000-)

$$m_0 + m_1 = \overline{wxyz} + \overline{wxyz} = \overline{wxy} :$$

Համեմատման գործընթացի վերջնական արդյունքը ներկայացված է չորս խմբով (բ սյունակ):

Քայլ 3. (բ) սյունակի տարրերը բաղկացած են միայն երեք փոփոխականից: Տվյալ կարգում 1-ը նշանակում է, որ այդ փոփոխականն ուղիղ է (չշրջված), 0-ն՝ շրջված, իսկ զծիկը՝ փոփոխականը չի ներառված իմպլիկանտի մեջ: Որոնման և համեմատման գործընթացը կրկնվում է (բ) սյունակի իմպլիկանտների միջև՝ ձևավորելով երկու փոփոխականով իմպլիկանտներ (գ) սյունակում: (000-) մինթերմը չի սոսնձվել, նրա կողքը չի դրվել 'V' նշանը:

Աղյուսակ 1.8

(ա)		(բ)		(գ)	
#	wxyz	#	wxyz	#	wxyz
0	0000 V	0,1	000- V	0,1,8,9	-00-
1	0001 V	0,2	00-0	0,8,1,9	-00-
2	0010 V	0,8	-000 V	8,9,12,13	1-0-
8	1000 V	1,9	-001 V	8,12,9,13	1-0-
9	1001 V	8,9	100- V		
12	1100 V	8,12	1-00 V		
13	1101 V	9,13	1-01 V		
		12,13	110- V		

Քայլ 4. Աղյուսակում 'V' -ով չնշված իմպլիկանտները պարզ իմպլիկանտներ են: Այս օրինակում առկա է  $\overline{w x z}$  (00-0) իմպլիկանտը (բ) սյունակում և  $\overline{x y}$  (-00-) ու  $\overline{w y}$  (1-0-) իմպլիկանտները (գ) սյունակում: Նշենք, որ (գ) սյունակի իմպլիկանտներից յուրաքանչյուրը որոշված է երկու անգամ: Կրկնվող իմպլիկանտներից պետք է վերցնել միայն մեկը: Պարզ իմպլիկանտների տրամաբանական գումարը ֆունկցիայի պարզեցված բանաձևն է՝  $f = \overline{w x z} + \overline{x y} + \overline{w y}$ :

Պարզ իմպլիկանտների ցուցակը այս օրինակի համար ներկայացված է աղյուսակ 1.9 –ում:

Աղյուսակ 1.9

Պարզ իմպլիկանտներ			Մինթերմներ						
			0000	0001	0010	1000	1001	1100	1101
			0	1	2	8	9	12	13
V	$\overline{w x z}$	0,2	X		X				
V	$\overline{x y}$	0,1,8,9	X	X		X	X		
V	$\overline{w y}$	8,9,12,13				X	X	X	X
			V	V	V	V	V	V	V

Պարզ իմպլիկանտներով մինթերմների ծածկումը նշում են 'X' նշաններով համապատասխան սյուներում: Եթե որևէ սյունում կա միայն մեկ 'X', ապա համապատասխան իմպլիկանտը կոչվում է անհրաժեշտ պարզ իմպլիկանտ: Անհրաժեշտ պարզ իմպլիկանտների բազմությունը կազմում է ֆունկցիայի միջուկը: Վերջնական պարզեցված արտահայտության մեջ պետք է ներառել, առաջին հերթին, միջուկը: Եթե միջուկով ծածկվում են ոչ բոլոր մինթերմները, ապա միջուկին պետք է ավելացնել մյուս իմպլիկանտներից, մինչև բոլոր մինթերմները ծածկվեն: Իմպլիկանտների ստացված բազմությունը համապատասխանում է ֆունկցիայի նվազարկված բանաձևին: Աղյուսակում պարզ իմպլիկանտների մոտ դրված ստուգիչ 'V' նշանը ցույց է տալիս, որ դրանք ընտրված են: Այնուհետև ստուգվում է յուրաքանչյուր սյուն, որի մինթերմը ծածկված է ընտրված պարզ իմպլիկանտներով: Ստուգումը նշվում է սյան ներքևում 'V' նշանով: Այսպիսով, ստացված բոլոր պարզ իմպլիկանտները մտնում են ֆունկցիայի միջուկի մեջ և ընտրության անհրաժեշտություն չկա: Նվազարկված ֆունկցիան պարունակում է բոլոր պարզ իմպլիկանտները՝

$$f = \overline{w x z} + \overline{x y} + \overline{w y} \quad (1.48)$$

Սա մասնակի դեպք է, երբ նվազարկված ֆունկցիան պարունակում է միայն միջուկը կազմող անհրաժեշտ պարզ իմպլիկանտներ: Դիտարկենք մեկ այլ օրինակ, որում ֆունկցիան ներկայացվում է հետևյալ մինթերմների ցուցակով՝

$$F = \Sigma(0, 3, 4, 5, 6, 7, 8, 10, 11) \quad (1.49)$$

Բոլոր պարզ իմպլիկանտների որոնման գործընթացը ներկայացված է աղյուսակ 1.10-ում: Վերջնական պարզ իմպլիկանտների ցուցակը բերված է աղյուսակ 1.11-ում: Պարզ իմպլիկանտների աղյուսակի հետազոտումը ցույց է տալիս, որ միջուկը կազմող

իմպլիկանտներն են՝  $\bar{w}x$ ,  $\bar{x}\bar{y}z$ ,  $w\bar{x}y$ : Դրանցով ծածկվում են բոլոր իմպլիկանտները, բացառությամբ 0011-ի: Աղյուսակ 1.11-ից միջուկը կազմող իմպլիկանտները և դրանցով ծածկվող միներմները հեռացնելուց հետո կստացվի աղյուսակ 1.12-ը:  $\bar{w}\bar{y}z$  իմպլիկանտը չի ծածկում 0011 միներմը, և այն միանգամայն ավելորդ է: F ֆունկցիայի պարզ իմպլիկանտների նվազագույն բազմությունն ստանալու համար անհրաժեշտ է միջուկին ավելացնել  $\bar{x}yz$  կամ  $\bar{w}yz$  իմպլիկանտը: Այսպիսով, F ֆունկցիայի համար գոյություն ունի երկու համարժեք նվազագույն ԴՆԶ՝

$$F = \bar{w}x + \bar{x}\bar{y}z + w\bar{x}y + \bar{x}yz, \quad (1.50)$$

$$F = \bar{w}x + \bar{x}\bar{y}z + w\bar{x}y + \bar{w}yz: \quad (1.51)$$

Աղյուսակ 1.10

(ա)			(բ)		(գ)	
#	WXYZ		#	WXYZ	#	WXYZ
0	0000	V	0,4	0-00		
			0,8	-000		
4	0100	V	4,5	010-	4,5,6,7	01—
8	1000	V	4,6	01-0	4,6,5,7	01--
3	0011	V	8,10	10-0		
5	0101	V	3,7	0-11		
6	0110	V	3,11	-011		
10	1010	V	5,7	01-1	V	
			6,7	011-	V	
			10,11	101-		
7	0111	V				
11	1011	V				

Պարզ իմպլիկանտներ աղյուսակից նվազագույն ենթաբազմության առանձնացումը կարելի է կատարել նաև հետևյալ ձևականացված եղանակով. յուրաքանչյուր միներմի վերագրվում է մեկ իդենտիֆիկատոր (աղյուսակ 1.11-ում դրանք նշված են իմպլիկանտների կոդին՝ A, B, C, D, E, F), այնուհետև, միներմների ծածկումների պայմանները գրվում են որպես համապատասխան իմպլիկանտների դիզյունկցիաներ: Բոլոր միներմների միաժամանակյա ծածկման պայմանը կգրվի որպես այդ դիզյունկցիաների կոնյունկցիա: Օրինակ, աղյուսակ 1.11-ից կարելի է գրել.

$$\begin{aligned}
 & (B + C) \& (E + F) \& (A + B) \& A \& A \& (A + F) \& C \& D \& (D + E) = \\
 & = ((B + C) \& (A + B)) \& ((E + F) \& (A + F)) \& A \& C (D \& (D + E)) = \\
 & = (B + AC) \& (F + AE) \& ACD = ACDF + ACDE
 \end{aligned} \quad (1.52)$$

Աղյուսակ 1.11

Պարզ իմպլիկանտներ			Միներմներ								
			0000	0011	0100	0101	0110	0111	1001	1010	1011
			0	3	4	5	6	7	8	10	11
V	$\bar{w}x$ , A	4,5,6,7			X	X	X	X			
	$\bar{w}\bar{y}\bar{z}$ , B	0,4	X		X						
V	$\bar{x}\bar{y}\bar{z}$ , C	0,8	X						X		
V	$w\bar{x}y$ , D	10,11								X	X
	$\bar{x}yz$ , E	3,11		X							X
	$\bar{w}yz$ , F	3,7		X				X			
			V		V	V	V	V	V	V	V

Աղյուսակ 1.12

Պարզ իմպլիկանտներ		Միներմներ
		0011
		3
$\bar{w}\bar{y}\bar{z}$	0,4	
$\bar{x}yz$	3,11	X
$\bar{w}yz$	3,7	X

Ըստ (1..52)-ի, աղյուսակ 1.11-ի նկարագրությանը համապատասխանում է երկու նվազագույն ԴՆՁ՝ կազմված {A, B, C, D, F} և {A, B, C, D, E} իմպլիկանտների ենթաբազմություններից, որոնք համապատասխանում են (1.50) և (1.51) բանաձևերին:

Եթե ֆունկցիան տրված է որևէ ոչ կատարյալ ԴՆՁ-ով, ապա նկարագրված աղյուսակային եղանակով նվազարկելու համար անհրաժեշտ է նախապես այն լրացնել մինչև ԿԴՆՁ: Օրինակ, ենթադրենք ունենք կրճատված ԴՆՁ-ով տրված հետևյալ ֆունկցիան՝

$$F = yz + x\bar{y}z + xyz : \quad (1.53)$$

Սա ԿԴՆՁ չէ, քանի որ  $yz$ -ը երեք փոփոխականի ֆունկցիայի համար միներմ չէ: Ֆունկցիայի ԿԴՆՁ-ն ստանալու համար անհրաժեշտ է  $yz$  իմպլիկանտը փոխարինել  $xyz$  և  $\bar{x}yz$  միներմներով, որոնց սուսնձուսից ստացվում է  $yz$  իմպլիկանտը:

Եթե նվազարկվող ֆունկցիան լրիվ որոշված չէ, ապա արգելված հավաքածուների վրա դրան պետք է վերագրել այնպիսի արժեքներ, որ նվազագույն ԴՆՁ-ի բանաձևն ունենա նվազագույն թվով տառեր՝ լիտերալներ: Որպես օրինակ դիտարկենք աղյուսակ 1.13-ով տրված ոչ լրիվ որոշված ֆունկցիան:

Աղյուսակ 1.13

xyz	F	xyz	F
000	X	100	0
001	0	101	1
010	0	110	1
011	1	111	x

Եթե 000 հավաքածուի վրա ֆունկցիային վերագրենք 1 արժեք, ապա ձեռք կբերենք ավելորդ միներմ, որը չի սոսնձվում մնացած՝ 011, 101, 110 միներմների հետ, հետևաբար ֆունկցիայի նվազագույն ԴՆԶ-ն կբարդանա: Ուստի 000 հավաքածուի վրա նպատակահարմար է ֆունկցիային վերագրել 0 արժեք: Իսկ 111 հավաքածուի վրա նպատակահարմար է ֆունկցիային վերագրել 1 արժեք, քանի որ 111 միներմը սոսնձվում է 011, 101, 110 միներմների հետ, որի հետևանքով ֆունկցիայի նվազագույն ԴՆԶ-ն կլինի՝

$$F = xy + xz + yz: \quad (1.54)$$

Ընդհանուր դեպքում, պետք է փորձել արգելված հավաքածուներին որոշակի արժեքներ վերագրելու բոլոր հնարավոր տարբերակները և յուրաքանչյուր տարբերակի համար կատարել նվազարկման ողջ գործընթացը: Այդ տարբերակներից ստացված նվազագույն ԴՆԶ-երից պետք է ընտրել ամենապարզը: Ակնհայտ է, որ մեծ թվով փոփոխականների դեպքում դա հնարավոր է միայն համակարգչային ծրագրերի միջոցով:

#### 1.4.2. Տրամաբանական ֆունկցիաների բանաձևերի նվազարկման Կառնոյի քարտերի եղանակ

Կառնոյի քարտը երկչափ աղյուսակ է, որի յուրաքանչյուր վանդակ համապատասխանում է ֆունկցիայի մեկ միներմի՝ փոփոխականների մեկ հավաքածուի: Քանի որ տրամաբանական ֆունկցիան կարող է ներկայացվել միներմների գումարով, Կառնոյի քարտում ֆունկցիան կարելի է տալ՝ նշելով ֆունկցիայի կազմի մեջ մտնող միներմները: Քարտը ակնառու դիագրամ է, որից կարելի է ստանալ ֆունկցիայի բոլոր հնարավոր բանաձևերը ԴՆԶ-ով: Օգտագործողը հնարավորություն ունի քարտից ստանալ ֆունկցիայի նվազագույն բանաձևը:

Երկու փոփոխականի ֆունկցիայի համար Կառնոյի քարտը ցույց է տրված նկ. 1.15-ում: Քարտն ունի չորս վանդակ, որոնցից յուրաքանչյուրը համապատասխանում է փոփոխականների մեկ հավաքածուի (միներմի): Նկ. 1.15բ-ում ցույց է տրված կապը քարտի վանդակների և ֆունկցիայի փոփոխականների միջև: Քարտի տողերի և սյուների համար նշված են  $x$  և  $y$  փոփոխականների 0 և 1 արժեքները: Նկատենք, որ  $x$ -ը 1-ով նշված տողի միներմներում մասնակցում է ուղիղ ձևով, իսկ 0-ով նշված տողում՝ ժխտված ձևով:

$m_0$	$m_1$
$m_2$	$m_3$

(ա)

	$y$	0	1
$x$			
0		$\overline{x}y$	$\overline{x}\overline{y}$
1		$x\overline{y}$	$xy$

(բ)

Նկ. 1.15. Երկու փոփոխականի Կառնոյի քարտ

Եթե քարտում 1-ով նշվեն այն վանդակները, որոնց համապատասխանող միներմները մասնակցում են ֆունկցիայի ԿՂՆՁ-ում, ապա կարելի է տեսնել, որ Կառնոյի քարտը տրամաբանական ֆունկցիայի աղյուսակային նկարագրության մեկ այլ եղանակ է: Օրինակ, նկ. 1.16-ում ցույց են տրված  $xy$  և  $x+y$  ֆունկցիաների քարտերը: Քանի որ  $xy$  ֆունկցիան հավասար է 1-ի միայն 11 հավաքածուի վրա՝ կազմված է միայն  $m_3$  միներմից, ապա նկ. 1.16ա-ի քարտում նշված է  $m_3=11$  վանդակը: Նույն եղանակով,  $x+y$  ֆունկցիան նկ. 1.16բ քարտում ներկայացված է երեք միներմով՝

$$x+y = \overline{x}y + x\overline{y} + xy = m_1 + m_2 + m_3$$

	$y$	0	1
$x$			
0			
1			1

(ա)  $xy$ 

	$y$	0	1
$x$			
0			1
1		1	1

(բ)  $x+y$ 

Նկ. 1.16. Տրամաբանական ֆունկցիաների ներկայացումը Կառնոյի քարտերով

Երեք փոփոխականի քարտը ներկայացված է նկ.1.17-ում: Երեք փոփոխականի դեպքում կա ութ միներմ: Հետևաբար աղյուսակը բաղկացած է ութ վանդակից: Նշենք, որ միներմները դասավորված են ոչ թե երկուական հաջորդականությամբ, այլ Գրեյի կոդի հաջորդականությամբ: Այս հաջորդականության առանձնահատկությունն այն է, որ մեկ հավաքածուից հաջորդին անցնելիս միայն մեկ բիթ է փոխվում 1-ից 0 կամ 0-ից 1: Նկ.3.1բ-ում պատկերված քարտի յուրաքանչյուր տող և յուրաքանչյուր սյուն նշված է թվերով, որոնք ցույց են տալիս երեք փոփոխականների և վանդակների միջև կապը: Օրինակ,  $m_5$  –ով նշված վանդակը համապատասխանում է 1 տողին և 01 սյանը: Երբ այս երկու թվերը միավորվում են, ձևավորվում է 101 երկուական թիվը, որի տասական համարժեքը 5 է:



	$m_0$	$m_1$	$m_3$	$m_2$
	$m_4$	$m_5$	$m_7$	$m_6$

(ա)

	$yz$			
$x$	00	01	11	10
0	$\overline{x}yz$	$\overline{x}yz$	$\overline{x}yz$	$\overline{x}yz$
1	$x\overline{y}z$	$x\overline{y}z$	$xyz$	$xy\overline{z}$

(բ)

Նկ. 1.17. Երեք փոփոխականի քարտ

Դիտարկենք հետևյալ ֆունկցիայի պարզեցումը.

$$F = \Sigma(2, 3, 4, 5) = \overline{x}y\overline{z} + \overline{x}yz + x\overline{y}z + x\overline{y}\overline{z} \quad (1.55)$$

Նախ, 1-երով նշենք այն վանդակները, որոնք համապատասխանում են ֆունկցիայի միներմների, ինչպես ցույց է տրված նկ. 1.18-ում: Դրա համար պետք է ձևափոխել յուրաքանչյուր միներմ իր համարժեք երկուական թվին, այնուհետև, համապատասխան վանդակում նշել 1: Հաջորդ քայլն է՝ 1-եր պարունակող հարևան 2, 4 կամ 8 վանդակները ընդգրկել կոնտուրներով: Կոնտուրում պարփակված վանդակներին համապատասխանող միներմները սոսնձվում են:

	$yz$			
$x$	00	01	11	10
0			1	1
1	1	1		

Նկ. 1.18.  $F = \Sigma(2, 3, 4, 5) = \overline{x}y\overline{z} + \overline{x}yz + x\overline{y}z + x\overline{y}\overline{z}$  ֆունկցիայի Կառնոյի քարտը

Օրինակ, վերին աջ կոնտուրը ներառում է 011 և 010 միներմները, որոնց սոսնձումից կստացվի  $\overline{x}y = 01$  իմպլիկանտը: Նույն եղանակով ստորին ձախ կոնտուրը կտա  $x\overline{y}$  իմպլիկանտը: Հարևան երկու վանդակների սոսնձումից կրճատվում է մեկ փոփոխական, այն փոփոխականը, որն այդ վանդակներում ունի տարբեր արժեքներ: Այս երկու արտադրյալների գումարը տալիս է լուծումը՝  $F = \overline{x}y + x\overline{y}$ :

Դիտարկենք պարզեցման մեկ այլ օրինակ.

$$F = \overline{x}yz + x\overline{y}z + xy\overline{z} + xyz \quad (1.56)$$

Այս ֆունկցիայի քարտը ցույց է տրված նկ. 1.19-ում: Չորս վանդակներ պարունակում են 1-եր՝ ֆունկցիայի յուրաքանչյուրը միներմի համար մեկ հատ 1: Երկու հարևան վանդակներ միավորվում (սոսնձվում) են երրորդ սյան մեջ՝ ձևավորելով երկու փոփոխականի  $yz$  արտադրյալը: Մնացած 1-երով երկու վանդակները ևս հարևան են, քանի որ տարբերվում են միայն մեկ փոփոխականի՝  $z$ -ի, մասնակցության ձևով: Նույն տողի կամ սյան ծայրագույն վանդակների սոսնձումը քարտում նշվում է կիսաբաց

կոնտուրի տեսքով: Այս երկու վանդակների սոսնձումից հետո ձևավորվում է երկու փոփոխականով  $x\bar{z}$  արտադրյալ: Պարզեցված ֆունկցիան կստանա հետևյալ տեսքը՝  $F = yz + x\bar{z}$ :

x \ yz	00	01	11	10
0			1	
1	1		1	1

Նկ. 1.19.  $F = \bar{x}yz + x\bar{y}\bar{z} + xy\bar{z} + xyz$  ֆունկցիայի Կառնոյի քարտը

Այժմ դիտարկենք չորս հարևան վանդակի սոսնձման դեպքը երեք փոփոխականի քարտում: Որպես օրինակ դիտարկենք հետևյալ ֆունկցիան.

$$F = \bar{x}yz + \bar{x}y\bar{z} + \bar{x}yz + x\bar{y}z + xyz : \quad (1.57)$$

Ֆունկցիայի պարզեցման քարտը ներկայացված է նկ. 1.20-ում: Բերված ֆունկցիան պարունակում է հինգ միմբերմ, այսինքն՝ հինգ վանդակներ պարունակում են 1-եր: Կենտրոնի չորս վանդակի համատեղ սոսնձումից կկրճատվեն փոփոխականներից երկուսը՝ x, y, կմնա միայն z փոփոխականը: Մնացած 010 վանդակի 1-ը սոսնձվում է հարևան վանդակի հետ, որն արդեն մեկ անգամ օգտագործվել էր: Դա թույլատրելի և ցանկալի է, քանի որ երկու վանդակի սոսնձումը տալիս է  $\bar{x}y$  արտադրյալը, մինչդեռ 010 վանդակին համապատասխանող միմբերմը երեք փոփոխականների արտադրյալ է՝  $\bar{x}yz$ : Պարզեցված ֆունկցիան հետևյալն է՝  $F = z + \bar{x}y$ :

x \ yz	00	01	11	10
0		1	1	1
1		1	1	

Նկ. 1.20.  $F = m_1 + m_2 + m_3 + m_5 + m_7 = \bar{x}yz + \bar{x}y\bar{z} + \bar{x}yz + x\bar{y}z + xyz$  ֆունկցիայի Կառնոյի քարտը

Դիտարկենք մեկ այլ օրինակ.

$$F(x, y, z) = \Sigma(0, 2, 4, 5, 6) : \quad (1.58)$$

Համապատասխան քարտը ներկայացված է նկ. 1.21-ում: Քարտից որոշվում է հետևյալ պարզեցված ֆունկցիան՝  $F = \bar{z} + x\bar{y}$ .

x \ yz	00	01	11	10
0	1			1
1	1	1		1

Նկ. 1.21.  $F = \Sigma(0, 2, 4, 5, 6)$  ֆունկցիայի Կառնոյի քարտը

Չորս փոփոխականի բուլյան ֆունկցիայի նվազարկումը կատարվում է նույն մեթոդով, ինչ երեք փոփոխականի ֆունկցիայի համար:

Դիտարկենք չորս փոփոխականով հետևյալ ֆունկցիայի պարզեցումը.

$$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14): \quad (1.59)$$

Ներկայացված միջթերմները նշված են «1»-երով Նկ. 1.22-ում բերված քարտում: 1-երով նշված ութ հարևան վանդակը կարող են սոսնձվել՝ ձևավորելով  $\bar{y}$  իմպլիկանտը: Որթ վանդակի սոսնձման արդյունքում կրճատվում է երեք փոփոխական:

wx \ yz	00	01	11	10
00	1	1		1
01	1	1		1
11	1	1		1
10	1	1		

Նկ. 1.22.  $F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$  ֆունկցիայի Կառնոյի քարտը

Նշենք, որ 4 փոփոխականի քարտում կարող են մեկ կոնտուրով սոսնձվել միայն 2, 4, 8 կամ 16 վանդակներ: Որքան մեծ է սոսնձվող վանդակների քանակը, այնքան փոքր է փոփոխականների քանակը արտադրյալում: Այս օրինակում վերին աջ կողմի երկու 1-երը սոսնձվում են վերին ձախ կողմի երկու 1-երի հետ՝ ստանալով  $\bar{w} \bar{z}$  արտադրյալը: Թույլատրելի է օգտագործել նույն վանդակը մեկից ավելի անգամներ: Այժմ մնացել է միայն մեկ 1 երրորդ տողում և չորրորդ սյունակում (1110 վանդակ): Այս վանդակը առանձին վերցնելու փոխարեն (որի արդյունքում կմնար չորս փոփոխականով արտադրյալ) այն սոսնձվում է արդեն օգտագործված չորս հարևան վանդակի հետ: Այս վանդակները ներառում են երկու միջին տողերը և երկու ծայրերի սյուները՝ ձևավորելով  $x \bar{z}$  արտադրյալը: Պարզեցված ֆունկցիան կլինի՝  $F = \bar{y} + \bar{w} \bar{z} + x \bar{z}$ :

Հինգ փոփոխականների քարտի համար անհրաժեշտ է 32 վանդակ, վեց փոփոխականի համար՝ 64 վանդակ: Յոթ և ավելի փոփոխականների համար անհրաժեշտ են ավելի մեծ թվով վանդակներ, քարտը խիստ մեծանում է. իսկ դրա գործնական կիրառությունը՝ խիստ բարդանում:

Հինգ և վեց փոփոխականների քարտերը ներկայացված է նկ. 1.23 և 1.24-ում: Տողերը և սյուները համարակալված են Գրեյի կոդի հաջորդականությամբ:

cde		00	001	011	010	110	111	101	100
ab	00	m0	m1	m3	m2	m6	m7	m5	m4
	01	m8	m9	m11	m10	m14	m15	m13	m12
	11	m24	m25	m27	m26	m30	m31	m29	m28
	10	m16	m17	m19	m18	m22	m23	m21	m20

Նկ. 1.23. Հինգ փոփոխականներով ֆունկցիայի Կառնոյի քարտը

Հինգ փոփոխականի քարտը կարելի է դիտարկել որպես երկու չորս փոփոխականի քարտերի միավորում: Այս չորս փոփոխականի քարտերը միմյանցից տարբերվում են միայն *c* փոփոխականի արժեքով: Յուրաքանչյուր վանդակ հարևան է ոչ միայն իր կողքի չորս վանդակին, այլ նաև կրկնակի գծով ցույց տրված առանցքի նկատմամբ իրեն սիմետրիկ վանդակին: Օրինակ, m31 միներմի հարևաններն են m30, m15, m29, m23 և m27: Այդ նույն միներմը վեց փոփոխականի քարտում հարևան է այդ բոլոր միներմներին՝ ավելացրած նաև m63 միներմը:

def		00	001	011	010	110	111	101	100
abc	00	m0	m1	m3	m2	m6	m7	m5	m4
	001	m8	m9	m11	m10	m14	m15	m13	m12
	011	m24	m25	m27	m26	m30	m31	m29	m28
	010	m16	m17	m19	m18	m22	m23	m21	m20
110	110	m48	m49	m51	m50	m54	m55	m53	m52
	111	m56	m57	m59	m58	m62	m63	m61	m60
	101	m40	m41	m43	m42	m46	m47	m45	m44
	100	m32	m33	m35	m34	m38	m39	m37	m36

Նկ. 1.24. Վեց փոփոխականներով ֆունկցիայի Կառնոյի քարտը

Դիտարկենք հինգ փոփոխականներով ֆունկցիայի պարզեցման հետևյալ օրինակը.

$$F = \Sigma(0, 2, 4, 6, 9, 11, 13, 15, 17, 21, 25, 27, 29, 31): \quad (1.60)$$

Այս ֆունկցիայի քարտը բերված է նկ. 1.25-ում:

	cde	00	001	011	010	110	111	101	100
ab									
00		1			1	1			1
01			1	1			1	1	
11			1	1			1	1	
10			1					1	

Նկ. 1.25.  $F = \Sigma(0, 2, 4, 6, 9, 11, 13, 15, 17, 21, 25, 27, 29, 31)$  ֆունկցիայի Կառնոյի քարտը

Պարզեցված ֆունկցիայի բանաձևն է՝

$$F = be + ade + \overline{abe} : \quad (1.61)$$

Եթե նվազարկվող ֆունկցիան լրիվ որոշված չէ, ապա նրան արգելված հավաքածուների վրա պետք է վերագրել այնպիսի արժեքներ, որ նվազագույն ԴՆԶ-ի բանաձևը լինի պարզագույնը: Մենք չենք անհանգստանում, թե ի՞նչ պետք է լինի ֆունկցիայի ելքում փոփոխականների այս հավաքածուների համար, քանի որ երաշխավորված է, որ դրանք չեն հանդիպելու: Այս անորոշ վիճակները կարող են օգտագործվել քարտում՝ ապահովելով ֆունկցիայի հետագա պարզեցումը: Քարտում այս հավաքածուներին համապատասխանող վանդակներում չի կարելի նշել 1, քանի որ դա կնշանակի, որ ֆունկցիան ընդունում է 1 արժեք մուտքային այդ հավաքածուների վրա: Նմանապես, 0-ի տեղադրումը ևս կնշանակի, որ ֆունկցիան ընդունում է 0 արժեք: 1-երից և 0-ներից տարբերելու համար, անորոշ վիճակը քարտում նշվում է X-ով:

Երբ ֆունկցիայի պարզեցման համար քարտում ընտրվում են սոսնձվող հարևան վանդակները, X-երը կարող են համարվել 0 կամ 1, կախված, թե դրանցից որը կբերի արտահայտության պարզեցմանը: Բացի այդ, X-երը կարող են ընդհանրապես չօգտագործվել, եթե դրանք չեն օժանդակում ավելի մեծ տիրույթների ծածկմանը: Ցանկացած դեպքում ընտրությունը կախված է միայն պարզեցումից, որին հնարավոր է հասնել:

Դիտարկենք մեկ օրինակ՝

$$F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15, 0^*, 2^*, 5^*): \quad (1.62)$$

1, 3, 7, 11, 15 հավաքածուները համապատասխանում են մինթերմներին, իսկ «\*»-ով նշված հավաքածուները՝ անորոշ վիճակներին: Նվազարկումը ներկայացված է նկ. 1.26-ում: 1-երը և X-երը սոսնձվում են որևէ հարմար եղանակով այնպես, որ ներգրավվեն առավելագույն թվով հարևան վանդակներ: Կարիք չկա ներառել բոլոր X-երը,

Ներառվում են միայն արտադրյալների պարզեցմանը օժանդակող վանդակները: Տվյալ օրինակում նպատակահարմար է ընդգրկել մեկ X և բաց թողնել մյուս երկուսը: Արդյունաբար պարզեցված ֆունկցիան հետևյալն է.  $F = \bar{w}z + yz$ .

wx \ yz	00	01	11	10
00	X	1	1	X
01		X	1	
11			1	
10			1	

Նկ. 1.26.  $F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15, 0^*, 2^*, 5^*)$  ֆունկցիայի Կառնոյի քարտը

Ելնելով հարևան վանդակների սոսնձման հատկությունից՝ կարելի է նկատել, որ  $n$ -փոփոխականի քարտում հարևան  $2^k$  վանդակների սոսնձումից ստացվում է  $n-k$  փոփոխականներից կազմված իմպլիկանտ: Երբ քարտի բոլոր վանդակները սոսնձվում են՝  $n=k$ , արդյունքում կստացվի 1 հաստատունի ֆունկցիա՝  $F=1$ :

Եթե սոսնձումներից ստացված իմպլիկանտները չեն պարունակում ավելորդ ծածկումներ, արդյունքում ստացված պարզեցված բանաձևը կլինի որևէ փակուղային ԴՆՁ, որից հնարավոր չէ դնել զգել որևէ իմպլիկանտ: Նվազագույն ԴՆՁ ստանալու համար սոսնձումները պետք է կատարել այնպես, որ ստացված իմպլիկանտները ամենաարդյունավետ կերպով ծածկեն ֆունկցիայի միներմները: Սոսնձումների նման ընտրությունը միշտ չէ, որ ակնհայտ է: Հետևաբար, Կառնոյի քարտերի եղանակը համակարգված չէ՝ պարզեցման արդյունքը շատ բանով կախված է կատարողի էվրիստիկ ենթադրություններից: Սակայն Կառնոյի քարտերի եղանակը չափազանց հարմար է ձեռքով աշխատելու համար՝ 5-6-ից ոչ ավել փոփոխականների դեպքում:

## 1.5. Խնդիրներ

**Խնդիր 1.1.** Հետևյալ տրամաբանական ֆունկցիաների համար կառուցել իսկության աղյուսակները.

ա)  $xyz + \bar{x}\bar{y}z$ , բ)  $x(yz + \bar{x}y)$ , գ)  $xyz + x\bar{y}z + \bar{x}y\bar{z}$ , դ)  $(x+y)(x+z)(\bar{x} + z)$ :

**Խնդիր 1.2.** Օգտվելով դե Մորգանի բանաձևերից գրել  $\bar{F}$  ֆունկցիայի բանաձևը.

ա)  $F(x, y, z) = x(\bar{y} + z)$ ,

բ)  $F(x, y, z) = xy + \bar{x}z + x\bar{z}$ ,

գ)  $F(w, x, y, z) = xyz(\bar{y}z + \bar{x}) + (\bar{w}yz + \bar{x})$ :

**Խնդիր 1.3.** Ցույց տալ, որ՝  $xz = (x + y)(x + \bar{y})(\bar{x} + z)$ ,

ա) օգտվելով իսկության աղյուսակից,

բ) օգտվելով բուլյան ֆունկցիաների հատկություններից:

**Խնդիր 1.4.** Պարզեցնել հետևյալ բանաձևերը՝ օգտվելով բուլյան ֆունկցիաների հատկություններից: Թվարկել յուրաքանչյուր քայլում օգտագործվող հատկությունները

ա)  $F(x, y, z) = \bar{x}y + xy\bar{z} + xyz$ ,

բ)  $F(w, x, y, z) = (x\bar{y} + \bar{w}z)(w\bar{x} + y\bar{z})$ ,

գ)  $F(x, y, z) = \overline{(x + y)(\bar{x} + \bar{y})}$ :

**Խնդիր 1.5.** Պարզեցնել հետևյալ բանաձևերը՝ օգտվելով բուլյան ֆունկցիաների հատկություններից: Թվարկել յուրաքանչյուր քայլում օգտագործվող հատկությունները

ա)  $(ab + c + df)ef$ ,

բ)  $x + xy$ ,

գ)  $(x\bar{y} + \bar{x}z)(w\bar{x} + y\bar{z})$ ,

դ)  $xy + x\bar{y}$ ,

ե)  $\bar{x}yz + xz$ ,

զ)  $wx + w(xy + y\bar{z})$ :

**Խնդիր 1.6.** Ճշմարտացի՞ է արդյոք հետևյալ նույնությունը: Պատասխանը հիմնավորեք:  
 $yz + xy\bar{z} + \bar{x}\bar{y}z = xy + \bar{x}z$  :

**Խնդիր 1.7.** Օգտվելով բուլյան ֆունկցիաների հատկություններից՝ ցույց տալ, որ:

ա)  $x(\bar{x} + y) = xy$ ,

բ)  $x + \bar{x}y = x + y$ ,

գ)  $xy + \bar{x}z + yz = xy + \bar{x}z$  :

**Խնդիր 1.8.** Օգտվելով բուլյան ֆունկցիաների հատկություններից ցույց տալ, թե հետևյալ նույնություններից որն է ճիշտ, որը ոչ.

ա)  $\bar{a}c + ab\bar{c} + \bar{a}b + a\bar{b} = \bar{b}c + a\bar{c} + b\bar{c} + \bar{a}bc$ ,

բ)  $a\bar{c} + bc + \bar{b}\bar{c} = (a + \bar{b} + c)(a + b + \bar{c})(\bar{a} + b + \bar{c})$ ,

գ)  $(a + c)(\bar{a} + \bar{b} + \bar{c})(\bar{a} + b) = (a + b)(b + c)(\bar{a} + \bar{c})$  :

**Խնդիր 1.9.** Աղյուսակ 1.14-ում բերված իսկության աղյուսակից ստանալ բուլյան ֆունկցիայի բանաձևը՝ արտադրյալների գումարի տեսքով:

Աղյուսակ 1.14

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

**Խնդիր 1.10.** Աղյուսակ 1.15-ում բերված իսկության աղյուսակից ստանալ բուլյան ֆունկցիայի բանաձևը՝ գումարների արտադրյալի տեսքով:

Աղյուսակ 1.15

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

**Խնդիր 1.11.** Հետևյալ բանաձևը ներկայացնել իբրև երկու արտադրյալների գումարի ժխտում.  $x\bar{z} + \bar{y}z + \bar{x}y$  :

**Խնդիր 1.12.** Տրված  $F(x, y, z) = \bar{x}y + xy\bar{z}$  բուլյան ֆունկցիայի համար՝

ա) ստանալ ժխտված ֆունկցիայի բանաձևը՝ արտադրյալների գումարի տեսքով,

բ) ցույց տալ, որ՝  $F\bar{F} = 0$ ,

գ) ցույց տալ, որ՝  $F + \bar{F} = 1$ :

**Խնդիր 1.13.** Օգտվելով հանրահաշվական ձևափոխություններից՝ երեք փոփոխականի համար ցույց տալ.

$$\sum m(1,2,3,4,5,6,7) = a + b + c :$$

**Խնդիր 1.14.** Օգտվելով հանրահաշվական ձևափոխություններից երեք փոփոխականի համար ցույց տալ.

$$\prod M(0,1,2,3,4,5,6) = abc :$$

**Խնդիր 1.15.** Հետևյալ քարտերից դուրս գրել պարզեցված ֆունկցիաների բանաձևերը:

yz x 0 00 01 11 10 1 1 0 0 1	yz x 0 00 01 11 10 1 1 0 0 0	yz x 0 00 01 11 10 1 1 1 1 1
---------------------------------------	---------------------------------------	---------------------------------------

(ա)

(բ)

(գ)

Նկ. 1.27. Խնդիր 1.15-ին վերաբերող քարտերը

**Խնդիր 1.16.** Հանրահաշվորեն ցույց տալ, թե ինչպես հետևյալ քարտի չորս նշված անդամներ պարզեցվելով բերվում են մեկ անդամի:

yz x 0 00 01 11 10 1 1 1 1 1
---------------------------------------

Նկ. 1.28. Խնդիր 1.16-ին վերաբերվող քարտը



**Խնդիր 1.17.** Հետևյալ քարտերից դուրս գրել պարզեցված ֆունկցիաների բանաձևերը:

$\backslash yz$ $wx$	00	01	11	10
00	1	1	1	1
01		x	1	x
11	1		x	
10	1		x	1

(ա)

$\backslash yz$ $x$	00	01	11	10
0	1	1		x
1	x	1	1	1

(բ)

Նկ. 1.29. Խնդիր 1.17-ին վերաբերող քարտերը

**Խնդիր 1.18.** Հետևյալ քարտերից դուրս գրել պարզեցված ֆունկցիաների բանաձևերը:

$\backslash yz$ $wx$	00	01	11	10
00	1			1
01	1			1
11			1	
10	1		1	

(ա)

$\backslash yz$ $wx$	00	01	11	10
00		1		1
01		1	1	1
11	1	1		
10	1	1		1

(բ)

$\backslash yz$ $wx$	00	01	11	10
00	1	1	1	1
01			1	1
11	1	1	1	1
10	1			1

(գ)

Նկ. 1.30. Խնդիր 1.18-ին վերաբերող քարտերը

**Խնդիր 1.19.** Ստուգել արդյոք նազագու՞յնն է հետևյալ ԴՆՁ-ն՝

$$F(t, u, v, w, x, y, z) = \bar{t}uvwx + \bar{t}u\bar{v}xz + \bar{t}uwx\bar{y}z :$$

**Խնդիր 1.20.** Նվազարկել հետևյալ ֆունկցիաների բանաձևերը Կառնոյի քարտերով: Ֆունկցիայի 1-երը լրացնել քարտում միանգամից տրված արտահայտություններից՝ առանց իսկության աղյուսակ կառուցելու.

ա)  $F(x, y, z) = \bar{x}z + xy + x\bar{y}z ,$

բ)  $F(w, x, y, z) = \bar{w}\bar{y}z + \bar{x}yz + w\bar{y}z + xyz ,$

գ)  $F(w, x, y, z) = wx\bar{z} + w\bar{x}yz + xz ,$

դ)  $F(w, x, y, z) = wx\bar{y}\bar{z} + \bar{w}x\bar{y} + wxz + \bar{w}yz + xy\bar{z} :$

## Գլուխ 2. Համակցական տրամաբանական շղթաներ

## 2.1. Համակցական շղթաներ

Թվային համակարգերում տրամաբանական շղթաները կարող են լինել համակցական կամ հաջորդական: Համակցական շղթաները կառուցվում են տրամաբանական տարրերից և դրանց ելքերը ժամանակի ցանկացած պահի միարժեք որոշվում են մուտքային փոփոխականների արժեքներով ժամանակի այդ նույն պահին, անկախ դրանց արժեքներից ժամանակի նախորդ պահերին: Որևէ համակցական շղթա իրագործում է տվյալների մշական որոշակի գործողություն, որն ամբողջությամբ նկարագրվում է բուլյան ֆունկցիաներով: Հաջորդական տրամաբանական շղթաները տրամաբանական տարրերի հետ միասին պարունակում են նաև հիշողության տարրեր (երկուական բջիջներ): Հիշողության տարրերի վիճակները կախված են մուտքերի արժեքներից ժամանակի նախորդ պահերին: Որպես հետևանք հաջորդական շղթաների ելքերը կախված են մուտքերի արժեքներից ոչ միայն տվյալ պահին, այլ նաև նախորդ պահերին:

Համակցական շղթան նկարագրվում է մուտքային փոփոխականներով, տրամաբանական տարրերով և ելքային փոփոխականներով: Մուտքային փոփոխականները կիրառվում են տրամաբանական տարրերի մուտքերին, իսկ դրանց ելքերում գեներացվում են անհրաժեշտ ելքային ազդանշանները: Մուտքային և ելքային տվյալները ներկայացվում են երկուական ազդանշաններով:

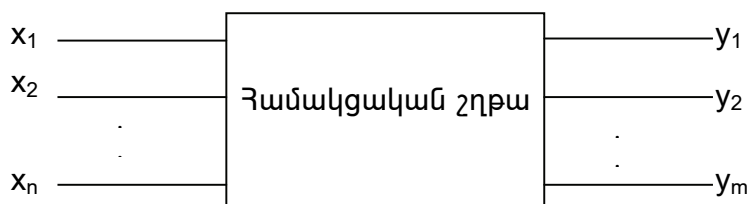
Համակցական շղթայի բլոկ սխեման ցույց է տրված նկ. 2.1-ում: Մուտքային ո փոփոխականները տրվում են արտաքին աղբյուրից, ելքային ո փոփոխականները տրվում են սպառիչին: Շատ կիրառություններում աղբյուրը և/կամ սպառիչը տվյալների պահպանման ռեգիստրներ են: ո մուտքային փոփոխականների դեպքում հնարավոր են 2<sup>ո</sup> հավաքածու: Համակցական շղթան նկարագրվում է ո բուլյան ֆունկցիաներով՝ յուրաքանչյուր ելք՝ մեկ ֆունկցիայով: Յուրաքանչյուր ելքային ֆունկցիա կախված է ո փոփոխականից:

$$y_1 = f_1(x_1, x_2, \dots, x_n)$$

$$y_2 = f_2(x_1, x_2, \dots, x_n) \quad (2.1)$$

.....

$$y_m = f_m(x_1, x_2, \dots, x_n)$$



Նկ. 2.1. Համակցական շղթայի բյուկ սխեման

Համակցական շղթայի յուրաքանչյուր մուտքային փոփոխական կարող է տրվել մեկ կամ երկու ազդանշանային գծով: Երբ առկա է միայն մեկ գիծ, այն կարող է կրել փոփոխականի միայն ուղիղ (չժխտված) կամ ժխտված ձևը: Քանի որ բուլյան արտահայտության մեջ փոփոխականը կարող է մասնակցել ուղիղ և ժխտված ձևով, ապա յուրաքանչյուր փոփոխականի համար անհրաժեշտ է ունենալ շրջիչ՝ մուտքային գծում փոփոխականի բացակայող ձևի համար: Մյուս կողմից՝ մուտքային փոփոխականը կարող է տրվել երկու գծով՝ ազդանշանի ուղիղ և ժխտված ձևերի համար: Այս դեպքում մուտքային փոփոխականների համար շրջիչի կիրառման անհրաժեշտությունն չկա: Թվային համակարգերում օգտագործվող հիմնական երկուական բջիջները տրիգերներն են, որոնք ունեն ուղիղ և ժխտված ելքեր: Հետագայում կհամարենք, որ համակցական շղթաների մուտքերում առկա են փոփոխականների և՛ ուղիղ, և՛ ժխտված ձևերը: Պետք է նկատի ունենալ, որ անհրաժեշտության դեպքում միշտ էլ կարելի օգտագործել շրջիչ տարր՝ ազդանշանի արժեքը շրջելու համար:

## 2.2. Համակցական տրամաբանական շղթաների սինթեզ

Համակցական շղթաների նախագծումը, որը կոչվում է նաև սինթեզ, սկսվում է խնդրի բառացի նկարագրությունից և ավարտվում է տրամաբանական շղթայի սխեմայով կամ բուլյան ֆունկցիաների համակարգով, որից կարող է կառուցվել կամ ծրագրավորվել (երբ օգտագործվում է ծրագրավորվող տրամաբանական սարք՝ ԾՏՍ) տրամաբանական շղթայի սխեման: Սինթեզի գործընթացը բաղկացած է հետևյալ քայլերից.

1. Նկարագրել խնդիրը:
2. Որոշել մուտքային և ելքային փոփոխականների թիվը:
3. Մուտքային և ելքային փոփոխականներին տալ սիմվոլիկ անվանումներ:
4. Կազմել մուտքերի և ելքերի տրամաբանական կապը նկարագրող իսկության աղյուսակը:
5. Նվազարկել իսկության աղյուսակով որոշվող բուլյան ֆունկցիաները (հիմնականում ԴՆԶ-ով):
6. 5-րդ կետում ստացված բուլյան արտահայտությունները ձևափոխել տրված տիպերի տրամաբանական տարրերի միջոցով սխեման իրականացնելու համար հարմար տեսքի:
7. Կառուցել համակցական շղթայի սխեման:

Բերված գործընթացի 1-ից 4 քայլերը կազմում են վերացական համադրման (սինթեզի) փուլը, իսկ 5-ից 7 քայլերը՝ կառուցվածքային համադրման փուլը:

Իսկության աղյուսակի մուտքային  $n$  փոփոխականների սյուներում գրվում են հնարավոր  $2^n$  հավաքածուները: Ելքային փոփոխականների արժեքները որոշվում են խնդրի նկարագրությունից: Յուրաքանչյուր մուտքային հավաքածուի համար ելքային փոփոխականը կարող է ունենալ 1 կամ 0 արժեք: Սակայն հնարավոր է, որ մուտքային որոշ հավաքածուներ խնդրի նկարագրությունում իմաստ չունենան՝ դրանք արգելված հավաքածուներ են: Կարևոր է, որ իսկության աղյուսակում ներկայացված ֆունկցիաները ճշգրիտ համապատասխանեն խնդրի նկարագրությանը: Խնդրի բառացի նկարագրությունը հազվադեպ է լինում լիարժեք և ճշգրիտ: Ցանկացած անճշտություն կամ

բացթողում խնդրի նկարագրության մեջ կհանգեցնի տրամաբանական շղթայի, որը չի իրագործի մտահղացված ֆունկցիաները: Բոլոր դեպքերում, երբ բառացի նկարագրությունում հայտնաբերվում են երկիմաստություններ կամ բացթողումներ, պետք է վերադառնալ խնդրի պայմանների վերաիմաստավորմանը և խմբագրմանը: Շատ դեպքերում նախագծողն առաջնորդվում է իր բնագոյով և նախորդ նախագծերից կուտակված փորձով:

Դիտարկենք համակցական շղթաների նախագծման և սխեմաների կառուցման օրինակներ:

**Օրինակ 2.1.** Նախագծել օդափոխանակության պարզ համակարգի կառավարման տրամաբանական շղթան: Աշխատավայրի օդափոխանակությունը իրականացվում է երկու՝ ցածր և բարձր հզորության օդափոխիչներով: Կառավարման խնդրի բառացի նկարագրությունը հետևյալն է.

-եթե ջերմաստիճանը աշխատավայրում  $20^{\circ}\text{C}$ -ից ցածր է, երկու օդափոխիչներն էլ պետք է անջատված լինեն,

-եթե ջերմաստիճանը աշխատավայրում բարձր է կամ հավասար  $20^{\circ}\text{C}$ -ի, և միաժամանակ ցածր է  $25^{\circ}\text{C}$ -ից, ցածր հզորության օդափոխիչը պետք է միացված լինի, իսկ բարձր հզորությանը՝ անջատված,

-եթե ջերմաստիճանը աշխատավայրում բարձր է կամ հավասար  $25^{\circ}\text{C}$ -ի, և միաժամանակ ցածր է  $30^{\circ}\text{C}$ -ից, բարձր հզորության օդափոխիչը պետք է միացված լինի, իսկ ցածր հզորությանը՝ անջատված,

-եթե ջերմաստիճանը աշխատավայրում բարձր է կամ հավասար  $30^{\circ}\text{C}$ -ի, և միաժամանակ ցածր է  $35^{\circ}\text{C}$ -ից, երկու օդափոխիչներն էլ պետք է միացված լինեն,

-եթե ջերմաստիճանը աշխատավայրում բարձր է  $35^{\circ}\text{C}$ -ից, պետք է միացվի տազնապի ազդանշան:

Ենթադրենք տարածքում տեղադրված են ջերմաստիճանի չափման զգայուն անալոգային սարք և անալոգ-թիվ կերպափոխիչ (ԱԹԿ), որի  $x_3$ ,  $x_2$ ,  $x_1$  երկուական ելքերում ստացվում է ջերմաստիճանի կարգավորման միջակայքերի կոդը, որը ցույց է տրված աղյուսակ 2.1-ում: Քանի որ առկա են ջերմաստիճանի կարգավորման 5 միջակայքեր, ուստի դրանց կոդավորման համար պետք է ունենալ նվազագույնը 3 երկուական փոփոխականներ՝  $\lceil \log_2 5 \rceil = 3$ , յալ նշանակում է  $a$ -ից ոչ փոքր ամբողջ թիվ:

Ըստ աղյուսակի, խնդրի բառացի նկարագրությունը լրիվ չէ՝ օդափոխիչների վիճակը որոշված չէ, երբ ջերմաստիճանը աշխատավայրում գերազանցում է  $35^{\circ}\text{C}$ : Ենթադրենք՝ խնդրի պայմանների վերագնահատումից հետո որոշվում է օդափոխիչները թողնել միացված, երբ ջերմաստիճանը աշխատավայրում գերազանցում է  $35^{\circ}\text{C}$ : Ելքային կառավարման ֆունկցիաներին վերագրենք հետևյալ սիմվոլները՝  $y_1$  բարձր հզորության օդափոխիչի միացման ֆունկցիան,  $y_2$  ցածր հզորության օդափոխիչի միացման ֆունկցիան,  $y_3$  տազնապի ազդանշանի միացման ֆունկցիան: Կառավարման համակարգի ելքային ֆունկցիաների իսկությունը ցույց է տրված աղյուսակ 2.2-ում, 101, 110, 111 հավաքածուները երբեք չեն գեներացվում, հետևաբար, դրանք արգելված հավաքածուներ են:

Աղյուսակ 2.1

Ջերմաստիճան, °C	$x_3$ $x_2$ $x_1$	Բարձր հզորության օդափոխիչ	Ցածր հզորության օդափոխիչ	Տազնապի ազդանշան
$T < 20\text{ }^{\circ}\text{C}$	0 0 0	անջատված	անջատված	անջատված
$20\text{ }^{\circ}\text{C} \leq T < 25\text{ }^{\circ}\text{C}$	0 0 1	անջատված	միացված	անջատված
$25\text{ }^{\circ}\text{C} \leq T < 30\text{ }^{\circ}\text{C}$	0 1 0	միացված	անջատված	անջատված
$30\text{ }^{\circ}\text{C} \leq T < 35\text{ }^{\circ}\text{C}$	0 1 1	միացված	միացված	անջատված
$35\text{ }^{\circ}\text{C} \leq T$	1 0 0	որոշված չէ	որոշված չէ	միացված

Աղյուսակ 2.2

$x_3$ $x_2$ $x_1$	$y_1$	$y_2$	$y_3$
0 0 0	0	0	0
0 0 1	0	1	0
0 1 0	1	0	0
0 1 1	1	1	0
1 0 0	1	1	1
1 0 1	x	x	x
1 1 0	x	x	x
1 1 1	x	x	x

Այսպիսով, կատարվեցին վերացական համադրման բոլոր չորս քայլերը: Հաջորդը, անհրաժեշտ է նվազարկել աղյուսակ 2.2-ի բուլյան ֆունկցիաները: Այդ ֆունկցիաների նվազարկման Կառնոյի քարտերը ցույց են տրված նկ. 2.2-ում: Նվազարկումից ստացված արտահայտությունները հետևյալն են.

$$y_1 = x_3 + x_2 ,$$

$$y_2 = x_1 + x_3 ,$$

$$y_3 = x_3 :$$

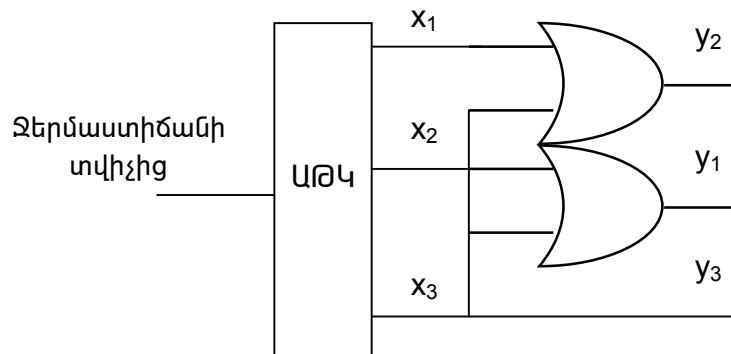
Այս արտահայտություններով կառուցված սխեման ցույց է տրված նկ. 2.3-ում:

$y_1$	$x_2 x_1$	00	01	11	10
$x_3$					
0				1	1
1		1	X	X	X

$y_2$	$x_2 x_1$	00	01	11	10
$x_3$					
0			1	1	
1		1	X	X	X

$y_3$	$x_2 x_1$	00	01	11	10
$x_3$					
0					
1		1	X	X	X

Նկ. 2.2. Օրինակ 2.1-ի կառավարման ֆունկցիաների Կառնոյի քարտերը



Նկ. 2.3. Կառավարման համակարգի տրամաբանական սխեման

Նվազարկված ֆունկցիաները հիմնականում ներկայացվում են ԴՆԶ-ով (արտադրյալների գումարի ձևով), որը հեշտությամբ իրականացվում է ԵՎ, ԿԱՄ, ՈՉ բազիսում: Շատ դեպքերում սխեմայի իրագործման համար առկա տրամաբանական տարրերի տիպերը կարող են սահմանափակված լինել: Նման դեպքերում նվազարկումից ստացված ԴՆԶ-ն պետք է ձևափոխել տրված տարրային բազիսում իրականացման համար հարմար տեսքի: Դրա համար կարելի է օգտվել տրամաբանական արտահայտությունների համարժեք ձևափոխություններից:

ԵՎ-ՈՉ տարրային բազիսում տրամաբանական շղթաների իրականացման համար նվազարկված ԴՆԶ-ն պետք է դե Մորգանի բանաձևերով ձևափոխել հետևյալ կերպ.

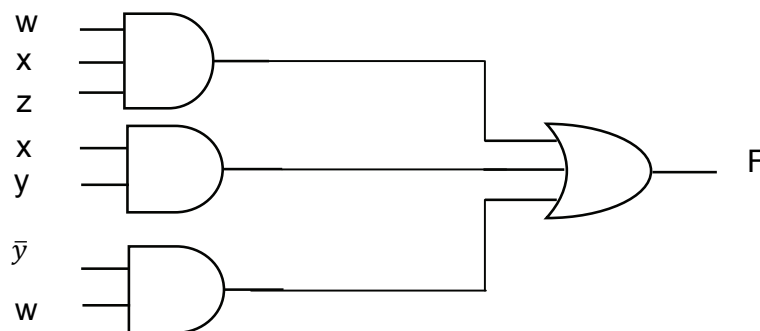
$$y = P_1 + P_2 + \dots + P_r = \overline{\overline{P_1} \cdot \overline{P_2} \dots \overline{P_r}} \quad (2.2)$$

որտեղ՝  $P_1, P_2, \dots, P_r$  մուտքային փոփոխականների ուղիղ կամ ժխտված ձևերից կազմված արտադրյալներ են: (2.2)-ի յուրաքանչյուր  $\overline{P_i}$  ժխտված արտադրյալ կարող է իրականացվել ԵՎ-ՈՉ տարրով, իսկ այդ տարրերի ելքերը պետք է միացվեն  $r$ -մուտք ԵՎ-ՈՉ տարրի մուտքերին: Այս  $r$ -մուտք ԵՎ-ՈՉ տարրի ելքը կլինի կառուցվող տրամաբանական շղթայի ելքը:

**Օրինակ 2.2.** Նվազարկված բուլյան ֆունկցիան տրվում է հետևյալ ԴՆԶ-ով՝

$$F = wxz + xy + w\bar{y} \quad (2.3)$$

ԵՎ, ԿԱՄ, ՈՉ բազիսում իրականացված սխեման ցույց է տրված նկ. 2.4-ում:



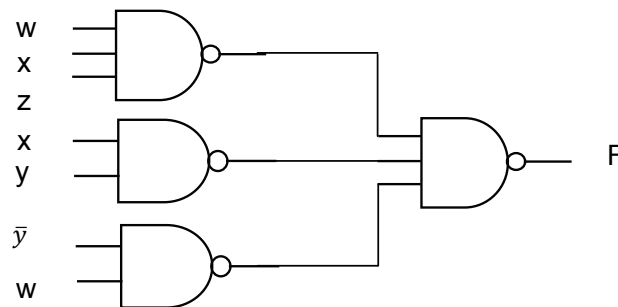
Նկ. 2.4. Տրամաբանական շղթայի սխեման՝ կառուցված ըստ (2.3) արտահայտության

Այժմ կառուցենք նույն շղթան ԵՎ-ՈՉ տարրի միջոցով: Կիրառելով (2.2) բանաձևը (2.3)-ի նկատմամբ՝ կարող ենք ստանալ հետևյալը՝

$$y = \overline{\overline{wxz} \cdot \overline{xy} \cdot \overline{yw}} : \quad (2.4)$$

Այս բանաձևին համապատասխանող սխեման ցույց է տրված նկ. 2.5-ում: Համեմատելով նկ. 2.4-ի և նկ. 2.5-ի սխեմաները՝ կարելի է տեսնել, որ ԵՎ, ԿԱՄ, ՈՉ բազիսում տրված սխեմայից ԵՎ-ՈՉ բազիսի համապատասխան սխեմայի անցնելու համար բավարար է ԵՎ և ԿԱՄ տարրերը փոխարինել ԵՎ-ՈՉ տարրով:

ԿԱՄ-ՈՉ բազիսում տրամաբանական շղթայի կառուցման համար նվազարկումից ստացված ԴՆՉ-ն պետք է ձևափոխել դե Մորգանի բանաձևերով հետևյալ կերպ.



Նկ. 2.5. Տրամաբանական շղթայի սխեման՝ կառուցված ըստ (2.4) արտահայտության

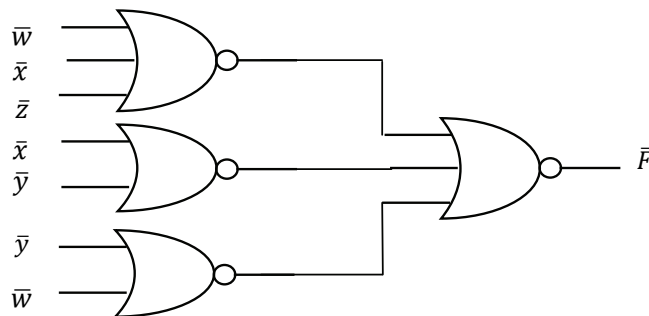
$$\bar{y} = \overline{P_1 + P_2 + \dots + P_r} = \overline{S_1 + S_2 + \dots + S_r}, \quad (2.5)$$

որտեղ  $S_i = \bar{P}_i$ , այսինքն՝  $S_i$ -ին արտադրյալի ժխտումն է, որն ըստ դե Մորգանի բանաձևերի ներկայացվում է փոփոխականների ժխտված ձևերի գումարով

**Օրինակ 2.3.** Կառուցել (2.3) արտահայտությամբ տրված շղթան ԿԱՄ-ՈՉ բազիսում: Կիրառելով (2.5) բանաձևը (2.3)-ի նկատմամբ, կստացվի՝

$$\bar{y} = \overline{\bar{w} + \bar{x} + \bar{z} + \bar{x} + \bar{y} + y + \bar{w}} : \quad (2.6)$$

Համապատասխան տրամաբանական շղթան ցույց է տրված նկ. 2.6-ում:



Նկ. 2.6. Տրամաբանական շղթայի սխեման՝ կառուցված ըստ (2.6) արտահայտության

Համեմատելով նկ. 2.4-ի և նկ. 2.6-ի սխեմաները՝ կարելի է տեսնել, որ ԵՎ, ԿԱՄ, ՈՉ բազիսում տրված սխեմայից ԿԱՄ-ՈՉ բազիսի համապատասխան սխեմային անցնելու համար բավարար է ԵՎ և ԿԱՄ տարրերը փոխարինել ԿԱՄ-ՈՉ տարրով և ժխտել

բոլոր մուտքային և ելքային փոփոխականները:

Նվազարկված բուլյան արտահայտությունը նվազագույն թվով տրամաբանական տարրերով իրականացնելու համար շատ դեպքերում օգտակար է կատարել նաև հետագա ձևափոխություններ:

**Օրինակ 2.4.** Նախագծել տրամաբանական շղթա, որը ելքում պետք է գեներացնի տրամաբանական 1, երբ մուտքային  $n$  փոփոխականների մասշտաբավորված գումարը փոքր չէ տրված  $S$  շեմի արժեքից.

$$y = \begin{cases} 1, & \text{if } \sum_{i=1}^n \alpha_i x_i \geq S \\ 0, & \text{if } \sum_{i=1}^n \alpha_i x_i < S \end{cases} \quad (2.7)$$

որտեղ՝  $\alpha_1, \dots, \alpha_n$  մուտքերի մասշտաբավորման գործակիցներն են: Ենթադրենք  $n=5$ ,  $S=5$ , և  $\alpha_1=\alpha_2=1$ ,  $\alpha_3=\alpha_4=\alpha_5=2$ .

Նկարագրված ֆունկցիայի իսկության աղյուսակը ցույց է տրված աղյուսակ 2.3-ում,  $y$ -ի արժեքները հաշվվում են (2.7) բանաձևով: Նկ. 2.7-ի Կառնոյի քարտով նվազարկված բանաձևն ունի հետևյալ տեսքը.

$$y = x_1 x_4 x_5 + x_1 x_3 x_5 + x_1 x_3 x_4 + x_3 x_4 x_5 + x_2 x_4 x_5 + x_2 x_3 x_5 + x_2 x_3 x_4 : \quad (2.8)$$

Կիրառելով դե Մորգանի բանաձևերը (2.7)-ի նկատմամբ՝ կստացվի ֆունկցիայի բանաձևը ԵՎ-ՈՉ բազիսում.

$$y = \overline{x_1 x_4 x_5} \& \overline{x_1 x_3 x_5} \& \overline{x_1 x_3 x_4} \& \overline{x_3 x_4 x_5} \& \overline{x_2 x_4 x_5} \& \overline{x_2 x_3 x_5} \& \overline{x_2 x_3 x_4} : \quad (2.9)$$

Աղյուսակ 2.3

#	$x_1 x_2 x_3 x_4 x_5$	$y$	#	$x_1 x_2 x_3 x_4 x_5$	$y$	#	$x_1 x_2 x_3 x_4 x_5$	$y$	#	$x_1 x_2 x_3 x_4 x_5$	$y$
0	0 0 0 0 0	0	8	0 1 0 0 0	0	16	1 0 0 0 0	0	24	1 1 0 0 0	0
1	0 0 0 0 1	0	9	0 1 0 0 1	0	17	1 0 0 0 1	0	25	1 1 0 0 1	0
2	0 0 0 1 0	0	10	0 1 0 1 0	0	18	1 0 0 1 0	0	26	1 1 0 1 0	0
3	0 0 0 1 1	0	11	0 1 0 1 1	1	19	1 0 0 1 1	1	27	1 1 0 1 1	1
4	0 0 1 0 0	0	12	0 1 1 0 0	0	20	1 0 1 0 0	0	28	1 1 1 0 0	0
5	0 0 1 0 1	0	13	0 1 1 0 1	1	21	1 0 1 0 1	1	29	1 1 1 0 1	1
6	0 0 1 1 0	0	14	0 1 1 1 0	1	22	1 0 1 1 0	1	30	1 1 1 1 0	1
7	0 0 1 1 1	1	15	0 1 1 1 1	1	23	1 0 1 1 1	1	31	1 1 1 1 1	1

		$x_3 x_4 x_5$							
		000	001	011	010	110	111	101	100
$x_1 x_2$	00	0	0	0	0	0	1	0	0
	01	0	0	1	0	1	1	1	0
	11	0	0	1	0	1	1	1	0
	10	0	0	1	0	1	1	1	0

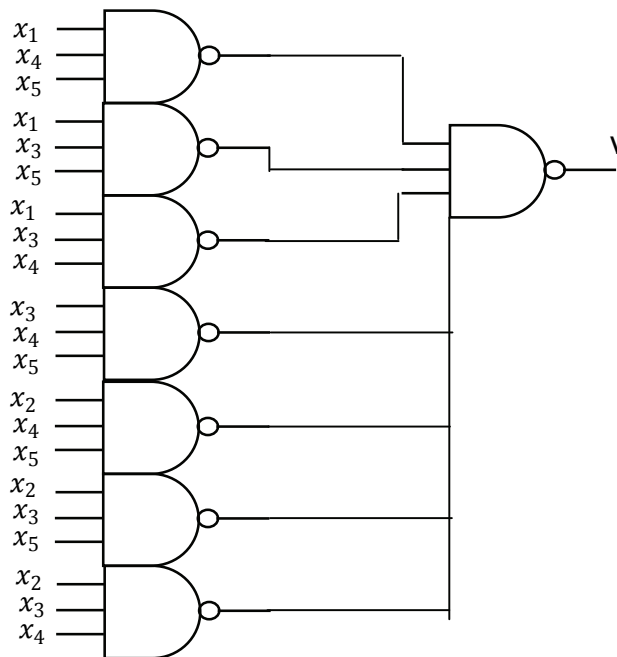
Նկ. 2.7. Աղյուսակ 2.3-ով նկարագրված ֆունկցիայի Կառնոյի քարտը



Եթե սխեման կառուցվի ըստ (2.8) բանաձևի, ապա ԵՎ և ԿԱՄ տարրեր պարունակող ցածր ինտեգրացման աստիճանի ինտեգրալ սխեմաների օգտագործման դեպքում կպահանջվի 7 հատ 3ԵՎ տարրեր և մեկ հատ 7ԿԱՄ տարր: Ուստի սխեման ԵՎ-ՈՉ բազիսում ըստ (2.9) բանաձևի կառուցելու համար կպահանջվի 7 հատ 3ԵՎ-ՈՉ և մեկ հատ 7ԵՎ-ՈՉ տարրեր: Կան ցածր ինտեգրացման աստիճանի ԻՍ-եր, որոնք մեկ պատյանում պարունակում են 3 հատ 3ԵՎ-ՈՉ տարրեր, և ԻՍ-եր, որոնք պարունակում են մեկ հատ 8ԵՎ-ՈՉ տարր: 8ԵՎ-ՈՉ տարրը 7ԵՎ-ՈՉ տարրի փոխարեն օգտագործման դեպքում ավելորդ՝ 8-րդ մուտքը կարելի է միացնել հաստատուն տրամաբանական 1 մակարդակի (VDD սնման դողին հաջորդաբար միացված դիմադրության միջոցով) կամ օգտագործված մուտքերից որևէ մեկին: Այսպիսով, սխեմայի իրականացման համար կպահանջվի 4 ԻՍ: Համապատասխան սխեման ցույց է տրված նկ. 2.8-ում: Մուտքային ազդանշանը որևէ մուտքից ելք հասնելու համար անցնում է երկու տրամաբանական տարրերով: Ազդանշանի տարածման հապաղումը որևէ մուտքից ելք ներկայացվում է հետևյալ կերպ՝  $t_{\text{pNAND3}} + t_{\text{pNAND8}}$ .

Սխեմայի կառուցման համար անհրաժեշտ տրամաբանական տարրերի թիվը կարելի է կրճատել (2.8) բանաձևի ձևափոխությունների միջոցով: Ձևափոխությունների ընթացքում անհրաժեշտ է բանաձևում նվազեցնել սիմվոլների թիվը.

$$y = x_4x_5(x_1 + x_2) + x_3x_5(x_1 + x_2) + x_3x_4(x_1 + x_2) + x_3x_4x_5 = (x_1 + x_2)(x_4x_5 + x_3x_5 + x_3x_4) + x_3x_4x_5 = \overline{x_1} \overline{x_2} \overline{x_4x_5} \overline{x_3x_5} \overline{x_3x_4} \overline{x_3x_4x_5} : \quad (2.10)$$

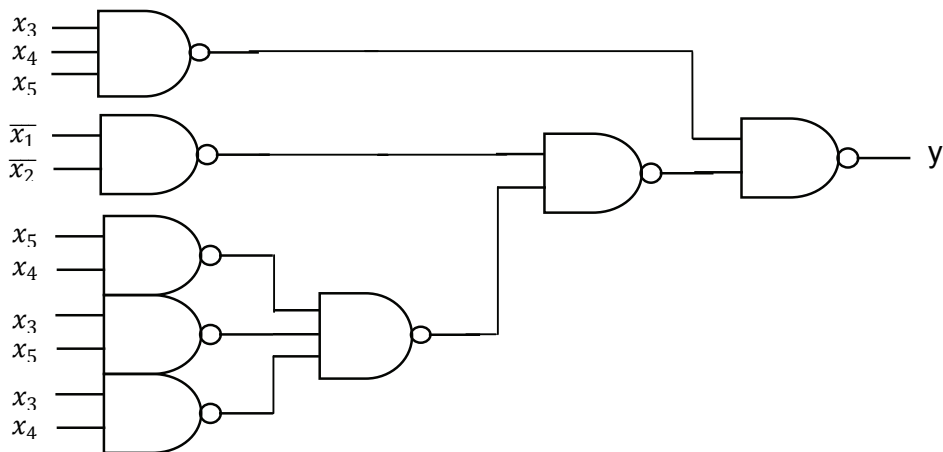


Նկ. 2.8. (2.9) բանաձևով նկարագրվող տրամաբանական շղթայի սխեման

(2.10) բանաձևը պարունակում է 11 սիմվոլ, այն դեպքում, երբ սկզբնական (2.8) բանաձևը պարունակում է 21 սիմվոլ: (2.10) բանաձևով նկարագրվող տրամաբանական շղթայի իրականացման համար անհրաժեշտ է ութ 2ԵՎ-ՈՉ և մեկ 7ԵՎ-ՈՉ

տարրեր: Համապատասխան տրամաբանական սխեման ցույց է տրված նկ. 2.9-ում: Նման իրականացման համար բավարար է ունենալ 3 ԻՍ: Նկ. 2.9-ի սխեմայի իրականացման համար պահանջվում է ավելի քիչ թվով ԻՍ-եր, քան նկ. 2.8-ի սխեմայի իրականացման համար: Սակայն նկ. 2.9-ի շղթայում ազդանշանի տարածման վատագույն դեպքի հապաղումը կազմում է  $4t_{\text{pNAND2}}$ , որը գրեթե երկու անգամ մեծ է, քան նկ. 2.8-ի շղթայում, որտեղ ազդանշանը ցանկացած մուտքից ելք տարածվում է անցնելով երկու տրամաբանական տարրով: Նկ. 2.9-ի սխեմայում առկա է ևս մեկ թերութուն. ազդանշանները տարբեր մուտքերից ելք տարածվում են՝ անցնելով տարբեր քանակությամբ տարրերով, հետևաբար, կունենան տարբեր հապաղումներ, որի պատճառով սխեմայի ելքում կարող են առաջանալ կարճատև կեղծ իմպուլսներ:

Բուլյան ֆունկցիաների նվազարկման և համարժեք ձևափոխությունների մեթոդները հնարավորություն են տալիս կառուցել շղթաները նվազագույն թվով կամ համարյա նվազագույն թվով տրամաբանական տարրերով: Սակայն հաճախ անհրաժեշտ է հաշվի առնել նաև այլ սահմանափակումներ և չափանիշներ՝ մուտքից-ելք ազդանշանի տարածման նվազագույն հապաղում, տրամաբանական տարրերի մուտքերի թվի սահմանափակում, նվազագույն թվով միջտարրային միացումներ, տրամաբանական տարրերի ելքերի ճյուղավորումների սահմանափակում, սխեմայում ցրվող նվազագույն հզորություն: Բոլոր այս չափանիշներին հնարավոր չէ բավարարել միաժամանակ: Յուրաքանչյուր որոշակի նախագծում անհրաժեշտ է սահմանել այդ չափանիշների առաջնահերթությունների սանդղակ: Այնուամենայնիվ, շատ դեպքերում նախագծումն սկսվում է բուլյան ֆունկցիաների նվազարկումից: Հիմնվելով նվազարկված տարրերակի վրա՝ կատարվում է հետագա վերլուծություն՝ լրացուցիչ պահանջներին բավարարելու նպատակով:



Նկ. 2.9. (2.10) բանաձևով նկարագրվող տրամաբանական շղթայի սխեման

Ավտոմատացված նախագծման ժամանակ նախագծման ծրագրային գործիքները համակցական սարքի աշխատանքի վարքագծային նկարագրությունից՝ իսկության աղյուսակից, գեներացնում են նվազարկված և ըստ լրացուցիչ չափանիշների օպտիմալացված շղթայի տրամաբանական տարրերի և դրանց միջմիացումների ցուցակը (կապերի ցուցակ, netlist): Կապերի ցուցակը տրամաբանական նախագծման

արդյունքն է՝ ներկայացված տեքստային տեսքով, որը հետագայում օգտագործվում է սարքի տոպոլոգիայի ավտոմատացված նախագծման համար: Սակայն միշտ պետք է հիշել, որ տրամաբանական սխեման չափազանց օգտակար է՝ սարքի կառուցվածքը տեսանելի դարձնելու և աշխատանքը վերլուծելու համար: Դրա համար ավտոմատ սինթեզի (նախագծման) ծրագրային գործիքները հնարավորություն ունեն կապերի ցուցակի հետ միաժամանակ նաև գեներացնել տրամաբանական սխեմաները:

### 2.3. Բազմաելք համակցական շղթաների նախագծում

Բազմաելք համակցական շղթան ունի  $n$  մուտքեր և  $m$  ելքեր: Հետևելով համակցական շղթաների սինթեզի եղանակին՝  $m$  ելքերից յուրաքանչյուրը նկարագրող ֆունկցիան պետք է նվազարկվի: Բազմաելք շղթաների նախագծման համար առանձին ելքային ֆունկցիաների ինքնուրույն նվազարկումը հնարավոր է, որ չտա նվազագույն լուծում՝ ստացված պարզեցված ֆունկցիաների համակարգում նվազագույն թվով լիտերալներ ստանալու տեսանկյունից:

Կարևոր է հաշվի առնել, որ միևնույն արտադրյալը՝ իմպլիկանտը, կարող է օգտագործվել մեկից ավելի ֆունկցիաներում: Հետևաբար, ելքային ֆունկցիաների նվազարկման ժամանակ կարևոր է ելքային ֆունկցիաների բանաձևերում ստանալ հնարավորինս մեծ թվով միանման իմպլիկանտներ. պարտադիր չէ, որ դրանք լինեն պարզ իմպլիկանտներ: Դիտարկենք նկ. 2.10ա,բ-ի քարտերում ներկայացված երկու ֆունկցիաների նվազարկման օրինակը: Այդ ֆունկցիաների նվազարկումից կստանանք.

F1		yz			
		00	01	11	10
wx	00		X		
	01	1	X		
	11	1	X	1	1
	10	1	X		

F2		yz			
		00	01	11	10
wx	00		X		
	01		X	1	
	11		X		
	10		X	1	

(ա)  $F1 = x\bar{y} + w\bar{y} + wx$

(բ)  $F2 = \bar{w}xz + w\bar{x}z$

F1

		yz			
		00	01	11	10
wx	00		X		
	01	1	X		
	11	1	X	1	1
	10	1	X		

(գ)  $F1 = \bar{w}x\bar{y} + w\bar{x}\bar{y} + wx$

Նկ. 2.10. F1 և F2 ֆունկցիաների քարտերը

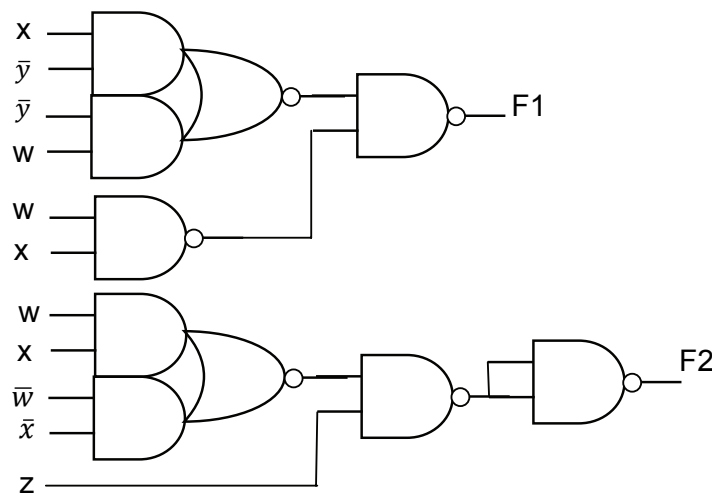
$$F1 = \overline{x}\overline{y} + \overline{w}\overline{y} + wx = \overline{\overline{\overline{\overline{x}\overline{y} + \overline{w}\overline{y} + wx}}}, \quad (2.11)$$

$$F2 = \overline{wx}z + \overline{wx}z = z(\overline{wx} + \overline{wx}) = z(\overline{\overline{wx}} + \overline{wx}) = z(\overline{wx} + wx) : \quad (2.12)$$

Այդ բանաձևերով կառուցված սխեման ցույց է տրված նկ. 2.11-ում:

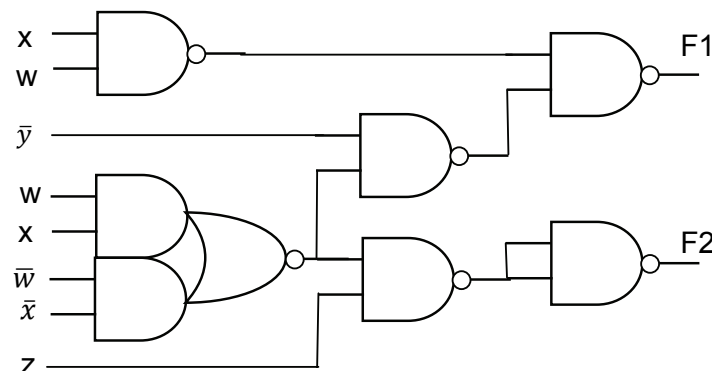
Եթե F1 ֆունկցիան նվազարկվի այնպես, ինչպես ցույց է տրված նկ. 2.10-ում, ապա կստացվի՝

$$F1 = \overline{wx}\overline{y} + \overline{wx}\overline{y} + wx = wx + \overline{y}(\overline{wx} + \overline{wx}) = wx + \overline{y}(\overline{\overline{wx}} + \overline{wx}) = \overline{wx} \& \overline{y}(\overline{wx} + wx) \quad (2.13)$$



Նկ. 2.11. (2.11) և (2.12) բանաձևերով իրականացված տրամաբանական շղթան

(2.12) և (2.13) բանաձևերի հիման վրա կառուցված սխեման ցույց է տրված նկ. 2.12-ում, որն ավելի պարզ է, քան նկ. 2.11-ինը, քանի որ նույն ԵՎ-ԿԱՄ-ՈՉ տարրը, որն իրագործում է  $\overline{wx} + wx$  արտահայտությունը, օգտագործվում է F1 և F2 ֆունկցիաներում:



Նկ. 2.12. (2.12) և (2.13) բանաձևերով իրականացված տրամաբանական շղթան:

## 2.4. Տրամաբանական շղթաների վերլուծություն

Տրամաբանական շղթաների սինթեզն առաջ գնացող գործընթաց է. սկսվում է իսկության աղյուսակից և ավարտվում տրամաբանական սխեմաներով: Վերլուծությունը սինթեզի հակադարձ գործընթացն է, որն սկսվում է տրամաբանական սխեմայից և ավարտվում է այդ սխեման նկարագրող իսկության աղյուսակի կամ բուլյան արտահայտությունների դուրս բերմամբ: Սկզբում կքննարկենք տրամաբանական սխեմայից բուլյան բանաձևերի ստացումը, այնուհետև՝ իսկության աղյուսակի ստացումը: Այդ բանաձևերը պետք է ներկայացնեն սխեմայի առաջնային ելքերը՝ որպես ֆունկցիաներ առաջնային մուտքերից: Առաջնային մուտքերն արտաքինից տրվող մուտքերն են, առաջնային ելքերը՝ սխեմայից դուրս գնացող ելքերն են:

Տրամաբանական սխեմայից բուլյան բանաձևերի ստացման ընթացակարգն ընդգրկում է հետևյալ քայլերը.

1. բոլոր տրամաբանական տարրերի ելքերը նշվում են միմյանցից տարբերվող սիմվոլներով:
2. դուրս են բերվում այն տարրերի ելքերի բանաձևերը, որոնց մուտքերին կիրառված են սխեմայի առաջնային մուտքերը:
3. հաջորդաբար դուրս են բերվում մնացած տրամաբանական տարրերի ելքերի ֆունկցիաների բանաձևերը, որոնց մուտքերն արդեն որոշվել են ստացված բանաձևերով:
4. կրկնել 3-րդ քայլը մինչև որ ստացվեն բոլոր առաջնային ելքերի բանաձևերը:
5. առաջնային ելքերի բուլյան ֆունկցիաների բանաձևերում հաջորդաբար փոխարինել փոփոխականները, մինչև որ մնան միայն առաջնային մուտքերի փոփոխականները:
6. ստացված բանաձևերը ձևափոխել  $\Gamma\text{N2}$ -ի (եթե դա անհրաժեշտ է):

Դիտարկենք նկ. 2.13-ում ցույց տրված սխեման: Սխեմայում բոլոր տարրերի ելքերը, որոնք առաջնային ելքեր չեն, նշված են  $T_i$  սիմվոլներով:

Դուրս բերենք այն տարրերի ելքերի ֆունկցիաները, որոնք ունեն միայն առաջնային մուտքեր.

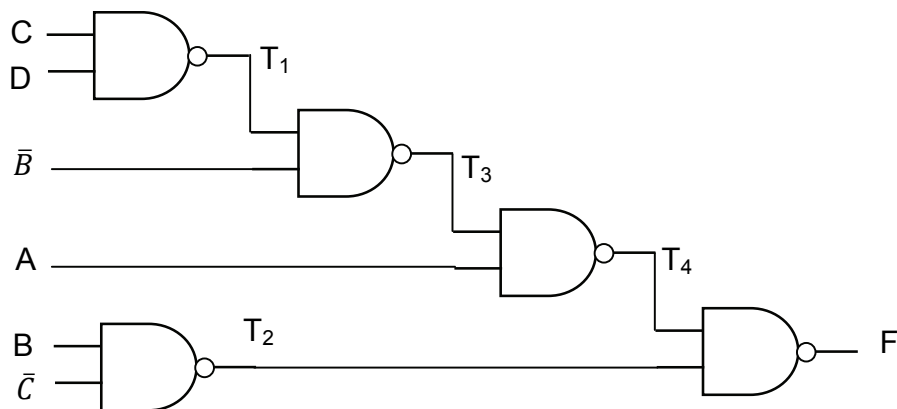
$$T_1 = \overline{CD} , \quad (2.14)$$

$$T_2 = \overline{\overline{C}B} : \quad (2.15)$$

Հաջորդաբար որոշենք մնացած բոլոր տարրերի ելքային ֆունկցիաների բանաձևերը.

$$T_3 = \overline{T_1 B} , \quad (2.16)$$

$$T_4 = \overline{T_3 A} : \quad (2.17)$$



Նկ. 2.13. Տրամաբանական սխեմայի վերլուծության օրինակ

Սխեմայի առաջնային ելքի ֆունկցիայի համար կունենանք.

$$F = \overline{T_2 T_4} : \quad (2.18)$$

Ստացված բանաձևում հաջորդաբար  $T_i$  ֆունկցիաները փոխարինենք իրենց բանաձևերով, մինչև մնան միայն առաջնային մուտքային փոփոխականները.

$$F = \overline{T_2 T_4} = \overline{\overline{CB} \overline{T_3 A}} = \overline{\overline{CB} \overline{T_1 \overline{BA}}} = \overline{\overline{CB} \overline{\overline{CDBA}}} : \quad (2.19)$$

Ձևափոխենք ստացված բանաձևը համարժեք ԴՆՁ-ի.

$$F = \overline{\overline{CB} \overline{\overline{CDBA}}} = \overline{CB} + \overline{CDBA} = \overline{CB} + A(CD + B) = \overline{CB} + ACD + AB : \quad (2.20)$$

Եթե պետք է շարունակել հետազոտությունը և որոշել շղթայով իրագործվող տվյալների մշակման խնդիրը, ապա կարելի է ստացված բանաձևից ստանալ ֆունկցիայի իսկության աղյուսակը: Դրա համար պետք է կառուցել աղյուսակ ու մուտքային փոփոխականների համար, լրացնել այն մուտքային փոփոխականների բոլոր հավաքածուներով, ստացված բանաձևով հաշվել ֆունկցիայի արժեքները բոլոր հավաքածուների համար և լրացնել աղյուսակի ֆունկցիայի սյունակում: Աղյուսակ 2.4-ում ցույց է նկ. 2.13-ի սխեմային համապատասխանող տրված իսկության աղյուսակը՝ հաշվված (2.20) բանաձևով:

Աղյուսակ 2.4

#	ABCD	F	#	ABCD	F
0	0000	0	8	1000	0
1	0001	0	9	1001	0
2	0010	0	10	1010	0
3	0011	0	11	1011	1
4	0100	1	12	1100	1
5	0101	1	13	1101	1
6	0110	0	14	1110	1
7	0111	0	15	1111	1

Այն դեպքերում, երբ սխեմայի վերլուծությունից պետք է որոշել միայն իսկության աղյուսակը, ապա դա կարելի կատարել միանգամից՝ առանց բանաձևի դուրս բերման: Դա կարելի է իրականացնել հետևյալ քայլերով.

1. որոշել սխեմայի առաջնային մուտքերի թիվը՝  $n$ : լրացնել մուտքային բոլոր  $2^n - 1$  հավաքածուները իսկության աղյուսակում:
2. տրամաբանական տարրերի ելքերը նշել սինվոլներով:
3. իսկության աղյուսակում լրացնել այն տարրերի ելքերի արժեքները, որոնց մուտքերը միայն սխեմայի առաջնային մուտքերն են:
4. շարունակել, լրացնել այն տարրերի ելքերի արժեքները, որոնց մուտքերն արդեն որոշվել են, մինչև որ լրացվեն սխեմայի առաջնային ելքերի սյունակները:

Աղյուսակ 2.5-ում ցույց է տրված նկ. 2.13-ի սխեմայի իսկության աղյուսակի լրացման գործընթացը:

Հարկ է նշել, որ նույնիսկ եթե սկզբնական տրամաբանական ֆունկցիան, որից սինթեզվել է վերլուծվող սխեման, պարունակել է մուտքային փոփոխականների արգելված հավաքածուներ, սխեմայի վերլուծությունից ստացված իսկության աղյուսակը արգելված հավաքածուներ չի ունենա: Ավելին, շատ դեպքերում սխեմայի վերլուծությունն իրականացվում է հենց արգելված հավաքածուների համար սխեմայի իրական ելքերը որոշելու նպատակով: Դա մասնավորապես անհրաժեշտ է լինում, երբ պետք է վերլուծել սխեմայի աշխատանքը՝ անսարքությունները հայտնաբերելու համար:

Աղյուսակ 2.5

#	ABCD	T1	T2	T3	T4	F
0	0000	1	1	0	1	0
1	0001	1	1	0	1	0
2	0010	1	1	0	1	0
3	0011	0	1	1	1	0
4	0100	1	0	1	1	1
5	0101	1	0	1	1	1
6	0110	1	1	1	1	0
7	0111	0	1	1	1	0
8	1000	1	1	0	1	0
9	1001	1	1	0	1	0
10	1010	1	1	0	1	0
11	1011	0	1	1	0	1
12	1100	1	0	1	0	1
13	1101	1	0	1	0	1
14	1110	1	1	1	0	1
15	1111	0	1	1	0	1

**Օրինակ 2. 5.** Դիտարկենք օրինակ 2.1-ում սինթեզված սխեման, որը ցույց է տրված նկ. 2.3-ում: Այդ սխեման սինթեզվել է աղյուսակ 2.2-ի իսկության աղյուսակից, որը պարունակում է  $(x_3x_2x_1)=101, 110, 111$  արգելված հավաքածուները: Նկ. 2.3-ի սխեմայի վերլուծությունից կստացվի աղյուսակ 2.6-ում ցույց տրված իսկության աղյուսակը:

Աղյուսակ 2.6

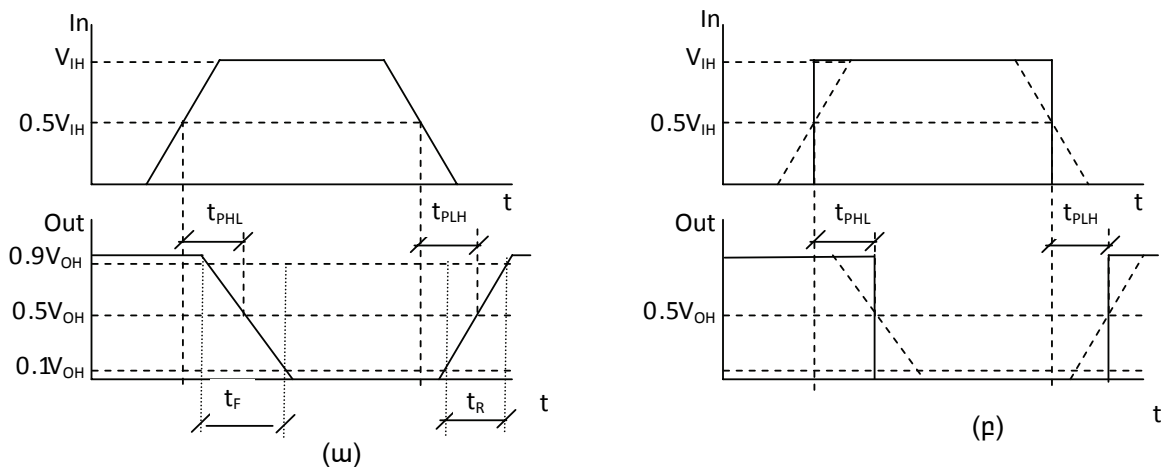
$x_3$	$x_2$	$x_1$	$y_1$	$y_2$	$y_3$
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

## 2.5. Հապաղումները և մրցումները համակցական շղթաներում

Նախագծվող տրամաբանական շղթաները պետք է համակողմանի ուսումնասիրել, համոզված լինելու համար, նրա աշխատանքը ճշգրիտ է ֆիզիկական իրականացման ժամանակ: Բացի սարքի տրամաբանության ստուգումից, պետք է ստուգել նաև նրա արագագործությունը՝ անսխալ աշխատանքի կարողությունը տրված աշխատանքային հաճախականությունների միջակայքում:

Իրական շղթաներում ազդանշանները փոփոխվում են վերջավոր ժամանակներում՝ որոշակի ժամանակ է անհրաժեշտ որպեսզի ազդանշանը տրամաբանական մեկմակարդակից փոխանջատվի մյուսին: Այդ անցումները բնութագրվում են աճի և անկման ժամանակներով, որոնք ցույց են տրված նկ. 2.14ա-ում: Աճի ժամանակը,  $t_R$  այն ժամանակահատվածն է, որի ընթացքում ազդանշանը  $0.1V_{OH}$  մակարդակից աճում է մինչև  $0.9V_{OH}$  մակարդակը,  $V_{OH}$ -ն տրամաբանական 1-ի հաստատված լարումն է: Անկման ժամանակը,  $t_F$  այն ժամանակահատվածն է, որի ընթացքում ազդանշանը  $0.9V_{OH}$  մակարդակից նվազում է մինչև  $0.1V_{OH}$  մակարդակը:

Թվային շղթաների արագագործությունը հիմնականում սահմանափակվում է տրամաբանական տարրերում ազդանշանների հապաղումներով: Նկ.2.14ա-ում ցույց են տրված շրջիչի մուտքային և ելքային ազդանշանները և հապաղումների որոշման սահմանումները:



Նկ. 2.14. Տրամաբանական շրջիչի մուտքային և ելքային ազդանշանները



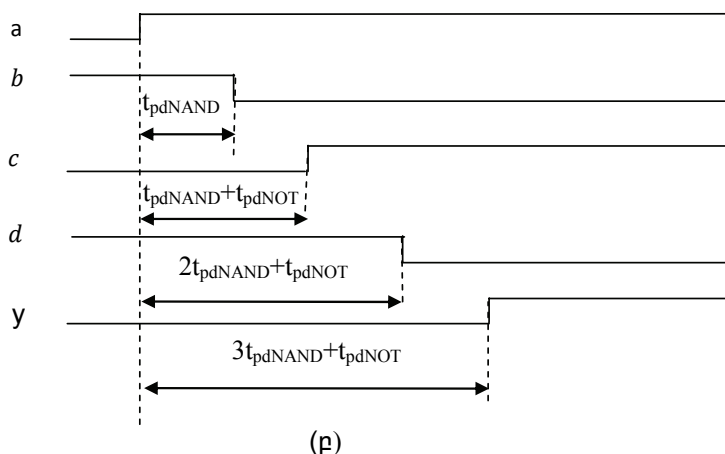
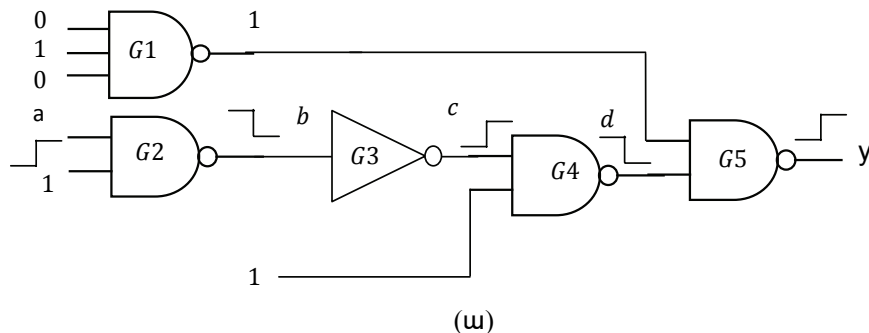
Ազդանշանի ցածրից բարձր անցման տարածման հապաղումը,  $t_{PLH}$ , մուտքային և ելքային ազդանշանների  $0.5V_{OH}$  մակարդակը հատելու պահերի տարբերությունն է, երբ ելքային ազդանշանը անցնում 0-ից 1: Բարձրից ցածր անցման տարածման հապաղումը,  $t_{PHL}$ , մուտքային և ելքային ազդանշանների  $0.5V_{OH}$  մակարդակը հատելու պահերի տարբերությունն է, երբ ելքային ազդանշանը անցնում 1-ից 0:

Երբեմն, պարզության համար, բավարար է տարրի հապաղման միջին ժամանակը.

$$t_{pd} = \frac{t_{PLH} + t_{PHL}}{2} : \quad (2.21)$$

Հաճախ տրամաբանական մնանակիչներում ազդանշանների իրական տեսքերը փոխարինվում են ուղղանկյունաձև տեսքերով, որտեղ աճի և անկման ժամանակները 0 են (նկ. 2.14բ):

Թվային շղթայում ազդանշանի փոխանցման տրամաբանական ուղու հապաղումը հաշվվում է իբրև առանձին տարրերի հապաղումների գումար: Նկ. 2.15ա-ում ցույց է տրված ազդանշանի փոխանցման ուղու օրինակ՝ G2-G3-G4-G5: Ազդանշանի տեսքը շղթայի հանգույցներում ցույց է տրված նկ. 2.15բ-ում: Շրջիչի հապաղումը նշանակված է  $t_{pdNOT}$ -ով, իսկ ԵՎ-ՈՉ տարրինը՝  $t_{pdNAND}$ -ով:

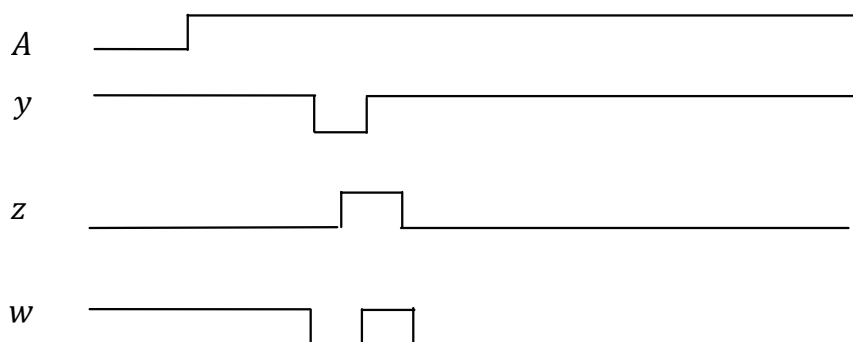


Նկ. 2.15. Ազդանշանի փոխանցման տրամաբանական ուղու օրինակ և ժամանակային դիագրամները

Տրամաբանական շղթաներում ազդանշանի տարածման տարբեր ուղիներում, հապաղումների տարբեր արժեքների պատճառով, կարող են ի հայտ գալ ազդանշանի անցանկալի փոխանջատումներ, որոնք կոչվում են մրցումներ (hazard): Մրցումները կարող են շղթայի սխալ աշխատանքի պատճառ դառնալ: Համակացական շղթաներում մրցումները ի հայտ են գալիս ելքային ազդանշանի կարճատև կեղծ մակարդակների տեսքով: Այդ կարճատև կեղծ մակարդակները ինքնին վտանգավոր չեն, սակայն եթե դրանք կիրառվեն ասինքրոն հաջորդական շղթաների, ապա առաջ կբերեն անցում չնախատեսված վիճակի և աշխատանքի խոտորում նախանշված ալգորիթմից: Հետևաբար, պետք է ստուգել մրցումների հնարավորությունը և վերլուծել դրանց ազդեցությունը շղթայի աշխատանքի ալգորիթմի վրա: Եթե մրցումը կարող է բերել շղթայի ոչ ճիշտ աշխատանքի, ուրեմն պետք է միջոցներ ձեռնարկել՝ մրցումների բացասական հետևանքները վերացնելու համար:

Պարզագույն դեպքում կհամարենք, որ ժամանակի որևէ պահի շղթայի մուտքերից միայն մեկը կարող է փոխանջատվել: Կա երկու տիպի մրցում՝ ստատիկ և դինամիկ: Ստատիկ մրցումն այն դեպքն է, երբ մուտքային մեկ փոփոխականի փոխանջատումը շղթայի ելքում առաջ է բերում մակարդակի կարճատև փոխանջատում, այն դեպքում, երբ ելքը պետք է չփոխանջատվեր:

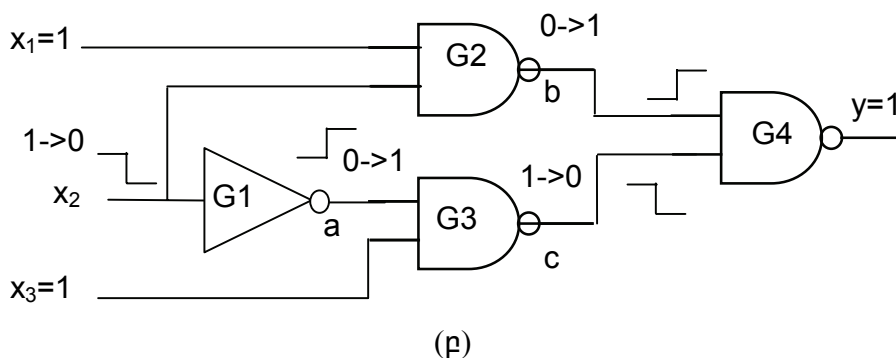
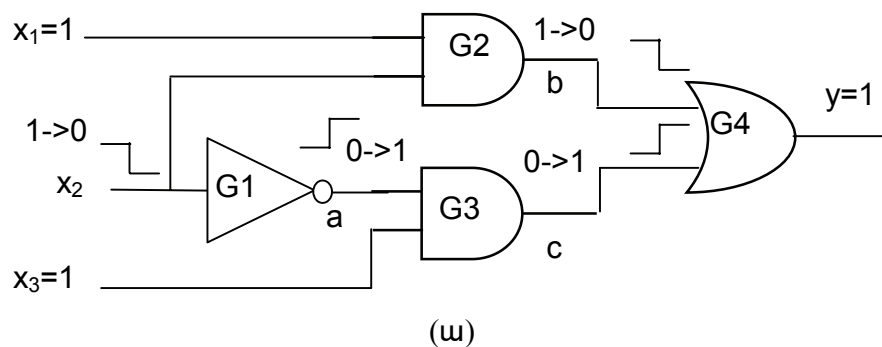
Նկ. 2.16-ում ցույց տրված ժամանակային դիագրամում ելքային  $y$  ազդանշանը պետք է մնա տրամաբանական 1 մակարդակի վրա, երբ մուտքային  $A$  ազդանշանը փոխանջատվում է 0-ից 1: Բայց դրա փոխարեն  $y$ -ը կատարում է կարճատև անցում 0-ի մակարդակ և վերադառնում 1 արժեքի: Այս դեպքում, երբ ելքը պետք է չփոխվեր՝ մնար 1 մակարդակի վրա, մրցումը կոչվում է ստատիկ-1 մրցում: Նկ. 2.16-ի  $z$  ազդանշանի տեսքը համապատասխանում է ստատիկ-0 մրցման դեպքին. ազդանշանը պետք է մնար 0 մակարդակի վրա, բայց կարճատև անցում է կատարում 1 մակարդակի: Նկ. 2.16-ում ցույց տրված  $w$  ելքային ազդանշանի փոփոխությունները համապատասխանում են դինամիկ մրցմանը: Ելքային ազդանշանը պետք է կատարի 1-ից 0 անցում, բայց մեկ անցում կատարելու փոխարեն ազդանշանը կատարում է մեկից ավելի կենտ թվով փոխանջատումներ (տվյալ դեպքում 3) մինչև հաստատվի վերջնական 0 մակարդակի վրա:



Նկ. 2.16. Մրցումները համակցական շղթաներում՝ ստատիկ-1 մրցում  $y$  ելքում, ստատիկ-0 մրցում  $z$  ելքում և դինամիկ մրցում  $w$  ելքում

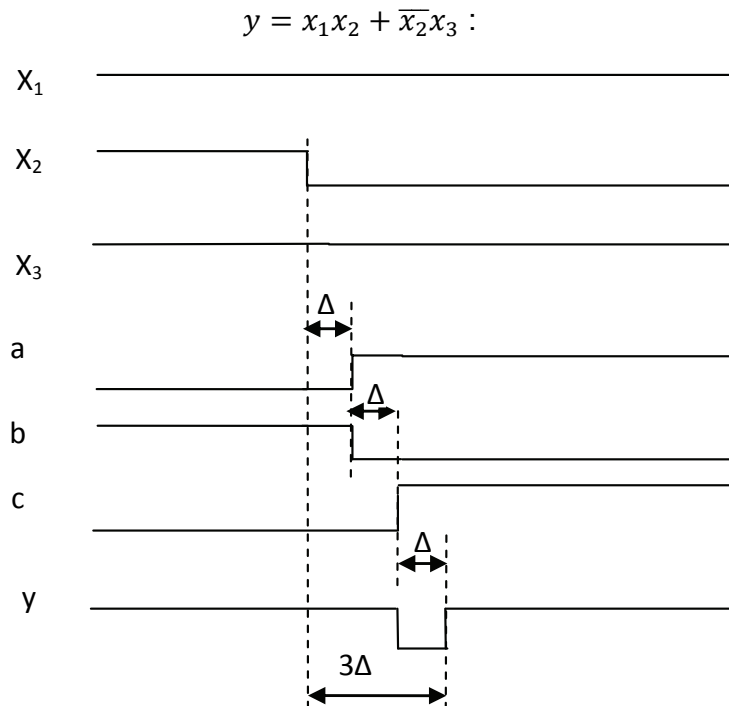
Նկ. 2.17ա-ում ցույց տրված շղթայում առկա է ստատիկ-1 մրցում: Ենթադրենք բոլոր երեք մուտքերին սկզբում տրված է եղել տրամաբանական 1 արժեք, ուրեմն G1 տարրի ելքը կունենա 0 արժեք, G2-ինը՝ 1, G3-ինը՝ 1, G4-ինը՝ 1: Այժմ ենթադրենք,  $x_2$ -ը փոխանջատվում է 1-ից 0: G1-ի ելքը փոխվում է 0-ից 1, G2-ինը՝ 1-ից 0, G3-ինը՝ 0-ից 1, թողնելով  $y$  ելքը 1 մակարդակի վրա: Եթե հաշվի առնվեն հապաղումները տրամաբանական տարրերում, կարելի է տեսնել, որ  $y$  ելքը կարճատև անցնում է 0 մակարդակ: Պարզության համար կհամարենք, որ բոլոր տարրերը ունեն հավասար հապաղումներ (կնշանակենք  $\Delta$ -ով): Շրջիչի հապաղման պատճառով G2-ի ելքը փոխվում է 0-ից 1 ավելի շուտ, քան G3-ինը կանցնի 0-ից 1: Այդ դեպքում G4-ի երկու մուտքերը միաժամանակ կունենան 0 արժեք և որպես հետևանք շղթայի  $y$  ելքը կընդունի 0 արժեք շրջիչի  $\Delta$  հապաղմանը հավասար ժամանակահատվածում: Նկ. 2.18-ում բերված են նկ 2.17ա շղթայի ժամանակային դիագրամները, որոնց վրա հեշտ է հետևել ստատիկ-1 մրցման առաջացմանը:

Նկ. 2.17բ-ում ցույց է տրված նույն շղթայի իրականացումը ԵԿ-ՈԶ տարրերի օգնությամբ: Երբ  $x_2$ -ը փոխվում է 1-ից 0, G4-ի երկու մուտքերը միաժամանակ կունենան 1 արժեք, որի հետևանքով շղթայի  $y$  ելքը կարճատև կընդունի 0 արժեք (ստատիկ-0 մրցում) :



Նկ. 2.17.  $y = x_2x_1 + \bar{x}_2x_3$  ֆունկցիայով նկարագրվող շղթան՝ իրականացված ԵԿ և ԿԱՄ տարրերով (ա) և ԵԿ-ՈԶ տարրերով (բ)

Նկ. 2.17ա,բ-ում ցույց տրված շղթաներն իրականացնում են միևնույն տրամաբանական ֆունկցիան արտադրյալների գումարի տեսքով՝



Նկ. 2.18. Նկ.2.17ա-ում ցույց տրված շղթայի ժամանակային դիագրամները, երբ  $x_2$ -ն անցնում է 1-ից 0

Այս տիպի իրականացման դեպքում (ՌՆՁ իրականացման դեպքում) հնարավոր է ստատիկ-0 մրցում:

Եթե տրամաբանական շղթան իրականացվում է գումարների արտադրյալի ձևով՝

$$y = (x_1 + x_2)(\overline{x_2} + x_3) ,$$

եւքը կարող է կարճատև անցնել 1 վիճակի, այն դեպքում, երբ պետք է մնար 0 մակարդակի վրա (ստատիկ-1 մրցում):

Մրցումների առկայությունը կարելի է հայտնաբերել՝ քննարկելով իրականացվող շղթայի Կառնոյի քարտը: Ցույց տանք դա նկ. 2.19ա-ում բերված քարտի օգնությամբ, որը համապատասխանում է նկ. 2.17ա շղթային:  $x_2$ -ի փոխանջատումը 1-ից 0 շղթայի վիճակը 111 միներմից տեղափոխում է 101 միներմ: Այդ անցման ժամանակ տեղի կունենա մրցում, քանի որ փոխվում են իմպլիկանտները, որոնք ծածկում են այդ միներմներին: 111 միներմը ծածկված է  $x_1x_2$  արտադրյալով, որն իրականացվում է G2 տարրով: 101 միներմը ծածկված է  $\overline{x_2}x_3$  արտադրյալով, որն իրականացվում է G3 տարրով: Այն դեպքերում, երբ շղթայի եւքը պետք է անցնի մեկ իմպլիկանտից մյուսին, հնարավոր է ստատիկ-0 մրցում:

		$x_2x_3$			
		00	01	11	10
$y_1$	$x_1$				
	0		1		
	1		1	1	1

$$y = x_1x_2 + \overline{x_2}x_3$$

(ա)

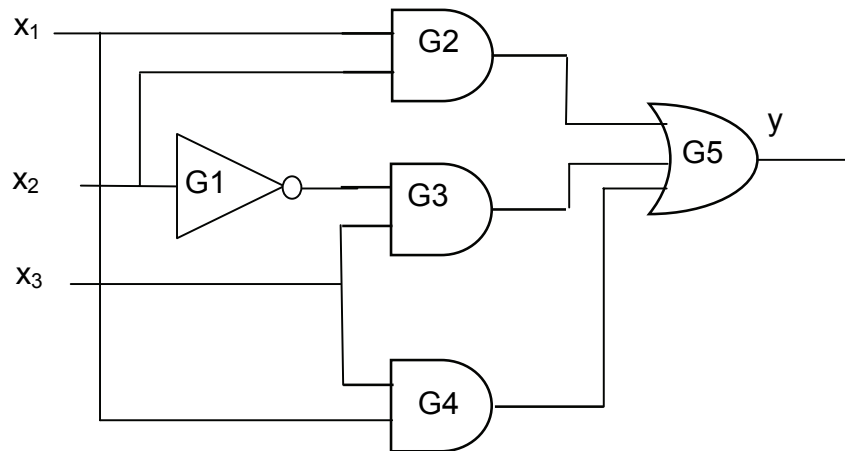
		$x_2x_3$			
		00	01	11	10
$y_1$	$x_1$				
	0		1		
	1		1	1	1

$$y = x_1x_2 + \overline{x_2}x_3 + x_1x_3$$

(բ)

Նկ. 2.19. Մրցումների վերացումը լրացուցիչ իմպլիկանտ մտցնելու միջոցով

Մրցումներից ազատվելու համար պետք է ֆունկցիայում ավելացնել ևս մեկ իմպլիկանտ, որը կծածկի իմպլիկանտների այդ երկու խմբերը: Դա ցույց է տրված նկ. 2.19բ-ի քարտում, որտեղ ավելացված իմպլիկանտը միավորում է 111 և 101 միներմները: Մրցումներից ազատ շղթան ցույց է տրված նկ. 2.20-ում: Ավելացված տարրը ձևավորում է  $x_1x_3$  իմպլիկանտը: Ընդհանուր դեպքում, համակցական շղթաներում մրցումներից կարելի է ազատվել, եթե ավելացվի իմպլիկանտ, որը կծածկի մրցման պատճառ հանդիսացող միներմներին: Մրցումներից խուսափելու համար շղթային պետք է ավելացվեն ավելցուկային տարրեր:

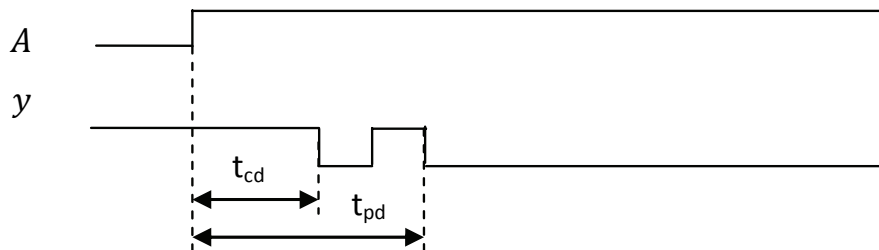


Նկ. 2.20. Մրցումներից ազատ շղթա

Ինքնուրույն օգտագործվող համակցական շղթաներում և սինթրոն հաջորդական շղթաներում օգտագործվող համակցական շղթաներում մրցումներն անհարմարություն չեն հանդիսանում, քանի որ դրանք չեն բերում աշխատանքի խափանման: Բայց եթե ազդանշանի կարճատև սխալ արժեքը հետադարձ կապով մտցվի ասինթրոն հաջորդական շղթա, այն կանցնի սխալ վիճակի, որից հետո ասինթրոն շղթայի աշխատանքը կշեղվի նախանշված ընթացքից: Հետևաբար մրցումներն ասինթրոն հաջորդական շղթաներում սխալի պոտենցիալ աղբյուրներ են և պետք է խնամքով ուսումնասիրվեն և կանխվեն: Մեծ քանակությամբ վիճակներով համակարգերում դա այնքան

էլ դյուրին խնդիր չէ, որի պատճառով ասինքրոն հաջորդական շղթաները ներկայումս մեծ կիրառություն չունեն թվային մեծ համակարգերում:

Թվային համակարգերի հուսալի աշխատանքի համար անհրաժեշտ է պահպանել որոշակի ժամանակային առնչություններ տարբեր հանգույցների աշխատանքում: Երբ թվային շղթայի մուտքում ազդանշանը փոխանջատվում է, որոշակի ժամանակահատված հետո ելքային ազդանշանը սկսում է փոխվել: Այդ պահից սկսած մինչև ելքային ազդանշանի կայունացումը նոր վիճակում ելքային ազդանշանի արժեքը համարվում է ոչ պիտանի նպատակային օգտագործման համար: Համակցական շղթայի ելքային ազդանշանը կարելի է օգտագործել միայն նրա հաստատումից հետո: Մուտքային ազդանշանի փոփոխության պահից մինչև ելքի փոփոխության սկիզբը ընկած ժամանակահատվածը կոչվում է ազդանշանի սկզբնական հապաղում (contamination delay,  $t_{cd}$ ): Մուտքի փոփոխության պահից սկսած մինչև ելքի հաստատման պահն ընկած ժամանակահատվածը կոչվում է ազդանշանի տարածման հապաղում (propagation delay,  $t_{pd}$ ): Այդ հապաղումների սահմանումները պարզաբանված են նկ. 2.21-ում:



Նկ. 2.21. Թվային ազդանշանների հապաղումները և հաստատման ժամանակը

## 2.6. Համակցական թվային ֆունկցիաներ

Կան տրամաբանական ֆունկցիաներ, որոնք հաճախ են կիրառվում շատ թվային համակարգերում: Այդպիսիք են կոդավորող սարքերը (շիֆրատորները), վերծանիչները (դեշիֆրատորները, դեկոդերները), մուլտիպլեքսորները, դեմուլտիպլեքսորները, կոդերի համեմատիչները (թվային կոմպարատորները), գումարիչները, բազմապատկիչները և տվյալների մշակման այլ ֆունկցիաներ: Այս ֆունկցիաներն իրականացվում են միջին ինտեգրացման աստիճանի ինտեգրալ սխեմաների տեսքով, որոնք օգտագործվում են տպասալի մակարդակով իրականացվող թվային համակարգերում՝ որպես պատրաստի թվային հանգույցներ: Դրանք իրականացվում են նաև թվային բջիջների տեսքով որպես կիրառությանը յուրահատուկ ինտեգրալ սխեմայով իրականացվող թվային հմակարգերի թվային հանգույցների բաղադրիչներ: Այս ֆունկցիաների կիրառությունը հնարավորություն է տալիս կրճատել նախագծման ժամկետները և պահանջվող ջանքերը:

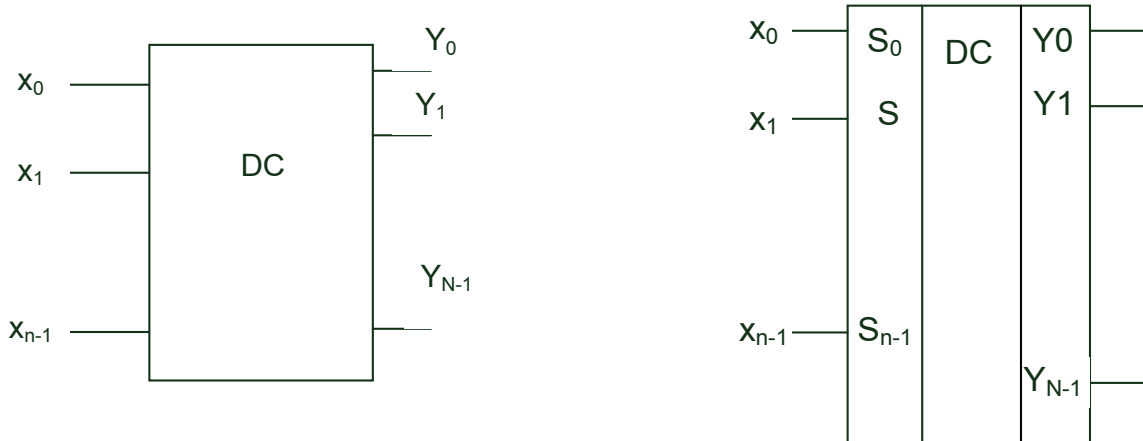
### 2.6.1. Վերծանիչ (դեկոդեր)

Վերծանիչը  $n$  մուտքերով և  $m$  ելքերով համակցական շղթա է, որի յուրաքանչյուր ելք իրականացնում է մուտքային փոփոխականների մեկ մինթերմ.

$$y_i = m_i(x_0, x_1, \dots, x_{n-1}), \quad i = 0, 1, \dots, N - 1, \quad (2.22)$$

որտե՛  $m_i, i=0, 1, \dots, N-1$  մինթերմներ են:

Վերծանիչի գրաֆիկական սիմվոլները ցույց են տրված նկ. 2.22-ում:



Նկ. 2.22.  $n$  մուտքերով և  $N$  ելքերով ( $n \times N$ ) վերծանիչի գրաֆիկական սիմվոլները

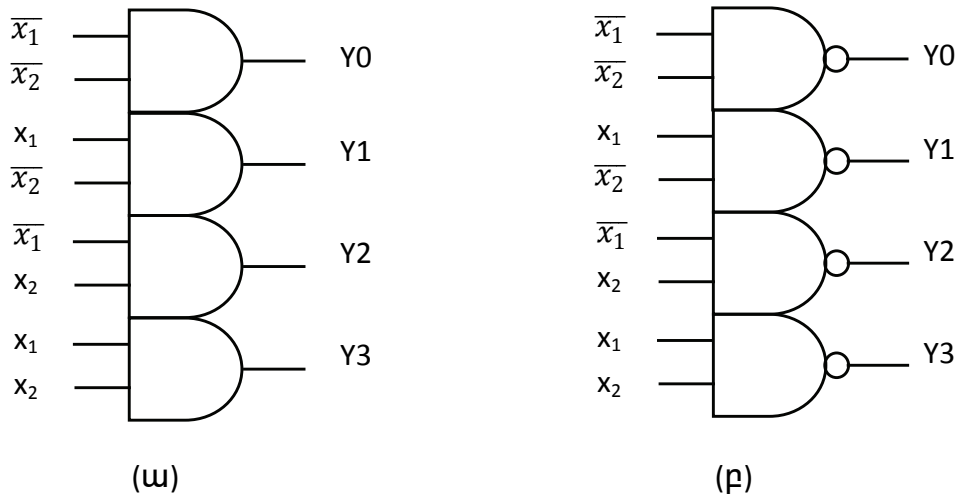
Վերծանիչի յուրաքանչյուր ելք համապատասխանում է մուտքային փոփոխականների մեկ հավաքածուի: Լրիվ վերծանիչն ունի  $N=2^n$  ելքեր, որոնցից յուրաքանչյուրը համապատասխանում է մուտքային փոփոխականների  $N=2^n$  հավաքածուներից մեկին:

Եթե վերծանիչի մուտքային փոփոխականների հավաքածուները դիտարկվեն որպես երկուական թվեր (կոդեր), ապա լրիվ վերծանիչի ելքերն իրականացնում են կոդի վերծանման ֆունկցիաներ՝ ժամանակի որևէ պահի ընտրված կամ ակտիվ ելքը որոշում է մուտքերին կիրառված հավաքածուն: Աղյուսակ 2.7-ում ցույց է տրված երկու մուտքերով և չորս ելքերով ( $2 \times 4$ ) վերծանիչի իսկության աղյուսակը: Մուտքային փոփոխականները կոչվում են նաև հասցեային փոփոխականներ, քանի որ յուրաքանչյուր ելք ընտրվում է մեկ հասցեով՝ հավաքածուով: Օրինակ,  $Y1$  ելքին համապատասխանում է  $x_2x_1=01$  հասցեն:

Աղյուսակ 2.7

$x_2x_1$	$Y0$	$Y1$	$Y2$	$Y3$
00	1	0	0	0
01	0	1	0	0
10	0	0	1	0
11	0	0	0	1

ԵՎ և ԵՎ-ՈՉ տարրերի վրա կառուցված  $2 \times 4$  վերծանիչի տրամաբանական սխեմաները ցույց են տրված նկ. 2.23-ում: Նկ. 2.23բ սխեմայում վերծանիչի ելքերը ժխտված են՝ ընտրված ելքն ունի տրամաբանական ցածր մակարդակ, իսկ մնացած ելքերն ունեն տրամաբանական բարձր մակարդակ:

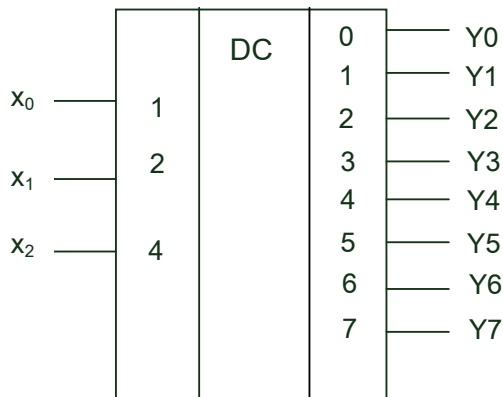


Նկ. 2.23.  $2 \times 4$  վերծանիչ: (ա) կառուցված ԵՎ տարրերով, (բ) կառուցված ԵՎ-ՈՉ տարրերով

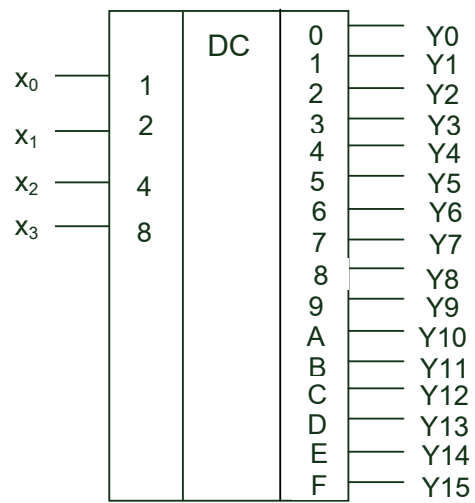
$3 \times 8$  վերծանիչը կարելի է դիտարկել որպես երկուական կոդը ութականի ձևափոխիչ (նկ. 2.24ա): Վերծանիչի երկուական մուտքերը հաճախ նշվում են երկուական քաշային գործակիցներով: Նույն կերպ,  $4 \times 16$  վերծանիչը կարելի է դիտարկել որպես երկուական կոդը տասնվեցական կոդի ձևափոխիչ (նկ. 2.24բ):

Գործնականում շատ հաճախ անհրաժեշտ է լինում երկուական-կոդավորված-տասական (ԵԿՏ, BCD (binary-coded-decimal)) կոդը ձևափոխել տասական թվանշանների: Այդպիսի ձևափոխիչը կարելի է իրագործել որպես  $4 \times 10$  ոչ լրիվ վերծանիչ (նկ. 2.25). Նկատենք, որ վերծանիչի միայն տասը առաջին մուտքային հավաքածուներն են պիտանի, մնացած վեց հավաքածուները՝ 1010, 1011, 1100, 1101, 1110, 1111 ավելցուկային են, դրանք երբեք չեն կիրառվելու վերծանիչի մուտքերին: Համենայն դեպս, շատ դեպքերում վերծանիչը կառուցվում է այնպես, որ նրա մուտքերին ավելցուկային հավաքածուներից որևէ մեկը կիրառելիս ելքերից ոչ մեկը չի ընտրվում:



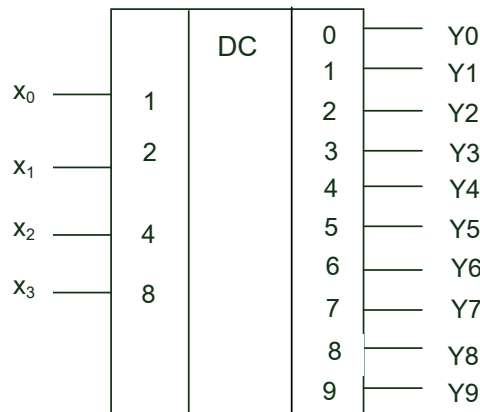


(ա)



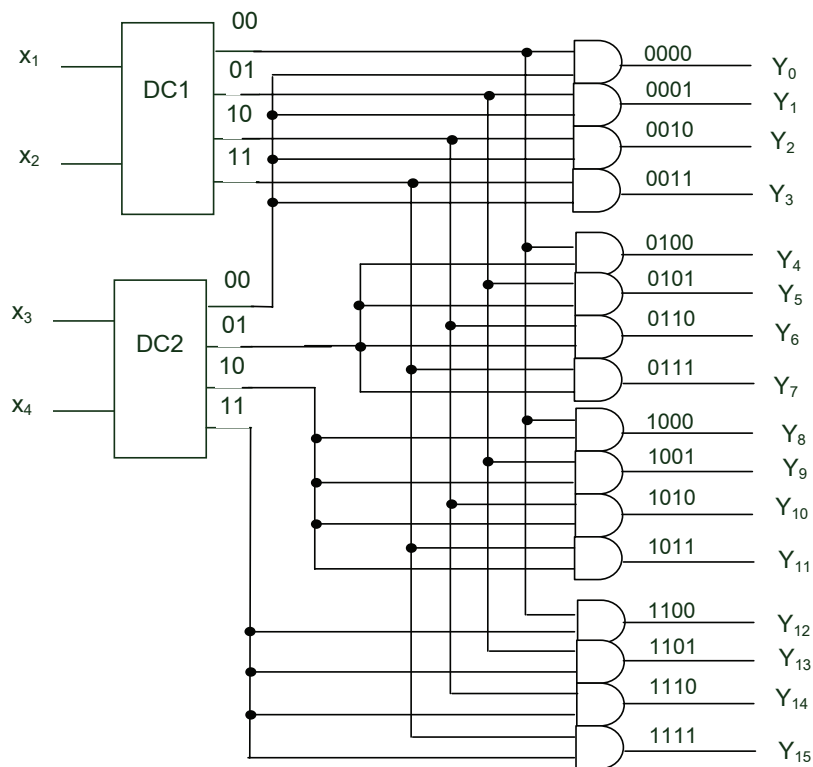
(բ)

Նկ. 2.24. (ա) 3x8 վերժանիչը կարելի է դիտարկել որպես երկուական կոդը ութականի ձևափոխիչ, (բ) 4x16 վերժանիչը կարելի է դիտարկել որպես երկուական կոդը տասնվեցական կոդի ձևափոխիչ



Նկ. 2.25. 4x10 ոչ լրիվ վերժանիչ, որը հաճախ օգտագործվում է որպես երկուական-կոդավորված-տասական (ԵԿՏ) կոդը տասական թվանշանների ձևափոխիչ

Այն կիրառությունները, որոնցում վերժանիչներն ունեն մեծ թվով մուտքեր և ելքեր, ինչպես օրինակ հիշասարքերի հասցեների վերժանիչները, վերժանիչի սխեման իրականացվում է բազմաստիճան կառուցվածքով: Նկ. 2.26-ում ցույց է տրված երկաստիճան կառուցվածքով 4x16 վերժանիչը: Առաջին աստիճանը պարունակում է երկու հատ 2x4 վերժանիչ՝ DC1 և DC2. Երկրորդ աստիճանը բաղկացած է 16 հատ 2ԵԿ տարրերից, որոնք իրականացնում են ԵԿ գործողություն DC1 և DC2 վերժանիչների ելքերի միջև:



Նկ. 2.26. Երկաստիճան 4x16 վերծանիչի սխեման

Բազմաստիճան վերծանիչները կարող են ունենալ ավելի մեծ հապաղումներ, քան մեկ աստիճանով կառուցվածները, քանի որ ազդանշանները մուտքից ելք են հասնում՝ անցնելով ավելի մեծ թվով տարրերով: Բայց բազմաստիճան վերծանիչներն ունեն ավելի պարզ սխեմաներ: Օրինակ, մեկ աստիճանով կառուցված 4x16 վերծանիչը բաղկացած է 16 հատ 4ԵՎ տարրերից: Սովորական ԿՄՕԿ իրականացման դեպքում 4ԵՎ տարրը պարունակում 10 տրանզիստոր: Հետևաբար, մեկ աստիճան 4x16 ԿՄՕԿ վերծանիչը պարունակում է  $16 \times 10 = 160$  տրանզիստոր: Երկաստիճան վերծանիչի առաջին աստիճանը բաղկացած է 8 հատ 2ԵՎ-ՈՉ տարրերից, երկրորդ աստիճանը՝ 16 հատ 2ԵՎ-ՈՉ տարրերից: Հետևաբար, վերծանիչը բաղկացած է 24 2ԵՎ-ՈՉ տարրից: Մեկ 2ԵՎ-ՈՉ տարրը պարունակում է 4 տրանզիստոր: Տրանզիստորների ընդհանուր թիվը երկաստիճան ԿՄՕԿ վերծանիչում կլինի  $24 \times 4 = 96$ :

Երկաստիճան վերծանիչում տրանզիստորների թվի կրճատումն ավելի արդյունավետ է դառնում վերծանիչի մուտքերի թվի մեծացման հետ: Դիտարկենք 6 մուտքով վերծանիչը: Մեկ աստիճանով իրականացման դեպքում  $6 \times 64$  վերծանիչը կպարունակի 64 հատ 6ԵՎ տարրեր, որոնցից յուրաքանչյուրն ընդգրկում է 14 տրանզիստոր: Տրանզիստորների ընդհանուր թիվը կլինի՝  $64 \times 14 = 896$ : Երկաստիճան կառուցվածքում առաջին աստիճանն ունի երկու հատ  $3 \times 8$  վերծանիչներ, իսկ երկրորդ աստիճանը՝ 64 հատ 2ԵՎ տարրեր: Յուրաքանչյուր  $3 \times 8$  վերծանիչ ունի 8 հատ 3ԵՎ տարրեր, որոնցից յուրաքանչյուրը բաղկացած է 8 տրանզիստորից: Տրանզիստորների ընդհանուր թիվը երկաստիճան վերծանիչում կլինի՝  $2 \times 8 \times 8 + 64 \times 6 = 514$ :

## 2.6.2. Դեմոլտիպլեքսոր

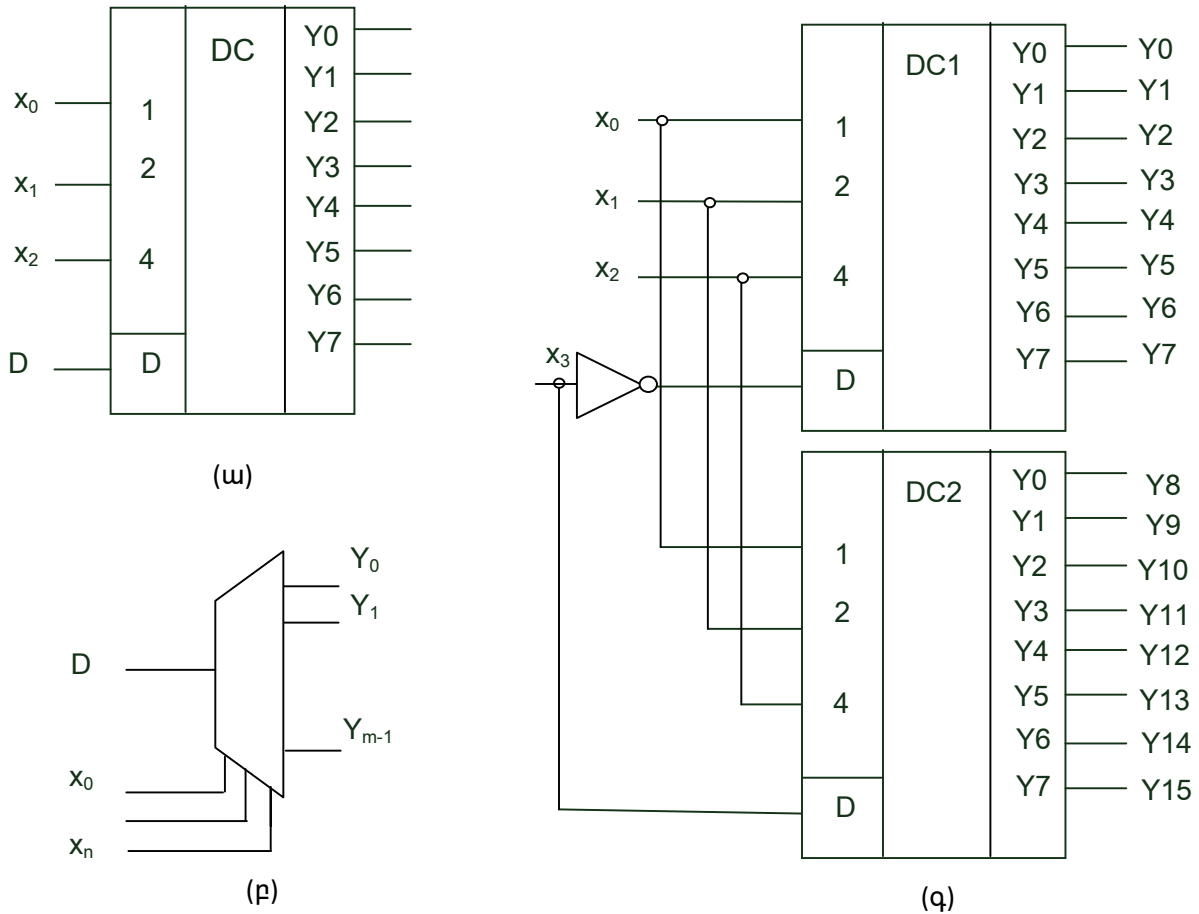
Դեմոլտիպլեքսորը թվային ֆունկցիա է, որն ունի  $n$  հասցեային մուտքեր, մեկ տվյալների մուտք և  $N=2^n$  ելքեր: Նկ. 2.27ա-ում ցույց է տրված  $3 \times 8$  դեմոլտիպլեքսորի գրաֆիկական սիմվոլը.  $x_2x_1x_0$  հասցեային փոփոխականները ընտրում են այն ելքը, որին պետք է փոխանցվի տվյալների  $D$  մուտքին կիրառված ազդանշանը: Օրինակ, եթե  $x_2x_1x_0=110$ , ապա  $D$  մուտքի տվյալը փոխանցվում է  $Y_6$  ելք: Դեմոլտիպլեքսորի ելքերի բուլյան ֆունկցիաները կարող են ներկայացվել հետևյալ կերպ.

$$y_i = D \& M_i(x_0, x_1, \dots, x_{n-1}), \quad i = 0, 1, \dots, N - 1: \quad (2.23)$$

Դեմոլտիպլեքսորների հիմնական կիրառությունը տվյալները մեկ մուտքային գծից  $2^n$  ելքային գծերի՝ հասցեատերերի, փոխանցելն է: Այս ֆունկցիան չափազանց օգտակար է թվային մոդուլների միջև տվյալների փոխանցման գծերի անհրաժեշտ թվի կրճատման համար: Տվյալները մեկ մոդուլից մյուսին փոխանցվում է ընդամենը մեկ գծով: Ընդունող մոդուլում տեղադրելով դեմոլտիպլեքսոր, մուտքային գծից ստացված տվյալները կարող է փոխանցել մոդուլում գտնվող  $2^n$  հասցեատերերից ցանկացածին:

Դեմոլտիպլեքսորը կարելի է օգտագործել որպես վերծանիչ: Դրա համար բավարար է, որ նրա տվյալների մուտքին կիրառվի հաստատուն տրամաբանակն  $1$ ՝  $D=1$ : Այս դեպքում տվյալների մուտքը կարելի է դիտել որպես թույլտվության մուտք: Եթե  $D=1$ , ապա դեմոլտիպլեքսորը գործում է որպես վերծանիչ, իսկ երբ  $D=0$ , ապա դեմոլտիպլեքսորի բոլոր ելքերը գտնվում են պասիվ՝  $0$  վիճակում, անկախ հասցեային փոփոխականների արժեքներից: Դեմոլտիպլեքսորի  $D$  մուտքը կարելի է օգտագործել վերծանիչի սխեմայի ընդարձակման համար՝ օգտագործելով նույնատիպ երկու դեմոլտիպլեքսորներ կարելի կառուցել երկու անգամ մեծ թվով ելքերով դեմոլտիպլեքսոր:

Նկ. 2.27գ-ում ցույց է տրված  $4 \times 16$  վերծանիչի սխեման՝ կառուցված երկու  $3 \times 8$  դեմոլտիպլեքսոր: Տվյալների մուտքերը դառնում են չորրորդ հասցեային մուտք՝  $x_3$ : Երբ  $x_3=0$ , ապա թույլ է տրվում  $DC1$ -ի աշխատանքը՝ կախված  $x_2x_1x_0$  մուտքերից ընտրվում է  $Y_0$ -ից  $Y_7$  ելքերից որևէ մեկը: Իսկ երբ  $x_3=1$ , ապա թույլ է տրվում  $DC2$ -ի աշխատանքը՝ կախված  $x_2x_1x_0$  մուտքերից ընտրվում է  $Y_8$ -ից  $Y_{15}$  ելքերից որևէ մեկը: ԻՍ արտադրողները ներկայացնում են ընդարձակվող դեմոլտիպլեքսոր-վերծանիչների մեծ բազմազանություն:



Նկ. 2.27. (ա) 3x8 դեմուլտիպլեքսոր, (բ)  $n \times 2^n$  դեմուլտիպլեքսորի պարզեցված սիմվոլը՝  $m=2^n$ , (գ) 3x8 դեմուլտիպլեքսորի սխեմայի ընդարձակումը 4x16-ի

Նկ. 2.28ա-ում ցույց է տրված K555ИД4 դեմուլտիպլեքսորի գրաֆիկական սիմվոլը: Այս ԻՍ-ը պարունակում է երկու հատ 2x4 դեմուլտիպլեքսոր, որոնք ունեն ընդհանուր  $x_0x_1$  ընտրության (հասցեային) մուտքեր: A դեմուլտիպլեքսորի տվյալների մուտքերն են DA, SA, ելքերը՝ A0,..., A3.

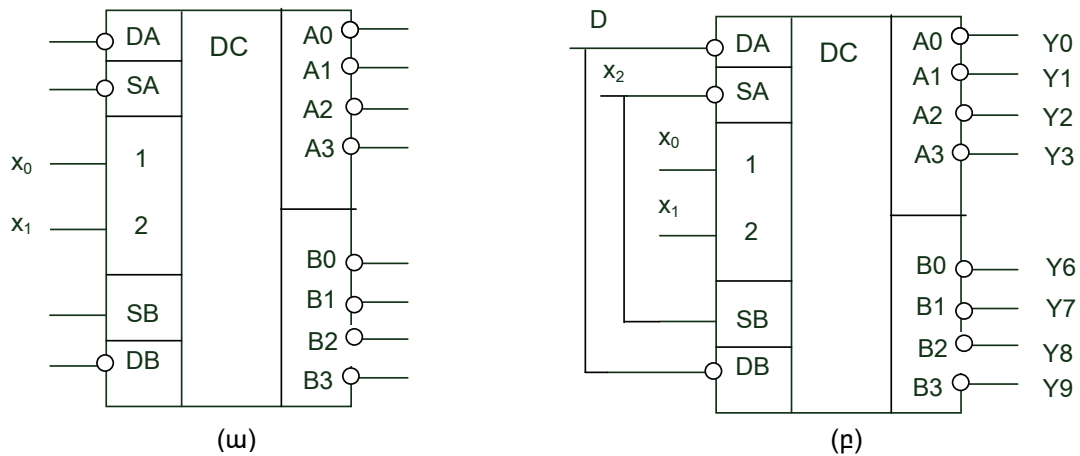
$$A_i = \overline{DA} \& \overline{SA} \& M_i(x_0, x_1), \quad i = 0, \dots, 3 : \quad (2.24)$$

B դեմուլտիպլեքսորի տվյալների մուտքերն են DB, SB, ելքերը՝ B0,..., B3.

$$B_i = \overline{DB} \& \overline{SB} \& M_i(x_0, x_1), \quad i = 0, \dots, 3 : \quad (2.25)$$

Դեմուլտիպլեքսորի շղթայի ընդարձակման հարմարության համար SB մուտքը ժխտված չէ: Նկ. 2.28բ-ում ցույց է տրված դեմուլտիպլեքսորի ընդարձակումը SA և SB մուտքերից: Դրանք օգտագործվում են որպես երրորդ հասցեային մուտք՝  $x_2$ : Եթե  $x_2=0$ , ապա A0,..., A3 ելքերը թույլատրվում են, իսկ B0,..., B3 ելքերն արգելվում են: Եթե  $x_2=1$ , ապա B0,..., B3 ելքերն թույլատրվում են, իսկ A0,..., A3 ելքերը արգելվում են: Արդյունադար շղթան 3x8 դեմուլտիպլեքսոր է  $x_0, x_1, x_2$  ընտրության մուտքերով, տվյալների D մուտքով և Y0,..., Y7 ելքերով.

$$Y_i = \overline{D} \& M_i(x_0, x_1, x_2), \quad i = 0, \dots, 7 : \quad (2.26)$$



Նկ. 2.28. K555MD4 տիպի դեմուլտիպլեքտոր. (ա) գրաֆիկական սիմվոլը, (բ) 3x8 ընդարձակված մուլտիպլեքտոր

### 2.6.3. Համակցական շղթաների իրագործումը դեմուլտիպլեքտորների և վերծանիչների միջոցով

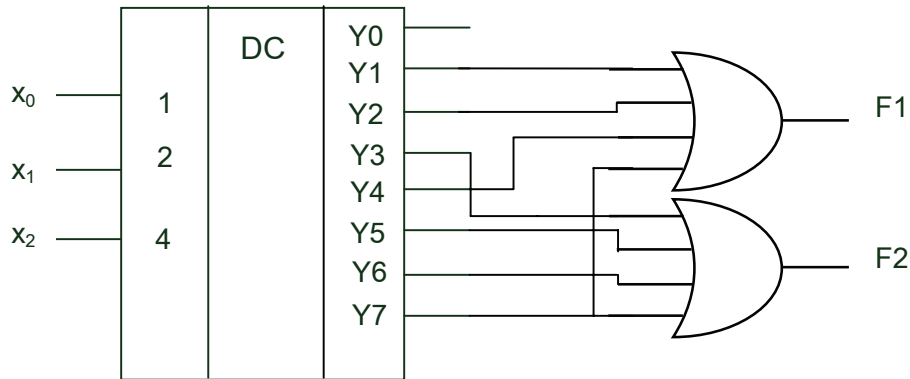
Վերծանիչը տրամադրում է  $2^n$  մինիստերմներ  $n$  մուտքային փոփոխականներից: Քանի որ ցանկացած բուլյան ֆունկցիա կարող է ներկայացվել ԿԴՆԶ-ով՝ մինիստերմների գումարով, ապա ֆունկցիայի իրականացման համար անհրաժեշտ է մինիստերմներին համապատասխանող վերծանիչի ելքերը միավորել ԿԱՄ տարրով: Այս եղանակով  $n \times 2^n$  վերծանիչի և ԿԱՄ տարրով կարելի է իրականացնել ցանկացած  $n$  փոփոխականի ֆունկցիա՝ կազմված  $m$  մինիստերմներից: Համակցական շղթայի կառուցման համար անհրաժեշտ է ԿԱՄ տարրի մուտքերին միացնել վերծանիչի այն ելքերը, որոնք համապատասխանում են իսկության աղյուսակում ֆունկցիայի 1 արժեքներին:

**Օրինակ 2.6.** Կառուցել համակցական շղթա, որը նկարագրվում է հետևյալ բուլյան ֆունկցիաներով.

$$F1 = \Sigma(1, 2, 4, 7), \quad (2.28)$$

$$F2 = \Sigma(3, 5, 6, 7): \quad (2.29)$$

Քանի որ մինիստերմների քանակը 8-ը չի գերազանցում, ուրեմն առկա են 3 մուտքային փոփոխականներ: Շղթան կարելի է կառուցել 3x8 վերծանիչով և երկու 4ԿԱՄ տարրերով: Նկ. 2.29-ում ցույց է տրված համակցական շղթան:

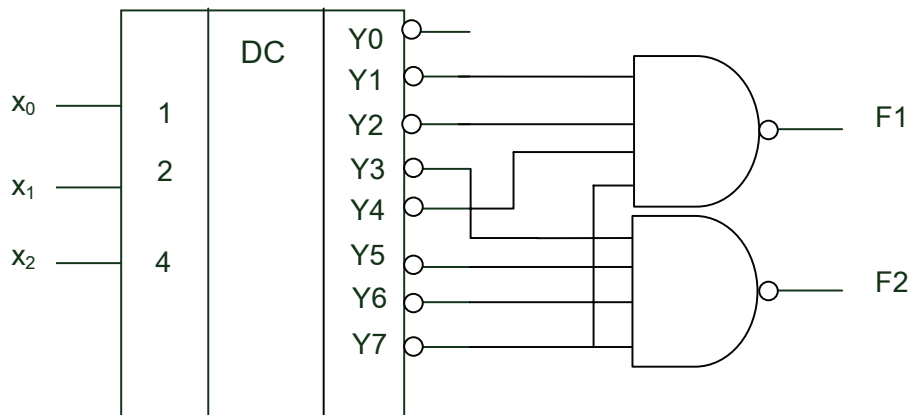


Նկ. 2.29. Վերծանիչի և ԿԱՄ տարրերի միջոցով կառուցված համակցական շղթա

Այն դեպքում, երբ վերծանիչի ելքերը ժխտված են (օրինակ, ինչպես K555ИД4-ի դեպքում), ԿԱՄ տարրի փոխարեն պետք է օգտագործել ԵՎ-ՈՉ տարր՝ վերծանիչի համապատասխան ելքերը միավորելու համար:

$$F = M_1 + M_2 + \dots + M_k = \overline{\overline{M_1} + \overline{M_2} + \dots + \overline{M_k}} : \quad (2.30)$$

Վերը բերված (2.28) և (2.29) ֆունկցիաներով նկարագրվող սարքի իրականացումը ժխտված ելքերով վերծանիչի և ԵՎ-ՈՉ տարրերի միջոցով ցույց է տրված նկ. 2.30-ում:



Նկ. 2.30. Ժխտված ելքերով վերծանիչի և ԵՎ-ՈՉ տարրերի միջոցով կառուցված համակցական շղթա

Մեծ թվով մինսերմներից բաղկացած ֆունկցիայի դեպքում կպահանջվի մեծ թվով մուտքերով ԿԱՄ տարր: Եթե  $F$  ֆունկցիան ունի  $k$  մինսերմ, ապա  $\bar{F}$  ժխտված ֆունկցիան կունենա  $2^n - k$  մինսերմ: Եթե  $k > 2^n/2$ , ապա  $\bar{F}$  ժխտված ֆունկցիան կարտահայտվի ավելի քիչ թվով մինսերմներով, քան  $F$ -ը՝  $2^n - k < k$ : Նման դեպքերում ավելի նպատակահարմար է օգտագործել ԿԱՄ-ՈՉ տարր  $\bar{F}$  ժխտված ֆունկցիայի մինսերմները միավորելու համար: ԿԱՄ-ՈՉ տարրի ելքում կձևավորվի  $F$  ուղիղ ֆունկցիան:

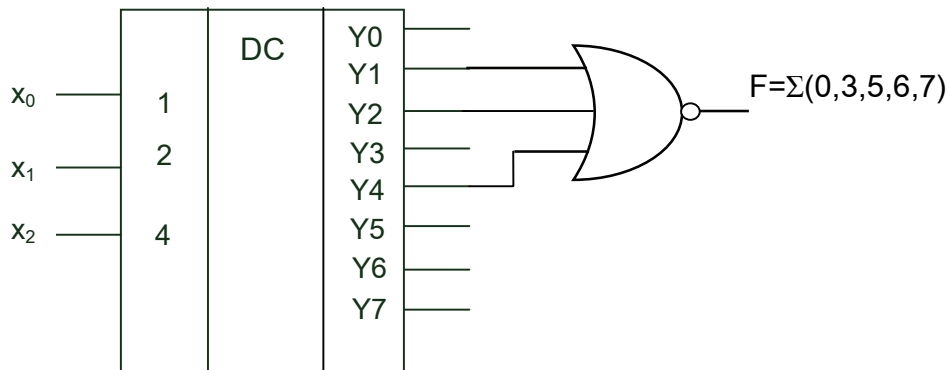
**Օրինակ 2.7.** Կառուցել համակցական շղթա, որը նկարագրվում է հետևյալ բուլյան ֆունկցիայով:

$$F = \Sigma(0,3,5,6,7): \quad (2.31)$$

Այս երեք փոփոխականի ֆունկցիան պարունակում է 5 միներմներ, ուստի ավելի նպատակահարմար է այն կառուցել ժխտված ֆունկցիայի միներմներով.

$$\bar{F} = \Sigma(1,2,4): \quad (2.32)$$

Նկ. 2.31-ում ցույց են տրված (2.31) կամ դրան համարժեք (2.32) բանաձևով նկարագրվող համակցական սարքի սխեման:



Նկ. 2.31. Համակցական սարքի սխեմայի իրականացումը ժխտված ֆունկցիայի միներմներով

#### 2.6.4. Կոդավորող սարք (կոդեր, շիֆրատոր)

Կոդավորող սարքը կատարում է վերծանիչի հակադարձ գործողությունը՝ որևէ տրված սիմվոլի համար գեներացնում է երկուական կոդ: Կոդավորող սարքի ամենատարածված կիրառությունը ստեղծաշարն է, որը յուրաքանչյուր ստեղծի համար գեներացնում է առանձին երկուական կոդ: Օրինակ, համակարգչի ստեղծաշարը գեներացնում է 8-բիթ կոդեր ASCII (American Standard Code for Information Interchange, տվյալների փոխանակաման ամերիկյան ստանդարտ կոդի ֆորմատով):

Կոդերը թվային ֆունկցիա է, որն ունի  $2^n$  մուտքեր և  $n$  ելքեր: Ելքերում ձևավորվում են մուտքային գծերին համապատասխանող երկուական կոդերը՝ յուրաքանչյուր մուտքին համապատասխանում է միայն մեկ հավաքածու, որը տարբեր է մնացածներից: Որպես օրինակ դիտարկենք 8 մուտքերով կոդերի կառուցումը: Ութ մուտքային գծեր կոդավորելու համար պետք է ունենալ 3 ելքային գծեր: Աղյուսակ 2.8-ում ցույց է տրված 8-մուտք կոդերի կրճատված իսկության աղյուսակը՝ համարելով, որ ցանկացած պահի ակտիվ է D0,...,D7 մուտքերից միայն մեկը (սեղմված է միայն մեկ ստեղծ): Ելքային եռաբիթ կոդի բիթերը նշանակված են A0, A1, A2: Ելքային A2A1A0=000 հավաքածուն համապատասխանում է և՛ D0 մուտքին, և այն դեպքին, երբ մուտքերից ոչ մեկն ակտիվ չէ (ոչ մի ստեղծ սեղմված չէ): Այս իրավիճակը շտկելու համար, սովորաբար, կոդերին ավելացվում է ևս մեկ ելք՝ G, որը հավասար է 0-ի, երբ ոչ մի մուտք ակտիվ չէ, և հավասար է 1-ի, երբ մուտքերից որևէ մեկն ակտիվ է: G ազդանշանն օգտագործվում է նաև սպասարկող սարքին տեղեկացնելու համար, որ պետք է արձագանքել ստեղծաշարից տրվող հրահանգներին: Կոդերի ելքային ֆունկցիաները կարելի է կառուցել հետևյալ դատողություններով: Ելքի ցածր կարգի A0 բիթը հավասար է 1-ի,

երբ ակտիվ է որևէ կենտ համարի մուտք՝  $D_1$ , կամ  $D_3$ , կամ  $D_5$ , կամ  $D_7$ : Ելքի  $A1$  բիթը հավասար է 1-ի, երբ ակտիվ է  $D_2$ ,  $D_3$ ,  $D_6$ ,  $D_7$  մուտքերից որևէ մեկը: Ելքի  $A2$  բիթը հավասար է 1-ի, երբ ակտիվ է  $D_4$ ,  $D_5$ ,  $D_6$ ,  $D_7$  մուտքերից որևէ մեկը: Կոդերը նկարագրող տրամաբանական ֆունկցիաների համակարգը կունենա հետևյալ տեսքը.

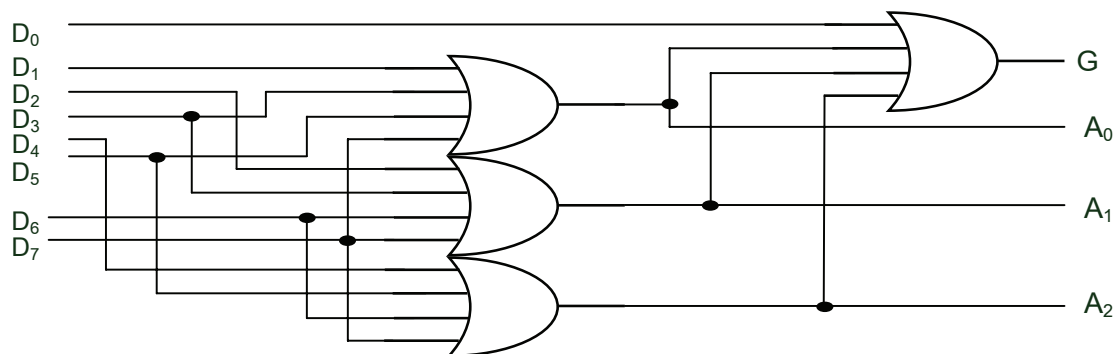
$$\begin{aligned} A0 &= D1 + D3 + D5 + D7, \\ A1 &= D2 + D3 + D6 + D7, \\ A2 &= D4 + D5 + D6 + D7, \end{aligned} \quad (2.33)$$

$$G = D0 + D1 + D2 + D3 + D4 + D5 + D6 + D7 = D0 + A0 + A1 + A2:$$

Դեկոդերի ելքային  $A2A1A0$  կոդային հավաքածուն կարելի է ընթերցել միայն, երբ  $G=1$ : Դեկոդերի սխեման կառուցած 4ԿԱՄ-ՈՉ տարրերի միջոցով ցույց է տրված նկ. 2.32-ում:

Աղյուսակ 2.8

D0	D1	D2	D3	D4	D5	D6	D7	A2	A1	A0	G
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	1	1	1
0	0	0	0	1	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	1	0	1	1
0	0	0	0	0	0	1	0	1	1	0	1
0	0	0	0	0	0	0	1	1	1	1	1



Նկ. 2.32. Ութ մուտքերով (8-ը 3-ի) դեկոդեր

Նկ. 2.32-ում ցույց տրված կոդերի ճիշտ աշխատանքի համար անհրաժեշտ է, որ ժամանակի ցանկացած պահի մուտքերից միայն մեկը հավասար լինի 1-ի (միայն մեկ ստեղծ կարող է սեղմված լինել), այլապես այդ շղթան որևէ իմաստ չունի: Օրինակ, եթե միաժամանակ  $D_2$ -ն ու  $D_5$ -ը հավասար լինեն 1-ի, ապա ելքում կստացվի  $A2A1A0=111$ , որը համապատասխանում է  $D_7$  մուտքին և տվյալ դեպքում ոչ մի իմաստ չունի: Ութ մուտքերի դեպքում առկա են  $2^8=256$  մուտքային հավաքածուներ, որոնցից միայն ութը տվյալ դեպքում իմաստ ունեն: Մնացած հավաքածուները ավելցուկային են: Այն դեպքերում, երբ հնարավոր է, որ մեկից ավելի մուտքեր միաժամանակ ունենան 1 ար-



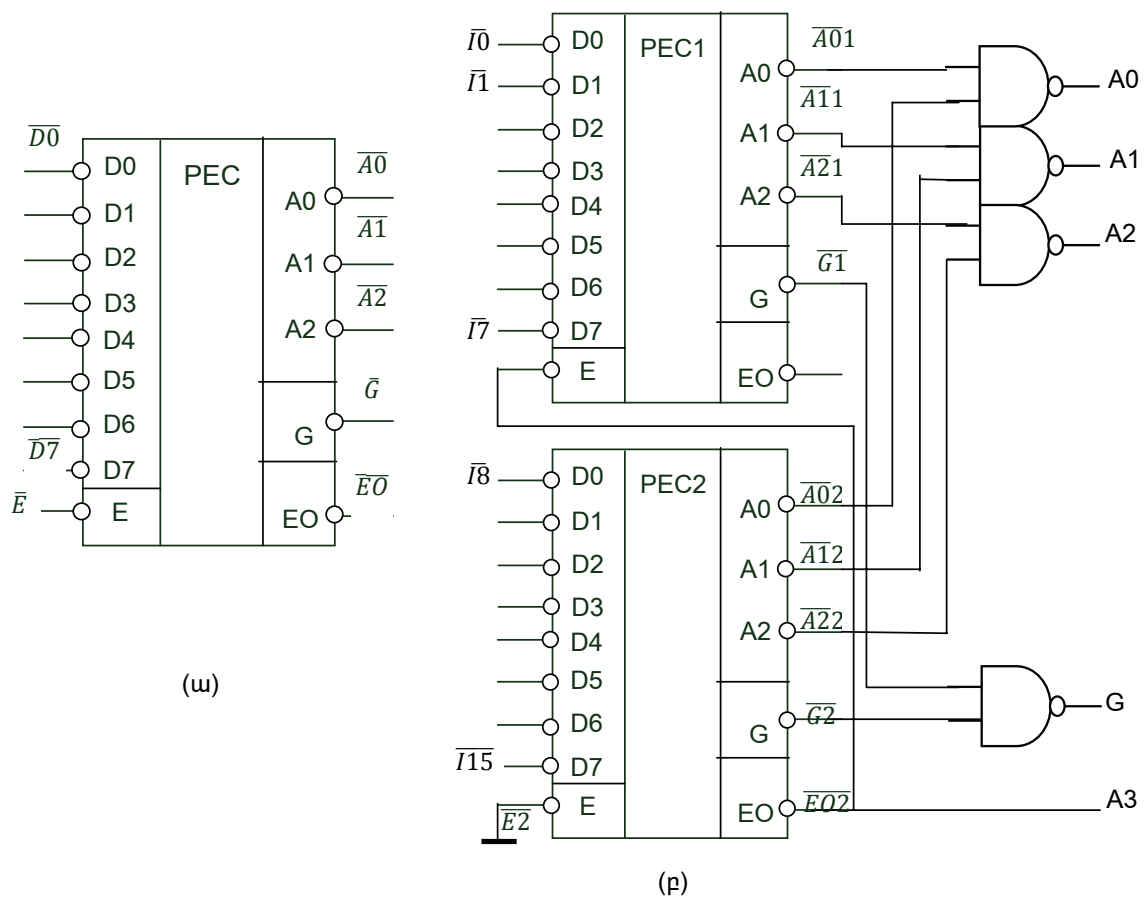
Ժեք, դեկոդերի ստույգ աշխատանքի համար պետք է սահմանվեն մուտքերի առաջնահերթություններ: Երբ մի քանի մուտքեր միաժամանակ ունեն 1 արժեք, ելքերում ձևավորվում է դրանցից ամենաբարձր առաջնահերթություն ունեցող մուտքի կոդը: Այդպիսի կոդավորմամբ սարքը կոչվում է առաջնահերթությամբ դեկոդեր: Աղյուսակ 2.9-ում ցույց է տրված 8-մուտք առաջնահերթությամբ դեկոդերի իսկության աղյուսակը, որտեղ ամենաբարձր առաջնահերթություն ունի ամենաբարձր համարով մուտքը՝ D7-ը: “X” սիմվոլը համապատասխանում է կամայական տրամաբանական արժեքի: Օրինակ, եթե D<sub>2</sub>-ն ու D<sub>5</sub>-ը միաժամանակ հավասար լինեն 1-ի, ապա ելքում կստացվի A2A1A0=111, քանի որ D<sub>5</sub>-ի առաջնահերթությունը բարձր է D<sub>2</sub>-ինից:

Աղյուսակ 2.9

D0	D1	D2	D3	D4	D5	D6	D7	A2	A1	A0
1	0	0	0	0	0	0	0	0	0	0
X	1	0	0	0	0	0	0	0	0	1
X	X	1	0	0	0	0	0	0	1	0
X	X	X	1	0	0	0	0	0	1	1
X	X	X	X	1	0	0	0	1	0	0
X	X	X	X	X	1	0	0	1	0	1
X	X	X	X	X	X	1	0	1	1	0
X	X	X	X	X	X	X	1	1	1	1

Ութ մուտքերով K555UB1 (SN74LS148N) տիպի առաջնահերթությամբ կոդերն ունի ժխտված  $\overline{A0}$ ,  $\overline{A1}$ ,  $\overline{A2}$  ու  $\overline{G}$  ելքեր և  $\overline{D0} \dots \overline{D7}$  մուտքեր(նկ. 2.33ա):  $\overline{E}$  մուտքը թույլատրում է կոդերի աշխատանքը՝ երբ  $\overline{E} = 0$ , կոդերը գտնվում է նորմալ աշխատանքային ռեժիմում, իսկ երբ  $\overline{E} = 1$ , կոդերը գտնվում է անջատված վիճակում՝  $\overline{A0}$ ,  $\overline{A1}$ ,  $\overline{A2}$  ու  $\overline{G}$  ելքերը գտնվում են 1 վիճակում (պասիվ վիճակում) անկախ  $\overline{D0} \dots \overline{D7}$  մուտքերի վիճակներից:  $\overline{EO}$  ելքը ցույց է տալիս կոդերի վիճակը՝  $\overline{EO} = 0$ , եթե  $\overline{D0} \dots \overline{D7}$  մուտքերից ոչ մեկը ակտիվ չէ, և  $\overline{EO} = 1$ , եթե  $\overline{D0} \dots \overline{D7}$  մուտքերից որևէ մեկն ակտիվ է: Երբ  $\overline{E} = 1$ ,  $\overline{EO} = 1$  անկախ մյուս մուտքերի վիճակներից:  $\overline{EO}$  ելքն օգտագործվում է կոդերի շղթայի ընդարձակման համար: Նկ. 2.33բ-ում ցույց է տրված 16 մուտքերով կոդերի սխեման կառուցված երկու 8-մուտք կոդերների միջոցով: Ամենաբարձր առաջնահերթություն ունի  $\overline{I15}$  մուտքը:

Երբ ակտիվ է  $\overline{I8} \dots \overline{I15}$  մուտքերից որևէ մեկը, ապա  $\overline{EO2} = 1$ , որն արգելում է PEC1 կոդերի աշխատանքը՝  $\overline{A01} = \overline{A11} = \overline{A21} = \overline{G1} = 1$ : 2ԵՎ-ՈՉ տարրերի A2, A1, A0, G ելքերը միարժեք որոշվում են PEC2 կոդերի ելքերի վիճակներով՝  $A0 = \overline{\overline{A02}}$ ,  $A1 = \overline{\overline{A12}}$ ,  $A2 = \overline{\overline{A22}}$ ,  $G = \overline{\overline{G2}}$ :



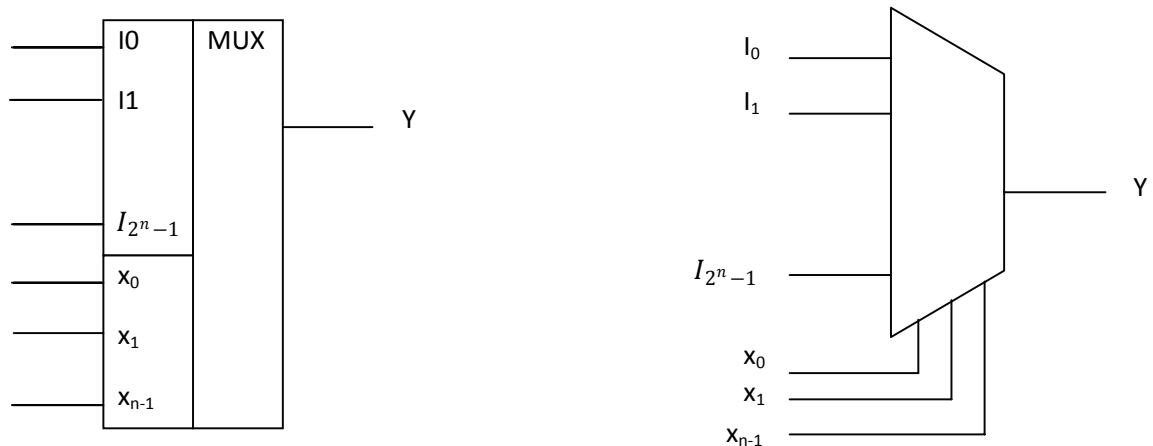
Նկ. 2.33. Ութ մուտքերով K555UB1 (SN74LS148N) տիպի առաջնահերթությամբ կոդեր (ա), կոդերի սխեմայի ընդարձակումը 16 մուտքերի համար (բ)

Երբ PEC2-ի մուտքերից ոչ մեկն ակտիվ չէ, ապա  $\overline{EO}2 = 0$  (քանի որ  $\overline{E}2 = 1$ ), որը թույլատրում է PEC1 կոդերի աշխատանքը, շեվ-ՈՉ տարրերի A2, A1, A0, G ելքերը միարժեք որոշվում են PEC1 կոդերի ելքերի վիճակներով: Ելքային կոդի A3 բարձր բիթը հավասար է 1-ի, երբ աշխատում է PEC2 կոդերը՝  $A3 = \overline{EO}2 = 1$ : Իսկ երբ աշխատում է PEC1 կոդերը, ապա  $A3 = \overline{EO}2 = 0$ :

## 2.6.5. Մուլտիպլեքսոր

Մուլտիպլեքսորն նշանակում է՝ ինֆորմացիայի մեծ թվով միավորների փոխանցում փոքր թվով հոսքուղիներով կամ կապի գծերով: Թվային մուլտիպլեքսորը համակցական սարք է, որն ընտրում է բազմաթիվ մուտքերից մեկը և այն միացնում միակ ելքին: Ընդհանուր դեպքում մուլտիպլեքսորն ունի թվով  $2^n$  տվյալների մուտքեր, մեկ ելք և  $n$  ընտրության մուտքեր, որոնց հավաքածուները որոշում են, թե որ մուտքը պետք է միացնել ելքին: Առկորաբար այդպիսի պարամետրերով մուլտիպլեքսորը կոչվում է “ $2^n$ -ից 1” մուլտիպլեքսոր: Մուլտիպլեքսորի գրաֆիկական սիմվոլը ցույց է տրված նկ. 2.34-ում: “ $2^n$ -ից 1” մուլտիպլեքսորի ելքային ֆունկցիան տրվում է հետևյալ բանաձևով.

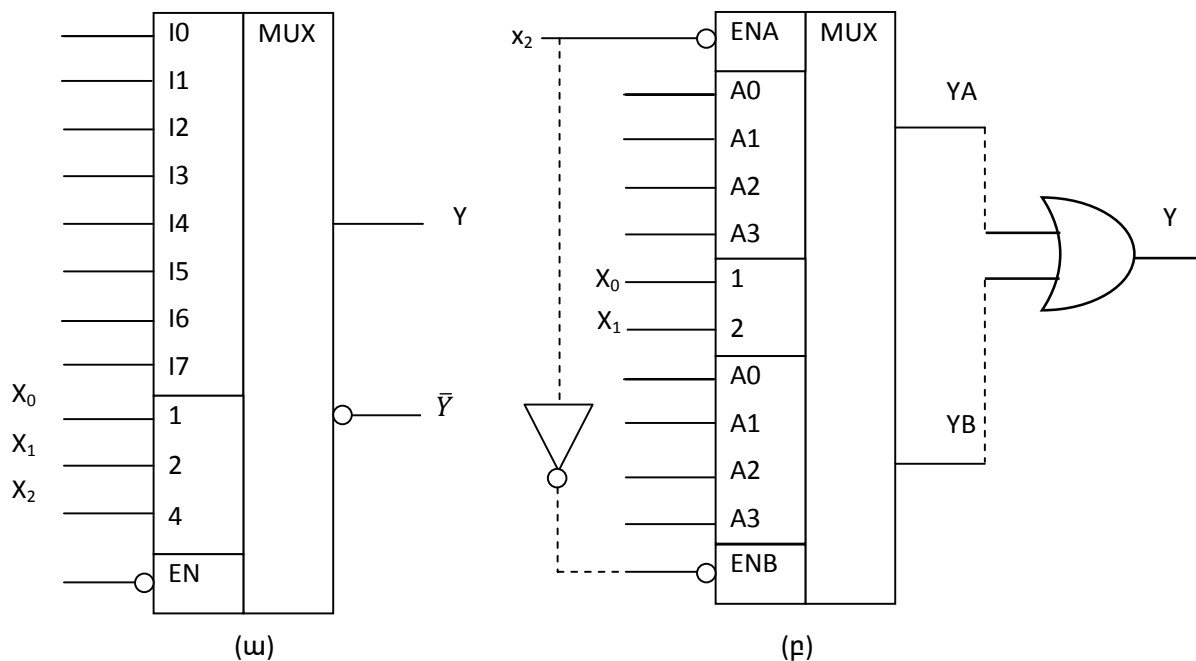
$$Y = \sum_{k=0}^{2^n-1} I_k m_k(x_0, x_1, \dots, x_{n-1}), \quad (2.24)$$



Նկ. 2.34. “ $2^n$ -ից 1” մուլտիպլեքսորի գրաֆիկական սիմվոլը

որտեղ  $I_0, \dots, I_{2^n-1}$  տվյալների մուտքերն են,  $x_1$ -ից  $x_{n-1}$ -ը ընտրության մուտքերն են,  $Y$ -ը ելքն է: Ընտրության մուտքերի յուրաքանչյուր հավաքածու ընտրում է մեկ տվյալի մուտք, որը պետք է փոխանցվի ելք: Ընտրված տվյալի մուտքի համարը հավասար է ընտրության մուտքերի հավաքածուով որոշվող թվին: Օրինակ, եթե ընտրության մուտքերին տրված հավաքածուն 101 է, ապա ելք է փոխանցվում 15 մուտքի տվյալը:

Մուլտիպլեքսորը հիմնական տրամաբանական ֆունկցիաներից է, և արտադրությունում թողարկվում են այդ ֆունկցիան իրականացնող բազմաթիվ ԻՍ-եր: Այսպես, նկ. 2.35-ում ցույց են տրված 74151 (K555KП7) և 74153 (K555KП2) ինտեգրալ մուլտիպլեքսորների սիմվոլները: 74151-ը “8-ից 1” մուլտիպլեքսոր է՝ նորմալ և ժխտված ելքերով: Այն ունի մաս թույլտվության EN մուտք՝ երբ  $EN=0$ , մուլտիպլեքսորն աշխատում է նորմալ ռեժիմում, իսկ երբ  $EN=1$ , մուլտիպլեքսորի ֆունկցիան արգելված է, և ելքը գտնվում 0 վիճակում: EN մուտքը սովորաբար օգտագործվում է մուլտիպլեքսորի ֆունկցիայի ընդարձակման համար՝ “8-ից 1” մուլտիպլեքսորի երկու 74151 ԻՍ-երով կարելի է կառուցել մեկ “16-ից 1” մուլտիպլեքսոր՝ EN մուտքն օգտագործելով որպես բարձր կարգի ընտրության  $x_3$  բիթի մուտք: Տվյալների ցածր կարգերի բիթերի մուտքերով ( $I_0, \dots, I_7$ ) 74151 ԻՍ-ի EN մուտքին պետք է կիրառել  $x_3$ -ը, իսկ տվյալների բարձր կարգերի բիթերի մուտքերով ( $I_8, \dots, I_{15}$ ) 74151 ԻՍ-ի EN մուտքին պետք է կիրառել  $\overline{x_3}$ : 74153 ԻՍ-ը ներառում է երկու “4-ից 1” մուլտիպլեքսորներ, որոնք ունեն ընդհանուր  $x_0, x_1$  ընտրության մուտքեր: EA և EB թույլտվության մուտքերը հնարավորություն են տալիս ընդարձակել մուլտիպլեքսորի ֆունկցիան: Կետագծերով ցույց է տրված, թե ինչպես կարելի է երկու “4-ից 1” մուլտիպլեքսորների հիման վրա կառուցել մեկ “8-ից 1” մուլտիպլեքսորներ:



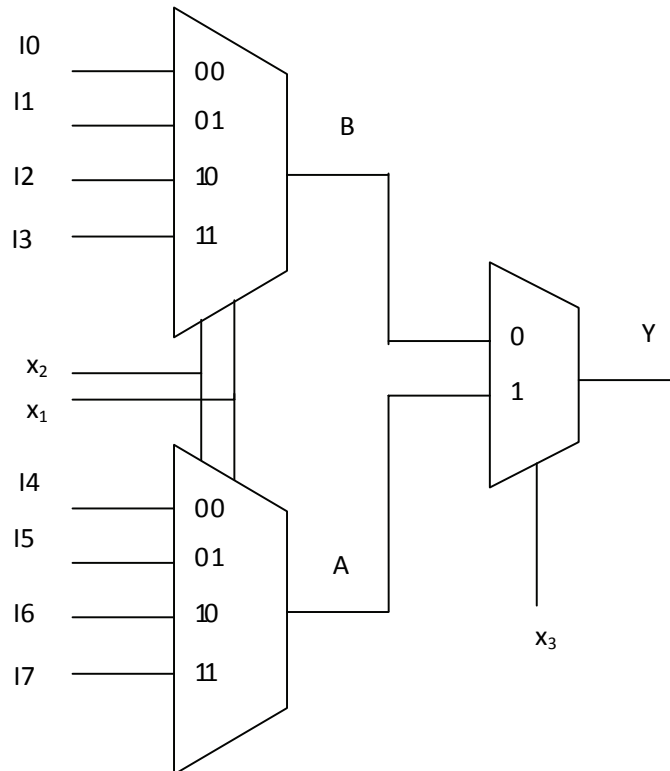
Նկ. 2.35. Ինտեգրալ մուլտիպլեքսորներ՝ (ա) 74151 (K555KП7), (բ) 74153 (K555KП2)

Լայն սպառման ԿՍՕԿ ԻՍ-երի շարքերում թողարկվող մուլտիպլեքսորները չորջեղի են. դրանք կարող են իրականացնել ինչպես  $I_0, \dots, I_{m-1}$  տվյալների մուտքերով և  $Y$  ելքով մուլտիպլեքսորի ֆունկցիան, այնպես էլ տվյալի  $Y$  մուտքով և  $I_0, \dots, I_{m-1}$  տվյալների ելքերով դեմուլտիպլեքսորի ֆունկցիան: ԿՍՕԿ CD4051AE (K561KП2) մուլտիպլեքսորն ունի նույն սիմվոլը, ինչպես 74153-ը (Նկ. 2.35բ), բայց երբ թույլտվության մուտքերին տրվում է արգելիչ մակարդակ, ելքերը գտնվում են Hi-Z վիճակում: Այս հատկությունը ավելի է հեշտացնում մուլտիպլեքսորի ընդարձակումը. ելքերը կարելի միացնել միմյանց ամիջապես՝ առանց լրացուցիչ ԿԱՄ տարրի:

Մուլտիպլեքսորի ֆունկցիայի ընդարձակման մեկ այլ ընդհանուր մոտեցում է “2<sup>n</sup>–ից 1” մուլտիպլեքսորի իրականացումը բազմաստիճան կառուցվածքով: Դրա համար ընտրության  $n$  մուտքերը պետք է բաժանել խմբերի: Յուրաքանչյուր խմբում ընտրության բիթերի թիվը հավասար է մուլտիպլեքսորի համապատասխան աստիճանի ընտրության գծերի թվին: Նկ. 2.36-ում ցույց է տրված երկաստիճան “8–ից 1” մուլտիպլեքսորների կառուցվածքը, այն բաղկացած է երկու “4–ից 1” (առաջին աստիճան) և մեկ “2–ից 1” (երկրորդ աստիճան) մուլտիպլեքսորներից:

Դիտարկենք ևս մեկ օրինակ, կառուցենք “32–ից 1” մուլտիպլեքսոր՝ օգտագործելով “4–ից 1” և “2–ից 1” մուլտիպլեքսորներ: Քանի որ ունենք տվյալների 32 մուտքեր, ապա պետք է ունենանք 5 ընտրության մուտքեր ( $2^5=32$ )՝  $x_5x_4x_3x_2x_1$  հաջորդականությամբ: Առաջին աստիճանում պետք է օգտագործել 8 հատ “4–ից 1” մուլտիպլեքսորներ  $x_2x_1$  ընտրության գծերով: Առաջին աստիճանի մուլտիպլեքսորների 8 ելքերը ծառայում են երկրորդ աստիճանի երկու “4–ից 1” մուլտիպլեքսորների տվյալների մուտքեր: Երկրորդ աստիճանի մուլտիպլեքսորների համար ընտրության մուտքեր են  $x_4$  և  $x_3$ -ը: Վերջապես, երրորդ աստիճանում օգտագործվում է մեկ “2–ից 1” մուլտիպլեքսոր՝ որի

համար տվյալների մուտքեր են երկրորդ աստիճանի մուլտիպլեքսորների ելքերը, իսկ ընտրության մուտքը  $x_5$ -ն է:

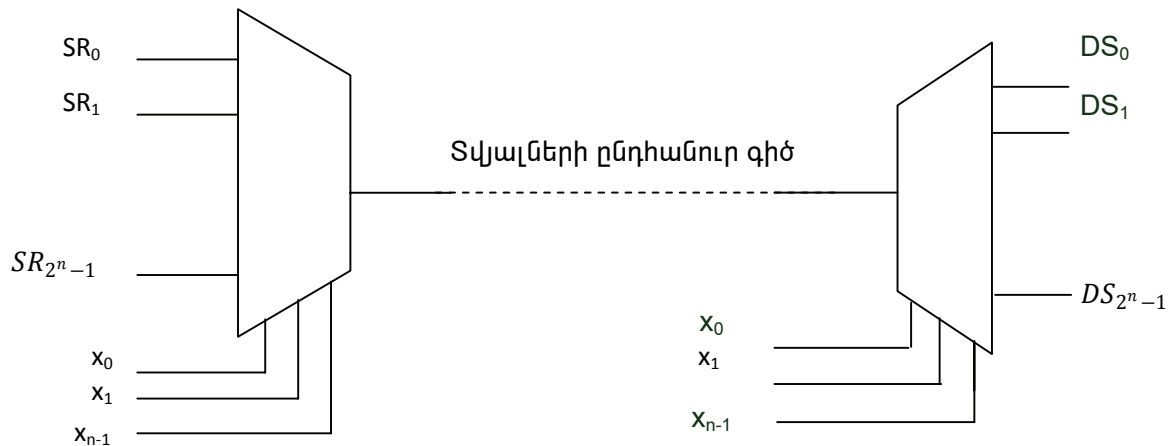


Նկ. 2.36. Երկաստիճան “8–ից 1” մուլտիպլեքսորներ՝ բաղկացած է երկու “4–ից 1” (առաջին աստիճան) և մեկ “2–ից 1” (երկրորդ աստիճան) մուլտիպլեքսորներից

Մուլտիպլեքսորները և դեմուլտիպլեքսորները լայնորեն օգտագործվում են տվյալների փոխանցման համակարգերում՝ կապի գծերի թվի նվազեցման համար: Այդպիսի համակարգի պարզեցված սխեման ցույց է տրված նկ. 2.37-ում: Հաղորդող կողմում  $x_0, \dots, x_{n-1}$  ընտրության փոփոխականներով ընտրվում է տվյալների ( $SR_k$ ) աղբյուրներից մեկը և փոխանցվում տվյալների ընդհանուր գիծ: Ընդունող կողմում ընտրության փոփոխականներն ընտրում են համապատասխան ընդունիչը՝  $DS_k$ , որը պետք է միացնել տվյալների ընդհանուր գծին: Եթե տվյալների ընդհանուր գիծը պետք է հաջորդաբար տրամադրել բոլոր “ $SR_i$ -ից- $DS_i$ ” տվյալների հոսքուղիներին, ապա ընտրության  $x_0, \dots, x_{n-1}$  փոփոխականները հաղորդող և ընդունող կողմերում պետք է փոխել սինքրոն եղանակով 00...00-ից մինչև 11...11. Տվյալների հոսքուղիների ընտրության պարզեցված ժամանակային դիագրամները ցույց են տրված նկ. 2.38-ում:

Տվյալների մշակման համակարգերում բազմաթիվ տվյալների փոխանցումը տարբեր հանգույցների միջև հաճախ կատարվող գործողություն է: Դրա համար տվյալների մուլտիպլեքսումը կարելի է իրագործել՝ օգտագործելով նկ. 2.37-ում ցույց տրված համակարգից  $m$  հատ՝ միացված զուգահեռ՝  $m$ -թիվ բառի յուրաքանչյուր բիթի համար մեկ համակարգ: Օրինակ, երկու բառաբիթ տվյալների մուլտիպլեքսման համար կարելի է օգտագործել 74157 ԻՍ-ը, որը պարունակում է քառաբիթ տվյալների “2–ից 1”

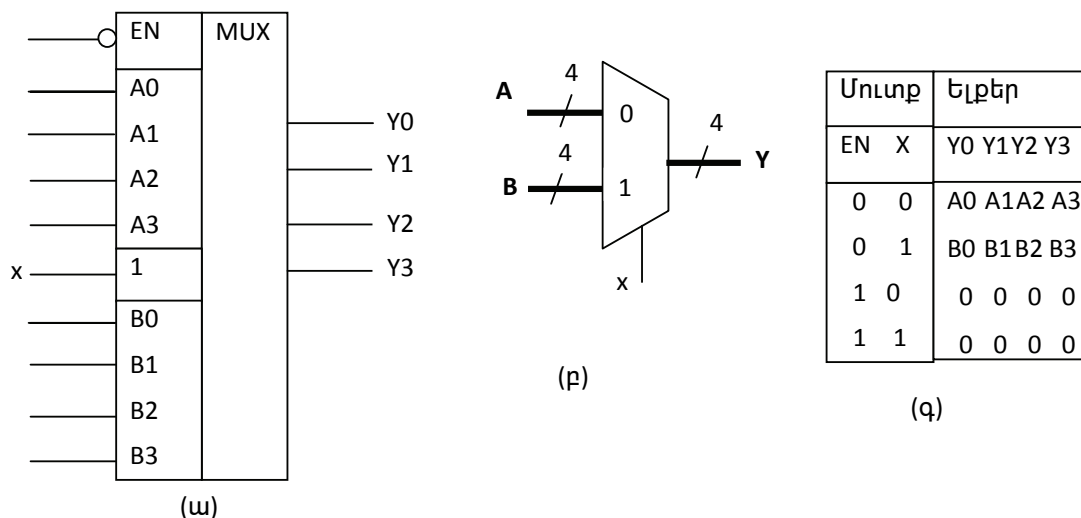
մուլտիպլեքսորներ: Այս մուլտիպլեքսորի գրաֆիկական սիմվոլները և իսկության աղ-  
յուսակը ցույց են տրված նկ. 2.39-ում: Պարզեցված գրաֆիկական սիմվոլի վրա բազմա-  
բիթ տվյալները ցույց են տրված մեկ հաստ գծով, որի վրա թեք գծով նշված է բիթերի  
քանակը:



Նկ. 2.37. Տվյալների փոխանցման բազմահոսքուղի համակարգի պարզեցված կառուցվածքը



Նկ. 2.38. Տվյալների փոխանցման բազմահոսքուղի համակարգի ժամանակային դիագրամները

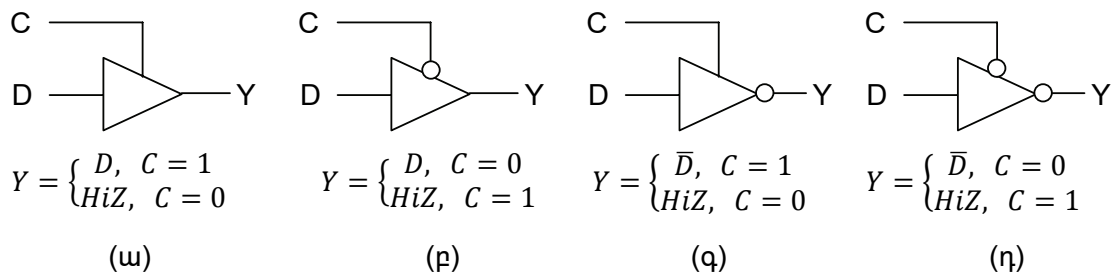


Նկ. 2.39. 74157 “2-ից 1” քառաբիթ մուլտիպլեքսոր: (ա) գրաֆիկական սիմվոլը; (բ) պարզեց-  
ված գրաֆիկական սիմվոլը; (գ) իսկության աղյուսակը

### 2.6.6. Եռավիճակ ելքով բուժեր և կոլեկտիվ օգտագործման կապի գիծ

Մի քանի մուտքային ազդանշաններից մեկի ընտրությունը և փոխանցումը ելքը՝ մուլտիպլեքսումը կարելի է իրականացնել նաև եռավիճակ ելքերով շղթաների միջոցով:

Եռավիճակ ելքերով հիմնական թվային շղթան եռավիճակ բուժերն է, որն ունի ազդանշանային մուտք և կառավարման մուտք: Բուժերի ելքը կարող գտնվել ակտիվ տրամաբանական վիճակում՝ 0 կամ 1, և բարձր դիմադրության վիճակում՝ HiZ վիճակ (ելքն անջատված է՝ խզված է շղթան դեպի VDD կամ VSS): HiZ վիճակում ելքի հանգույցի լարման մակարդակն անորոշ է: Նկ. 2.40-ում ցույց են տրված եռավիճակ բուժերի չորս տարբերակ՝ (ա) ուղիղ ելքով և ակտիվ բարձր մակարդակի կառավարումով, (բ) ուղիղ ելքով և ակտիվ ցածր մակարդակի կառավարումով, (գ) շրջված ելքով և ակտիվ բարձր մակարդակի կառավարումով, (դ) շրջված ելքով և ակտիվ ցածր մակարդակի կառավարումով:

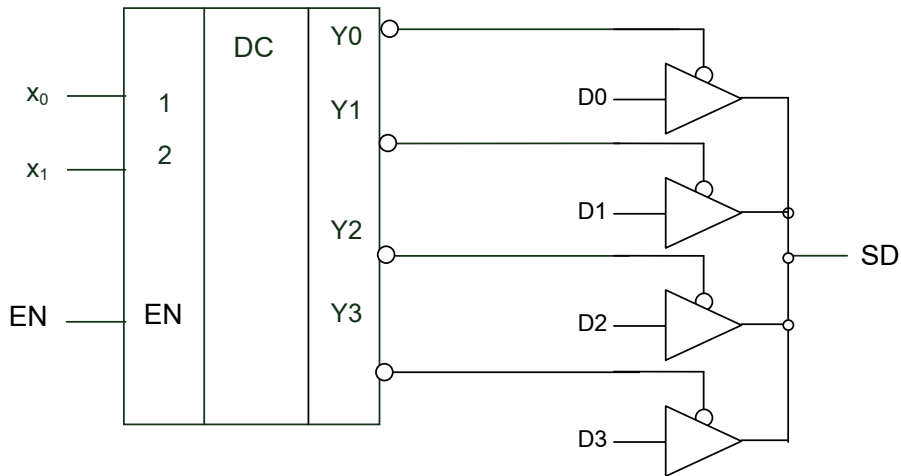


Նկ. 2.40. Եռավիճակ ելքերով բուժերներ

Եռավիճակ ելքերով սարքերը ազդանշանի մի քանի աղբյուրների թույլ են տալիս համատեղ օգտագործել ընդհանուր կապի գիծը, պայմանով, որ ժամանակի ցանկացած պահի դրանցից միայն մեկի ելքում ազդանշանը կարող է ունենալ ակտիվ տրամաբանական մակարդակ, իսկ մնացած ելքերը պետք է գտնվեն HiZ վիճակում: Նկ. 2.41-ում բերված օրինակում ցույց է տրված, թե դա ինչպես է արվում: Ընտրության երկու բիթերով, որոնք տրվում են վերծանիչի հասցեային մուտքերին, ընտրվում է ազդանշանի չորս աղբյուրներից որևէ մեկը, որի ազդանշանը փոխանցվում է կոլեկտիվ օգտագործման 1-բիթ կապի գիծ՝ SD:

Ժամանակի ցանկացած պահի վերծանիչի ելքերից միայն մեկն ունի թույլատրող 0 մակարդակ, որով հնարավորություն է տրվում եռավիճակ բուժերներից միայն մեկին մուտքային ազդանշանը փոխանցել ելք: Եթե վերծանիչի թույլատրող ազդանշանը ակտիվ չէ՝ EN=0, ապա վերծանիչի բոլոր ելքերում կունենանք տրամաբանական 1 մակարդակ, բուլոր չորս եռավիճակ բուժերների ելքերը կլինեն HiZ վիճակում, իսկ կապի գծում ազդանշանի տրամաբանական մակարդակը կլինի անորոշ:

$$SD = \begin{cases} \sum_{i=0}^3 D_i \& K_i(x_0, x_1), & EN = 1 \\ \text{HiZ}, & EN = 0 \end{cases} \quad (2.25)$$



Նկ. 2.41. Եռավիճակ ելքով չորս ազդանշանների աղբյուրներ, որոնք օգտագործում են մեկ ընդհանուր կապի գիծ

Եթե այս ֆունկցիան մուլտիպլեքսորով իրականացնելիս ելքում միշտ ունենք ազդանշանի որևէ տրամաբանական մակարդակ, ապա եռավիճակ ելքերի դեպքում ելքում կարելի ունենալ նաև պարապ վիճակ՝ մուտքային ազդանշաններից, և ոչ մեկն էլ ելք չի փոխանցվում: Երբ մուտքային ազդանշաններից ոչ մեկը չի ընտրվում, ելքում հաստատվում է բարձր դիմադրության՝ HiZ, վիճակ:

#### 2.6.7. Համակցական շղթաների նախագծում մուլտիպլեքսորների միջոցով

Մուլտիպլեքսորները լայնորեն կիրառվում են նաև համակցական սարքեր կառուցելու համար: Եթե տրամաբանական ֆունկցիան, որը պետք է իրականացվի մուլտիպլեքսորի միջոցով, տրված է ԿԴՆԶ-ով և փոփոխականների թիվը չի գերազանցում մուլտիպլեքսորի ընտրության մուտքերի թվին, ապա համապատասխան համակցական սարքավորումը իրականացվում է տրիվիալ եղանակով: Համաձայն (2.24) բանաձևի, տրված ֆունկցիան իրականացնելու համար, բավարար է ֆունկցիայի իսկության աղյուսակից որոշել այն  $m_k(x_1, x_2, \dots, x_n)$  միներմները, որոնց վրա ֆունկցիան ընդունում է 1 արժեք և մուլտիպլեքսորի համապատասխան  $I_k$  մուտքերին կիրառել ազդանշանի 1 մակարդակ, իսկ մյուս մուտքերին կիրառել ազդանշանի 0 մակարդակ:

Ցույց տանք մուլտիպլեքսորի միջոցով համակցական սարքի կառուցման եղանակը հետևյալ 3 փոփոխականի բուլյան ֆունկցիայով նկարագրվող համակցական սարքի օրինակով.

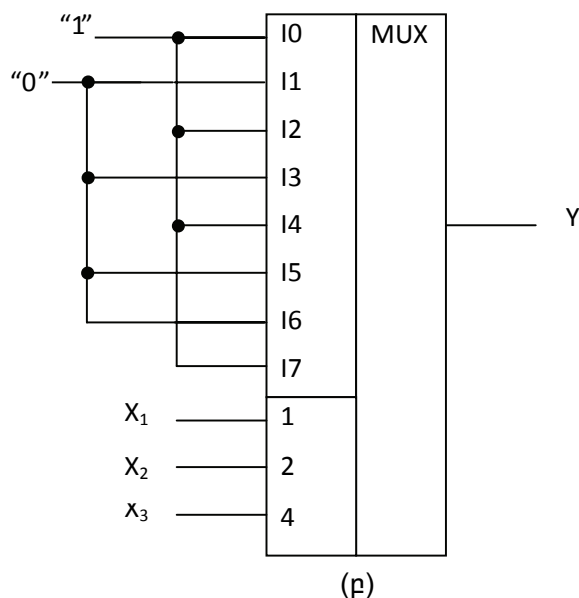
$$F(x_1, x_2, x_3) = \Sigma(0, 2, 4, 7): \quad (2.26)$$

Նկ. 2.42-ում ցույց է տրված այդ ֆունկցիայի իսկության աղյուսակը և նկարագրվող համակցական սարքի սխեման՝ կառուցված “2<sup>3</sup>–ից 1” մուլտիպլեքսորի միջոցով:



#	$x_3x_2x_1$	Y
0	0 0 0	1
1	0 0 1	0
2	0 1 0	1
3	0 1 1	0
4	1 0 0	1
5	1 0 1	0
6	1 1 0	0
7	1 1 1	1

(ա)



(բ)

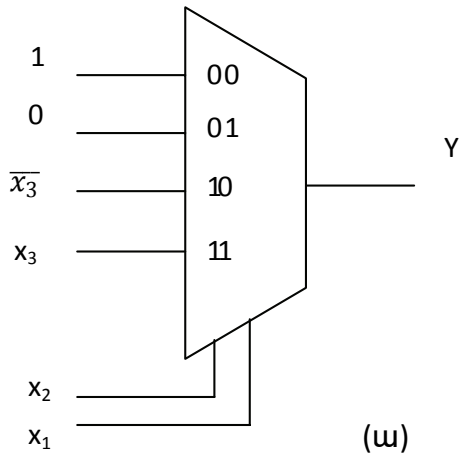
Նկ. 2.42. (2.26) ֆունկցիայով նկարագրվող համակցական սարքի իրականացումը “8-ից 1” մուլտիպլեքսորի միջոցով

Այն դեպքերում, երբ ֆունկցիան տրված է որևէ կրճատված ձևով, այն կարելի է լրացնել մինչև ԿԴՆՁ և օգտվել նկարագրված եղանակից:

Դիտարկենք մուլտիպլեքսորի վրա համակցական սարքերի սինթեզի եղանակը, երբ սինթեզվող սարքավորման մուտքերի թիվը գերազանցում է մուլտիպլեքսորի ընտրության մուտքերի թվին: Սկզբում դիտարկենք այն դեպքը, երբ մուլտիպլեքսորի մուտքերի  $n$  թիվը՝ մեկով պակաս է ֆունկցիայի փոփոխականների թվից՝  $n+1$ : Այդ  $n+1$  փոփոխականներից  $n$ -ը կիրառվում է մուլտիպլեքսորի ընտրության մուտքերին: Մնացած մեկ փոփոխականից ձևավորվում են մուլտիպլեքսորի տվյալների մուտքերի ազդանշանները: Եթե այդ փոփոխականը  $x_{n+1}$ -ն է, ապա մուլտիպլեքսորի տվյալների մուտքերին տրվող ազդանշանները կարող է լինել հետևյալներից որևէ մեկը՝  $x_{n+1}$ ,  $\overline{x_{n+1}}$ , 1, 0:

Դիտարկենք (2.26) ֆունկցիայով նկարագրվող սարքի կառուցումը “4-ից 1” մուլտիպլեքսորի միջոցով, ինչպես դա ցույց է տրված նկ. 2.43-ում:  $x_2$  և  $x_1$  փոփոխականները կիրառված են ընտրության մուտքերին, իսկ տվյալների մուտքերին տրվում են՝ 0, 1,  $x_3$  և  $\overline{x_3}$ : Երբ  $x_2x_1=00$ , մուլտիպլեքսորի ելքը՝  $Y=1$ , քանի որ  $I_0=1$ : Հետևաբար,  $m_0 = \overline{x_3} \overline{x_2} \overline{x_1}$  և  $m_4 = \overline{x_3} \overline{x_2} x_1$  միներմների համար (000 և 100 հավաքածուների համար) կունենանք  $Y=1$ , անկախ  $x_3$  արժեքից (տե՛ս նկ. 2.42ա-ի իսկության աղյուսակը և նկ. 2.43բ-ի մուլտիպլեքսորի մուտքերի աղյուսակը):  $m_1 = \overline{x_3} \overline{x_2} x_1$  և  $m_5 = x_3 \overline{x_2} x_1$  միներմների համար (001 և 101 հավաքածուների համար)  $x_2x_1=01$  և  $Y=0$ ՝ անկախ  $x_3$ -ի արժեքից: Հետևաբար,  $I_1$  մուտքին պետք է կիրառել 0: Երբ  $x_2x_1=10$ , ընտրվում է  $I_2$  մուտքը:  $x_2x_1=10$ -ին համապատասխանում են  $m_2 = \overline{x_3} x_2 \overline{x_1}$  և  $m_6 = x_3 x_2 \overline{x_1}$  միներմները (010 և 110 հավաքածուները): Իսկության աղյուսակից հետևում է, որ 010 հավաքածուի համար  $Y=1$ , իսկ 110 հավաքածուի համար  $Y=0$ : Այստեղից

կարելի է տեսնել, որ  $Y = \overline{x_3}$ : Հետևաբար  $I_2$  մուտքին պետք է կիրառել  $\overline{x_3}$ : Երբ  $x_2x_1=11$ , ընտրվում է  $I_3$  մուտքը:  $x_2x_1=11$ -ին համապատասխանում են  $m_3 = \overline{x_3}x_2x_1$  և  $m_7 = x_3x_2x_1$  միներմները (011 և 111 հավաքածուները): Իսկության աղյուսակից հետևում է, որ 011 հավաքածուի համար  $Y=0$ , իսկ 111 հավաքածուի համար  $Y=1$ : Այստեղից կարելի է տեսնել, որ  $Y=x_3$ : Հետևաբար  $I_3$  մուտքին պետք է կիրառել  $x_3$ :



	10	11	12	13
$\overline{x_3}$	<u>0</u>	1	<u>2</u>	3
$x_3$	<u>4</u>	5	6	<u>7</u>
	1	0	$\overline{x_3}$	$x_3$

(ա)

(բ)

Նկ. 2.43. (2.26) ֆունկցիայով նկարագրվող համակցական սարքի իրականացումը “4-ից 1” մուլտիպլեքսորի միջոցով և մուլտիպլեքսորի մուտքերի աղյուսակը

Մուլտիպլեքսորի միջոցով համակցական շղթայի սինթեզի հաջորդականությունը, երբ սինթեզվող սարքավորման մուտքերի թիվը գերազանցում է մուլտիպլեքսորի ընտրության մուտքերի թվին մեկով, նկարագրված է ստորև:

Նախ՝ պետք է ֆունկցիան ներկայացնել ԿՂՆՁ-ով (իսկության աղյուսակով): Կհամարենք, որ փոփոխականները տրվում են հետևյալ հաջորդականությամբ՝

$x_n x_{n-1} \dots x_2 x_1$ . Միացնել  $x_{n-1} \dots x_1$  փոփոխականները մուլտիպլեքսորի ընտրության մուտքերին՝  $x_{n-1}$  -ը միացնել ամենաբարձր կարգի մուտքին,  $x_1$  -ը՝ ամենացածր կարգի մուտքին: Մնացած  $x_n$  -ը ամենաբարձր կարգի փոփոխականն է, այն ունի 0 արժեք (ժխտված է) իսկության աղյուսակի առաջին կեսի հավաքածուների համար՝ 0-ից  $(2^n/2)-1$  կարգաթվով միներմների համար: Իսկության աղյուսակի երկրորդ կեսի հավաքածուների (միներմների) համար  $x_n$ -ն ունի 1 արժեք (ժխտված չէ):

Կառուցել մուլտիպլեքսորի մուտքերի աղյուսակը: Վերևի տողում թվարկել մուլտիպլեքսորի մուտքերը, իսկ դրա տակ՝ երկու տողով թվարկել բոլոր միներմները (հավաքածուները): Միներմների առաջին տողում գտնվում են  $x_n$  -ի ժխտված արժեքին համապատասխանող հավաքածուների համարները, իսկ երկրորդ տողում՝  $x_n$  -ի չժխտված արժեքին համապատասխանողները (ինչպես ցույց է տրված նկ. 2.43բ-ում): Ընդգծել ֆունկցիայի 1 արժեքին համապատասխանող միներմները:

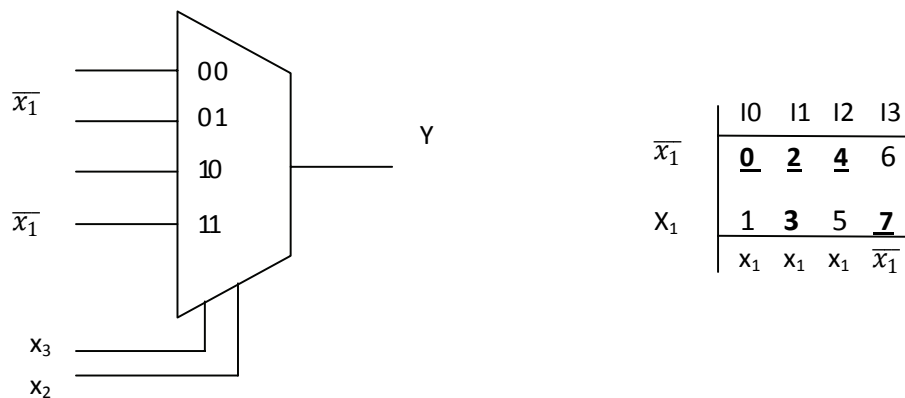
Եթե նույն սյան երկու միներմներն էլ ընդգծված չեն, մուլտիպլեքսորի համապատասխան մուտքին պետք է կիրառել 0:

Եթե նույն սյան երկու միներմներն էլ ընդգծված են, մուլտիպլեքսորի համապատասխան մուտքին պետք է կիրառել 1:

Եթե երկրորդ տողի միներմն ընդգծված է, իսկ առաջինինը՝ ոչ, մուլտիպլեքսորի համապատասխան մուտքին պետք է կիրառել  $x_n$ :

Եթե առաջին տողի միներմն ընդգծված է, իսկ երկրորդինը՝ ոչ, մուլտիպլեքսորի համապատասխան մուտքին պետք է կիրառել  $\overline{x_n}$ :

Գործնականում կարևոր չէ, որ մուլտիպլեքսորի տվյալների մուտքերի ձևավորման համար ընտրվի ամենաբարձր կարգի փոփոխականը. կարելի է ընտրել փոփոխականներից ցանկացածը: Նկ. 2.44-ում ցույց է տրված (2.26) ֆունկցիայով նկարագրվող համակցական սարքի իրականացումը “4-ից 1” մուլտիպլեքսորի միջոցով, երբ  $x_3x_2$ -ն կիրառվում մուլտիպլեքսորի ընտրության մուտքերին, իսկ տվյալների մուտքերը որոշվում են  $x_1$ -ով:



Նկ. 2.44. (2.26) ֆունկցիայով նկարագրվող համակցական սարքի այլընտրանքային իրականացումը “4-ից 1” մուլտիպլեքսորի միջոցով

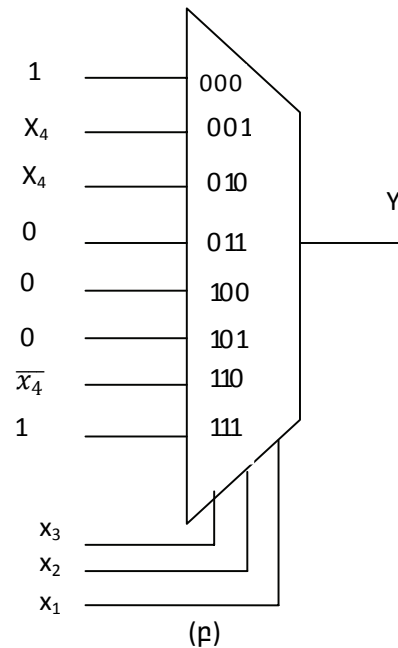
**Օրինակ 2.8.** Հետևյալ ֆունկցիայով նկարագրվող համակցական սարքը կառուցել մուլտիպլեքսորի միջոցով.

$$Y = \Sigma(0, 6, 7, 8, 9, 10, 15): \quad (2.27)$$

Սա 4 փոփոխականի ֆունկցիա է, որի իրականացման համար պետք է ունենալ 3 ընտրության մուտքերով և 8 տվյալների մուտքերով մուլտիպլեքսոր: Ընտրության մուտքերին կիրառենք  $x_3x_2x_1$  փոփոխականները: Տվյալների մուտքերը որոշվում են  $x_4$ -ով: Մուլտիպլեքսորի միջոցով համակցական սարքի իրականացումը ցույց է տրված նկ. 2.45-ում:

	I0	I1	I2	I3	I4	I5	I6	I7
$\bar{x}_4$	<u>0</u>	1	2	3	4	5	<u>6</u>	<u>7</u>
$x_4$	<u>8</u>	<u>9</u>	<u>10</u>	11	12	13	14	<u>15</u>
	1	$x_4$	$x_4$	0	0	0	$\bar{x}_4$	1

(ա)



(բ)

Նկ. 2.45. (2.27) ֆունկցիայով նկարագրվող համակցական սարքի իրականացումը “8-ից 1” մուլտիպլեքսորի միջոցով

Մուլտիպլեքսորի վրա համակցական սարքի սինթեզը կարելի է իրականացնել նաև Կառնոյի քարտերով: Թող սինթեզվող համակցական շղթան նկարագրվի նկ. 2.46ա-ում ցույց տրված քարտի միջոցով: Ֆունկցիայի նվազագույն ԴՆՁ-ն ունի հետևյալ տեսքը.

$$Y = \bar{x}_4 \bar{x}_3 \bar{x}_2 + \bar{x}_3 \bar{x}_2 x_1 + x_4 x_2 \bar{x}_1 + x_4 x_3 x_2: \quad (2.28)$$

Որպես ընտրության մուտքեր վերցնենք  $x_3 x_2 x_1$ , իսկ տվյալների մուտքերը կդիտարկվեն իբրև ֆունկցիաներ  $x_4$ -ից: Նկ. 2.46բ-ում քարտի վանդակներում նշված են մուլտիպլեքսորի տվյալների մուտքերը, որոնք համապատասխանում են  $x_3 x_2 x_1$ -ի արժեքներով: Մուլտիպլեքսորի մույն մուտքին համապատասխանող վանդակները կազմում են  $I_k = f(x_4)$  ֆունկցիայի Կառնոյի քարտը՝ որոշված միայն  $x_4$ -ից: Այդ քարտերից (նկ. 2.46բ) կարելի է որոշել՝  $I_0 = \bar{x}_4$ ,  $I_1 = 1$ ,  $I_2 = x_4$ ,  $I_3 = 0$ ,  $I_4 = 0$ ,  $I_5 = 0$ ,  $I_6 = x_4$ ,  $I_7 = x_4$ : Նկ. 2.47ա-ում ցույց է տրված սինթեզված սարքավորման սխեման իրականացված 74151 (K555KП7) մուլտիպլեքսորի վրա:

Y	$x_2x_1$			
	00	01	11	10
$x_4x_3$				
00	1	1		
01				
11			1	1
10		1		1

(ա)

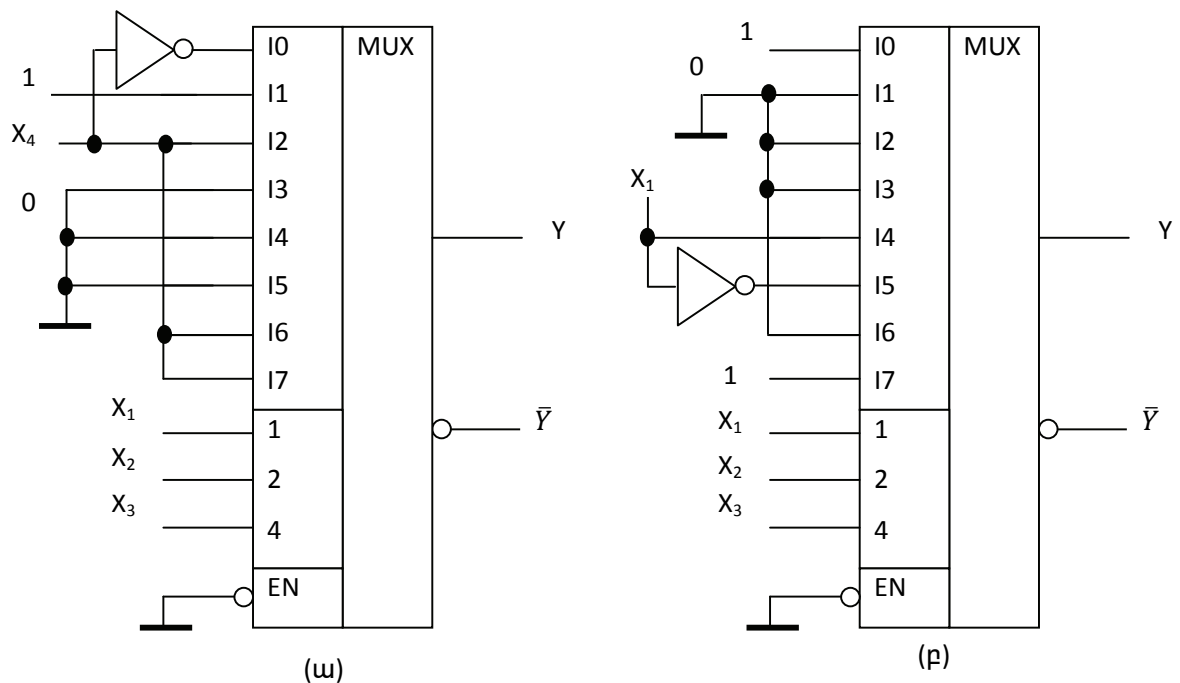
Y	$x_2x_1$			
	00	01	11	10
$x_4x_3$				
00	I0	I1	I3	I2
01	I4	I5	I7	I6
11	I4	I5	I7	I6
10	I0	I1	I3	I2

(բ)

Y	$x_2x_1$			
	00	01	11	10
$x_4x_3$				
00	I0	I0	I1	I1
01	I2	I2	I3	I3
11	I6	I6	I7	I7
10	I4	I4	I5	I5

(գ)

Նկ. 2.46. (2.28) ֆունկցիայի կառուցման քարտեզը ու մուլտիպլեքսորի մուտքերի աղյուսակները



Նկ. 2.47. Սինթեզված սարքավորման սխեման՝ իրականացված 74151 (K555KП7) մուլտիպլեքսորի վրա

Մուլտիպլեքսորի միջոցով համակցական շղթաների իրականացման բարդությունը կախված է ընտրության մուտքերին տրվող փոփոխականների ընտրությունից: Այդ փոփոխականները պետք է ընտրել այնպես, որ մուլտիպլեքսորի նվազագույն թվով մուտքեր որոշված լինեն  $x_4$ -ով կամ  $\bar{x}_4$ -ով: Այդ դեպքում մուլտիպլեքսորի մուտքերը ավելի քիչ են բեռնավորում  $x_4$  ազդանշանի աղբյուրը, և կպահանջվեն այդ աղբյուրից դեպի մուլտիպլեքսորի մուտքեր քիչ թվով ծրման գծեր: Այս սկզբունքի իրականացման համար պետք է որպես ընտրության մուտքեր ընտրել այն փոփոխականները, որոնք տանում են մեծ տրամաբանական բեռնվածք՝ որոնք մասնակցում են ֆունկցիայի նվազագույն ԴՆԶ-ում ավելի շատ անգամ: Որպես օրինակ նորից դիտարկենք (2.28)

բանաձևը: Այդ բանաձևում ամենաքիչ անգամ մասնակցում է  $x_1$  փոփոխականը, հետևաբար որպես ընտրության մուտքեր կընտրվեն  $x_4, x_3, x_2$  փոփոխականները: Նկ. 2.46գ-ում քարտի վաղակները նշված են մուլտիպլեքսորի մուտքերը, որոնք համապատասխանում են  $x_4x_3x_2$ -ի արժեքներին: Ըստ այդ նշումների նկ. 2.46ա-ի քարտը կտրոհվի երկվանդակ 8 քարտերի, որոնցից կստացվի՝  $I_0=1, I_1=0, I_2=0, I_3=0, I_4=x_1, I_5=\bar{x}_1, I_6=0, I_7=1$ : Համապատասխան սխեման ցույց է տրված նկ. 2.47բ-ում: Այստեղ ութ մուտքերից վեցը հավասար են 0 կամ 1 հաստատունների, իսկ նախորդում դեպքում՝ միայն չորսը:

Ընդհանուր դեպքում, երբ անհրաժեշտ է ու ընտրության մուտքերով մուլտիպլեքսորի վրա կառուցել  $m$  մուտքերով համակցական շղթա՝  $m > n$ , անհրաժեշտ է մուլտիպլեքսորի տվյալների մուտքերը դիտարկել որպես տրամաբանական ֆունկցիաներ մնացած  $(m-n)$  փոփոխականներից, որոնք կարող են իրականացվել տրամաբանական տարրերի միջոցով: Որպես օրինակ դիտարկենք նկ. 2.48ա-ի քարտով նկարագրվող համակցական շղթայի իրականացումը “4-ից 1” մուլտիպլեքսորի միջոցով: Որպես ընտրության մուտքեր վերցնենք  $x_3x_2$ -ը, տվյալների մուտքերը կլինեն ֆունկցիա  $x_4$ -ից և  $x_1$ -ից: Նկ. 2.48բ-ում ցույց է տրված քարտի նշումը ըստ մուլտիպլեքսորի տվյալների  $I_0, \dots, I_3$  մուտքերի: Նկ. 2.48ա-ում կրկնված է ֆունկցիայի Կառնոյի քարտը նկ. 2.46ա-ից, հարմարության համար:

Y

x<sub>2</sub>x<sub>1</sub>

00

01

11

10

x<sub>4</sub>x<sub>3</sub>

00

01

11

10

1	1		
		1	1
	1		1

(ա)

Y

x<sub>2</sub>x<sub>1</sub>

00

01

11

10

x<sub>4</sub>x<sub>3</sub>

00

01

11

10

I0	I0	I1	I1
I2	I2	I3	I3
I2	I2	I3	I3
I0	I0	I1	I1

(բ)

I<sub>0</sub>

x<sub>1</sub>

0

1

x<sub>4</sub>

0

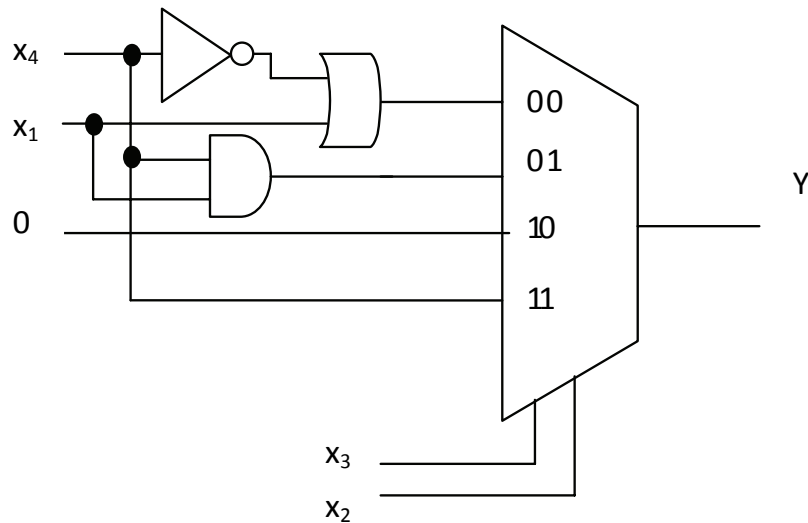
1

1	1
	1
	1

(գ)

Նկ. 2.35. (2.28) ֆունկցիայի Կառնոյի քարտն ու “4-ից 1” մուլտիպլեքսորի մուտքերի աղյուսակները

$I_0$  ֆունկցիայի քառավանդակ Կառնոյի քարտը նկ. 2.48ա-ի քարտում համապատասխանում է նկ. 2.48բ-ում  $I_0$ -ով նշված վանդակներին: Նկ. 2.48գ-ում ցույց է տրված  $I_0$ -ի և  $I_1$ -ի քարտերը: Այդ քարտերից կստացվեն  $I_0$  և  $I_1$  նվազարկված ֆունկցիաները՝  $I_0 = x_1 + \bar{x}_4$ ;  $I_1 = x_1 \& x_4$ : Համանման եղանակով կարելի է ստանալ  $I_2$  և  $I_3$  նվազարկված ֆունկցիաների բանաձևերը՝  $I_2=0, I_3=x_4$ : Նկ. 2.49-ում ցույց է տրված սինթեզված շղթայի սխեման իրականացված “4-ից 1” մուլտիպլեքսորի միջոցով:



Նկ. 2.49. (2.28) ֆունկցիայով նկարագրվող համակցական սարքի սխեման՝ իրականացված “4-ից 1” մուլտիպլեքսորի միջոցով

Նկարագրված եղանակով կարելի է կառուցել սինթեզվող սարքի սխեման ցանկացած ( $m > n$ )-ի դեպքում: Սակայն  $m - n > 2$  դեպքում մուլտիպլեքսորի տվյալների մուտքերի ֆունկցիաները բարդ տեսք կունենան, և մուլտիպլեքսորի կիրառությունն այլևս արդյունավետ չի լինի: Նման իրավիճակներում, հնարավոր է, որ ավելի արդյունավետ լինի մուլտիպլեքսորի սխեմայի ընդարձակումը ըստ մուտքերի:

Այժմ համեմատենք համակցական սարքավորումների վերծանիչով և մուլտիպլեքսորով իրականացման եղանակները: Վերծանիչի դեպքում պահանջվում է մեկական ԿԱՍ տարր յուրաքանչյուր ֆունկցիայի համար, բայց միայն մեկ վերծանիչ է պետք բոլոր մինթերմների ձևավորման համար: Մուլտիպլեքսորի տարբերակում օգտագործվում է ավելի փոքր թվով տրամաբանական տարրեր, բայց պահանջվում է առանձին մուլտիպլեքսոր յուրաքանչյուր ֆունկցիայի համար: Այսպիսով, մուլտիպլեքսորի կիրառությունն արդյունավետ է փոքր թվով ֆունկցիաների դեպքում, իսկ վերծանիչի կիրառություն՝ մեծ թվով ֆունկցիաների դեպքում: Նկատենք, որ մուլտիպլեքսորներն ու վերծանիչներն լայնորեն օգտագործվում են կիրառությունում ծրագրավորվող փականների զանգվածի ԻՍ-երում (ԿԾՓՁ, Field Programmable Gate Array, FPGA):

Այնուամենայնիվ, պետք է հիշել, որ վերծանիչների հիմնական ֆունկցիան երկուական տվյալների վերծանումն է, իսկ մուլտիպլեքսորներինը՝ բազմաթիվ աղբյուրներից միակ նպատակակետը տվյալների փոխանցման ուղի ստեղծելը:

## 2.7. Համակցական սարքավորումների կառուցումը ծրագրավորվող ինտեգրալ սխեմաներով

### 2.7.1. Ընդհանուր տեղեկություններ ծրագրավորվող մեծ ինտեգրալ սխեմաների վերաբերյալ

Ծրագրավորվող մեծ ինտեգրալ սխեմաների կիրառումը հնարավորություն է տալիս կրճատել օգտագործվող միկրոսխեմաների քանակը և նախագծման ժամկետները: Համակցական սարքավորումների կառուցման համար մեծ կիրառություն ունեն հաստատուն հիշասարքերը՝ ՀՀՍ (կամ այլ կերպ միայն ընթերցվող հիշասարքերը ՄԸՀ, Read Only Memory, ROM) և ծրագրավորվող տրամաբանական մատրիցները (ԾՏՍ, Programmable Logic Array, PLA):

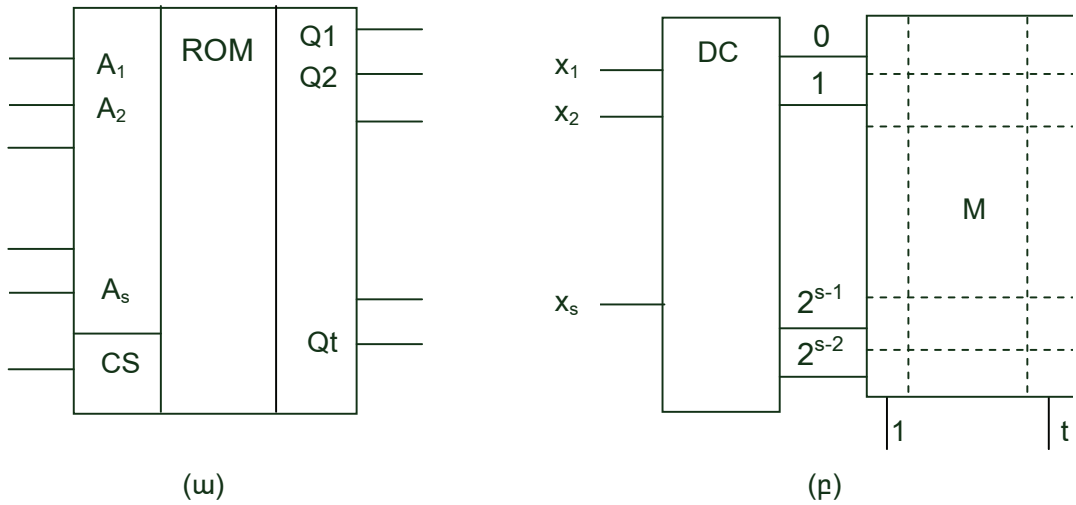
ՀՀՍ-ի պայմանական գրաֆիկական նշանը ցույց է տրված նկ. 2.50ա-ում, որտեղ  $A_1, \dots, A_s$ -ն հասցեային մուտքերն են,  $Q_1, \dots, Q_r$ -ն ելքերը: Սովորաբար ՀՀՍ-ն ունի նաև ծրագրավորման մուտքեր, որոնք նկ. 2.50ա-ում ցույց չեն տրված: Երբ ԻՍ-ի ընտրության CS մուտքին տրվում է ազդանշանի ակտիվ մակարդակ, միկրոսխեման կատարում է իր վրա դրված ֆունկցիան, երբ CS-ին տրվում է ազդանշանի պասիվ մակարդակ, միկրոսխեմայի ելքերը գտնվում են բարձր դիմադրության վիճակում: ՀՀՍ-ի կառուցվածքը ցույց է տրված նկ. 2.50բ-ում, այն բաղկացած է  $s \times 2^s$  լրիվ վերձանիչից և  $2^s \times t$  բիթ ունակությամբ հիշողության մատրիցից: Մուտքային հասցեային փոփոխականների յուրաքանչյուր համակացություն ընտրում է վերձանիչի մեկ ելք, որի միջոցով M մատրիցից կարդացվում է համապատասխան բառն իր t կարգերով: Սովորաբար ՀՀՍ-ն ներկայացվում է երկու պարամետրով՝ (s, t): Ծրագրավորման, այսինքն անհրաժեշտ ինֆորմացիայի գրանցման, եղանակից կախված ՀՀՍ-երը կարող են լինել՝ (ա) դիմակային ծրագրավորումով՝ ծրագրավորվում են արտադրության պրոցեսում, և այնտեղ գրված ինֆորմացիան այլևս չափագործման ժամանակ հնարավոր չէ փոխել, (բ) մեկ անգամ ծրագրավորվող՝ ծրագրավորվում է օգտագործողի կողմից, ծրագրավորված ԻՍ-ում այլևս փոփոխություններ կատարել հնարավոր չէ, (գ) օգտագործողի կողմից բազմաթիվ անգամ վերածրագրավորվող-ջնջվող:

ԾՏՍ-ն բաղկացած է երկու մատրիցներից (նկ. 2.51ա)՝ M1-ն իրականացնում է q կոնյունկցիաներ s փոփոխականներից, իսկ M2-ը՝ t դիզյունկցիաներ q կոնյունկցիաներից: Փաստորեն ԾՏՍ-ի կառուցվածքը համապատասխանում է ԴԵԶ-ով՝ կոնյունկցիաների դիզյունկցիայով տրված տրամաբանական ֆունկցիաների իրականացմանը: ԾՏՍ-ն ներկայացվում է s մուտքերի թվով, t ելքերի թվով և q կոնյունկցիաների գծերի թվով: (s, t, q) ԾՏՍ-ի պայմանական գրաֆիկական նշանը ցույց է տրված նկ. 2.38բ-ում: ԾՏՍ-ի ինֆորմացիոն ունակությունը գնահատվում է հետևյալ կերպ՝

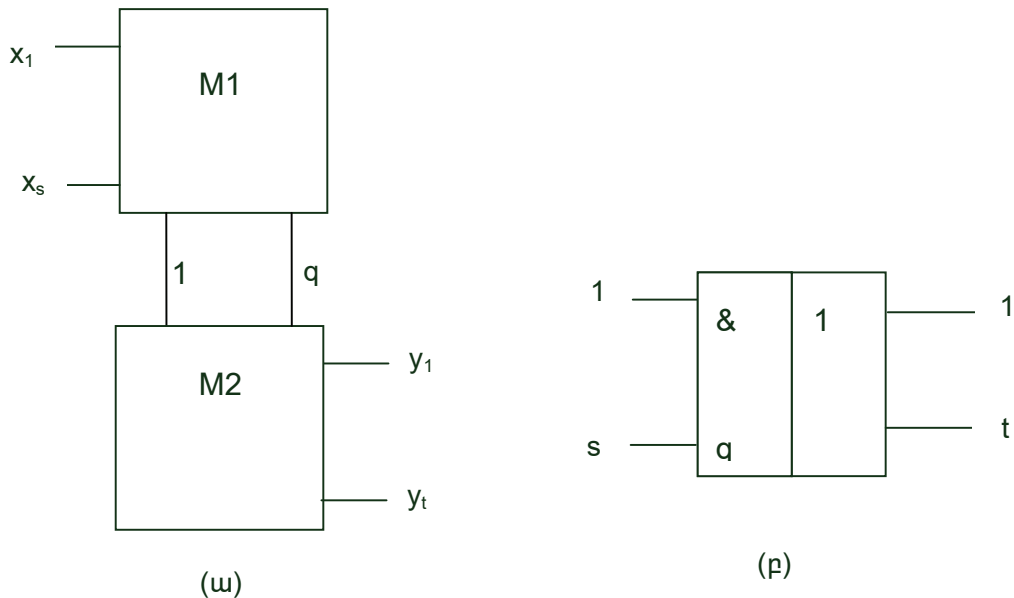
$$C(M) = C(M_1) + C(M_2) = 2s \cdot q + q \cdot t, \text{ բիթ:} \quad (2.29)$$

s գործակիցը հաշվի է առնում այն հանգամանքը, որ s մուտքերի դեպքում մատրիցն ունի  $2s$  հորիզոնական գծեր՝  $x_1, \overline{x_1}, \dots, x_s, \overline{x_s}$ : Գոյություն ունեն ԾՏՍ-երի մի շարք տարատեսակներ, որոնք տարբերվում են կառուցվածքով և տրամաբանական հնարավորություններով:





Նկ. 2.50. ԴՀՄ-ի գրաֆիկական նշանակումը և կառուցվածքը



Նկ. 2.51. ԾՏՄ-ի կառուցվածքը և գրաֆիկական նշանակումը

Ծրագրավորվող տրամաբանական ԻՍ-երը, բացի համակցական հնարավորություններից՝ կոնյունկցիաների և դիզյունկցիաների մատրիցներից, կարող են պարունակել նաև հիշողության ռեգիստրներ՝ թվային ավտոմատներ կառուցելու համար: Այդպիսի ԻՍ-երը կոչվում են ծրագրավորվող տրամաբանական սարքեր՝ ԾՏՄ (Programmable Logic Device, PLD): Էլ ավելի մեծ հնարավորություններ է ընձեռում կիրառությունում ծրագրավորվող փականների զանգվածը (ԿԾՓԶ): ԿԾՓԶ-ն առանձին կիրառվող, ծրագրավորվող տրամաբանական ԻՍ է, որը թույլ է տալիս արագ իրականացնել բարդ թվային ամբողջական համակարգեր:

### 2.7.2. Համակցական սարքավորումների սինթեզ ծրագրավորվող ԻՍ-ով

Ենթադրենք սինթեզվող տրամաբանական սարքավորումը նկարագրվում է  $n$  փոփոխականի  $m$  տրամաբանական ֆունկցիաներով: Եթե  $n < s$ ,  $m < t$ , ապա համակցական սարքավորումը կարող է տրիվիալ եղանակով իրականացվել ( $s$ ,  $t$ ) ՌՅՍ-ով: Որպես օրինակ դիտարկենք աղյուսակ 2.10-ով նկարագրվող համակցական սարքավորման սինթեզը (4,4) ՌՅՍ-ով: Սինթեզի արդյունքը ներկայացվում է ԻՍ-ի ծրագրավորման աղյուսակով, որտեղ նշվում է ՌՅՍ-ի յուրաքանչյուր հասցեի բառի պարունակությունը: ՌՅՍ-ի հասցեները համապատասխանում են իսկության աղյուսակի հավաքածուներին, իսկ հիշողության համապատասխան հասցեների բջիջներում գրվում են ֆունկցիաների արժեքները: Փաստորեն ՌՅՍ-ի ծրագրավորման աղյուսակը համընկնում է իրականացվող տրամաբանական ֆունկցիաների համակարգի իսկության աղյուսակի հետ: Տրամաբանական ֆունկցիաների այսպիսի իրականացումը կոչվում է աղյուսակային (Look up Table, LUT):

Աղյուսակ 2.10

#	$x_4x_3x_2x_1$	$y_4y_3y_2y_1$	#	$x_4x_3x_2x_1$	$y_4y_3y_2y_1$
0	0000	0000	8	1000	0111
1	0001	0011	9	1001	0110
2	0010	0000	10	1010	0100
3	0011	0011	11	1011	0000
4	0100	1100	12	1100	0001
5	0101	1111	13	1101	0000
6	0110	1100	14	1110	0001
7	0111	1111	15	1111	1000

Աղյուսակ 2.11

Հասցե	Բառ	Հասցե	Բառ
0	0000	8	0111
1	0011	9	0110
2	0000	10	0100
3	0011	11	0000
4	1100	12	0001
5	1111	13	0000
6	1100	14	0001
7	1111	15	1000

ՌՅՍ-ով համակցական սարքն իրականացվում է ըստ այն նկարագրող իսկության աղյուսակի՝ առանց որևէ ձևափոխության կամ պարզեցուման: Եթե սինթեզվող սարքավորումը նկարագրող ֆունկցիաները տրված են որևէ կրճատված ԴՆԶ-ով, ապա ՌՅՍ-ով իրականացման համար անհրաժեշտ է տրված ԴՆԶ-ն ընդարձակել մինչև ԿԴՆԶ: Այս առումով ՌՅՍ-ով համակցական սարքավորումների իրականացումը պարունակում է որոշակի ավելցուկություն: Որոշ դեպքերում, օգտվելով իսկության աղյուսակի որոշակի հատկություններից, համակցական սարքը կարելի է կառուցել ավելի փոքր ծավալի ՌՅՍ-ով, քան պահանջվում է իսկության աղյուսակից:

**Օրինակ 2.9.** ՀՀՍ-ի օգնությամբ կառուցել համակցական սարք, որի ելքերում ստացվում է մուտքային եռաբիթ երկուական թվի քառակուսին:

Սինթեզվող սարքավորման իսկության աղյուսակը ցույց է տրված աղյուսակ 2.12.-ում: Ըստ այս աղյուսակի ՀՀՍ-ն պետք է ունենա երեք հասցեային մուտքեր, որոնց կիրառվում են մուտքային  $a$  թվի բիթերը, և վեց ելքեր, որոնցում ստացվում են ելքային  $b$  թվի բիթերը: Իսկության աղյուսակից երևում է, որ  $b_0$ -ն միշտ կրկնում է  $a_0$ -ի արժեքը, իսկ  $b_1$ -ը միշտ 0 է: Հետևաբար կարիք չկա ՀՀՍ-ով գեներացնել  $b_1$  և  $b_0$ : ՀՀՍ-ն կարող է ունենալ միայն 4 ելքեր: Սինթեզվող սարքավորումն իրականացնող (3,4) ՀՀՍ-ի ծրագրավորման աղյուսակը ցույց է տրված աղյուսակ 2.13-ում, իսկ սարքի սխեման՝ նկ. 2.52-ում: ՀՀՍ-ն ունի  $8 \times 4 = 32$  բիթ ծավալ:

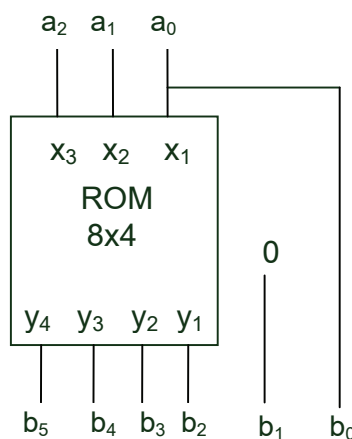
ԾՏՄ-ներով համակցական սարքավորումների իրականացման համար անհրաժեշտ է, որ  $L \leq s$ ,  $N \leq t$  և հավասարումների տրված համակարգում բոլոր իրարից տարբեր կոնյունկցիաների թիվը՝  $B$ -ն մեծ չլինի ԾՏՄ-ի կոնյունկցիաների գծերի թվից՝  $B \leq q$ : Վերջին պայմանից հետևում է, որ ԾՏՄ-ի վրա սինթեզի համար անհրաժեշտ է ֆունկցիաները ներկայացնել նվազագույն թվով կոնյունկցիաներ պարունակող ԴՆԶ-երով, այսինքն՝ կարճագույն ԴՆԶ-երով: Դիտարկենք աղյուսակ 2.10-ով տրված համակցական սարքավորման սինթեզը ԾՏՄ-ի վրա: Աղյուսակի ֆունկցիաների նվազարկումից կստացվի՝

Աղյուսակ 2.12

$a_2 a_1 a_0$	$b_5 b_4 b_3 b_2 b_1 b_0$	Տասական
000	0 0 0 0 0 0	0
001	0 0 0 0 0 1	1
010	0 0 0 1 0 0	4
011	0 0 1 0 0 1	9
100	0 1 0 0 0 0	16
101	0 1 1 0 0 1	25
110	1 0 0 1 0 0	36
111	1 1 0 0 0 1	49

Աղյուսակ 2.13

Մուտքեր	Ելքեր
$x_3 x_2 x_1$	$y_4 y_3 y_2 y_1$
$a_2 a_1 a_0$	$b_5 b_4 b_3 b_2$
000	0000
001	0000
010	0001
011	0010
100	0100
101	0110
110	1001
111	1100



Նկ. 2.52. Եռաբիթ թվերի քառակուսի բարձրացնող սարքը՝ ՀՀՍ-ով իրականացմամբ

$$\begin{aligned}
y_1 &= \bar{x}_4 x_1 + x_4 \bar{x}_1, \\
y_2 &= \bar{x}_4 x_1 + x_4 \bar{x}_3 \bar{x}_2, \\
y_3 &= \bar{x}_4 x_3 + x_4 \bar{x}_3 \bar{x}_2 + x_4 \bar{x}_3 x_1, \\
y_4 &= \bar{x}_4 x_3 + x_3 x_2 x_1.
\end{aligned}
\tag{2.30}$$

Այս բանաձևերում առկա են վեց իրարից տարբեր կոնյունկցիաներ, հետևաբար սինթեզվող սարքավորումը կարող է իրականացվել (4, 4, 6) պարամետրերով ԾՏՄ-ով: ԾՏՄ-ն ծրագրավորվում է ըստ աղյուսակ 2.14-ի, որտեղ յուրաքանչյուր տող համապատասխանում է մեկ կոնյունկցիայի: Եթե տվյալ կոնյունկցիայում  $x_i$  փոփոխականը մասնակցում է ուղիղ ձևով, նրա դիրքում գրվում է 1, եթե  $x_i$ -ն մասնակցում է բացասված ձևով, նրա դիրքում գրվում է 0, իսկ եթե  $x_i$  փոփոխականը չի մասնակցում տվյալ կոնյունկցիայում, նրա դիրքում գծիկ է դրվում: Եթե տվյալ կոնյունկցիան մասնակցում է  $y_k$  ֆունկցիայում,  $y_k$  սյունակի համապատասխան տողում գրվում է 1, եթե ոչ՝ գծիկ է դրվում:

Աղյուսակ 2.14

Տող	Մուտքեր	Ելքեր
	$x_1 x_2 x_3 x_4$	$y_1 y_2 y_3 y_4$
1	1 - - 0	1 1 - -
2	0 - - 1	1 - - -
3	- 0 0 1	- 1 1 -
4	- - 1 0	- - 1 1
5	0 - 0 1	- - 1 -
6	1 1 1 -	- - - 1

Շատ ԿԾՓՁ-ներում համակցական շղթաներն իրականացվում են տրամաբանական աղյուսակների (LUT) միջոցով: Այդ աղյուսակները կարող են իրականացվել ՀՀ-Ս-ի կամ մուլտիպլեքսորի կառուցվածքով: Յուրաքանչյուր ծրագրավորվող տրամաբանական բլոկ պարունակում է որոշակի թվով տրամաբանական աղյուսակ: Տրամաբանական աղյուսակների մուտքերի թիվը սահմանափակ է: Տրամաբանական աղյուսակների համակցման միջոցով կարելի է իրականացնել ավելի մեծ թվով մուտքերից կախված ֆունկցիաներ:

Ենթադրենք տրված ԿԾՓՁ-ի տրամաբանական բլոկներում տրամաբանական աղյուսակների մուտքերի թիվը 3 է (3-LUT): Այսպիսի աղյուսակով կարելի է իրականացնել 3 մուտքերով ցանկացած համակցական շղթա: Ավելի մեծ թվով մուտքերով շղթաների իրականացման համար կարելի է օգտվել Շենոնի վերլուծության բանաձևից: Դիտարկենք չորս փոփոխականի հետևյալ ֆունկցիան

$$F = \bar{x}_2 x_3 + \bar{x}_1 x_2 \bar{x}_3 + x_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 \bar{x}_4. \tag{2.31}$$

Վերլուծենք այս ֆունկցիան ըստ  $x_1$ -ի՝

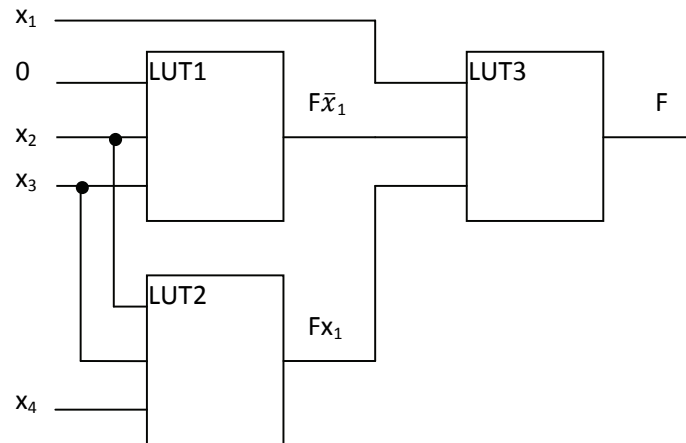
$$\begin{aligned}
F &= \bar{x}_1 F_{\bar{x}_1} + x_1 F_{x_1} = \bar{x}_1 (\bar{x}_2 x_3 + x_2 \bar{x}_3 + x_2 \bar{x}_3 x_4) + x_1 (\bar{x}_2 x_3 + x_2 \bar{x}_3 x_4 + \bar{x}_2 \bar{x}_4) = \\
&= \bar{x}_1 (\bar{x}_2 x_3 + x_2 \bar{x}_3) + x_1 (\bar{x}_2 x_3 + x_2 \bar{x}_3 x_4 + \bar{x}_2 \bar{x}_4):
\end{aligned}
\tag{2.32}$$

Այս արտահայտությունը կարելի է իրականացնել երեք հատ 3-LUT աղյուսակներից բաղկացած շղթայով, ինչպես ցույց է տրված նկ. 2.53-ում:

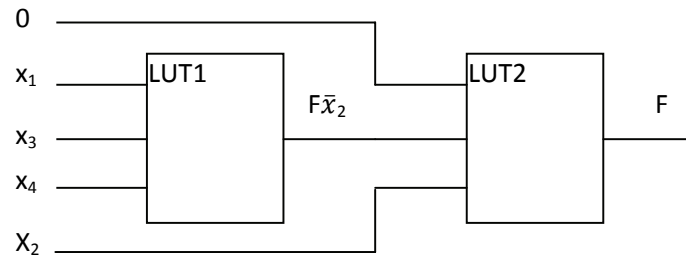
Ըստ  $x_2$ -ի վերլուծությունից կստանանք

$$F = \bar{x}_2 F_{\bar{x}_2} + x_2 F_{x_2} = \bar{x}_2(x_3 + x_1 \bar{x}_4) + x_2(\bar{x}_1 \bar{x}_3 + \bar{x}_3 x_4): \quad (2.33)$$

Կարելի է նկատել, որ  $\bar{F}_{\bar{x}_2} = F_{x_2}$ : Այս հանգամանքն օգտագործելով՝ կարելի է իրագործել ֆունկցիան միայն երկու 3-LUT աղյուսակներով, ինչպես ցույց է տրված նկ. 2.54-ում:



Նկ. 2.53. (2.31) բանաձևով նկարագրվող ֆունկցիայի իրականացումը ըստ (2.32) վերլուծության



Նկ. 2.54. (2.31) բանաձևով նկարագրվող ֆունկցիայի իրականացումը ըստ (2.33) վերլուծության

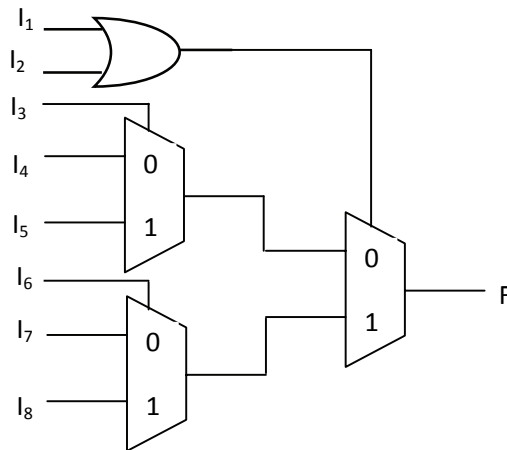
Աղյուսակ 2.15

LUT1		LUT2		LUT3	
$0x_2x_3$	$F_{\bar{x}_1}$	$x_2x_3x_4$	$F_{x_1}$	$x_1 F_{\bar{x}_1} F_{x_1}$	$F$
000	0	000	1	000	0
001	1	001	0	001	0
010	1	010	1	010	1
011	0	011	1	011	1
000	0	100	0	100	0
001	1	101	1	101	1
010	1	110	0	110	0
011	0	111	0	111	1

Աղյուսակ 2.16

LUT1		LUT2	
$x_1x_3x_4$	$F\bar{x}_2$	$0 F\bar{x}_2x_2$	$F$
000	0	000	0
001	0	001	1
010	1	010	1
011	1	011	0
100	1	000	0
101	0	001	1
110	1	010	1
111	1	011	0

**Օրինակ 2.10.** Actel ընկերության Act1 ԿԾՓԶ-ի տրամաբանական բլոկը հիմնված է մուլտիպլեքսորների վրա և ունի նկ. 2.55-ում ցույց տրված տեսքը:



Նկ. 2.55. Act1 ԿԾՓԶ-ի տրամաբանական բլոկը

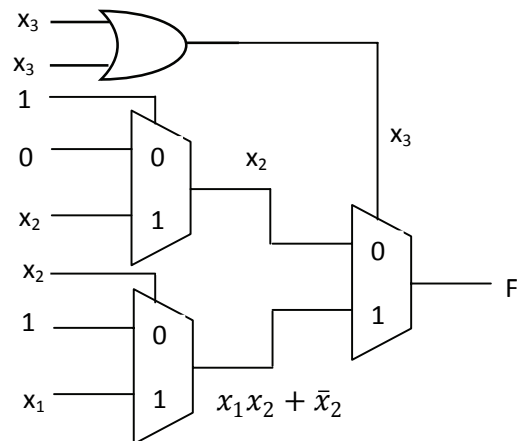
Ցույց տալ, որ

$$F = x_2\bar{x}_3 + x_1x_3 + \bar{x}_2x_3 \quad (2.34)$$

Ֆունկցիան կարող է իրագործվել մեկ այդպիսի տրամաբանական բլոկով: Նկատի ունեցեք, որ տրամաբանական բլոկը ՈՉ տարրեր չունի, հետևաբար փոփոխականների ժխտված արժեքները պետք է ստանալ մուլտիպլեքսորներով:

$$F = x_2\bar{x}_3 + x_1x_3 + \bar{x}_2x_3 = \bar{x}_3x_2 + x_3(x_1 + \bar{x}_2) = \bar{x}_3x_2 + x_3(x_1x_2 + \bar{x}_2): \quad (2.35)$$

Նկ. 2.56-ում ցույց է տրված ծրագրավորված տրամաբանական բլոկը:



Նկ. 2.56. (2.35) ֆունկցիայի իրականացումը Act1 ԿԾՓԶ-ի տրամաբանական բլոկով

## 2.8. Խնդիրներ

**Խնդիր 2.1.** Հետևյալ ֆունկցիաների բանաձևերը նվազարկել Կառնոյի քարտերով.

ա)  $F(x, y, z) = \bar{x}\bar{y}z + \bar{x}yz + \bar{x}y\bar{z},$

բ)  $F(x, y, z) = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}z + xy\bar{z},$

գ)  $F(x, y, z) = \bar{y}z + \bar{y}z + xy\bar{z}:$

**Խնդիր 2.2.** Կառուցել Խնդիր 2.1-ի նվազարկված բանաձևերին համապատասխանող շղթաները՝ օգտագործելով միայն

ա) 2ԵՎ-ՈՉ տարրեր,

բ) 2ԿԱՄ-ՈՉ տարրեր:

**Խնդիր 2.3.** Հետևյալ ֆունկցիաների բանաձևերը նվազարկել Կառնոյի քարտերով.

ա)  $F(w, x, y, z) = \bar{w}\bar{x}\bar{y}z + \bar{w}\bar{x}y\bar{z} + \bar{w}x\bar{y}z + \bar{w}xyz + \bar{w}xy\bar{z} + w\bar{x}\bar{y}z + w\bar{x}y\bar{z},$

բ)  $F(w, x, y, z) = \bar{w}\bar{x}\bar{y}z + \bar{w}\bar{x}y\bar{z} + w\bar{x}\bar{y}z + w\bar{x}y\bar{z} + w\bar{x}\bar{y}z,$

գ)  $F(w, x, y, z) = \bar{y}z + w\bar{y} + \bar{w}xy + \bar{w}\bar{x}y\bar{z} + w\bar{x}y\bar{z}:$

**Խնդիր 2.4.** Կառուցել Խնդիր 2.3-ի նվազարկված բանաձևերին համապատասխանող շղթաները՝ օգտագործելով միայն

ա) ԵՎ-ՈՉ տարրեր,

բ) ԿԱՄ-ՈՉ տարրեր:

**Խնդիր 2.5.** Տրված  $F(x, y, z) = x\bar{y}z + \bar{x}\bar{y}z + xyz$  ֆունկցիայի համար՝

ա) Կազմել իսկության աղյուսակը,

բ) Կառուցել թվային շղթան՝ օգտագործելով տիպային տրամաբանական տարրեր,

գ) Նվազարկել բանաձևը՝ օգտվելով բուլյան ֆունկցիաների հատկություններից,

դ) Համեմատել նվազարկված ֆունկցիայի իսկության աղյուսակը սկզբնականի հետ,

ե) Կառուցել թվային շղթան՝ ըստ նվազարկված բանաձևի:

**Խնդիր 2.6.** Կառուցել Բացառող-ԿԱՄ շղթա՝ օգտագործելով միայն 2ԿԱՄ-ՈՉ տարրեր:

**Խնդիր 2.7.** Կառուցել Բացառող-ԿԱՄ շղթա՝ օգտագործելով միայն 2ԵՎ-ՈՉ տարրեր:

**Խնդիր 2.8.** Կառուցել թվային շղթա՝ որն ունի երեք մուտք ( $x$ ,  $y$  և  $z$ ): Շղթայի ելքում ստացվում է 1, եթե երկու կամ ավելի մուտքային փոփոխականներ հավասար են 1-ի, մնացած դեպքերում ելքը պետք է լինի 0:

**Խնդիր 2.9.** Կառուցել թվային շղթա, որն ունի երեք մուտք ( $x$ ,  $y$  և  $z$ ): Շղթայի ելքում ստացվում է 1, եթե մուտքային փոփոխականներից ճիշտ մեկը կամ երկուսը հավասար են 1-ի, մնացած դեպքերում ելքը պետք է լինի 0:

**Խնդիր 2.10.** Կառուցել թվային շղթա, որն ունի չորս մուտք ( $w$ ,  $x$ ,  $y$  և  $z$ ): Շղթայի ելքում ստացվում է 1, եթե երեք կամ ավելի մուտքային փոփոխականներ հավասար են 1-ի, մնացած դեպքերում ելքը պետք է լինի 0:

**Խնդիր 2.11.** Կառուցել թվային շղթա, որի մուտքային երեք փոփոխականները երկուական թվի բիթերն են: Շղթան ունի երեք ելք ( $a$ ,  $b$  և  $c$ ), որոնք նույնպես երկուական թվի բիթեր են: Երբ մուտքային թիվը 0, 1, 2 կամ 3 է, ելքային թիվը մեկով մեծ է մուտքայինից: Երբ մուտքային թիվը 4, 5, 6 կամ 7 է, ելքային թիվը մեկով պակաս է մուտքայինից: Ցույց տալ իսկության աղյուսակը, ֆունկցիաների պարզեցման քայլերը և վերջնական շղթան:

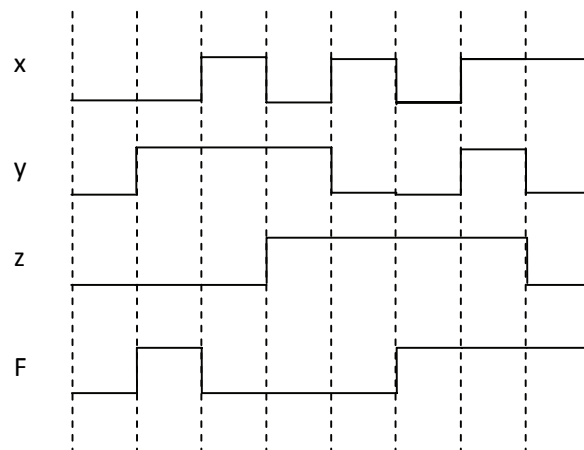
**Խնդիր 2.12.** Կառուցել տրամաբանական շղթա, որն իրականացնում է հետևյալ բուլյան ֆունկցիան.

$$F(x, y, z) = xz + (xy + \bar{z}) :$$

**Խնդիր 2.13.** Կառուցել տրամաբանական շղթա, որն իրականացնում է հետևյալ բուլյան ֆունկցիան.

$$F(x, y, z) = (xy \oplus (y + \bar{z})) + \bar{x}z:$$

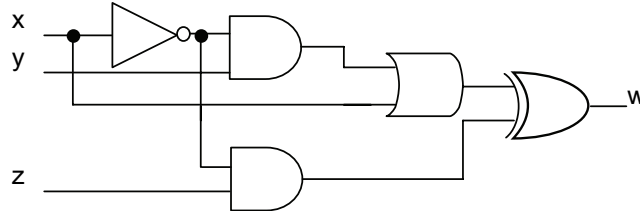
**Խնդիր 2.14.** Ստորև ցույց են տրված տրամաբանական շղթայի մուտքերի ( $x$ ,  $y$  և  $z$ ) և ելքի ( $F$ ) ժամանակային դիագրամները: Ստանալ  $F$  ֆունկցիայի բանաձևը դիզյունկտիվ նորմալ ձևով:



Նկ. 2.57. Խնդիր 2.14-ին վերաբերող ժամանակային դիագրամները

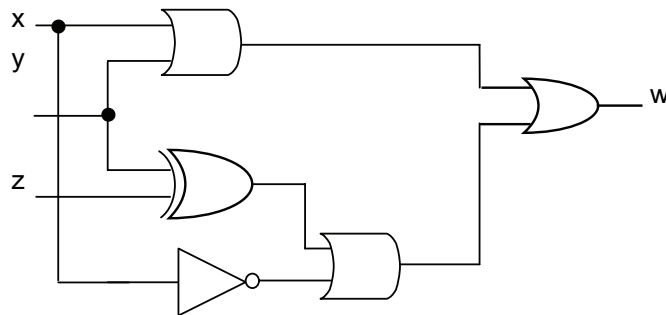


**Խնդիր 2.15.** Ստանալ հետևյալ շղթան նկարագրող իսկության աղյուսակը: Շղթայի մուտքերին հաջորդաբար կիրառելով փոփոխականների բոլոր հավաքածուները՝ կառուցել շղթայի բոլոր հանգույցների ազդանշանների ժամանակային դիագրամները:



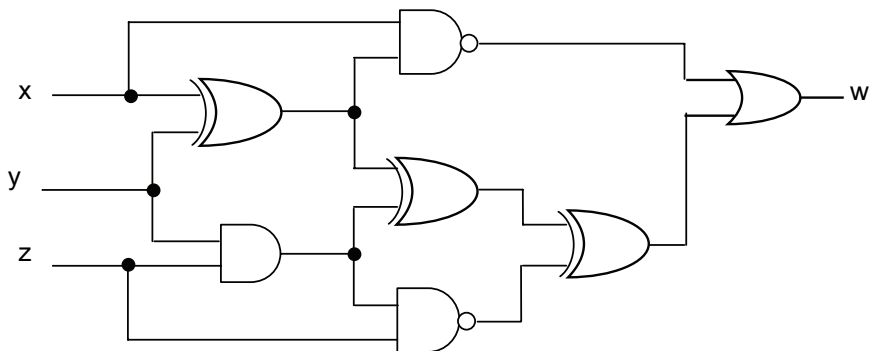
Նկ. 2.58. Խնդիր 2.15-ին վերաբերող տրամաբանական սխեման

**Խնդիր 2.16.** Ստանալ հետևյալ շղթան նկարագրող իսկության աղյուսակը: Շղթայի մուտքերին հաջորդաբար կիրառելով փոփոխականների բոլոր հավաքածուները՝ կառուցել շղթայի բոլոր հանգույցների ազդանշանների ժամանակային դիագրամները:



Նկ. 2.59. Խնդիր 2.16-ին վերաբերող տրամաբանական սխեման

**Խնդիր 2.17.** Ստանալ հետևյալ շղթան նկարագրող իսկության աղյուսակը: Շղթայի մուտքերին հաջորդաբար կիրառելով փոփոխականների բոլոր հավաքածուները՝ կառուցել շղթայի բոլոր հանգույցների ազդանշանների ժամանակային դիագրամները:



Նկ. 2.60. Խնդիր 2.17-ին վերաբերող տրամաբանական սխեման

**Խնդիր 2.18.** Համատեղ պարզեցնել հետևյալ ֆունկցիաները նվազագույն տարրերով 2 ելքով թվային շղթա կառուցելու համար.

$$F_1 = \sum m(0, 2, 4, 6, 7, 9, 10^*, 11^*),$$

$$F_2 = \sum m(2, 4, 9, 10, 15, 0^*, 13^*, 14^*):$$

Նվազարկել այդ ֆունկցիաները առանձին-առանձին և կառուցել համապատասխան թվային շղթաները: Համարել, որ մուտքային փոփոխականների ուղիղ և ժխտված ձևերը առկա են:

Համեմատել այդ երկու տարբերակների շղթաների բարդությունները՝ տարրերի և տարրերի մուտքերի թվերի գումարը:

**Խնդիր 2.19.** Կրկնել խնդիր [նդիր 2.18-ը հետևյալ ֆունկցիաների համար.

$$F_1 = \Sigma m(1,4,5,11, 27,28,10^*, 12^*, 14^*, 15^*, 20^*, 31^*),$$

$$F_2 = \Sigma m(0,1,2,4,5,8,14, 15, 16,18,20,24,26,28,31,10^*, 11^*, 12^*, 27^*):$$

**Խնդիր 2.20.** Նվազարկել հետևյալ ֆունկցիաների բանաձևերը Կառնոյի քարտերով, ստանալ նվազագույն ԴՆԶ-ն, կառուցել նվազարկված շղթան ԵՎ-ՈՉ տարրերի բազիսում.

ա)  $F(x, y, z) = \Sigma(0,1,2,4),$

բ)  $F(w, x, y, z) = \Sigma(1,4,5,6,11,12,13,14),$

դ)  $F(w, x, y, z) = \Sigma(0,1,2,3,7,8,10,11,15),$

ե)  $F(w, x, y, z) = \Sigma(1,2,4,7,8,11,13,14):$

**Խնդիր 2.21.** Նվազարկել հետևյալ ոչ լրիվ որոշված ֆունկցիաների բանաձևերը Կառնոյի քարտերով, ստանալ նվազագույն ԴՆԶ-ն, կառուցել նվազարկված շղթան ԿԱՄ-ՈՉ տարրերի բազիսում.

ա)  $F(w, x, y, z) = \Sigma(0,1,3,5,14, 8^*, 15^*),$

բ)  $F(w, x, y, z) = \Sigma(0,1,2,8,11, 3^*, 9^*, 15^*),$

դ)  $F(w, x, y, z) = \Sigma(1,5,6,7,9,13, 4^*, 15^*):$

**Խնդիր 2.22.** Երեք ելքով համակցական շղթան նկարագրվում է հետևյալ տրամաբանական ֆունկցիաներով.

$$F(x, y, z) = \Sigma(0,1,2),$$

$$G(x, y, z) = \Sigma(1,4,6),$$

$$H(x, y, z) = \Sigma(0,1,2,4,6):$$

Նվազարկել այս համակարգը Կառնոյի քարտերով: Շղթան կառուցել ԵՎ-ՈՉ տարրերի միջոցով:

**Խնդիր 2.23.** Նվազարկել հետևյալ 5 փոփոխականի ֆունկցիաների բանաձևերը Քվայն-Մաք-Կլասկիի եղանակով, ստանալ նվազագույն ԴՆԶ-ն, կառուցել նվազարկված շղթան.

ա)  $F(v, w, x, y, z) = \Sigma(5,7,13,15,16,20,25,27,29,31),$

բ)  $F(v, w, x, y, z) = \Sigma(0,7,8,9,12,13,15,16,22,23,30,31),$

գ)  $F(v, w, x, y, z) = \Sigma(0,1,2,3,4,5,10,11,14,20,21,24,25,26,27,28,29,30),$

դ)  $F(v, w, x, y, z) = \Sigma(0,2,4,6,7,8,10,11,12,13,14,16,18,19,29,30):$

**Խնդիր 2.24.** Նվազարկել խնդիր 2.23-ի 5 փոփոխականի ֆունկցիաների բանաձևերը Կառնոյի քարտերով:

**Խնդիր 2.25.** Նվազարկել հետևյալ ոչ լրիվ որոշված 5 փոփոխականի ֆունկցիայի բանաձևը Կառնոյի քարտով, ստանալ նվազագույն ԴՆԶ-ն, կառուցել նվազարկված շղթան.

$$F(v, w, x, y, z) = \Sigma(4,6,7,9,11,12,13,14,15,20,22,25,27,28,30, 1^*, 5^*, 29^*, 31^*)$$

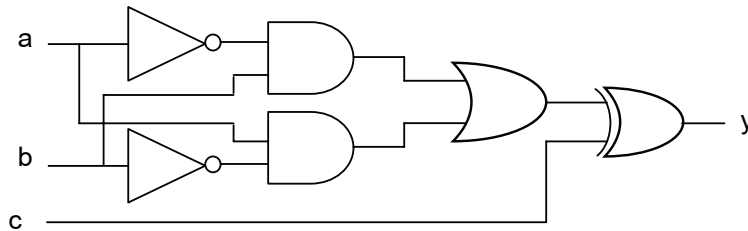
**Խնդիր 2.26.** Նվազարկել հետևյալ 6 փոփոխականի ֆունկցիաների բանաձևերը Կառնոյի քարտերով. ստանալ նվազագույն ԴԼՁ-ն, կառուցել նվազարկված շղթան.

ա)  $F(u, v, w, x, y, z) = \Sigma(1,5,9,13,21,23,29,31,37,45,53,61),$

բ)  $F(u, v, w, x, y, z) = \Sigma(0,4,8,16,24,32,34,36,37,39,40,48,50,56),$

գ)  $F(v, w, x, y, z) = \Sigma(2,4,5,6,12,13,14,15,16,17,18,19,20,21,28,29,30,31,34,38,50,51,60,61,62,63):$

**Խնդիր 2.27.** Որոշել հետևյալ շղթայում մուտքից ելք առավելագույն հապաղումը, եթե տարրերի հապաղումները հետևյալն են՝  $t_{\text{and}}=2.5, t_{\text{or}}=3.5, t_{\text{inv}}=1.5, t_{\text{xor}}=5:$



Նկ. 2.61. Խնդիր 2.27-ին վերաբերող տրամաբանական սխեման

**Խնդիր 2.28.** Կառուցել հետևյալ տրամաբանական ֆունկցիաներով որոշվող մեկ և երկու ելքերով համակցական սխեմաները ժխտված ելքերով 3x8 վերժանիչների և ԵՎ-ՈՉ տարրերի միջոցով:

ա)  $F = \Sigma(2,4,7), G = \Sigma(1,3,5,6):$

բ)  $F = \Sigma(2,3, 4,7):$

գ)  $F = \Sigma(0,1,3, 5), G = \Sigma(2,5,7):$

**Խնդիր 2.29.** Վերժանիչի և տրամաբանական տարրերի միջոցով կառուցել հետևյալ ֆունկցիաներով որոշված համակցական սխեման:

$F_1 = \bar{a}\bar{b} + ab\bar{c}, F_2 = \bar{a} + b, F_2 = \bar{a}\bar{b} + ab:$

**Խնդիր 2.30.** Վերժանիչի և ԵՎ-ՈՉ տարրերի միջոցով կառուցել հետևյալ ֆունկցիաներով որոշված համակցական սխեման:

$F = \Sigma(0, 3), G = \Sigma(1, 2, 3):$

**Խնդիր 2.31.** Կառուցել 5x32 վերժանիչ՝ օգտագործելով 3x8 դեմուլտիպլեքսորներ և մեկ 2x4 վերժանիչ:

**Խնդիր 2.32.** Կառուցել 2x4 դեմուլտիպլեքսոր՝ օգտագործելով միայն ԿԱՄ-ՈՉ տարրեր:

**Խնդիր 2.33.** Կառուցել 4-ը 2-ի առաջնահերթությամբ կոդեր, որն ունի G ելք, որը ցույց է տալիս, որ որևէ մուտքը 1 է:

**Խնդիր 2.34.** Կառուցել  $F = \Sigma(2,3, 4,7)$  ֆունկցիայով նկարագրվող սխեման 8x1 մուլտիպլեքսորի օգնությամբ:

**Խնդիր 2.35.** Կառուցել  $F = \Sigma(2, 3, 4,7)$  ֆունկցիայով նկարագրվող սխեման 4x1 մուլտիպլեքսորի օգնությամբ:

**Խնդիր 2.36.** Կառուցել հետևյալ ֆունկցիաներով նկարագրվող սխեմաները 8x1 մուլտիպլեքսորի օգնությամբ:

բ)  $F(w, x, y, z) = \Sigma(1,4,5,6,11,12,13,14),$

դ)  $F(w, x, y, z) = \Sigma(0,1,2,3,7,8,10,11,15),$

ե)  $F(w, x, y, z) = \Sigma(1,2,4,7,8,11,13,14):$

### Գլուխ 3. Տվյալների մշակման տրամաբանական սարքեր

#### 3.1. Թվերի դիրքային համակարգեր

Տասական որևէ թվի արժեք որոշվում է նրա թվանշաններից: Օրինակ, 5634 տասական թվի արժեքը հավասար է 5 հատ հազարների, 6 հատ հարյուրների, 3 հատ տասնյակների և 4 միավորների գումարի.  $5 \cdot 10^3 + 6 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0$ : Հազարը, հարյուրը, տասը, միավորը տասի ամբողջ աստիճաններ են, որոնք թվի սովորական գրառման մեջ անբացահայտ որոշվում են համապատասխան թվանշանների դիրքով: Թվի կոտորակային մասի ներկայացման համար օգտագործվում են տասի բացասական աստիճանները, օրինակ.  $185.68 = 1 \cdot 10^2 + 8 \cdot 10^1 + 5 \cdot 10^0 + 6 \cdot 10^{-1} + 8 \cdot 10^{-2}$ :

Տասական թիվը ընդհանուր դեպքում ներկայացվում է հետևյալ կերպ.

$$D = \sum_{i=-n}^p d_i 10^i : \quad (3.1)$$

Թվերի տասական համակարգը կոչվում է 10 հիմքով, քանի որ օգտագործվում են տասը թվանշաններ՝ 0, 1, 2, 3, ..., 9: Թվանշանները, կախված թվանշանի դիրքից, բազմապատկվում են տասի համապատասխան աստիճաններով:

Թվերի երկուական համակարգում օգտագործվում է միայն երկու թվանշան՝ 0 և 1: Կախված դիրքից՝ թվանշանները բազմապատկվում են երկուսի համապատասխան աստիճաններով: Երկուական թվի ընդհանուր տեսքը ներկայացվում է հետևյալ կերպ.

$$B = \sum_{i=-n}^p b_i 2^i : \quad (3.2)$$

Օրինակ, 11010.11 երկուական թվի տասական համարժեքը 26.75 է: Դա կարելի է ստանալ երկուական թվի  $b_i$  թվանշանները բազմապատկելով երկուսի համապատասխան աստիճաններով.  $11010.11 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} = 26.75$ :

Ընդհանուր դեպքում որևէ թիվ  $r$ -հիմքով համակարգում ներկայացվում է հետևյալ կերպ՝

$$D = \sum_{i=-n}^p d_i r^i, \quad (3.3)$$

որտեղ  $d_i$ -ները թվանշանների թույլատրելի արժեքներն են՝ 0, 1, 2, ...,  $(r-1)$ :

Երկուական թվում ամենաձախ դիրքի բիթը կոչվում է ավագ բիթ, իսկ ամենաաջ դիրքի բիթը կոչվում է կրտսեր բիթ:

Տվյալների անմիջական մշակումը թվային համակարգերում իրականացվում է երկուական համակարգում:

Թվերի տասից փոքր հիմքով համակարգերում, օգտագործվում են 0-ից մինչև  $(r-1)$  թվանշանները: Իսկ տասից մեծ հիմքով համակարգերում, բացի տասական թվանշաններից, օգտագործվում են նաև լատինական այբուբենի տառերը: Օրինակ, տասնվեցական համակարգում (հիմքը հավասար 16-ի) առաջին տասը թվանշանները վերցված են տասական համակարգից, իսկ մնացած վեցի համար՝ 10, 11, 12, 13, 14, 15, օգտագործվում են A, B, C, D, E, F տառերը: Օրինակ, B65F տասնվեցական թիվը ներկայացվում է հետևյալ կերպ.  $(B65F)_{16} = 11 \cdot 16^3 + 6 \cdot 16^2 + 5 \cdot 16^1 + 15 \cdot 16^0 = (46687)_{10}$ :

Տասական, երկուական, ութական և տասնվեցական համակարգերի առաջին 16 թվերը ցույց են տրված աղյուսակ 3.1-ում:

*Աղյուսակ 3.1.*

Տասական (10 հիմքով)	Երկուական (2 հիմքով)	Ութական (8 հիմքով)	Տասնվեցական (16 հիմքով)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

$r$  հիմքով համակարգում թվաբանական գործողությունները կատարվում են այն նույն կանոններով, ինչ տասական համակարգում: Բայց պետք է օգտագործել միայն  $r$  թույլատրելի թվանշաններ: Երկուական թվերի հետ գումարման, հանման և բազմապատկման գործողությունների օրինակներ ցույց են տրված նկ. 3.1-ում:

$$\begin{array}{rcl}
 C: & 1011110 & B: \quad 000220 \\
 & \backslash \quad \backslash \quad \backslash \quad \backslash & // \\
 X: & 101101 & X: \quad 101101 \\
 Y: & +100111 & Y: \quad -100111 \\
 \hline
 X+Y: & 1010100 & X-Y: \quad 000110 \\
 & & \\
 & & \quad \quad \quad 1011 \\
 & & \hline
 & & X \times Y: \quad 110111
 \end{array}$$

*Նկ. 3.1. Երկուական թվերի հետ գումարման, հանման և բազմապատկման գործողությունների օրինակներ*

Երկուական թվերի գումարը հաշվվում է այն նույն կանոններով, ինչ տասականի դեպքում, բացառությամբ, որ ցանակացած դիրքում թվանշանները կարող են լինել միայն 0 կամ 1: Որևէ կարգում բիթերի գումարումից առաջացած  $C$  փոխանցումն օգտագործվում է հաջորդ կարգի բիթերի գումարման ժամանակ: Հանման գործողությունն ավելի բարդ է, բայց կանոնները նույնն են, ինչ տասական համակարգում, բացառությամբ, որ որևէ կարգում բիթերի հանման ժամանակ փոխառնված  $B$  բիթը տվյալ

կարգի նվազելիին գումարում է 2 միավոր (տասական համակարգում փոխառությունը նվազելիին գումարում է տասը). Բազմապատկումը շատ պարզ է՝ բազմապատկիչ թվանշանները միայն 0 կամ 1 են: Հետևաբար, մասնակի գումարները կամ 0 են, կամ հավասար են բազմապատկելիին:

### 3.1.2. Անցում մեկ թվային համակարգից մյուսին

Ցանկացած  $r$ -հիմքով թիվը կարելի է նախադասել տասականի՝ բազմապատկելով թվանշանները  $r$ -ի համապատասխան աստիճաններով և գումարելով դրանք ինչպես դա ցույց է տրված (3.2) բանաձևում: Ստորև բեված են երկուականից և ութականից տասականի անցման օրինակներ.

$$(1010.011)_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = (10.375)_{10},$$

$$(630.4)_8 = 6 \cdot 8^2 + 3 \cdot 8^1 + 0 \cdot 8^0 + 4 \cdot 8^{-1} = (408.5)_{10}:$$

Տասական համակարգից մեկ այլ  $r$  հիմքով համակարգի անցումն ավելի հարմար է իրականացնել առանձին ամբողջ մասի և կոտորակային մասի համար: Ցույց տանք այդ ձևափոխությունները օրինակների օգնությամբ:

**Օրինակ 3.1.** Ձևափոխել տասական 41 թիվը երկուականի: Սկզբում 41-ը բաժանվում է 2-ի՝ տալով քանորդը 20, մնացորդը՝ 1: Քանորդը նորից բաժանվում է 2-ի՝ տալով նոր քանորդ և մնացորդ: Այս գործընթացը կրկնվում է այնքան, մինչև քանորդում ստացվի 0: Որոշելի երկուական թվի գործակիցները ստացվում են մնացորդներից, ինչպես ցույց է տրված ստորև:

Քանորդ	Մնացորդ	
41		
20	1	↑ պատասխան՝ 101001
10	0	
5	0	
2	1	
1	0	
0	1	

Նկ. 3. 2. Տասական 41 թվի ձևափոխումը երկուականի՝ հաջորդաբար 2-ի բաժանումով

Տասական ամբողջ թվերի ձևափոխումը ցանկացած  $r$  հիմքով թվի կատարվում է ճիշտ նույն եղանակով, ինչպես դա ցույց տրվեց վերը բերված օրինակում, բացառությամբ, որ տասական թիվը բաժանվում է  $r$ -ի, 2-ի փոխարեն:

**Օրինակ 3.2.** Ձևափոխել տասական 153 թիվը ութականի: Պահանջվող հիմքը 8 է: Սկզբում 153-ը բաժանվում է 8-ի՝ տալով քանորդը 19, մնացորդը՝ 1: Այնուհետև 19-ը բաժանվում է 8-ի՝ տալով նոր քանորդը 2 և մնացորդը՝ 3: Վերջապես, 2-ը բաժանվում է 8-ի՝ տալով նոր քանորդը 0 և մնացորդը՝ 2: Որոշելի ութական թվի գործակիցները ստացվում են մնացորդներից, ինչպես ցույց է տրված ստորև:

Քանորդ	Մնացորդ	
153		
19	1	↑
2	3	
0	2	↘

պատասխան՝  $(231)_8$

Նկ. 3.3. Տասկան 153 թվի ձևափոխումը ութականի հաջորդաբար 8-ի բաժանումով

Տասնորդական կոտորակի ձևափոխումը երկուականի իրականացվում է բազմապատկումներով, իսկ արդյունքն ստացվում է բազմապատկումներից ստացված ամբողջ մասերից: Նորից ցույց տանք այդ ձևափոխությունները օրինակների օգնությամբ:

**Օրինակ 3.3.** Ձևափոխել  $(0,6875)_{10}$  տասական թիվը երկուականի: Սկզբում  $0.6875$ -ը բազմապատկվում է  $2$ -ով՝ տալով ամբողջ և կոտորակային մասեր: Նոր կոտորակային մասը բազմապատկվում է  $2$ -ով՝ տալով նոր ամբողջ և կոտորակային մասեր: Այս գործընթացը շարունակվում է այնքան, մինչև կոտորակային մասը դառնա  $0$  կամ մինչև կոտորակային մասն ունենա բավարար ճշտություն: Երկուական թվի գործակիցները ստացվում են ամբողջ մասերից, ինչպես ցույց է տրված ստորև:

Ամբողջ		Կոտորակ	Գործակից
$0.6875 \times 2 = 1$	+	$0.3750$	$b_{-1}=1$
$0.3750 \times 2 = 0$	+	$0.7500$	$b_{-2}=0$
$0.7500 \times 2 = 1$	+	$0.5000$	$b_{-3}=1$
$0.5000 \times 2 = 1$	+	$0.0000$	$b_{-4}=1$
Պատասխան: $(0,6875)_{10} = (0.b_{-1}b_{-2}b_{-3}b_{-4})_2 = (0.1011)_2$ :			

Տասնորդական կոտորակի ձևափոխումը  $r$  հիմքով թվի կատարվում ինչպես  $2$ -հիմքի դեպքում, այն տարբերությամբ, որ կոտորակային մասը բազմապատկվում է  $r$ -ով,  $2$ -ի փոխարեն: Բազմապատկումներից ստացված ամբողջ մասերը գտնվում են  $0$ -ից  $r-1$  միջակայքում:

**Օրինակ 3.4.** Ձևափոխել  $(0,513)_{10}$  տասական թիվը ութականի:

$$\begin{aligned}
 0.513 \times 8 &= 4.104 \\
 0.104 \times 8 &= 0.832 \\
 0.832 \times 8 &= 6.656 \\
 0.656 \times 8 &= 5.248 \\
 0.248 \times 8 &= 1.984 \\
 0.984 \times 8 &= 7.872
 \end{aligned}$$

Վեց միջերի ճշտությամբ պատասխանը ստացվում է բազմապատկումներից ստացված ամբողջ մասերից՝  $(0,513)_{10} = (0.406517...)_{8}$ .

Ամբողջ և կոտորակային մասերով տասական թվի ձևափոխումը կատարվում է ամբողջ և կոտորակային մասերի համար առանձին: Այնուհետև պատասխանը ստաց-

վում է այդ մասերի համակցությունից: Օգտագործելով օրինակ 3.1-ի և 3.3-ի արդյունքները՝ կստացվի.  $(41.68785)_{10} = (101001.1011)_2$ :

Օրինակներ 3.2 և 3.4-ից կստացվի.  $(153.513)_{10} = (231.406517)_8$ :

### 3.1.3. Ութական և տասնվեցական թվեր

Երկուական, ութական և տասնվեցական թվերի փոխադարձ ձևափոխությունները կարևոր նշանակություն ունեն թվային համակարգերում: Քանի որ  $2^3=8$  և  $2^4=16$ , ապա յուրաքանչյուր ութական թվանշանի համապատասխանում են երեք երկուական նիշեր և յուրաքանչյուր տասնվեցական թվանշանի համապատասխանում են չորս երկուական նիշեր: Երկուական թիվը ութականի ձևափոխելու համար երկուական թիվը բաժանվում է երեք նիշերով խմբերի՝ եռյակների, սկսած ամբողջ մասը կոտորակայինից բաժանող կետից դեպի աջ և դեպի ձախ: Այնուհետև, յուրաքանչյուր եռյակ փոխարինվում է համապատասխան ութական թվանշանով: Հետևյալ օրինակը բացատրում է նկարագրված ձևափոխությունը.

$(10\_110\_001\_101\_011.111\_100\_000\_110)_2 = (26153.7406)_8$ :

Երկուականից տասնվեցականի ձևափոխությամբ կատարվում է համանման եղանակով, բացառությամբ, որ ձևափոխվող երկուական թիվը բաժանվում է չորս նիշերով խմբերի՝ քառյակների.

$(10\_1100\_0110\_1011.1111\_0010)_2 = (2C6B.F2)_{16}$ :

Ութականից կամ տասնվեցականից երկուականի ձևափոխությունը կատարվում է վերը նկարագրվածի հակադարձ գործելակերպով: Յուրաքանչյուր ութական թվանշան ձևափոխվում է համարժեք 3-բիթ երկուական խմբերի: Նույն կերպ, տասնվեցական թվանշանները ձևափոխվում են համարժեք 4-բիթ երկուական խմբերի: Հետևյալ օրինակները պարզաբանում են նկարագրված գործելակերպերը.

$(673.124)_8 = (110\_111\_011.001\_010\_100)_2$ ,

$(306.D)_{16} = (0011\_0000\_0110.1101)_2$ :

Երկուական թվերը հարմար չեն ձեռքով աշխատելու համար, քանի որ դրանք երեքից չորս անգամ ավելի շատ նիշեր են պահանջում, քան համարժեք տասական թվերը: Ութական և տասնվեցական թվերն ավելի հարմար են, քանի որ հնարավորություն են տալիս թվերը ներկայացնել կոմպակտ տեսքով և հեշտությամբ ձևափոխվում են երկուականի:

### 3.1.4. Երկուական-կոդավորված-տասական կոդ (ԵԿՏ)

Տասական թվանշանների երկուական կոդերը պահանջում են չորս բիթ: Երկուական-կոդավորված-տասական կոդում (ԵԿՏ, Binary Coded Decimal, BCD) տասական թվանշանները ուղղակի փոխարինվում են երկուական համարժեքներով: ԵԿՏ կոդի քառյակն գործակիցներն են՝ 8, 4, 2, 1: Օրինակ, 0110 բիթերի քառյակն հա-



մապատասխանում է  $0 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 = 6$  տասական թվանշանը:

Թվային համակարգերում թվերը պահպանվում են երկուական կամ ԵԿՏ կոդերով: Տվյալների ներածման և արտածման համար հարմար է օգտվել ԵԿՏ ներկայացումից, իսկ թվաբանական գործողությունների համար ԵԿՏ կոդերը հարմար է ձևափոխել երկուականի: Թեկուզ հնարավոր է թվաբանական գործողությունները կատարել նաև ԵԿՏ թվերի հետ: Օրինակ, տասական 395 թվի երկուական կոդը հավասար է 110001011-ի և բաղկացած է 9 բիթերից: Այդ նույն թվի ԵԿՏ համարժեքը բաղկացած է 12 բիթերից՝ 0011 1001 0101: Առաջին քառյակը ներկայացնում է տասական 3-ը, հաջորդ քառյակը՝ 9-ը, և վերջին քառյակը՝ 5-ը: Շատ կարևոր է հասկանալ տասական թիվը երկուականի և ԵԿՏ-ի ձևափոխությունների տարբերությունը: Նշենք նաև, որ 1010, 1011, 1100, 1101, 1110, 1111 քառյակները չեն համապատասխանում տասական որևէ թվանշանի և ԵԿՏ կոդում անիմաստ են:

### 3.1.5. Նշանով թվերի ներկայացումը թվային համակարգերում

Մինչ այժմ քննարկվել են միայն դրական թվերը կամ ավելի ճիշտ առանց նշանի թվերը: Թվային համակարգերում հաճախ անհրաժեշտ է գործ ունենալ նշանով թվերի հետ: Բացասական թվերի ներկայացման համար կան տարբեր եղանակներ: Բայց թվային համակարգերում ավելի հաճախ օգտագործվում է բացասական թվերի ներկայացումը լրացուցիչ կոդով (դա կքննարկվի հետագայում):

Թվային համակարգերում նշանով թվերի նորմալ ներկայացումը նշան-արժեք չափաձևն է՝ թիվը բաղկացած է արժեքը ներկայացնող բիթերից և նշանը ցույց տվող բիթերից (կամ բիթից): Սովորաբար տվյալների ներկայացման համար օգտագործվում են պլուս և մինուս նշանները, ինչպես ցույց են տրված հետևյալ օրինակներում՝ +98, -57.3: Թվային համակարգերում ցանկացած տվյալ կամ ինֆորմացիա կոդավորվում է 0 կամ 1-ով: Նշանով երկուական թվերն ունեն լրացուցիչ բիթ՝ նշանի բիթ, որը ցույց է տալիս թվի նշանը: Սովորաբար նշանի բիթը գրվում է ամենաձախ դիրքում. 0=պլուս, 1=մինուս. Մնացած բիթերը ներկայացնում են թվի արժեքը: Ստորև բերված են նշանով երկուական թվերի օրինակներ՝ նշան-արժեք չափաձևով.

$$(01010101)_2 = (+85)_{10},$$

$$(11010101)_2 = (-85)_{10},$$

$$(01111111)_2 = (+127)_{10},$$

$$(11111111)_2 = (-127)_{10},$$

$$(00000000)_2 = (+0)_{10},$$

$$(10000000)_2 = (-0)_{10}:$$

Յուրաքանչյուր թվային համակարգում դրական և բացասական թվերի քանակները հավասար են:  $n$ -բիթ նշանով թվերի արժեքները ընկած են  $-(2^{n-1}-1)$ -ից  $+(2^{n-1}-1)$  միջակայքում:

Ենթադրենք ցանկանում ենք նախագծել նշան-արժեք չափաձևով նշանով թվերի գումարիչի տրամաբանական շղթա: Շղթան պետք է ստուգի գումարվող թվերի նշանները, որպեսզի որոշվի դրանց հետ գործողությունների կատարման հաջորդականությունը: Եթե երկու գումարելիներն էլ միևնույն նշանի են, ապա թվերը գումարվում են, և արդյունքին վերագրվում է գումարելիների նշանը: Եթե գումարելիների նշանները

տարբեր են, ապա դրանց արժեքները պետք է համեմատվեն, որպեսզի մեծից հանվի փոքրը և արդյունքին վերագրվի մեծի նշանը: Բոլոր այս “եթե”, “ապա”, “համեմատել” տրամաբանական պայմանները և գործողությունները բարդացնում են գումարիչի սխեման: Այդ պատճառով լրացուցիչ կոդով գումարիչներն ավելի մեծ տարածում ունեն և կքննարկվեն հաջորդ բաժիններում: Սակայն նշան-արժեք չափաձևով գումարիչներն ունեն այն առավելությունը, որ եթե գումարիչը ստեղծված է, ապա հանիչի կառուցումը դառնում է պարզ խնդիր, պետք է միայն որոշել հանելիի նշանը և նվազելիի հետ կիրառել գումարիչի մուտքերին:

### 3.1.6. Լրացուցիչ կոդեր

Թվային համակարգերում լրացուցիչ կոդերն օգտագործվում են նշանով թվերի հետ թվաբանական գործողությունների պարզեցման համար:  $r$  հիմքով համակարգերում գոյություն ունեն երկու տիպի լրացուցիչ կոդեր՝  $(a)$   $r$ -ի լրացում և  $(p)$   $(r-1)$ -ի լրացում: Օրինակ, երկուական համակարգում կլինեն 2-ի և 1-ի լրացում, տասական համակարգում՝ 10-ի և 9-ի լրացում:

$R$  հիմքով համակարգում  $n$  նիշերով դրական  $N$  թվի  $r$ -ի լրացումը որոշվում է իբրև  $r^n - N$ , եթե  $N \neq 0$  և 0 եթե  $N = 0$ : Հետևյալ թվային օրինակները պարզաբանում են այս սահմանումները.

$(52520)_{10}$  թվի 10-ի լրացումը կլինի՝  $10^5 - 52520 = 47480$ : Այս օրինակում թվի նիշերի թիվը՝  $n=5$ :

$(0.3267)_{10}$  թվի 10-ի լրացումը կլինի՝  $1 - 0.3267 = 0.6733$ : Այս օրինակում թվի ամբողջ մասը 0 է, հետևաբար՝  $10^n = 10^0 = 1$ :

$(25.639)_{10}$  թվի 10-ի լրացումը կլինի՝  $10^2 - 25.639 = 74.361$ :

$(101100)_2$  թվի 2-ի լրացումը կլինի՝  $(2^6) - (101100)_2 = (1000000 - 101100)_2 = 010100$ :

$(0.0110)_2$  թվի 2-ի լրացումը կլինի՝  $(1 - 0.0110)_2 = 0.1010$ :

10-ի լրացման սահմանումից և դիտարկված օրինակներից հետևում է, որ 10-ի լրացումը որոշելու համար պետք է ցածր կարգերի զրոները թողնել անփոփոխ, առաջին զրոյից տարբեր կարգում 10-ից հանել այդ կարգի նիշը, իսկ մնացած ավագ կարգերում 9-ից հանել այդ կարգերի թվանշանները: 2-ի լրացումը որոշելու համար պետք է անփոփոխ թողնել ցածր կարգերի զրոները և առաջին ոչ զրո նիշը՝ 1-ը, իսկ մնացած բարձր կարգերում փոխարինել 0-ն 1-ով, 1-ը 0-ով:  $r$ -ի լրացումը որոշելու ևս մեկ պարզ եղանակ կդիտարկենք  $(r-1)$ -ի լրացումը սահմանելուց հետո:

$r$ -հիմքով համակարգում  $n$  նիշերով ամբողջ մասով և  $m$  նիշերով կոտորակային մասով դրական  $N$  թվի  $(r-1)$ -ի լրացումը որոշվում է իբրև  $r^n - r^m - N$ : Հետևյալ թվային օրինակները պարզաբանում են այս սահմանումը.

$(52520)_{10}$  թվի 9-ի լրացումը կլինի՝  $10^5 - 1 - 52520 = 99999 - 52520 = 47479$ : Կոտորակային մասը բացակայում է՝  $10^{-m} = 10^{-0} = 1$ :

$(0.3267)_{10}$  թվի 9-ի լրացումը կլինի՝  $(1 - 10^{-4} - 0.3267) = 0.9999 - 0.3267 = 0.6732$ . Ամբողջ մասը բացակայում է՝  $10^n = 10^0 = 1$ :

$(25.639)_{10}$  թվի 9-ի լրացումը կլինի՝  $(10^2 - 10^{-3} - 25.639) = 99.999 - 25.639 = 74.360$ .

$(101100)_2$  թվի 1-ի լրացումը կլինի՝  $(2^6-1) - (101100)_2 = (111111-101100)_2 = 010011$ :

$(0.0110)_2$  թվի 1-ի լրացումը կլինի՝  $(1-2^{-4}-0.0110)_2 = (0.1111-0.1010)_2 = 0.1001$ :

Այս օրինակներից երևում է, որ 9-ի լրացումը ձևավորվում է՝ 9-ից հանելով յուրաքանչյուր նիշը: Երկուական թվի 1-ի լրացումը ստացվում է բիթերի արժեքների ժխտումով՝ 1-ը փոխարինվում է 0-ով, 0-ն՝ 1-ով: Քանի որ  $(r-1)$ -ի լրացումը շատ հեշտ է ստանալ, երբեմն ավելի հարմար է  $r$ -ի լրացումը ստանալ  $(r-1)$ -ի լրացումից:  $r$ -ի և  $(r-1)$ -ի լրացումների սահմանումներից և քննարկված օրինակներից հետևում է, որ  $(r-1)$ -ի լրացումից  $r$ -ի լրացումը ստանալու համար պետք է  $(r-1)$ -ի լրացմանը գումարել  $r^m$  կրտսեր բիթին: Այսպիսով, երկուական ամբողջ թվի 2-ի լրացումը ստացվում է՝ 1-ի լրացմանը 1 գումարելով: Օրինակ,  $10110100$ -ի 1-ի լրացումը  $01001011$  է, դրան գումարելով 1, կստացվի 2-ի լրացումը՝  $01001100$ :

Նշենք, որ լրացման լրացումը վերականգնում է թվի սկզբնական արժեքը:  $N$  թվի  $r$ -ի լրացումը  $(r^n-N)$  է, իսկ  $(r^n-N)$ -ի  $r$ -ի լրացումը կլինի՝  $r^n-(r^n-N)=N$ : Նույն կերպ նաև  $(r-1)$ -ի լրացման համար:

### 3.2. Նշանով թվերի գումարում

$n$ -բիթ նշանով թվերը ներկայացվում են նշանի բիթով և թվի արժեքն արտահայտող  $(n-1)$  բիթերով: Դրական թվերի նշանի բիթը հավասար է 0-ի, իսկ բացասական թվերինը՝ 1-ի: Թվային համակարգերում ընդունված թվաբանությունում դրական թվերն օգտագործվում են նորմալ տեսքով, իսկ բացասական թվերը՝ 2-ի լրացմամբ: Դա թույլ է տալիս օգտագործել գումարման նույն կանոնները՝ անկախ գումարվող թվերի նշաններից:  $n$ -բիթ ֆորմատով  $m$ -բիթ դրական թիվը՝  $m < n$ , ներկայացնելու համար ավագ  $n-m$  բիթերը լրացվում են նշանի բիթի արժեքով՝ 0-ով: Օրինակ, 5 թիվը երկուական 8-բիթ ֆորմատով ունի հետևյալ տեսքը՝  $0\_0000101$ :  $n$ -բիթ ֆորմատով  $m$ -բիթ բացասական թիվը՝  $m < n$ , ներկայացնելու համար ավագ  $n-m$  բիթերը լրացվում են նշանի բիթի արժեքով՝ 1-ով: Օրինակ, -5 թիվը երկուական 8-բիթ ֆորմատով ունի հետևյալ տեսքը՝  $1\_1111011$ , որտեղ ցածր 3 բիթերը՝  $011$  ներկայացնում են 5 թվի եռաբիթ 2-ի լրացումը: Ամենամեծ դրական  $n$ -բիթ թիվը  $(2^{n-1}-1)$ -ն է, ամենափոքր բացասական  $n$ -բիթ թիվը  $(-2^{n-1})$ -ն է, 0 թիվը ներկայացվում է  $n$  հատ զրոներով: Հետևաբար  $n$ -բիթ նշանով թվերը գտնվում են հետևյալ միջակայքում՝  $-2^{n-1} \leq N \leq 2^{n-1}-1$ , ընդամենը  $2^n$  թվեր:

Ստորև բերված օրինակները բացահայտում են նշանով թվերի երկուական գումարումը, երբ բացասական թվերը ներկայացված են 2-ի լրացումով:

**Օրինակ 3. 5.** Գումարել  $M=84_{10}=1010100_2$  և  $N=-68_{10}=-1000100_2$  8-բիթ թվերը (84-ից հանել 68): Նախ,  $M$  և  $N$ -ը պետք է ներկայացնել նշանով թվերի ֆորմատով՝  $M=0\_1010100$ ,  $N=1\_0111100$ , այստեղ՝  $0111100$ -ն  $1000100$ -ի 2-ի լրացումն է, որը կարելի հաշվել ինչպես՝  $0111011+1=0111100$ : Գումարումը կատարվում է հետևյալ կերպ.

$$\begin{array}{r} 0\_1010100 \\ + 1\_0111100 \\ \hline 10\_0010000 \end{array}$$

Ընդհանրապես նշանին տրամադրվում է մեկ բիթ, նշանային բիթից անցումը՝ լրացուցիչ բիթը, անտեսվում է: Այնպես որ պատասխանը կլինի՝  $0\_0010000=+16_{10}$ :

**Օրինակ 3.5.** Գումարել  $M=68_{10}=1000100_2$  և  $N=-84_{10}=-1010100_2$  8-բիթ թվերը (68-ից հանել 84): Նշանով թվերը կլինեն՝  $M=0\_1000100$ ,  $N=1\_0101100$ . Գումարումը կատարվում է հետևյալ կերպ.

$$\begin{array}{r} 0\_1000100 \\ + 1\_0101100 \\ \hline 1\_1110000 \end{array}$$

Պատասխանը  $1\_1110000$ -ն է, որը բացասակն թիվ է: Դրա բացարձակ արժեքը որոշվում է՝ վերցնելով 2-ի լրացումը՝  $0\_0001111+1=0\_010000=16$ .

Երկու n-բիթ նշանով թվերի գումարման ժամանակ հնարավոր է, որ արդյունքի արժեքը ստացվի (n-1)-ից շատ բիթերով, որի պատճառով գումարի արժեքը ճիշտ չի լինի: Դա բիթերի թվի գերլցում է: Ակնհայտ է, որ գերլցումը հնարավոր է միայն, երբ գումարվում են նույն նշանով թվեր:

**Օրինակ 3.7.** Գումարել  $M=68_{10}=1000100_2$  և  $N=84_{10}=1010100_2$  8-բիթ թվերը: Գումարը 152 է, որը մեծ է 127-ից՝ ամենամեծ 8-բիթ նշանով ֆորմատով ներկայացված դրական թվից: Հետևաբար, 8-բիթ արդյունքը ճիշտ չէ: Նշանով գումարելիներն են՝  $M=0\_1000100$ ,  $N=0\_1010100$ : Գումարումը կատարվում է հետևյալ կերպ.

$$\begin{array}{r} 0\_1000100 \\ + 0\_1010100 \\ \hline 1\_0011000 \end{array}$$

Գումարը՝  $1\_0011000$  բացասական թիվ է, իսկ գումարելիները դրական են, հետևաբար պատասխանը սխալ է. տեղի է ունեցել գերլցում:

**Օրինակ 3.8.** Գումարել  $M=-68_{10}=-1000100_2$  և  $N=-84_{10}=-1010100_2$  8-բիթ թվերը: Գումարը -152 է, որը փոքր է -128-ից՝ ամենափոքր 8-բիթ նշանով ֆորմատով ներկայացված բացասական թվից: Հետևաբար, 8-բիթ արդյունքը ճիշտ չէ: Նշանով գումարելիներն են՝  $M=1\_0111100$ ,  $N=1\_0101100$ : Գումարումը կատարվում է հետևյալ կերպ.

$$\begin{array}{r} 1\_0111100 \\ + 1\_0101100 \\ \hline 0\_1101000 \end{array}$$

Գումարը՝  $0\_1101000$  դրական թիվ է, իսկ գումարելիները բացասական են, հետևաբար պատասխանը սխալ է. տեղի է ունեցել գերլցում:

Համման համար կարելի է օգտագործել գումարման կանոնները, բացի այն բանից, որ հանելին միշտ պետք է ներկայացվի 2-ի լրացումով՝ անկախ նշանից՝  $M-N = M + (N\text{-ի երկուսի լրացումը})$ :

### 3.3. Երկուական գումարիչներ

Թվային համակարգերում ինֆորմացիայի մշակման հիմնական ֆունկցիաներից են թվաբանական գործողությունները: Ամենահիմնական թվաբանական գործողությունը երկու միաբիթ երկուական թվերի գումարումն է: Այս պարզագույն գումարումը բաղկացած է չորս տարրական գործողություններից  $0+0=0$ ,  $0+1=1$ ,  $1+0=1$  և  $1+1=10$ : Առաջին երեք գործողությունները տալիս են 1-բիթ արդյունք, բայց չորրորդը, երբ երկու գումարելիներն էլ 1-են, տալիս է երկբիթ արդյունք: Արդյունքի ավագ բիթը կոչվում է անցում: Երբ գումարվում են բազմաբիթ երկուական թվեր, ապա որևէ կարգի բիթերի գումարումից առաջացած անցումը գումարվում է հաջորդ բարձր կարգի բիթերին: Երկու բիթեր գումարող համակցական շղթան կոչվում է կիսագումարիչ: Երեք բիթեր՝ տվյալ կարգի երկու բիթերը և նախորդ կարգից առաջացած անցումը, գումարող համակցական շղթան կոչվում է լրիվ գումարիչ: Կիսագումարիչ անվանումը գալիս է այն բանից, որ լրիվ գումարիչ կարելի ստանալ երկու կիսագումարիչներից:

#### 3.3.1. Կիսագումարիչ

Կիսագումարիչն ունի երկու մուտք՝  $x$  և  $y$  գումարելիներ և երկու ելք՝  $S$  գումար և  $C$  անցում: Կիսագումարիչի իսկության աղյուսակը ցույց է տրված աղյուսակ 3.2-ում:

Աղյուսակ 3.2

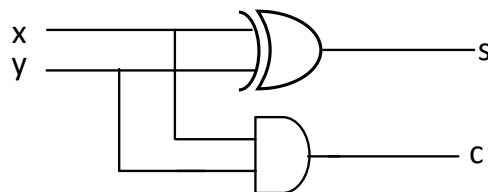
$x$	$y$	$C$	$S$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Անցման ելքը 1 է միայն այն դեպքում, երբ երկու մուտքերն էլ 1 են:  $S$  ելքը ներկայացնում է գումարի ցածր կարգի բիթը: Իսկության աղյուսակից որոշված ելքերի տրամաբանական ֆունկցիաներն ունեն հետևյալ տեսքերը.

$$S = x \oplus y = \bar{x}y + x\bar{y}, \quad (3.4)$$

$$C = x \& y: \quad (3.5)$$

Կիսագումարիչը կարելի կառուցել Բացառող-ԿԱՄ և ԵՎ տարրերով, ինչպես ցույց է տրված նկ. 3.4-ում:



Նկ. 3.4. Կիսագումարիչի տրամաբանական սխեման

### 3.3.2. Լրիվ գումարիչ

Լրիվ գումարիչը համակցական շղթա է, որը ձևավորում է երեք մուտքի բիթերի թվաբանական գումարը: Այն ունի երեք մուտք և երկու ելք: Մուտքի երկու փոփոխականները նշվում են  $x$  և  $y$ , որոնք գումարելիների բիթերն են: Երրորդ մուտքը նշվում է  $C_{in}$ , որը նախորդ կարգի գումարումից առաջացած անցումն է: Երկու ելքերը նշվում են՝  $S$  գումարի բիթի համար և  $C_{out}$  տվյալ կարգից հաջորդ կարգ անցման համար: Լրիվ գումարիչի ելքային ֆունկցիաների իսկության աղյուսակը բերված է աղյուսակ 3.3-ում:

Աղյուսակ 3.3

x	y	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Նկ. 3.5-ում ցույց են տրված ելքային ֆունկցիաների քարտերը:  $S$  ֆունկցիայի քարտում սոսնձվող վանդակներ չկան,  $C_{out}$  ֆունկցիան պարզեցվում է՝

$$S = \bar{x}\bar{y}C_{in} + \bar{x}y\bar{C}_{in} + x\bar{y}\bar{C}_{in} + xyC_{in} = x \oplus y \oplus C_{in}, \quad (3.6)$$

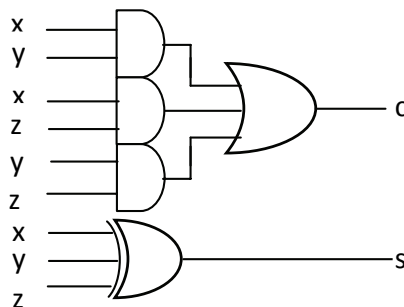
$$C_{out} = xy + xC_{in} + yC_{in}: \quad (3.7)$$

S	C <sub>in</sub>	xy			
		00	01	11	10
0	0		1		1
1	1	1		1	

C <sub>out</sub>	C <sub>in</sub>	xy			
		00	01	11	10
0	0			1	
1	1		1	1	1

Նկ. 3.5. Լրիվ գումարիչի ելքային ֆունկցիաների քարտերը

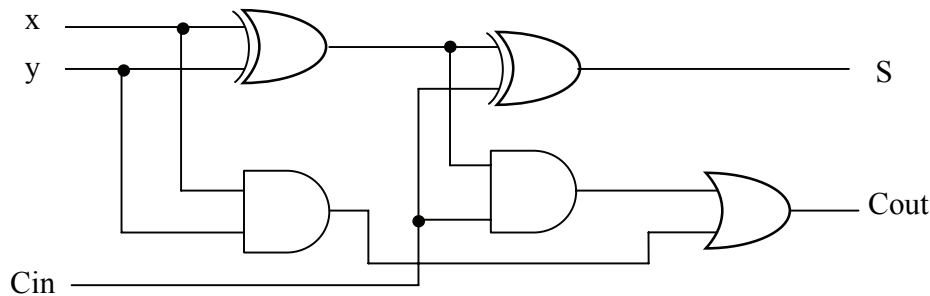
Լրիվ գումարիչի սխեմայի կառուցման մեկ օրինակ ցույց է տրված նկ. 3.6-ում: Այն ընդգրկում է մեկ 3-մուտք Բացառող-ԿԱՄ տարր  $S$  ֆունկցիայի իրականացման համար, երեք 2եվ տարրեր և մեկ 3ԿԱՄ տարր  $C_{out}$  ֆունկցիայի իրականացման համար:



Նկ. 3.6. Լրիվ գումարիչի տրամաբանական սխեման

Լրիվ գումարիչի սխեման կարելի է կառուցել երկու կիսագումարիչների և մեկ ԿԱՄ տարրի միջոցով, ինչպես ցույց է տրված նկ. 3.7-ում: Երկրորդ կիսագումարիչի S ելքը ստացվում է  $C_{in}$ -ի և առաջին կիսագումարիչի ելքի միջև Բացառող-ԿԱՄ գործողությամբ: Երկրորդ կիսագումարիչի  $C_{out}$  ելքն ստացվում է հետևյալ գործողությամբ.

$$C_{out} = C_{in}(x \oplus y) + xy = C_{in}(x\bar{y} + \bar{x}y) + xy = C_{in}x\bar{y} + C_{in}\bar{x}y + xy = x(C_{in}\bar{y} + y) + y(C_{in}\bar{x} + x) + xy = x(C_{in} + y)(y + \bar{y}) + y(C_{in} + x)(x + \bar{x}) + xy = xC_{in} + xy + C_{in}y + xy + xy = xC_{in} + yC_{in} + xy:$$



Նկ. 3.7. Լրիվ գումարիչի սխեման՝ կառուցված երկու կիսագումարիչների և մեկ ԿԱՄ տարրի միջոցով

### 3.4. Հանող տրամաբանական շղթա (հանիչ)

Երկու եկուսական թվերի հանումը կարելի է կատարել՝ գումարելով հանելիի 2-ի լրացումը նվազելիին: Այս եղանակով հանման գործողությունը դառնում է գումարման գործողություն, որը կարելի է կատարել լրիվ գումարիչով:

Կարելի է նաև հանման գործողություն կատարել ուղղակի եղանակով. հանելիի յուրաքանչյուր բիթ հանվում է նվազելիի համապատասխան բիթից, և ձևավորվում է տարբերության բիթը: Եթե նվազելիի բիթը փոքր է հանելիի բիթից, հաջորդ բարձր կարգի նվազելի բիթից փոխառնվում է 1: 1-ի փոխառնման երևույթը պետք է իրագործվի տվյալ կարգից հաջորդ կարգ փոխառնության երկուսական ազդանշանի փոխանցումով: Ինչպես որ կան կիսագումարիչ և լրիվ գումարիչ, այնպես էլ կան կիսահանիչ և լրիվ հանիչ:

#### 3.4.1. Կիսահանիչ

Կիսահանիչը համակցական շղթա է, որը հանում է երկու բիթեր և ձևավորում դրանց տարբերությունը: Այն ունի ևս մեկ ելք, որը ցույց է տալիս, որ 1 է փոխառնված: Նշանակենք նվազելիի բիթը x, իսկ հանելիին՝ y: x-y տարբերությունը ձևավորելու համար պետք է ստուգել x և y-ի արժեքները: Եթե  $x \geq y$ , ապա հնարավոր է երեք դեպք՝ 0-0=0, 1-0=1 և 1-1=0. Արդյունքը կոչվում է տարբերության բիթ: Եթե  $x < y$ , ապա ունենք 0-1, և անհրաժեշտ է հաջորդ կարգից փոխառնել 1: Հաջորդ կարգից փոխառնած 1-ը տվյալ կարգի նվազելիին ավելացնում է 2: Նվազելիի 2 արժեքի դեպքում տարբերությունը կորոշվի՝ 2-1=1: Կիսահանիչն ունի երկու ելք՝ մեկում գեներացվում է տարբերու-

թյունը՝ D, իսկ մյուսում՝ փոխառության B բիթը, որը հաջորդ կարգին տեղեկացնում է, որ 1 է փոխառված: Կիսահանիչի իսկության աղյուսակը ցույց է տրված աղյուսակ 3.4-ում:

Աղյուսակ 3.4

x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Փոխառության B բիթը 0 է, քանի դեռ  $x \geq y$ : Այն 1 է դառնում, երբ  $x=0$  և  $y=1$ : Այս դեպքում D ելքում ստացվում է  $2B+x-y$  գործողության արդյունքը: Իսկության աղյուսակից ստացված ֆունկցիաներն ունեն հետևյալ տեսքերը.

$$D = x \oplus y = \bar{x}y + x\bar{y}, \quad (3.8)$$

$$B = \bar{x} \& y: \quad (3.9)$$

Հետաքրքիր է նկատել, որ D ելքի ֆունկցիան համընկնում է կիսագումարիչի S ելքի ֆունկցիայի հետ:

### 3.4.2. Լրիվ հանիչ

Լրիվ հանիչը համակցական շղթա է, որն իրականացնում է երկու բիթերի հանում՝ հաշվի առնելով, որ նախորդող ցածր կարգը 1 է փոխառել: Այս շղթան ունի երեք մուտք և երկու ելք: Երեք մուտքերը հետևյալներն են x, y և B<sub>in</sub>, որոնցով նշանակված են նվազելին, հանելին և նախորդ փոխառությունը: Երկու ելքերն են D և B<sub>out</sub>, որոնցով նշանակվում են տարբերությունը և հաջորդ փոխառությունը: Իսկության աղյուսակը ցույց է տրված աղյուսակ 3.5-ում:

Աղյուսակ 3.5

x	y	B <sub>in</sub>	B <sub>out</sub>	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Այն տողերը, որոնցում B<sub>in</sub>=0, համընկնում են կիսահանիչի տողերի հետ: Երբ  $x=0$ ,  $y=0$  և B<sub>in</sub>=1, պետք է փոխառել հաջորդ բարձր կարգից, որի համար պետք է, որ B<sub>out</sub>=1: Դա 2 է ավելացնում x-ին: Քանի որ  $2-0-1=1$ , ապա D=1: Երբ  $x=0$ ,  $y=1$  և B<sub>in</sub>=1, նորից, պետք է փոխառել՝ B<sub>out</sub>=1 և x=2: Քանի որ  $2-1-1=0$ , ապա D=0: Երբ  $x=1$ ,  $y=0$  և B<sub>in</sub>=1, կստացվի  $x-y-B_{in}=0$ , որի դեպքում B<sub>out</sub>=0: Ի վերջո, երբ  $x=1$ ,  $y=1$ , B<sub>in</sub>=1, պետք է 1 փոխառել՝ դարձնելով  $x=3$ , և  $3-1-1=1$ , այսինքն՝ D=1. Նկ. 3.8-ի քարտերից ստացված տրամաբանական ֆունկցիաներն ունեն հետևյալ տեսքերը.



$$D = \bar{x}\bar{y}B_{in} + \bar{x}y\bar{B}_{in} + x\bar{y}\bar{B}_{in} + xyB_{in} = x \oplus y \oplus B_{in}, \quad (3.10)$$

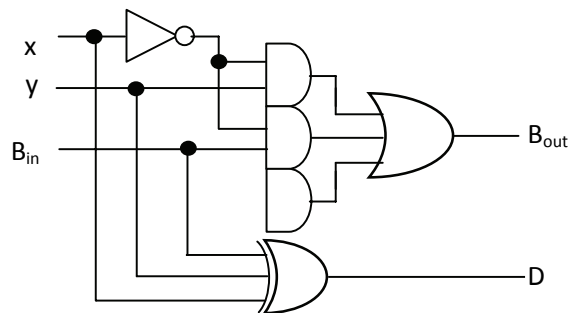
$$B_{out} = \bar{x}y + \bar{x}B_{in} + yB_{in}: \quad (3.11)$$

D B <sub>in</sub>	xy	00	01	11	10
0			1		1
1		1		1	

B <sub>out</sub> B <sub>in</sub>	xy	00	01	11	10
0				1	
1			1	1	1

Նկ. 3.8. Լրիվ հանիչի ելքային ֆունկցիաների քարտերը

Նորից նկատենք, որ լրիվ հանիչի D ելքային ֆունկցիան ճիշտ համընկնում է լրիվ գումարիչի S ելքային ֆունկցիայի հետ: Ավելին, B<sub>out</sub> ֆունկցիան նման է գումարիչի C<sub>out</sub> ելքային ֆունկցիային, տարբերվելով միայն նրանով, որ x-ը բացասված է: Այս նմանությունների շնորհիվ լրիվ գումարիչի սխեման կարելի ձևափոխել լրիվ հանիչի սխեմայի՝ միայն բացասելով անցման ելքը ձևավորող շղթայի x մուտքը:



Նկ. 3.9. Միակարգ լրիվ հանիչի սխեման

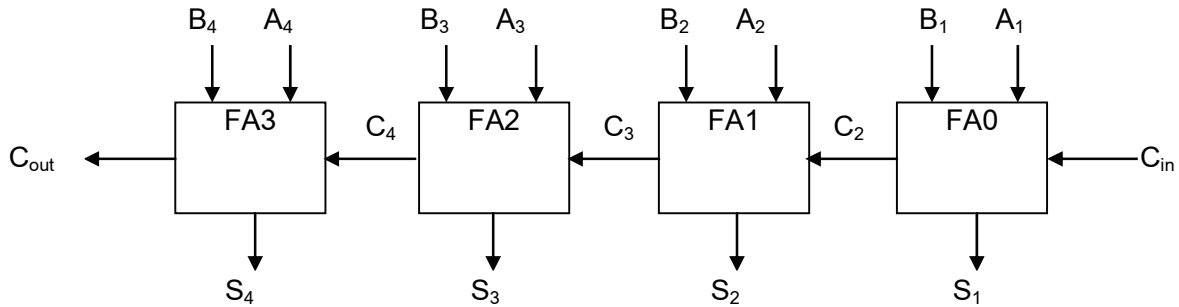
### 3.5. Երկուական զուգահեռ գումարիչ

Երկու n-բիթ թվերի գումարումը կարելի է իրականացնել զուգահեռ միացված n լրիվ գումարիչների միջոցով: Քննարկենք A=1011 և B=0011 թվերի գումարման օրինակը: Գումարը հավասար է՝ S=1110: Երբ երկու բիթ գումարվում է լրիվ գումարիչով, շղթան ձևավորում է անցման բիթ, որը պետք է օգտագործվի հաջորդ՝ մեն կարգ բարձր բիթերի գումարման ժամանակ: Դա ցույց է տրված աղյուսակ 3.6-ում և նկ. 3.10-ում:

Աղյուսակ 3.6

Ինդեքս, i	4321		Նկ. 3.7-ի լրիվ գումարիչը
Անցման մուտք	0110	C <sub>i</sub>	C <sub>in</sub>
Գումարելի	1011	A <sub>i</sub>	x
Գումարելի	0011	B <sub>i</sub>	y
Գումար	1110	S <sub>i</sub>	s
Անցման ելք	0011	C <sub>i+1</sub>	C <sub>out</sub>

Լրիվ գումարիչներով բիթերը գումարվում են՝ սկսած ցածր կարգի բիթերից, ինդեքսը՝  $i=1$ : Յուրաքանչյուր կարգում ձևավորվում են գումարի և անցման բիթերը: Ամենացածր կարգի մուտքի  $C_1$  անցումը պետք 0 լինի՝ հողանցվի: Տրված կարգի գումարիչի  $C_{i+1}$  ելքը միացվում է հաջորդ կարգի գումարիչի անցման մուտքին:

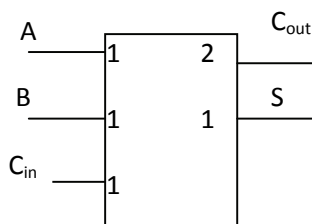


Նկ. 3. 10. Քառաբիթ զուգահեռ գումարիչ

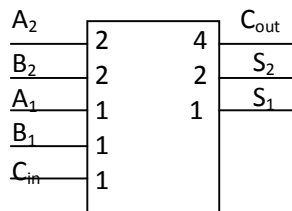
$n$ -բիթ զուգահեռ գումարիչը կառուցվում է  $n$  լրիվ գումարիչներով: Այն կարող է կառուցվել 4-բիթ, 2-բիթ, և 1-բիթ լրիվ գումարիչների ԻՍ-երի կասկադավորումից: Մեկ ԻՍ-ի անցման ելքը պետք է միացվի հաջորդ բարձր կարգի գումարիչի ԻՍ-ի անցման մուտքին:

Քառաբիթ զուգահեռ գումարիչը տիպային միջին ինտեգրացման աստիճանի ֆունկցիա է: Այն օգտագործվում է բազմաթիվ կիրառություններում, որտեղ անհրաժեշտ են թվաբանական գործողություններ: Նկատենք, որ դասական եղանակով քառաբիթ գումարիչ կառուցելու համար պետք է կառուցել  $2^9=512$  տողերով իսկության աղյուսակ, քանի որ քառաբիթ գումարիչն ունի 9 մուտք: Այն դեպքում, երբ հայտնի ֆունկցիաների կասկադացմամբ հաջողվեց կառուցել պարզ և լավ կարգավորված կառուցվածք:

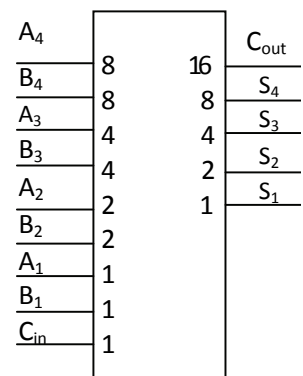
Նկ. 3.11-ում ցույց են տրված միակարգ, երկկարգ և քառակարգ գումարիչների գրաֆիկական սիմվոլները: Սիմվոլի ներսում գրված են համապատասխան մուտքերի և ելքերի քառային գործակիցները:



(ա)



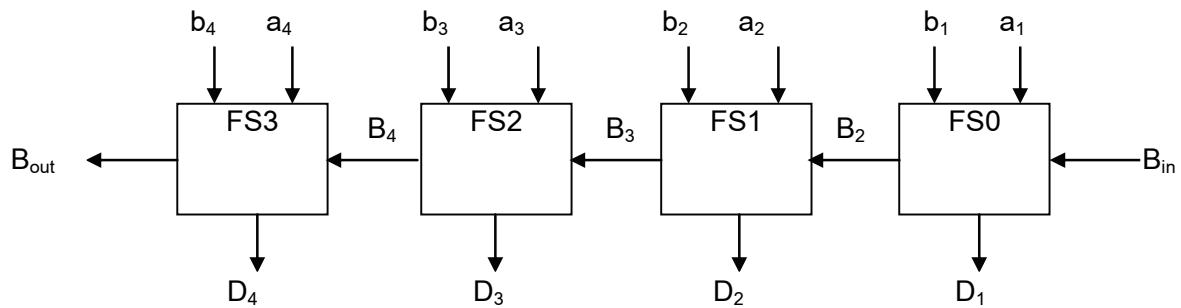
(բ)



(գ)

Նկ. 3.11. Միակարգ (ա), երկկարգ (բ), քառակարգ (գ) գումարիչների գրաֆիկական սիմվոլները

Նկատենք, որ ճիշտ նույն եղանակով կարելի է կառուցել բազմակարգ զուգահեռ հանիչ: Քառակարգ հանիչի սխեման ցույց է տրված նկ. 3.12-ում, որտեղ՝ FS0,...,FS3-ը միակարգ լրիվ հանիչներ են,  $a[4:1]$  և  $b[4:1]$ -ը քառակարգ նվազելին և հանելին են,  $B_{in}$  և  $B_{out}$ -ը փոխառության մուտքը և ելքն են,  $D[4:1]$ -ը քառակարգ տարբերությունն է: Երբ  $B_{out}=0$ , հանումը ճիշտ է կատարվել՝  $a \geq b$ : Իսկ երբ  $B_{out}=1$ , դա վկայում է, որ  $b$ -ն  $a$ -ից չի հանվում՝  $a < b$ :

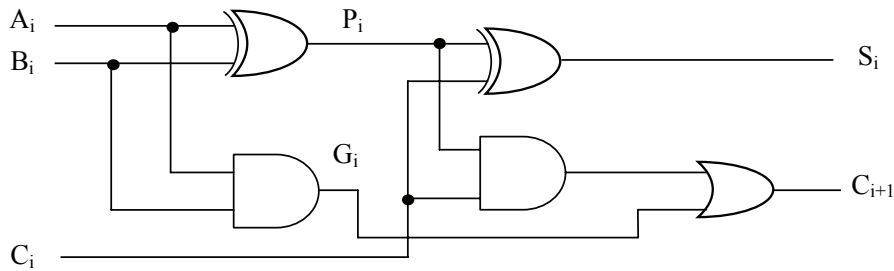


Նկ. 3. 12. Քառաբիթ զուգահեռ հանիչ

### 3.6. Փոխանցման կանխորոշման գումարիչ

Երկու թվերի զուգահեռ գումարումը ենթադրում է, որ գումարելիների բոլոր բիթերը մատչելի են միաժամանակ: Ինչպես և ցանկացած համակցական շղթայում, գումարիչում նույնպես ազդանշանը պետք է տարածվի տրամաբանական տարրերով, մինչև որ գումարի բիթերը հաստատվեն ելքերում: Տարածման լրիվ ժամանակը հավասար է մեկ տարրով հապաղման ժամանակի և շղթայում տրամաբանական տարրերի մակարդակների թվի արտադրյալին: Ամենաերկար հապաղումը զուգահեռ գումարիչում փոխանցման հաջորդաբար տարածման ժամանակն է: Քանի որ յուրաքանչյուր բիթում ելքի գումարի  $S_i$  բիթի ձևավորման համար անհրաժեշտ է մուտքի փոխանցման  $C_{in}$  բիթի առկայությունը, գումարի բիթը կհաստատվի միայն տվյալ գումարիչով փոխանցման տարածումից հետո: Դիտենք նկ. 3.9-ում ցույց տրված քառակարգ գումարիչի սխեմայում  $S_4$  ելքի ձևավորումը:  $A_4$  և  $B_4$  բիթերը առկա են այն պահից, երբ գումարիչներին կիրառվում են  $A$  և  $B$  գումարելիները, իսկ  $C_4$ -ը կհաստատվի միայն, երբ հաստատված լինի  $C_3$ -ը բիթը: Նույն կերպ  $C_3$ -ը կհաստատվի միայն, երբ հաստատված լինի  $C_2$ -ը բիթը, և այսպես՝ մինչև  $C_1$ -ը: Այսպիսով, որպեսզի հաստատվեն  $S_4$  և  $C_5$  ելքերը, պետք է  $C_1$ -ը փոխանցվի հաջորդաբար բոլոր գումարիչներով:

Փոխանցման տարածման ուղու վրա գտնվող տրամաբանական տարրերի թիվը կարելի է որոշել նկ. 3.13-ում բերված լրիվ գումարիչի սխեմայից: Չուգահեռ գումարիչի մեկ բիթի մուտքային և ելքային ազդանշանները նշված են /իմդեքսով:  $P_i$  և  $G_i$  ազդանշանները հաստատվում են  $A_i$  և  $B_i$ -ի համապատասխան տարրերով տարածումից հետո:  $P_i$  և  $G_i$  ազդանշանները կախված են միայն գումարելիների  $A_i$  և  $B_i$  բիթերից:  $C_i$  մուտքից  $C_{i+1}$  ելք ազդանշանը տարածվում է՝ անցնելով մեկ ԵՎ և մեկ ԿԱՄ տարրով, որոնք կազմում են տրամաբանական տարրերի երկու մակարդակ:



Նկ. 3.13. Ձուգահեռ գումարիչի մեկ բիթի սխեման

Քառաբիթ զուգահեռ գումարիչում  $C_5$  ելքի ձևավորմանը մասնակցում են բոլոր մուտքային ազդանշանները: Ազդանշանի տարածման ամենաերկար ուղին  $C_1$ -ից  $C_5$  տանողն է, որն ունի  $2 \times 4 = 8$  հաջորդաբար միացված տրամաբանական տարրեր: Գումարիչում հապաղման ընդհանուր ժամանակը հավասար կլինի մեկ կիսագումարիչի հապաղման և 8 տարրերի հապաղումների գումարին:  $n$ -բիթ զուգահեռ գումարիչում փոխանցման տարածման հապաղումը կորոշվի տրամաբանական տարրերի  $2n$  մակարդակներով:

Փոխանցման տարածման հապաղումը զուգահեռ գումարիչների արագագործությամբ սահմանափակող հիմնական պատճառն է: Հաշվի առնելով այն փաստը, որ բոլոր թվաբանական գործողությունները թվային սարքերում իրականացվում են հաջորդաբար գումարումների միջոցով, զուգահեռ գումարիչի հապաղման ժամանակը տվյալների մշակման համակարգերի նախագծման գերակա խնդիր է: Գոյություն ունի փոխանցման տարածման հապաղման փոքրացման մի քանի մոտեցում: Դրանցում գումարիչի սխեման բարդացվում է հապաղման փոքրացման նպատակով: Ամենատարածվածը փոխանցման կանխորոշմամբ գումարիչի սխեման է:

Նկ. 3.13-ում ցույց տրված լրիվ գումարիչի սխեմայում մտցնենք երկու նոր փոփոխական՝

$$P_i = A_i \oplus B_i, \quad (3.12)$$

$$G_i = A_i \& B_i: \quad (3.13)$$

Ելքում գումարի և փոխանցման բիթերը կորոշվեն՝

$$S_i = P_i \oplus C_i, \quad (3.14)$$

$$C_{i+1} = G_i + P_i C_i: \quad (3.15)$$

$G_i$ -ին կոչվում է փոխանցման գեներացում, այն ձևավորում է փոխանցման 1, երբ  $A_i=B_i=1$ , անկախ փոխանցման մուտքից՝  $C_i$ :  $P_i$ -ին կոչվում է փոխանցման տարածում, քանի որ այն ստեղծում է  $C_i$  մուտքից  $C_{i+1}$  փոխանցման ելք ազդանշանի տարածման շղթա: Գումարիչի յուրաքանչյուր բիթում փոխանցման ելքի բուլյան արտահայտություններում փոխարինելով  $C_i$  -ն իր արժեքով՝ ստացված նախորդ գումարիչից, փոխանցման բիթերը կարելի են րկայացնել դիզյունկտիվ նորմալ ձևով.

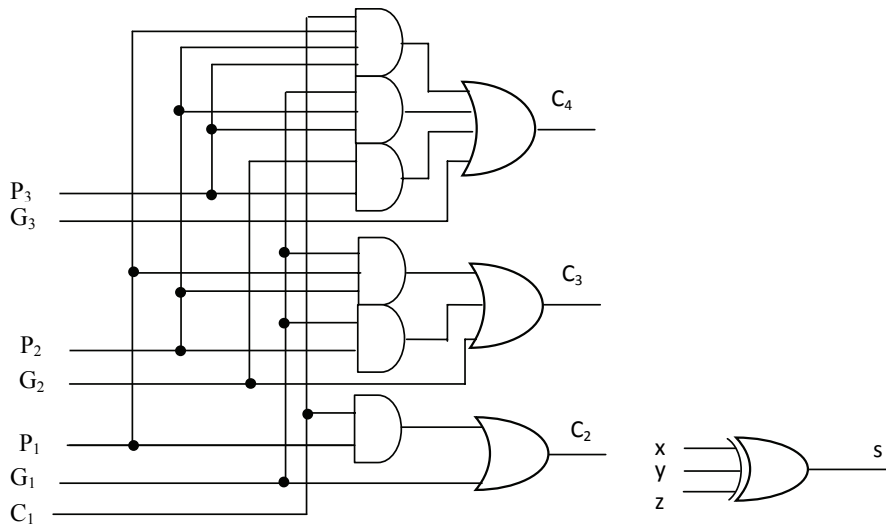
$$C_2 = G_1 + P_1 C_1,$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 C_1) = G_2 + P_2 G_1 + P_2 P_1 C_1, \quad (3.16)$$

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1,$$

$$C_5 = G_4 + P_4 C_4 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 C_1:$$

Նկ. 3.14.-ում ցույց տրված  $C_2$ ,  $C_3$ ,  $C_4$  ֆունկցիաներն իրականացնող շղթան կոչվում է փոխանցման կանխորոշման գեներատոր: Այն իրականացված է ԵՎ և ԿԱՄ տարրերի երկու մակարդակով:



Նկ. 3.14. Փոխանցման կանխորոշման գեներատորի (ՓԿԳ) տրամաբանական շղթան

Փոխանցման կանխորոշման սկզբունքով կառուցված քառաբիթ զուգահեռ գումարիչի սխեման ցույց է տրված նկ. 3.15-ում: Գումարի յուրաքանչյուր բիթ ձևավորվում է երկու Բացառող-ԿԱՄ տարրով: Բուլոր  $P_i$  և  $G_i$  ազդանշանները ձևավորվում են երկու տարրերով:  $P_i$  և  $G_i$  ազդանշանների հաստատումից հետո փոխանցման  $C_2$ ,  $C_3$ ,  $C_4$  ազդանշանները ձևավորվում են փոխանցման կանխորոշման գեներատորի շղթայով:  $C_5$ -ը նույնպես կարելի է ձևավորել ըստ (3.16) բանաձևի, սակայն ավելի նպատակահարմար է այն իրականացնել ըստ չձևավորված՝

$$C_5 = G_4 + P_4 C_4 \quad (3.17)$$

արտահայտության, որն ավելի պարզ է:

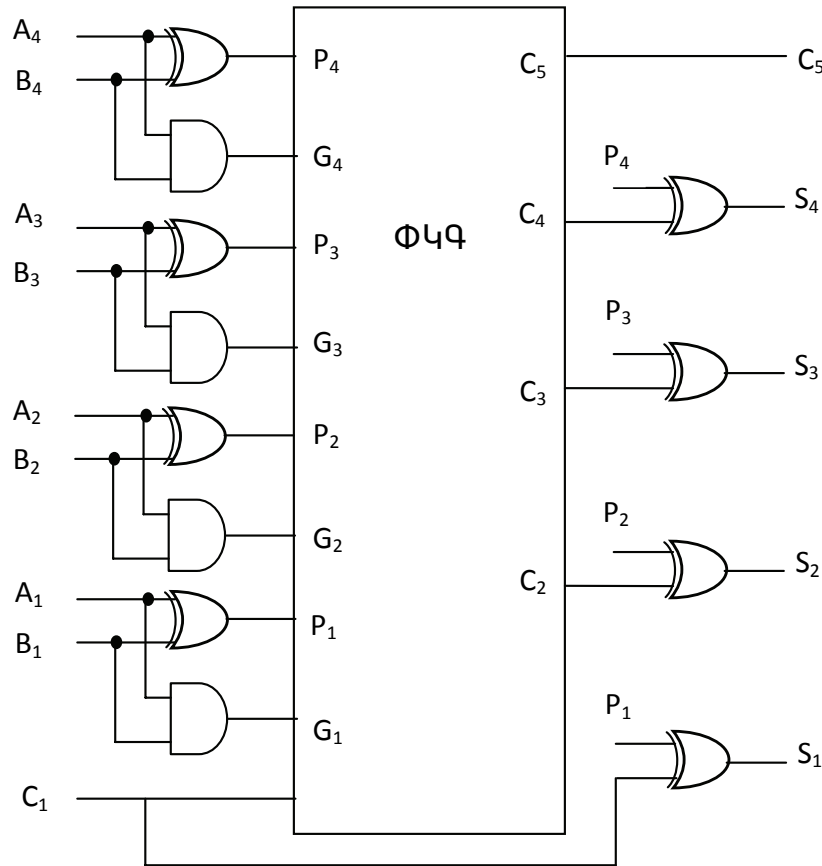
Վերջնական գումարի  $S_1$ ,  $S_2$ ,  $S_3$ ,  $S_4$  բիթերի ձևավորման հապաղման ժամանակը կլինի՝

$$t_{ds} = \max (2t_{dxor}, t_{dxor} + 2t_{dand} + t_{dor}), \quad (3.18)$$

որտեղ  $t_{dxor}$ ,  $t_{dand}$ ,  $t_{dor}$  համապատասխանաբար Բացառող-ԿԱՄ, ԵՎ, ԿԱՄ տարրերի հապաղումներն են:  $2t_{dxor}$ -ը  $(A_i, B_i)$ - $P_i$ - $S_i$  ուղու հապաղումն է,  $t_{dxor}+2t_{dand}+t_{dor}$ -ը՝  $(A_i, B_i)$ - $G_i$ - $C_i$ - $S_i$ -ինը: Փոխանցման կանխագուշակման գեներատորի հապաղումը կազմում է  $t_{dand}+t_{dor}$ :

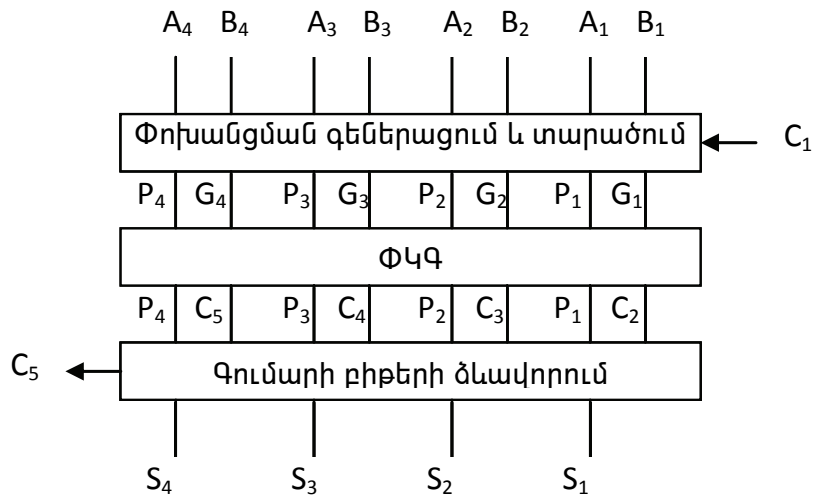
C<sub>5</sub>-ի հաստատման հապաղման ժամանակը կլինի՝

$$t_{dc5} = t_{dxor} + 2t_{dand} + 2t_{dor}: \quad (3.19)$$

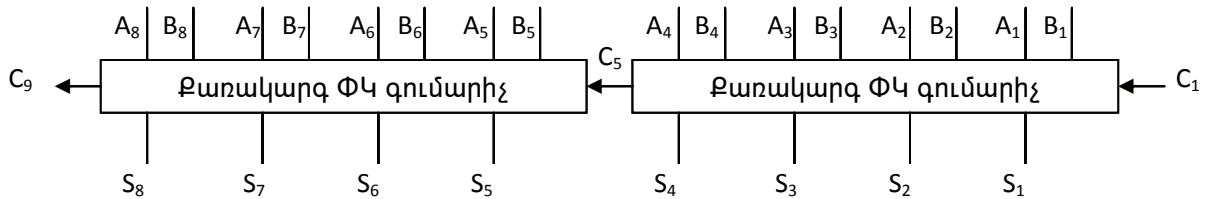


Նկ. 3.15. Փոխանցման կանխորոշման քառակարգ գումարիչի սխեման

Փոխանցման կանխորոշման գեներատորի սխեման 4-ից մեծ թվով մուտքերի դեպքում խիստ բարդանում է և գործնականում չի օգտագործվում: Բազմաբիթ երկուական թվերի զուգահեռ գումարիչները կառուցվում են քառաբիթ գումարիչների կասկադացմամբ: Նկ. 3.16-ում ցույց է տրված քառաբիթ գումարիչի պարզեցված կառուցվածքը, իսկ նկ. 3.17-ում՝ 8-կարգ գումարիչի սխեման, որը կառուցված է երկու քառակարգ գումարիչների հաջորդական միացմամբ: Պետք է նկատի ունենալ, որ փոխանցման բիթը առաջին քառյակից երկրորդին փոխանցվում է հաջորդաբար, այսինքն՝ գումարիչն անբողջությամբ վերցրած փոխանցման կանխորոշմամբ չէ:



Նկ. 3.16. Փոխանցման կանխորոշմամբ քառակարգ գումարիչի պարզեցված կառուցվածքը



Նկ. 3.17. 8-կարգ գումարիչ՝ կառուցված երկու քառակարգ գումարիչներով

### 3.7. Տասական գումարիչ

Թվային համակարգերում ուղղակի տասական թվերի հետ գործողություններ կատարելու համար սովորաբար տասական թվերը ներկայացվում են երկուական կոդավորումով՝ ԵԿՏ կոդով: Տասական գումարիչը պետք է ունենա 9 մուտք և 5 ելք, քանի որ յուրաքանչյուր տասական նիշ կոդավորվում է 4 բիթով, և շղթան պետք է ունենա նախորդ կարգից փոխանցման մուտք ու հաջորդ կարգ փոխանցման ելք: Ինը մուտքով և հինգ ելքով համակցական շղթայի դասական եղանակով նախագծման համար անհրաժեշտ կլինի կազմել  $2^9=512$  տողով իսկության աղյուսակ, որը կունենա շատ արգելված հավաքածուներ, քանի որ յուրաքանչյուր տասական մուտք ունի 6 չօգտագործվող համակցություն՝ 1010, 1011, 1100, 1101, 1110, 1111: Ելքային ֆունկցիաների պարզեցված արտահայտությունները կարելի է ստանալ ավտոմատացված նախագծման ծրագրերով: Այստեղ կքննարկենք մեկ այլ մոտեցում՝ օգտագործելով քառաբիթ երկուական գումարիչներ:

Դիտարկեք երկու տասական թվանշանների գումարումը, երբ նախորդ կարգից առկա է փոխանցում: Քանի որ յուրաքանչյուր թվանշան չի գերազանցում 9-ը, ապա ելքի գումարը չի կարող մեծ լինել  $9+9+1=19$ -ից, այստեղ 1-ը նախորդ կարգից փոխանցումն է: Ենթադրենք երկու ԵԿՏ թվանշանները կիրառվում են քառաբիթ երկուական գումարիչին: Գումարիչի կծնավորի երկուական գումարը 0-ից 19-ի սահմաններում: Այս երկուական թվերի բիթերը  $s_{4b}$ ,  $s_{3b}$ ,  $s_{2b}$ ,  $s_{1b}$  ցույց են տրված աղյուսակ 3.7-ի «երկուա-

կան գումար” սյուններում, ինչպես որ դրանք կհայտնվեն երկուական գումարիչի ելքերում: “ԵԿՏ գումար” սյուններում ցույց են տրված ԵԿՏ գումարի բիթերը՝  $S_{4d}$ ,  $S_{3d}$ ,  $S_{2d}$ ,  $S_{1d}$ , ինչպես որ դրանք պետք է հայտնվեն տասական գումարիչի ելքերում:

Աղյուսակ 3.7

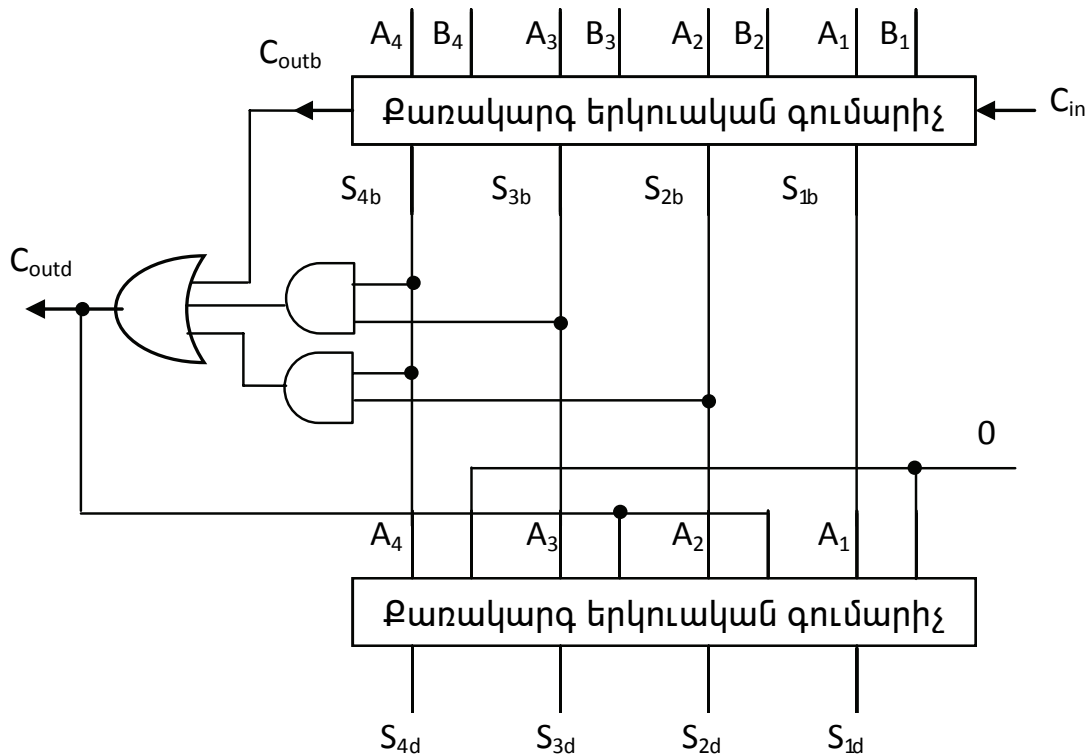
Տասական արժեքը	Երկուական գումար					ԵԿՏ գումար				
	$C_{outb}$	$S_{4b}$	$S_{3b}$	$S_{2b}$	$S_{1b}$	$C_{outd}$	$S_{4d}$	$S_{3d}$	$S_{2d}$	$S_{1d}$
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

Խնդիրն այն է, որ պետք է գտնել այնպիսի պարզ եղանակ որ “երկուական գումար” թվերը ձևափոխվեն “ԵԿՏ գումար” թվերի: Համեմատելով երկուական և տասական գումարները՝ կարելի է տեսնել, որ մինչև 1001 արժեքը դրանք համընկնում են, և ոչ մի ձևափոխություն չի պահանջվում: Երբ երկուական գումարը մեծ է 1001-ից, ստացվում է ոչ ԵԿՏ արժեք: Գումարելով 6 (0110) երկուական գումարին՝ այդ արժեքը ձևափոխվում է ԵԿՏ արժեքի և ձևավորվում է դեպի հաջորդ կարգ փոխանցում ( $C_{outd}=1$ ), ինչպես որ պահանջվում է: Երբ քառաբիթ երկուական գումարում ձևավորվում է դեպի հաջորդ կարգ փոխանցում, այդ փոխանցման քաշը 16 է: Իսկ ԵԿՏ գումարից փոխանցման քաշը տասն է: Հետևաբար երկուական գումարից փոխանցման դեպքում երկուական գումարը ԵԿՏ գումարի ձևափոխելու համար պետք է ավելացնել 6: Այսպիսով, երկուական գումարը ԵԿՏ-ի ձևափոխելու համար պետք է երկուական գումարին գումարել 0110, երբ առկա է փոխանցում դեպի հաջորդ կարգ ( $C_{outb}=1$ ) կամ երկուական գումարը հավասար է 1010, 1011, 1100, 1101, 1110, 1111 թվերից որևէ մեկին: Այս վեց հավաքածուներից առաջին երկուսում  $s_{4b}=1$  և  $s_{2b}=1$ , իսկ մնացած չորսում՝  $s_{4b}=1$  և  $s_{3b}=1$ : Երկուական գումարը ԵԿՏ-ի շտկման պայմանը և հաջորդ կարգ տասական փոխանցման արժեքը կարելի է նկարագրել հետևյալ ֆունկցիայով.

$$C_{outd} = C_{outb} + S_{4b}S_{2b} + S_{4b}S_{3b} : \quad (3.20)$$



Տասական գումարիչը պետք է ունենա (3.20) արտահայտությամբ նկարագրվող տասական շտկման շղթա և երկուական գումարին ավելացնի 6, երբ  $c_{outd}=1$ : Տասական գումարիչի սխեման ցույց է տրված նկ. 3.18-ում: Առաջին քառաբիթ երկուական գումարիչը գումարում է մուտքային ԵԿՏ թվերը և նախորդ կարգից փոխանցումը՝ ելքերում ձևավորվում է երկուական գումարը: Երկրորդ գումարիչն իրականացնում է տասական շտկում՝ երկուական գումարին գումարում է 6, երբ  $c_{outd}=1$ : Երբ  $c_{outd}=0$ , երկուական գումարին ոչինչ չի ավելացվում: Երկրորդ գումարիչից փոխանցումը կարելի է անտեսել, քանի որ փոխանցումն արդեն ձևավորված է  $c_{outd}$  ելքում:



Նկ. 3.18. Տասական (ԵԿՏ, BCD) գումարիչ՝ կառուցված երկուական գումարիչների միջոցով

Ինտեգրալ իրականացման դեպքում գումարիչի շղթայի պարզեցման համար կարելի է հաշվի առնել հետևյալ հանգամանքները՝ երկրորդ քառաբիթ գումարիչի առաջին բիթում գումարիչ պետք չէ, քանի որ  $s_{1d} = s_{1b}$ , իսկ երկրորդ և չորրորդ բիթերում կարող են լինել կիսագումարիչներ, քանի որ գումարվում է երկու բիթ: Լրիվ գումարիչ անհրաժեշտ է ունենալ միայն երրորդ բիթում:

Ջուգահեռ ո-կարգ տասական գումարիչը բաղկացած կլինի ո հատ տասական գումարիչներից, որոնցից յուրաքանչյուրի ելքի փոխանցման բիթը պետք է միացվի հաջորդի փոխանցման մուտքին:

### 3.8. Երկուական բազմաթիթ թվերի բազմապատկիչներ

Երկուական թվերի բազմապատկիչը տրամաբանական շղթա է, որն օգտագործվում է թվային համակարգերում երկու բազմաթիթ թվերի բազմապատկման համար: Այն կառուցվում է երկուական գումարիչների հիման վրա:

Օգտվելով սյունակով բազմապատկման եղանակից՝ երկու N-բիթ թվերի արտադրյալը կարելի է արտահայտել որպես թվով N հատ N-բիթ մասնակի արտադրյալների գումար, որը կունենա 2N բիթեր: Երկու քառակարգ a և b երկուական թվերի սյունակով բազմապատկման կարգը ցույց է տրված նկ. 3.19-ում:

$$\begin{array}{r}
 a_3 \quad a_2 \quad a_1 \quad a_0 \\
 \times \quad b_3 \quad b_2 \quad b_1 \quad b_0 \\
 \hline
 a_3b_0 \quad a_2b_0 \quad a_1b_0 \quad a_0b_0 \\
 \\
 a_3b_1 \quad a_2b_1 \quad a_1b_1 \quad a_0b_1 \\
 \\
 a_3b_2 \quad a_2b_2 \quad a_1b_2 \quad a_0b_2 \\
 \\
 a_3b_3 \quad a_2b_3 \quad a_1b_3 \quad a_0b_3 \\
 \hline
 p_7 \quad p_6 \quad p_5 \quad p_4 \quad p_3 \quad p_2 \quad p_1 \quad p_0
 \end{array}$$

Նկ. 3.19. Երկու քառակարգ երկուական թվերի սյունակով բազմապատկման կարգը

Մասնակի արտադրյալների  $a_i b_j$  բաղադրիչները կարելի է հաշվել 2ԵՎ տրամաբանական տարրերով՝  $a_i \times b_j = a_i \& b_j$ : Բազմապատկիչի բարդությունը պայմանավորված է մասնակի արտադրյալների գումարի հաշվմամբ:

Գոյություն ունեն երկուական բազմապատկիչների իրականացման մի շարք եղանակներ, որոնք կարելի է բաժանել երկու խմբի՝ համակցական կամ զուգահեռ բազմապատկիչներ և հաջորդական բազմապատկիչներ: Այստեղ մենք կքննարկենք միայն համակցական բազմապատկիչները, հաջորդական բազմապատկիչները կքննարկվեն հաջորդ բաժիններում:

Համակցական կամ բազմապատկիչը կառուցվում է բազմաթիվ N-բիթ երկուական գումարիչների օգտագործմամբ: Մասնակի արտադրյալները գումարվում են զույգերով, մինչև որ ստացվի ամբողջ արտադրյալը: Դրա համար անհրաժեշտ է ունենալ N-1 գումարիչներ:

Նկ. 3.20-ում ցույց է տրված առանց նշանի քառաբիթ թվերի բազմապատկիչի սխեման կառուցված  $4 \times 4 = 16$  հատ 2ԵՎ տրամաբանական տարրերի և երեք հատ քառաբիթ երկուական գումարիչների օգնությամբ:

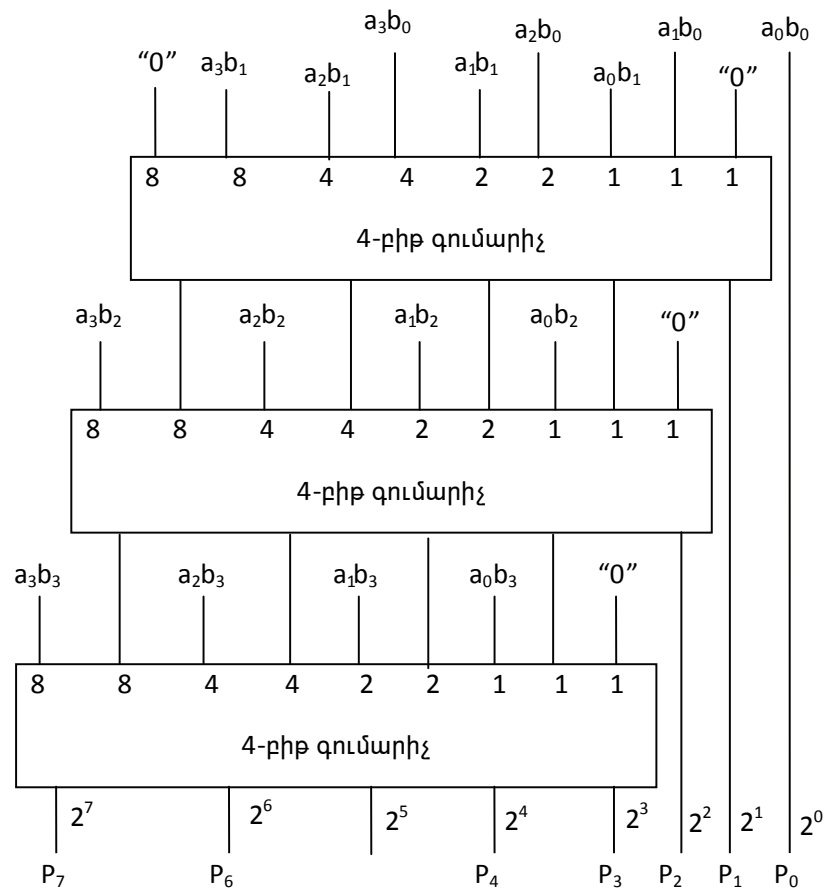
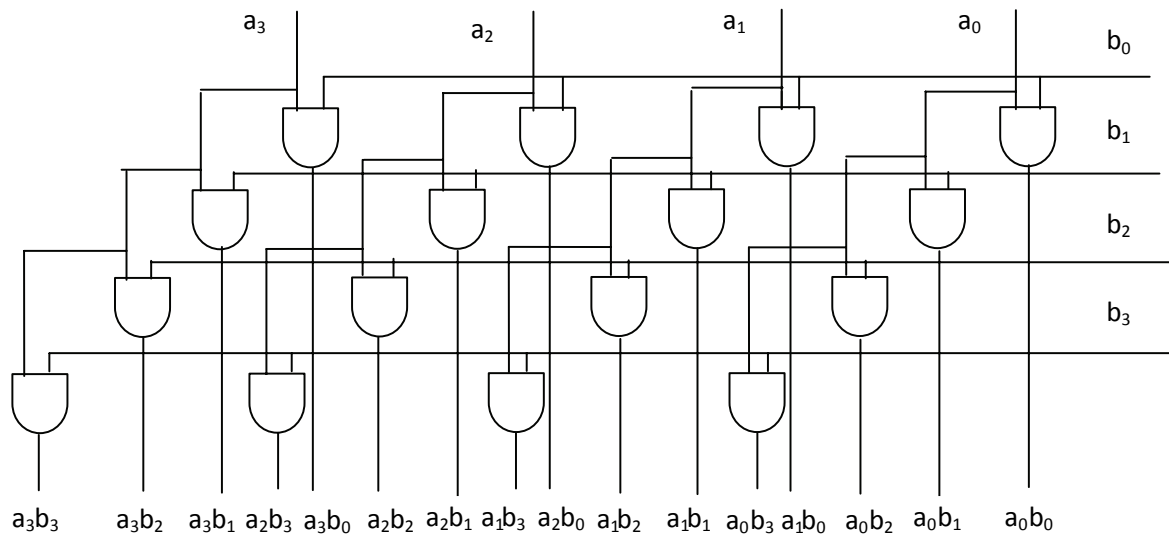
Բերված կառուցվածքով բազմապատկիչների հիմնական թերությունը ցածր արագագործությունն է՝ պայմանավորված մուտքային բիթերից ելքի բիթեր տանող ուղիների վրա գտնվող գումարիչներով ազդանշանների տարածման մեծ հապաղումներով: Օրինակ,  $a_0$  կամ  $b_0$  մուտքից  $p_7$  ելք տանող ուղու հապաղումը, հաշվի չառած 2 ԵՎ տարրերի հապաղումները, կլինի  $3t_{\text{dad}}$ , որտեղ  $t_{\text{dad}}$ –ն քառակարգ գումարիչի ցածր բիթի մուտքից մինչև փոխանցման ելք ազդանշանի տարածման հապաղումն է: Նշենք, որ ներկայումս թվային համակարգերում լայնորեն օգտագործվում են 8, 16, 32, 64-բիթ բազմապատկիչներ, որոնց դեպքում հապաղումները չափազանց մեծանում են և դառնում գործնականում անընդունելի ժամանակակից պահանջների տեսանկյունից:

Բազմաբիթ համակցական բազմապատկիչների արագագործության բարձրացման բազմաթիվ եղանակներ են առաջարկվել [9], որոնցից առավել տարածված են Վալասի և Դադայի կողմից առաջարկվածները: Դրանք միմյանց շատ նման են, բացառությամբ, որ Դադայի բազմապատկիչն ավելի արագ է արտադրիչների բարձր բիթայնության դեպքերում:

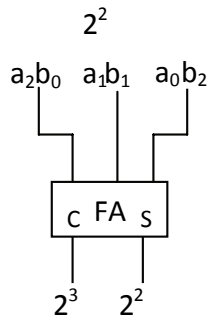
Դադայի և Վալասի բազմապատկիչներում արտադրյալը ձևավորվում է երեք քայլով.

1. Բազմապատկել արտադրիչներից մեկի յուրաքանչյուր բիթը մյուս արտադրիչի բոլոր բիթերով (կատարել  $a_i$  &  $b_j$  գործողություններ): Արդյունքում կձևավորվեն  $N^2$  մասնակի արտադրյալներ: Կախված բազմապատկված բիթերի դիրքերից՝ 2ԵՎ տարրերի ելքային գծերը կունենան տարբեր երկուական քաշեր: Օրինակ,  $a_2b_3$  արտադրյալին համապատասխանող գծի քաշը կլինի  $2^2 \times 2^3 = 32$ :
2. Կրճատել մասնակի արտադրյալների թիվը աստիճանաբար, մինչև որ ստացվի նույն քաշերով երկուսից ոչ ավելի գծեր:
3. Խմբավորել նույն քաշերով գծերը և դրանք գումարել սովորական  $(2N-1)$ -բիթ զուգահեռ գումարիչով:

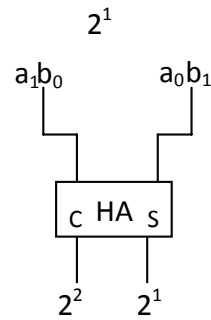
Այստեղ հիմնականում խնդիրը նույն քաշն ունեցող մասնակի արտադրյալների թվի կրճատումն է. քայլ 2: Դա կատարվում է 1-բիթ լրիվ գումարիչների և կիսագումարիչների միջոցով: Նկ. 3.19-ից կարելի տեսնել, որ  $a_2b_0$ ,  $a_1b_1$  և  $a_1b_2$  մասնակի արտադրիչներն ունեն միևնույն քաշը՝  $2^2 \times 2^0 = 2^1 \times 2^1 = 2^0 \times 2^2 = 4$ : Դրանք գումարելով լրիվ գումարիչով՝ կստանաք արդյունքի երկու գիծ՝  $s$  և  $c$ ,  $s$ -ը 4 քաշով,  $c$ -ն 8 քաշով: Այսինքն, լրիվ գումարիչը երեք մուտքի գծերից ստանում է երկու ելքի գիծ՝ հանդիսանում է 3:2 սեղմիչ (կոմպրեսոր): Նկ. 3.21ա-ում ցույց է տրված 3:2 սեղմիչի գրաֆիկական սիմվոլը: Եթե միևնույն քաշով գծերի թիվը երկուս է, ապա դրանք կարելի գումարել կիսագումարիչով: Օրինակ, 2 քաշով ունենք միայն  $a_1b_0$  և  $a_0b_1$  գծերը, դրանց գումարումը ցույց է տրված նկ. 3.21բ-ում: Չնայած կիսագումարիչի դեպքում ելքի գծերը նույնպես երկուսն են, այս գումարիչը նույնպես կոչվում է սեղմիչ՝ 2:2 սեղմիչ:



Նկ. 3.20. Քառաբիթ համակացական բազմապատկիչ



(ա)



(բ)

Նկ.3.21. 3:2 և 2:2 սեղմիչներ

Դադայի և Վալասի բազմապատկիչները տարբերվում են քայլ 2-ի իրականացմամբ: Վալասի գումարիչում այն իրականացվում է հետևյալ ալգորիթով.

- 2.1. Վերցնել միևնույն քաշով ցանկացած 3 գիծ և միացնել դրանք լրիվ գումարիչի մուտքերին: Գումարիչի ելքերում կստացվի մեկ գիծ մուտքային գծերի քաշով և մեկ գիծ երկու անգամ ավելի բարձր քաշով:
- 2.2. Եթե մնացել են միևնույն քաշով երկու գծեր և այդ քաշով ելքային գծերի թիվը հավասար է մոդուլ 3-ով 2-ի, ապա դրանք միացնել կիսագումարիչի մուտքերին: Հակառակ դեպքում դրանք տեղափոխել հաջորդ մակարդակ:
- 2.3. Եթե տվյալ քաշի միայն մեկ գիծ է մնացել, ապա այն տեղափոխել հաջորդ մակարդակ:

Այս քայլում օգտագործվում են այնքան գումարիչներ, որ միևնույն քաշով ելքային գծերի թիվը մնա մոտ 3-ի բազմապատկիչ: 3-ի բազմապատկիչը իդեալական դեպքն է, երբ լրիվ գումարիչներն օգտագործվում են իբրև 3:2 սեղմիչներ: Երբ որևէ մակարդակում մնացել է տվյալ քաշի ամենաշատը 3 գիծ, այդ մակարդակը տվյալ քաշի համար կլինի վերջինը:

Դադայի բազմապատկիչում կիսագումարիչներն օգտագործվում են այնպես, որ տվյալ մակարդակում մնա նույն քաշի միայն երկու գիծ: Դրա համար 2.2 քայլը փոխարինվում է հետևյալով.

- 2.2Դ. Եթե մնացել է տվյալ քաշի երկու գիծ և այդ նույն քաշով ելքային գծերի ընթացիկ թիվը հավասար է մոդուլ 2-ով 1 կամ 2, ապա դրանք միացնել կիսագումարիչի մուտքերի: Հակառակ դեպքում դրանք տեղափոխել հաջորդ մակարդակ:

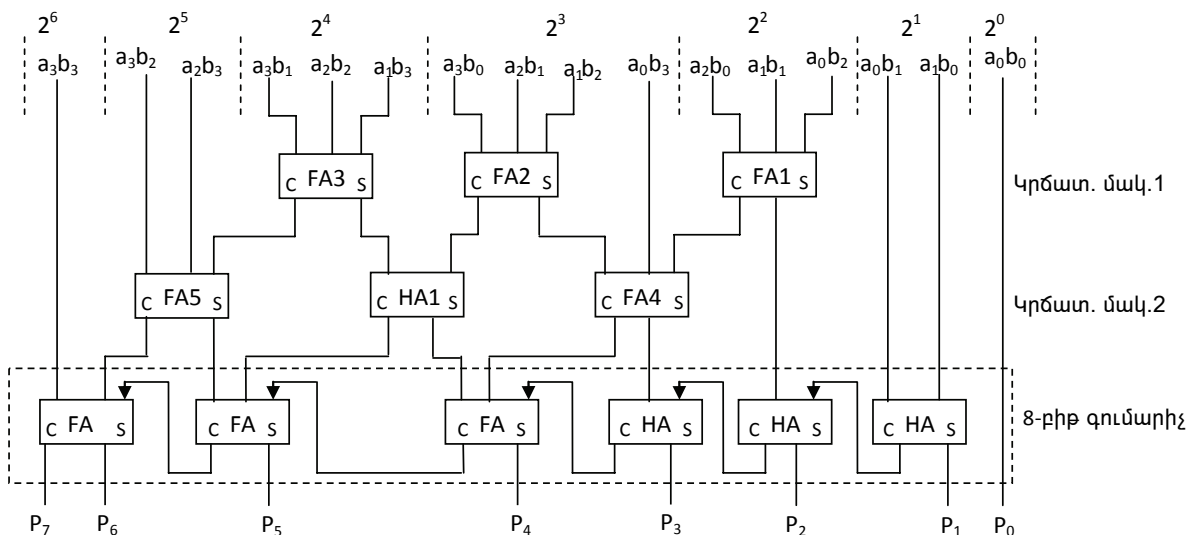
Մասնակի արտադրյալների թվի կրճատման նկարագրված ալգորիթնը կոչվում է Դադայի ծառ (գարֆ, որն ունի ծառի տեսք):

**Օրինակ 3.9.** Կառուցել Դադայի քառակարգ բազմապատկիչ՝  $n = 4$ , որը բազմապատկում է  $a_3a_2a_1a_0$  թիվը  $b_3b_2b_1b_0$  թվով:

Ստորև բերված է բազմապատկման իրականացման ալգորիթմի նկարագրությունը: Բազմապատկիչի համապատասխան կառուցվածքը ցույց է տրված նկ. 3.22-ում: Մասնակի արտադրյալների թվի կրճատման **մակարդակ 1**.

- միակ  $2^0=1$  քաշով  $a_0b_0$  արտադրյալն տեղափոխել հաջորդ մակարդակ: Ելք՝ 1 հատ 1 քաշով,

- $2^1=2$  քաշով  $a_0b_1$  և  $a_1b_0$  արտադրյալներն տեղափոխել հաջորդ մակարդակ: Ելքեր՝ 2 հատ 2 քաշով,
- օգտագործել մեկ հատ լրիվ գումարիչ  $2^2=4$  քաշով  $a_0b_2$ ,  $a_1b_1$  և  $a_2b_0$  արտադրյալները գումարելու համար (FA1): Ելքեր՝ 1 հատ 4 քաշով (գումարիչի  $s$  ելքը), 1 հատ 8 քաշով (գումարիչի  $c$  ելքը),
- օգտագործել մեկ հատ լրիվ գումարիչ  $2^3=8$  քաշով  $a_1b_2$ ,  $a_2b_1$  և  $a_3b_0$  արտադրյալները գումարելու համար (FA2), չորրորդ 8 քաշով արտադրյալը՝  $a_0b_3$  տեղափոխել հաջորդ մակարդակ: Ելքեր՝ 2 հատ 8 քաշով (գումարիչի  $s$  ելքը և  $a_0b_3$ -ը), 1 հատ 16 քաշով (գումարիչի  $c$  ելքը),
- օգտագործել մեկ հատ լրիվ գումարիչ  $2^4=16$  քաշով  $a_1b_3$ ,  $a_2b_2$  և  $a_3b_1$  արտադրյալները գումարելու համար (FA3): Ելքեր՝ 1 հատ 16 քաշով (գումարիչի  $s$  ելքը), 1 հատ 32 քաշով (գումարիչի  $c$  ելքը),
- $2^5=32$  քաշով  $a_2b_3$  և  $a_3b_2$  արտադրյալներն անց կացնել հաջորդ մակարդակ: Ելքեր՝ 2 հատ 32 քաշով,
- միակ  $2^6=64$  քաշով  $a_3b_3$  արտադրյալն տեղափոխել հաջորդ մակարդակ: Ելք՝ 1 հատ 64 քաշով:



Նկ. 3.22. Քառաբիթ քվերի Դադայի բազմապատկիչ ( $a, b$  մասնակի արտադրյալները ձևավորվում են 2ԵՎ տարրերով, ինչպես ցույց է տրված նկ. 3.20-ում)

### Մակարդակ 1-ի ելքերը՝

- քաշ 1 - 1 հատ
- քաշ 2 - 2 հատ
- քաշ 4 - 1 հատ
- քաշ 8 - 3 հատ
- քաշ 16 - 2 հատ
- քաշ 32 - 3 հատ
- քաշ 64 - 1 հատ

**Մասնակի արտադրյալների թվի կրճատման մակարդակ 2** (այս մակարդակը կլինի վերջինը, քանի որ ցանկացած քաշի մուտքային գծերի թիվը երեքից մեծ չէ)։

- 1, 2, 4, 64 քաշով գծերը տեղափոխել ելք,
- օգտագործել մեկ լրիվ գումարիչ 8 քաշով մուտքերի համար(FA4): Ելքեր՝ 1 հատ 8 քաշով (գումարիչի s ելքը), 1 հատ 16 քաշով (գումարիչի c ելքը),
- օգտագործել մեկ կիսագումարիչ 16 քաշով մուտքերի համար (HA1): Ելքեր՝ 1 հատ 16 քաշով (գումարիչի s ելքը), 1 հատ 32 քաշով (գումարիչի c ելքը): 8 քաշով գծերի այս մակարդակի լրիվ գումարիչն արդեն ձևավորել է 16 քաշով մեկ ելք: Օգտագործելով 16 քաշով մուտքերով մեկ կիսագումարիչ՝ Դադայի ալգորիթմը երաշխավորում է, որ վերջին մակարդակում կունենանք ցանկացած քաշի միայն երկու մուտք,
- օգտագործել մեկ լրիվ գումարիչ 32 քաշով մուտքերի համար(FA5): Ելքեր՝ 1 հատ 32 քաշով (գումարիչի s ելքը), 1 հատ 64 քաշով (գումարիչի c ելքը):

#### **Մակարդակ 2-ի ելքերը՝**

քաշ 1 - 1 հատ  
 քաշ 2 - 2 հատ  
 քաշ 4 - 1 հատ  
 քաշ 8 - 1 հատ  
 քաշ 16 - 2 հատ  
 քաշ 32 - 2 հատ  
 քաշ 64 - 2 հատ

Այսպիսով, քառակարգ բազմապատկիչի մասնակի արտադրյալների թվի կրճատման Դադայի ծառում օգտագործվում է հինգ լրիվ գումարիչ և մեկ կիսագումարիչ: Ազդանշանի առավելագույն հապաղումը ծառում հավասար է  $td_{tree}=2t_{dFA}$ , որտեղ՝  $t_{dFA}$  –ն լրիվ գումարիչի հապաղումն է:

Դադայի ծառի ելքերը տրվում են բազմապատկիչի  $p_i$  ելքերը ձևավորող 7-րիթ գուգահեռ գումարիչին: Նկատենք, որ այդ գումարիչը նույնպես կարելի է կառուցել արդյունավետ եղանակով, ինչպես ցույց է տրված նկ. 3.17-ում: Վերջնական արդյունքի ձևավորման հապաղումը կլինի՝  $t_d=t_{dAND}+t_{dtree}+t_{d8bitAdder}$ :

Նշանով թվերի բազմապատկման համար սովորաբար թվերը ներկայացվում են լրացուցիչ կոդով: Երբ նշանով թվերը ներկայացված են նշանի կարգով և մանտիսով, ապա կարելի է բազմապատկել մանտիսները որպես առանց նշանի թվեր, իսկ արդյունքի նշանը ձևավորել առանձին, ըստ հետևյալ արտահայտության.

$$S_P = S_{M1} \oplus S_{M2}, \quad (3.21)$$

որտեղ  $S_P$ -ն արդյունքի նշանային բիթն է,  $S_{M1}$  և  $S_{M2}$ -ը՝ արտադրիչների նշանային բիթերը:

### 3.8. Երկուական բազմաբիթ թվերի բաժանիչներ

Երկուական բազմաբիթ թվերի բաժանիչը տրամաբանական շղթա է, որն օգտագործվում է թվային համակարգերում երկու բազմաբիթ թվերի բաժանման համար: Այն կառուցվում է երկուական հանիչների հիման վրա:

Երկուական թվերի բաժանումը կատարվում է տասական բաժանման ալգորիթմով, ինչպես ցույց է տրված ստորև բերված քառաբիթ թվերի բաժանման օրինակում՝ բաժանելին՝ 1101, բաժանարարը՝ 0101 (նկ. 3.23): Պետք է նկատել, որ բաժանարարի բազմապատկումը քանորդի բարձր բիթով համարժեք է բազմապատկմանը 8-ով, այսինքն, 3 բիթով դեպի ձախ տեղաշարժին: Բաժանարարը պարբերաբար բազմապատկվում է 0-ով կամ 1-ով և հանվում է մասնակի արդյունքից այնքան ժամանակ, քանի դեռ մասնակի արդյունքը փոքր չէ հանելիից: Բազմապատկման 0 կամ 1 բիթն ընտրվում է այնպես, որ հանումից հետո մասնակի արդյունքը բացասական չլինի: Քանորդը կազմվում է բազմապատկման բիթերից, իսկ մնացորդը հավասար է մասնակի արդյունքի վերջնական արժեքին:

0001101	0101
-0101 x0	0010
0001101	
-0101 x0	
0001101	
-0101 x1	
0000011	
-0101 x0	
0000011	

Նկ. 3.23. Բազմաբիթ երկուական թվերի բաժանում

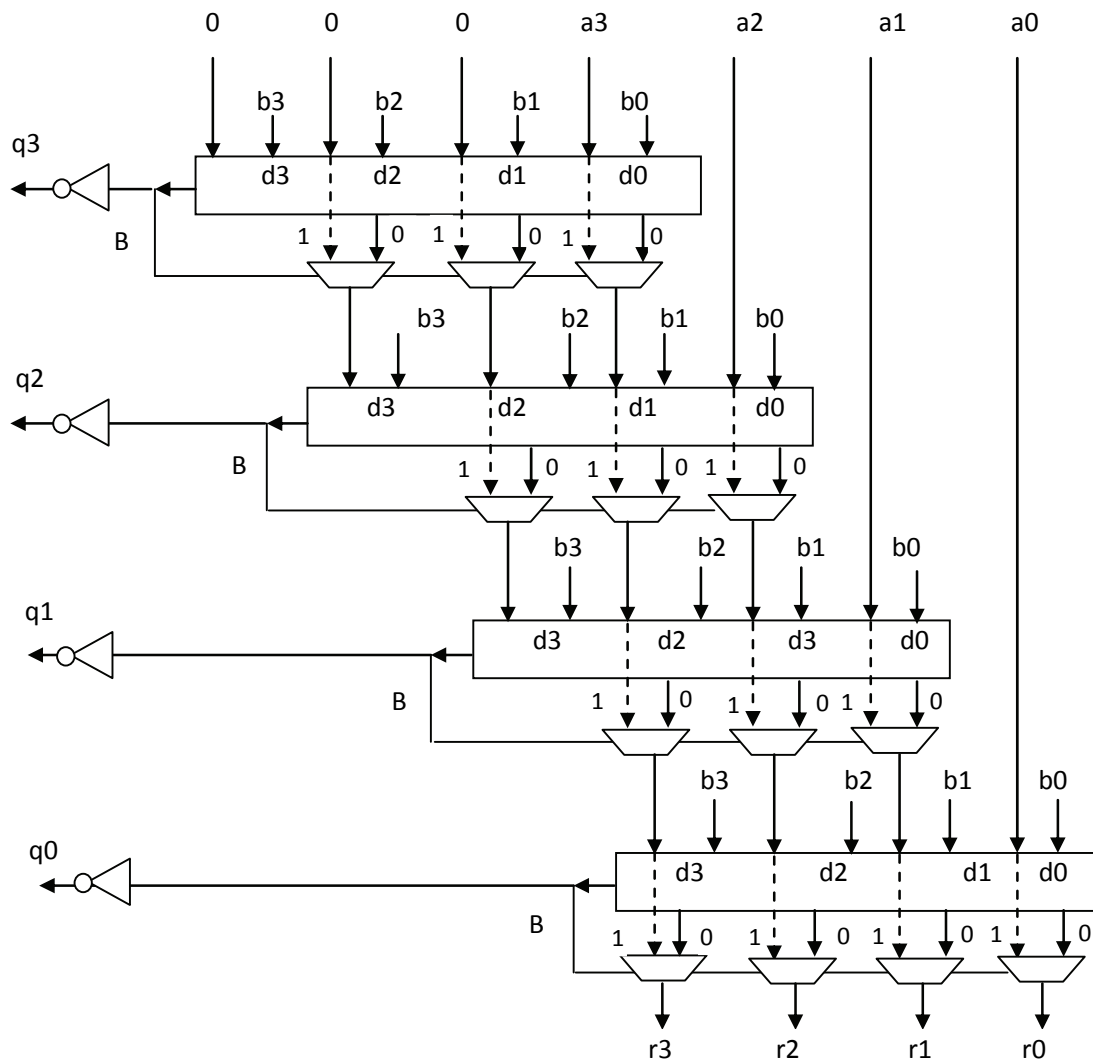
Աղյուսակ 3.8-ում ցույց են տրված բաժանման իրականացման քայլերը և տվյալների չափաձևերը: Այստեղ՝  $a = \{a_3, a_2, a_1, a_0\} = 1101$ -ը բաժանելին է,  $b = \{b_3, b_2, b_1, b_0\} = 0101$ -ը՝ բաժանարարը,  $q = \{q_3, q_2, q_1, q_0\} = 0010$ -ն՝ քանորդը,  $r = \{r_3, r_2, r_1, r_0\} = 0011$ -ը՝ մնացորդը: Նախ կազմվում է մասնակի քր արդյունքի սկզբնական արժեքը  $pr = (r, a)$  չափաձևով: Յուրաքանչյուր քայլում մասնակի արդյունքը մեկ բիթով տեղաշարժվում է ձախ և մասնակի արդյունքից հանվում է  $(q_i \times b)$ : Նկատենք, որ մասնակի արդյունքի ձախ տեղաշարժի փոխարեն  $(q_i \times b)$  արտադրյալը կարելի է տեղաշարժել աջ:



Աղյուսակ 3.8

Կատարվող գործողությունը	$q_i$	$pr_7$	$pr_6$	$pr_5$	$pr_4$	$pr_3$	$pr_2$	$pr_1$	$pr_0$
$pr$ -ի տեղաշարժ		0	0	0	0	1	1	0	1
$q_3xb$	0	0	0	0	0				
Հանում $pr-q_3xb$		0	0	0	1	1	0	1	0
$pr$ -ի տեղաշարժ		0	0	1	1	0	1	0	0
$q_2xb$	0	0	0	0	0				
Հանում $pr-q_2xb$		0	0	1	1	0	1	0	0
$pr$ -ի տեղաշարժ		0	1	1	0	1	0	0	0
$q_1xb$	1	0	1	0	1				
Հանում $pr-q_1xb$		0	0	0	1	1	0	0	0
$pr$ -ի տեղաշարժ		0	0	1	1	0	0	0	0
$q_0xb$	0	0	0	0	0				
Հանում $pr-q_0xb$		0	0	1	1	0	0	0	0

Նկարագրված ալգորիթմի ապարատուր իրականացման համար անհրաժեշտ է օգտագործել երկուական թվերի հանիչներ, որոնցով պետք է իրագործել  $pr - q_i \times b$  գործողությունները: Եթե հանումից ստացվի փոխառություն, այդ հանման գործողությունը պետք է բաց թողնվի, այսինքն՝ պետք է դրվի  $q_i = 0$ : Դա կարելի է իրագործել՝ օգտագործելով հանիչի փոխառության  $B$  ելքը. եթե  $B=0$ , ապա պետք է դրվի  $q_i = 1$  և հաջորդ մասնակի արդյունքը վերցնել հանիչի ելքերից՝  $pr - b$ , իսկ եթե  $B=1$ , ապա պետք է դրվի  $q_i = 0$  և հաջորդ մասնակի արդյունքը վերցնել հավասար նախորդին: Հանման գործողությունից առաջ  $pr$  մասնակի արդյունքը պետք է տեղաշարժել ձախ կամ  $q_i \times b$  արտադրյալը տեղաշարժել աջ: Նկ. 3.24-ում բերված է քառակարգ բաժանիչի սխեման՝ կառուցված չորս քառակարգ հանիչների և  $4 \times 3=12$  երկուդի մուլտիպլեքսորների միջոցով: Երբ հանիչի ելքային փոխառության բիթը՝  $B=1$ , ընտրվում է մուլտիպլեքսորի ձախ մուտքը՝ հաջորդ հանիչին փոխանցվում է նախորդ մասնակի արդյունքը՝ մեկ բիթով տեղաշարժված ձախ: Իսկ երբ  $B=0$ , ընտրվում է մուլտիպլեքսորի աջ մուտքը՝ հաջորդ հանիչին փոխանցվում է նախորդից ստացված տարբերության բիթերը մեկ բիթով տեղաշարժված ձախ և աջից ազատված բիթը լրացվում է  $a$ -ի հաջորդ կրտսեր բիթով: Բաժանման քանորդն ստացվում է հանիչների փոխառության ելքերի ժխտմամբ: Իսկ մնացորդը հավասար է վերջնական մասնակի արդյունքին: Պետք է հիշել, որ քննարկված բաժանիչները կիրառելի են առանց նշանի երկուական թվերի դեպքում:



Նկ. 3.24. Քառակարգ երկուական առանց նշանի թվերի բաժանիչ

Կարելի է նկատել, որ նկ. 3.24-ի բաժանիչի կառուցվածքը նման է է նկ. 3.20-ում բերված բազմապատկիչի կառուցվածքին, և դրանց բարդությունները համադրելի են: Ինչպես բազմապատկիչների դեպքում, բաժանիչների պարագայում նույնպես կարելի է կատարելագործել կառուցվածքը առավելագույն արագություն ապահովելու տեսանկյունից:

### 3.9. Երկուական գումարիչների կիրառությունը համակցական սխեմաներում

Երկուական գումարիչները, բացի ուղղակի կիրառությունից՝ երկուական թվերի գումարից, օգտագործվում են նաև համակցական սխեմաների մշակման համար: Դիտարկենք մեկ օրինակ:

**Օրինակ 3.10.** Կառուցել երկուական-տասական կոդը ավելացված-3 կոդի ձևափոխիչ: Ավելացված-3 կոդը ստացվում է երկուական-տասական կոդի հավաքածուներին գումարելով 3՝ 0011: Կոդի ձևափոխիչի իսկության աղյուսակը ցույց է տրված աղ-

յուսակ 3.9-ում: Երկուական-տասական կոդի բիթերը նշված են w, x, y, z, որոնք կոդի ձևափոխիչի մուտքերն են: Ավելացված-3 կոդի բիթերը նշված են a, b, c, d, որոնք ձևափոխիչի ելքերն են:

Աղյուսակ 3.9

#	wxyz	abcd
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Կոդի ձևափոխիչի ելքային ֆունկցիաների նվազարկման քարտերը ցույց են տրված նկ. 3.25-ում: Վեց արգելված հավաքածուները նշված են X-ով: Նվազարկված ֆունկցիաների բանաձևերը նշված են քարտերի տակ: Որոշակի ձևափոխություններից հետո ելքային ֆունկցիաները ստանում են հետևյալ տեսքը.

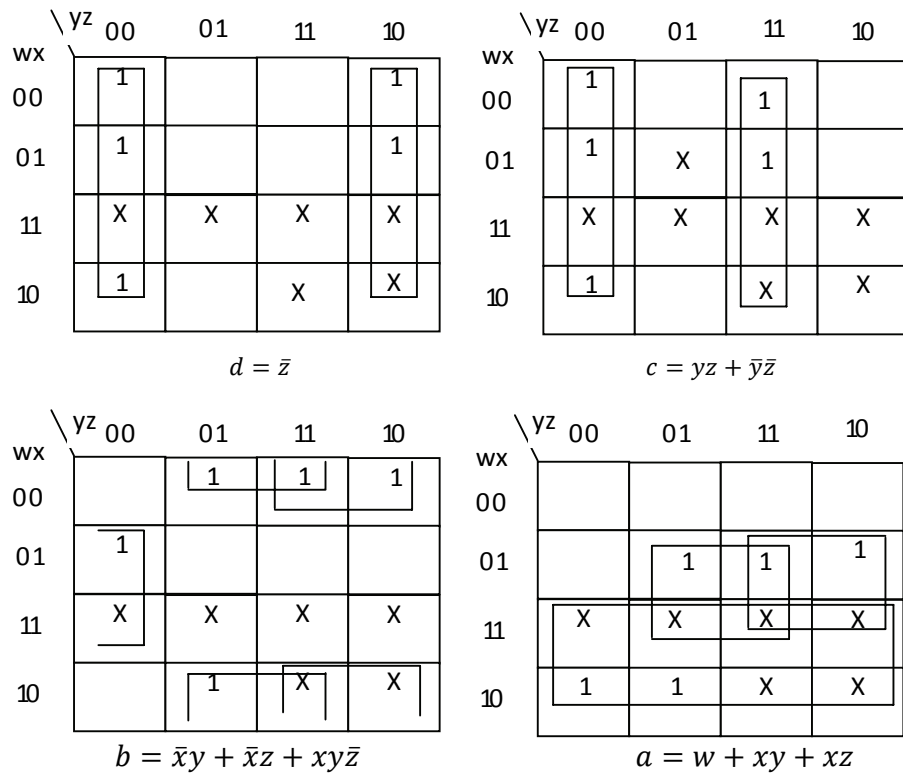
$$d = \bar{z} ,$$

$$c = yz + \overline{yz} = \overline{yz} + yz ,$$

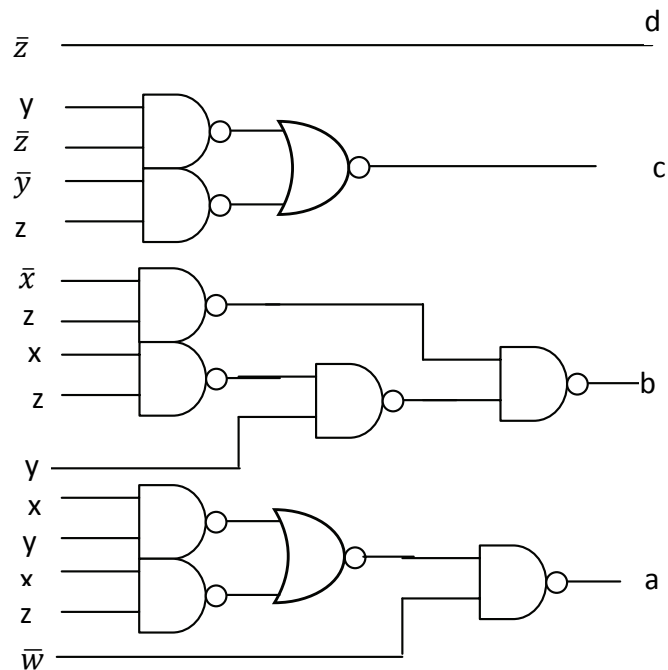
$$b = \bar{x}y + \bar{x}z + xy\bar{z} = \bar{x}z + y(\bar{x} + x\bar{z}) = \bar{x}z + y(\bar{x} + \bar{z}) = \overline{\overline{\bar{x}z} \& \overline{y\bar{x}z}} ,$$

$$a = w + xy + xz = \overline{\overline{w + xy + xz}} : \quad (3.22)$$

(3.22) արտահայտություններով կառուցված տրամաբանական սխեման ցույց է տրված նկ. 3.26-ում: Այս սխեմայի իրականացման համար անհրաժեշտ են 14 ելուստներով ցածր ինտեգրացման աստիճանի 3 ԻՍ-եր:

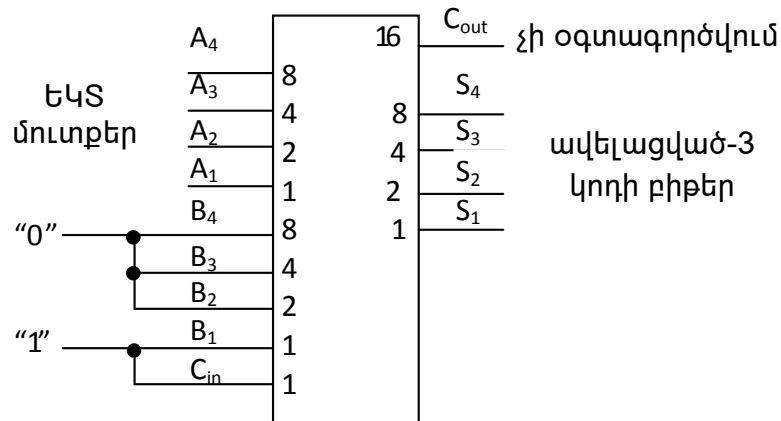


Նկ. 3.25. Երկուական-տասական կոդը ավելացված-3 կոդի ձևափոխիչի ելքային ֆունկցիաների քարտերը



Նկ. 3.26. Երկուական-տասական կոդը ավելացված-3 կոդի ձևափոխիչի սխեման

Այժմ կառուցենք երկուական-տասական կոդը ավելացված-3 կոդի ձևափոխիչի սխեման քառակարգ գուգահեռ գումարիչի օգնությամբ: Սխեմայի իրականացումը ուղղակի է (նկ. 3.27)՝ մուտքային երկուական-տասական կոդի հավաքածուներին պետք է գումարել  $3_{10}=0011_2$ : Գումարիչի մուտքերին որպես A թիվ տրվում են երկուական-տասական կոդի բիթերը, իսկ որպես B թիվ՝ 0011 բիթերը: Գումարիչի S ելքերում ստացվում են ավելացված-3 կոդի բիթերը: Այս իրականացման համար անհրաժեշտ է միայն մեկ 16 ելուստներով ԻՄ՝ քառակարգ գումարիչ:



Նկ. 3.27. ԵԿՏ կոդը ավելացված-3 կոդի ձևափոխիչ

### 3.10. Շեմային սխեմաների կառուցումը երկուական գումարիչներով

Երկուական գումարիչների կիրառությունը համակցական սարքերի կառուցման համար արդյունավետ է հատկապես այն դեպքերում, երբ կառուցվող սխեմայի ելքային ֆունկցիաները վատ են նվազարկվում դիզյունկտիվ նորմալ տեսքով: Այդպիսիք են շեմային սխեմաները: Շեմային սխեման ու մուտքերով և մեկ ելքով համակցական սարք է, որի ելքային ազդանշանը ընդունում է տրամաբանական 1 արժեք, եթե մուտքային փոփոխականների քաշավորված գումարը փոքր չէ տրված շեմից:

$$y = \begin{cases} 1, & \text{if } \sum_{i=1}^n \alpha_i x_i \geq S \\ 0, & \text{if } \sum_{i=1}^n \alpha_i x_i < S \end{cases} \quad (3.23)$$

որտեղ  $x_1, \dots, x_n$  մուտքային տրամաբանական փոփոխականներն են,  $\alpha_1, \dots, \alpha_n$  համապատասխան մուտքերի քաշային գործակիցներն են, որոնք կարող են լինել կանայական իրական թվեր: Շեմային սխեմայի կառուցման օրինակ քննարկվել է օրինակ 2.5-ում:

Մասնավոր դեպքում, երբ մուտքերի թիվը կենտ է, և բոլոր մուտքերի քաշային գործակիցները հավասար են 1-ի, իսկ շեմը՝

$$S = \frac{n+1}{2}, \quad (3.24)$$

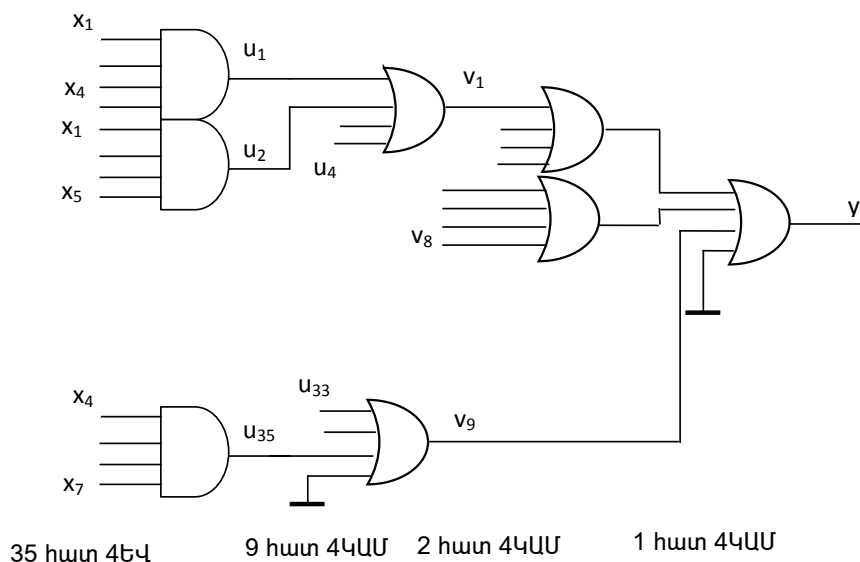
շեմային սխեման կոչվում է մեծամասնական տարր: Մեծամասնական տարրի ելքը հավասար է 1-ի, եթե մուտքային փոփոխականներից կեսից ավելին հավասար է 1-ի:

$$y = \begin{cases} 1, & \text{if } \sum_{i=1}^n x_i \geq S \\ 0, & \text{if } \sum_{i=1}^n x_i < S \end{cases} \quad (3.25)$$

Ենթադրենք անհրաժեշտ է կառուցել  $n=7$  մուտքերով մեծամասնական տարր: Այս դեպքում (3.24)-ից շենը կորոշվի՝  $S=(7+1)/2=4$ : Կանոնական սինթեզի եղանակով սխեմայի կառուցման համար անհրաժեշտ կլինի կառուցել  $2^7=128$  տողերով իսկուբյան աղյուսակ: Սակայն դրա կարիքը չկա, քանի որ ելքային ֆունկցիան կարելի է կառուցել հետևյալ դատողություններից: Սխեմայի ելքում պետք է ձևավորվի տրամաբանական 1, եթե մուտքերից գոնե 4-ը միաժամանակ հավասար լինեն 1-ի: Դիզյունկտիվ նորմալ ձևով այդ ֆունկցիան կներկայացվի 4 փոփոխականներից բաղկացած արտադրյալների գումարով:

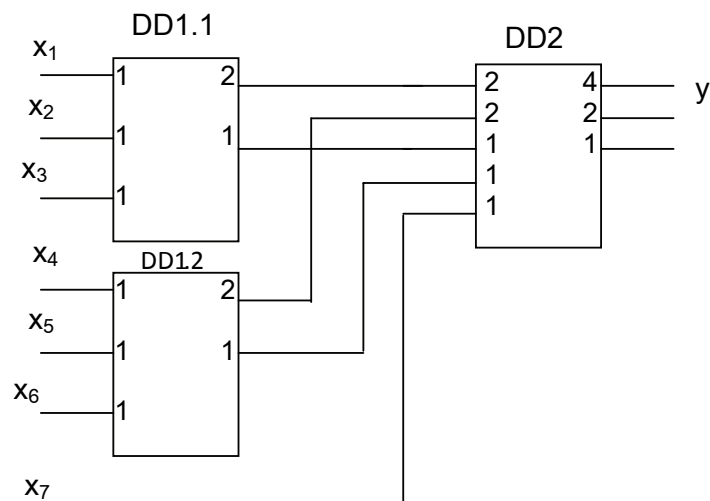
$$y = x_1x_2x_3x_4 + x_1x_2x_3x_5 + \dots + x_1x_2x_3x_7 + x_1x_2x_4x_5 + \dots + x_1x_2x_4x_7 + \dots + x_4x_5x_6x_7: (3.26)$$

Այս բանաձևում բացասված փոփոխականներ չկան, հետևաբար սոսնձվող արտադրյալներ չկան, և դա նվազագույն դիզյունկտիվ ձևն է: 4 փոփոխականներից արտադրյալների ընդհանուր թիվը կլինի՝  $C_7^4 = \frac{7!}{3!4!} = 35$ : Չետևաբար, այս մեծամասնական տարրի իրականացման համար անհրաժեշտ կլինի ունենալ 35 հատ 4ԵՎ և մեկ 35 մուտքերով ԿԱՄ տարր, կամ եթե ձևափոխվի ԵՎ-ՈՉ բազիսի, պետք է ունենալ 35 հատ 4 մուտքերով և մեկ հատ 35 մուտքերով ԵՎ-ՈՉ տարր: 35 մուտքերով տարրեր չեն արտադրվում: 35 մուտքերով ԿԱՄ տարրը կարող է իրագործվել 4 մուտքերով տարրերով, որոնց քանակը կլինի՝  $9+2+1=12$ : 7 մուտքերով մեծամասնական տարրի պարզեցված սխեման, կառուցված 4ԵՎ և 4ԿԱՄ տարրերով, ցույց է տրված նկ. 3.28-ում: Սխեմայի կառուցման համար անհրաժեշտ են 18 ԻՍ-եր, որոնցից յուրաքանչյուրը պարունակում է երկու 4ԵՎ տարրեր և 6 ԻՍ-եր, որոնցից յուրաքանչյուրը պարունակում է երկու 4ԿԱՄ տարրեր, ընդամենը 24 ԻՍ-եր:



Նկ. 3.28. 7 մուտքերով մեծամասնական տարրի սխեման կառուցված 4ԵՎ և 4ԿԱՄ տարրերի վրա

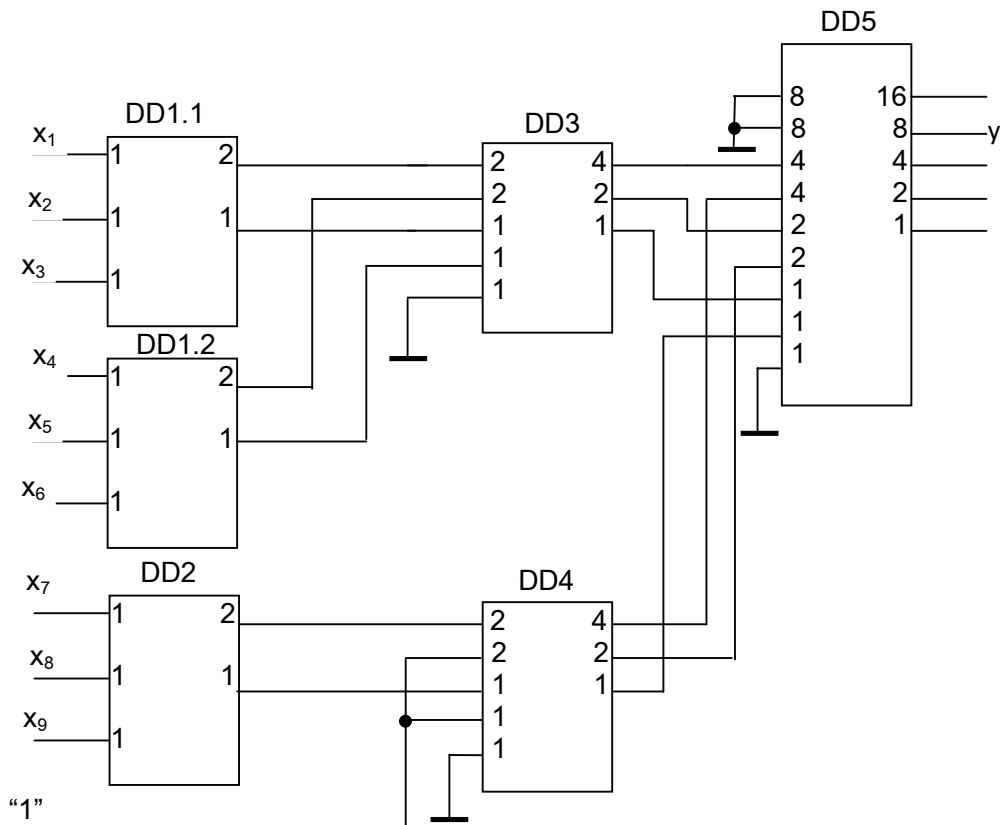
Այժմ դիտարկենք 7 մուտքերով մեծամասնական տարրի կառուցումը երկուական գումարիչներով: Դրա համար ուղղակի պետք է որոշել մուտքային փոփոխականների գումարը՝ դրանք կիրառելով գումարիչների 1 քաշով մուտքերի: Նկ. 3.29-ում ցույց է տրված մեծամասնական տարրի սխեման, կառուցված երկու միակարգ (DD1.1, DD1.2, դրանք գտնվում են մեկ ԻՍ-ի՝ DD1-ի պատյանում) և մեկ երկկարգ (DD2) գումարիչների վրա, ծախսելով ընդամենը 2 ԻՍ, նախորդ տարբերակի 24-ի փոխարեն: DD2 գումարիչի ելքերում ստացվում է մուտքային  $x_1, \dots, x_7$  փոփոխականների գումարը եռակարգ երկուական կոդով, որի ավագ բիթը հավասար է 1-ի, եթե այդ կոդի արժեքը մեծ կամ հավասար է 4-ի: Հետևաբար DD2 գումարիչի ելքի ավագ բիթը կարող է ծառայել մեծամասնական տարրի  $y$  ելք: Սակայն միշտ չէ, որ մուտքային փոփոխականների գումարի ավագ բիթը կարող է ծառայել մեծամասնական տարրի ելք: Դա տեղի ունի միայն այն դեպքում, երբ շեմի արժեքը երկուսի ամբողջ աստիճան է: Դիտարկենք ևս մեկ օրինակ:



Նկ. 3.29.  $n=7$  մուտքերով մեծամասնական տարրի սխեման կառուցված երկու ԻՍ-երի վրա՝ DD1 և DD2

**Օրինակ 3.11.** Կառուցել  $n=9$  մուտքերով մեծամասնական տարր երկուական գումարիչների միջոցով:

Շեմի արժեքը՝  $S=(9+1)/2=5$ : Որպեսզի մեծամասնական տարրի ելք ծառայի վերջին գումարիչի ելքի ավագ բիթը, պետք է շեմը արհեստականորեն բարձրացնել մինչև երկուսի մոտակա ամբողջ աստիճանը՝ 8: Դրա համար գումարիչների մուտքերին գումարվող փոփոխականներից բացի, պետք է կիրառել նաև հաստատուն 3՝  $5+3=8$ : Մեծամասնական տարրի կառուցման համար պետք է ունենալ 1 քաշով նվազագույնը 9 մուտք: Տարրի սխեման ցույց է տրված նկ. 3.30-ում: Սխեմայում կան մի շարք մուտքեր, որոնք չեն մասնակցում մուտքային փոփոխականների գումարի ձևավորմանը: Դրանցից մի մասը օգտագործվել է շեմը 8 դարձնելու համար՝ հաստատուն 3 ավելացնելով, իսկ մնացածները պետք է հողանցվեն: Հաստատուն 3 ավելացվում է DD4 գումարիչի 2 և 1 քաշերով մուտքերին տրամաբանական  $1*2+1*1=3$ :



Նկ. 3.30.  $n=9$  մուտքերով մեծամասնական տարրի սխեման՝ կառուցված երկուական գումարիչների միջոցով

### 3.11. Երկուական թվերի համեմատման սարքեր՝ թվային կոմպարատորներ

Առանց նշանի երկուական թվերի համեմատման սարքը որոշում է, թե համեմատվող թվերից մեկը մեծ է, հավասար կամ փոքր է մյուսից:

Երկու  $n$ -կարգ երկուական թվերի՝  $X=\{x_n x_{n-1} \dots x_2 x_1\}$ ,  $Y=\{y_n y_{n-1} \dots y_2 y_1\}$ , համեմատող սարքը նկարագրվում է հետևյալ ֆունկցիաներով.

$$f_n = \begin{cases} 1, & \text{if } Y > X \\ 0, & \text{if } Y \leq X \end{cases} \quad (3.27)$$

$$\varphi_n = \begin{cases} 1, & \text{if } Y = X \\ 0, & \text{if } Y \neq X \end{cases} \quad (3.28)$$

$$g_n = \begin{cases} 1, & \text{if } Y < X \\ 0, & \text{if } Y \geq X \end{cases} \quad (3.29)$$

Բերված բանաձևերից հետևում է, որ այդ երեք ֆունկցիաներն անկախ չեն՝ դրանցից ցանկացածը կարելի է արտահայտել մնացած երկուսով: Օրինակ՝

$$g_n = \overline{\varphi_n} \& \overline{f_n} \quad (3.30)$$



Ուստի թվային կոմպարատորի նախագծման համար բավարար է իրականացնել բերված ֆունկցիաներից որևէ երկուսը, օրինակ՝  $f_n$  և  $\varphi_n$  ֆունկցիաները:

Բազմակարգ դիրքային թվերի համեմատությունը կարելի է իրականացնել հաջորդաբար կարգերով: Ընդ որում, համեմատությունը կարելի է կատարել՝ սկսած բարձր կարգերից կամ ցածր կարգերից:

Բարձր կարգերից սկսելն ունի այն առավելությունը, որ եթե որևէ կարգում համեմատվող թվերի բիթերը տարբերվում են, համեմատման արդյունքը պարզ է, մնացած կարգերում համեմատումը դառնում է անհիմաստ: Օրինակ,  $X=0110\_1011$  և  $Y=0101\_1010$  թվերի համեմատումն սկսելով բարձր՝ 8-րդ կարգերից 6-րդ կարգում ստանում ենք  $y_6=0 < x_6=1$ , ուստի արդեն պարզ է, որ  $Y < X$  և կարիք չկա համեմատել մնացած ցածր կարգերը: Համեմատության ժամանակը կախված է համեմատվող թվերի արժեքներից: Վատագույն դեպքում, երբ թվերը հավասար են կամ տարբերվում են միայն ցածր կարգում, համեմատության համար կպահանջվեն  $n$  քայլեր:

Ամենացածր կարգից համեմատությունը սկսելու դեպքում արդյունքը հնարավոր չէ որոշել, քանի դեռ չեն համեմատվել բոլոր կարգերը՝ անկախ համեմատվող թվերի արժեքներից: Համեմատությունը բոլոր դեպքերում իրականացվում է  $n$  քայլերով: Համեմատության այս եղանակը միջին հաշվով երկու անգամ ավելի երկար ժամանակ կպահանջի, քան ավագ կարգից սկսելու դեպքում: Սակայն այս եղանակն ավելի տարածված է. այն ապահովում է կոմպարատորի կարգավորված կառուցվածք և կարգաթվի ընդարձակման հնարավորություն:

Նախ դիտարկենք միակարգ թվերի համեմատությունը: Համեմատության արդյունքը տրվում է  $f_1$  և  $\varphi_1$  ֆունկցիաներով, որոնց իսկության աղյուսակը ցույց է տրված աղյուսակ 3.10-ում: Աղյուսակից կարելի դուրս գրել  $f_1$  և  $\varphi_1$  ֆունկցիաների բանաձևերը՝

Աղյուսակ 3.10

$x_1$	$y_1$	$f_1$	$\varphi_1$
0	0	0	1
0	1	1	0
1	0	0	0
1	1	0	1

$$f_1 = \overline{x_1} y_1, \quad (3.31)$$

$$\varphi_1 = \overline{x_1} \overline{y_1} + x_1 y_1 = \overline{x_1 \oplus y_1} = \overline{x_1} \oplus y_1: \quad (3.32)$$

Այժմ դիտարկենք երկկարգ թվեր՝  $X=\{x_2 x_1\}$ ,  $Y=\{y_2 y_1\}$ , որոնց ցածր կարգերի համեմատումից արդեն հայտնի են  $f_1$  և  $\varphi_1$  ֆունկցիաների արժեքները: Կազմենք  $f_2$  և  $\varphi_2$  ֆունկցիաների իսկության աղյուսակները, ինչպես ցույց է տրված աղյուսակ 3.11-ում:  $f_2$  և  $\varphi_2$  ֆունկցիաների համար արգումենտներ են ծառայում  $f_1$ ,  $\varphi_1$ ,  $x_2$ ,  $y_2$ :

Աղյուսակ 3.11

#	$f_1$	$\varphi_1$	$x_2$	$y_2$	$f_2$	$\varphi_2$
0	0	0	0	0	0	0
1	0	0	0	1	1	0
2	0	0	1	0	0	0
3	0	0	1	1	0	0
4	0	1	0	0	0	1
5	0	1	0	1	1	0
6	0	1	1	0	0	0
7	0	1	1	1	0	1
8	1	0	0	0	1	0
9	1	0	0	1	1	0
10	1	0	1	0	0	0
11	1	0	1	1	1	0
12	1	1	0	0	X	X
13	1	1	0	1	X	X
14	1	1	1	0	X	X
15	1	1	1	1	X	X

Աղյուսակ 3.11-ի 12-15 տողերում  $f_2$  և  $\varphi_2$  ֆունկցիաների արժեքները որոշված չեն, քանի որ  $f_1$  և  $\varphi_1$ -ը չեն կարող միաժամանակ 1 լինել, այդ տողերում գրված մուտքային հավաքածուները արգելված հավաքածուներ են:  $f_2$ -ի սյունակում 1 արժեքը նշանակում է, որ  $Y > X$ , դա հնարավոր է, երբ  $y_2 > x_2$  կամ երբ  $y_2 = x_2$  և միաժամանակ  $f_1 = 1$ :  $\varphi_2$  -ի սյունակում 1 արժեքը նշանակում է, որ  $Y = X$ , դա հնարավոր է, երբ  $y_2 = x_2$  և միաժամանակ  $\varphi_1 = 1$ :  $f_2$  և  $\varphi_2$  ֆունկցիաների քարտերը ցույց են տրված նկ. 3.31-ում: Նկատենք, որ նկ. 3.31-ում ցույց տրված սոսնձումները չեն ապահովում  $f_2$ -ի նվազագույն բանաձև:

$f_2$		$f_1\varphi_1$			
		00	01	11	10
$x_2y_2$					
00				X	1
01		1	1	X	1
11				X	1
10				X	

$\varphi_2$		$f_1\varphi_1$			
		00	01	11	10
$x_2y_2$					
00			1	X	
01				X	
11			1	X	
10				X	

Նկ. 3.31.  $f_2$  և  $\varphi_2$  ֆունկցիաների Կառնոյի քարտերը

Սակայն այդ սոսնձումները տալիս են այնպիսի իմպլիկանտներ, որոնց միջոցով ստացվում է  $(\overline{x_2} \oplus y_2)$  արտահայտությունը, որը կրկնվում է  $f_2$  և  $\varphi_2$  ֆունկցիաների բանաձևերում: Քարտերից դուրս գրված պարզեցված բանաձևերը կլինեն՝

$$f_2 = \overline{x_2} y_2 + \overline{x_2} \overline{y_2} f_1 + x_2 y_2 f_1 = \overline{x_2} y_2 + f_1 (\overline{x_2} \overline{y_2} + x_2 y_2) = \overline{x_2} y_2 + f_1 (\overline{x_2} \oplus y_2), \quad (3.33)$$

$$\varphi_2 = \varphi_1 \overline{x_2} \overline{y_2} + \varphi_1 x_2 y_2 = \varphi_1 \overline{x_1} \oplus y_1 = \varphi_1 (\overline{x_2} \oplus y_2) : \quad (3.34)$$

Այժմ դիտարկենք եռակարգ թվեր՝  $X=\{x_3 x_2 x_1\}$ ,  $Y=\{y_3 y_2 y_1\}$ , որոնց ցածր երկու կարգերի համեմատումից արդեն հայտնի են  $f_2$  և  $\varphi_2$  ֆունկցիաների արժեքները: Եթե կազմենք  $f_3$  և  $\varphi_3$  ֆունկցիաների իսկության աղյուսակները, դրանք կունենան նույն տեսքը, ինչպես ցույց է տրված աղյուսակ 3.10-ում, միայն այն տարբերությամբ, որ  $f_2$  և  $\varphi_2$  ֆունկցիաների փոխարեն պետք է գրել  $f_3$  և  $\varphi_3$ , իսկ այդ ֆունկցիաների արգումենտները կլինեն  $f_2$ ,  $\varphi_2$ ,  $x_3$ ,  $y_3$ :  $f_3$  և  $\varphi_3$  ֆունկցիաների համար կստացվեն (3.33) և (3.34)-ին նման բանաձևեր.

$$f_3 = \overline{x_3} y_3 + f_2(\overline{x_3} \oplus y_3), \quad (3.35)$$

$$\varphi_3 = \varphi_2(\overline{x_3} \oplus y_3): \quad (3.36)$$

Ընդհանուր դեպքում,  $n$ -կարգ թվերի համեմատությունից, երբ նախորդ  $n-1$  կարգերի համեմատության արդյունքը՝  $f_{n-1}$  և  $\varphi_{n-1}$  հայտնի են, կստացվի՝

$$f_n = \overline{x_n} y_n + f_{n-1}(\overline{x_n} \oplus y_n), \quad (3.37)$$

$$\varphi_n = \varphi_{n-1}(\overline{x_n} \oplus y_n): \quad (3.38)$$

Որպեսզի ստացված (3.37) և (3.38) բանաձևերը իրավացի լինեն  $n$ -ակարգ կոմպարատորի կարգավորված կառուցվածքի բոլոր բիթերի համար, այն պետք իրավացի լինի նաև կրտսեր բիթի համար՝  $n=1$ .

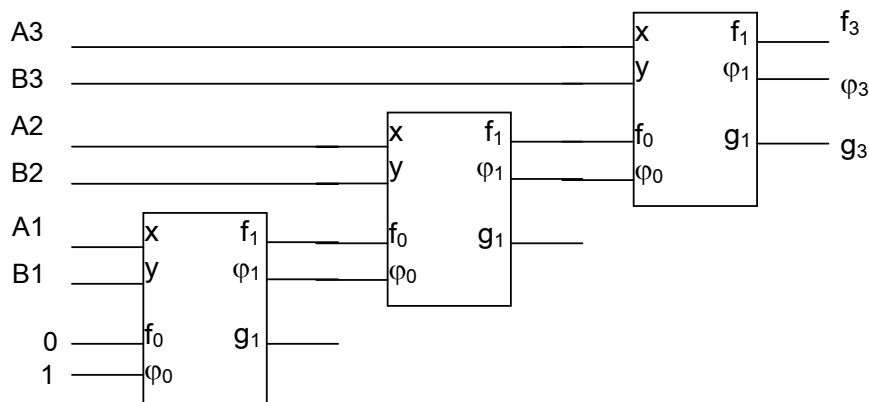
$$f_1 = \overline{x_1} y_1 + f_0(\overline{x_1} \oplus y_1), \quad (3.39)$$

$$\varphi_1 = \varphi_0(\overline{x_1} \oplus y_1): \quad (3.40)$$

Որպեսզի (3.39) և (3.40) բանաձևերը համընկնեն (3.31) և (3.32) բանաձևերի հետ, անհրաժեշտ է (3.39) և (3.40)-ում վերցնել  $f_0=0$  և  $\varphi_0=1$ :

Նկ. 3.32-ում ցույց է տրված եռակարգ թվերի համեմատման սխեման կառուցված միակարգ կոդեր համեմատող սարքերի վրա, որոնք նկարագրվում են (3.39) և (3.40) բանաձևերով:

Երկկարգ և քառակարգ կոդեր համեմատող սարքեր թողարկվում են միջին ինտեգրացման աստիճանի ԻՍ-երի տեսքով: Նկ. 3.33-ում ցույց է տրված ութակարգ կոդերի համեմատման սխեման՝ կառուցված 564ИПТ2 քառակարգ համեմատող սարքերով: Երկուական կոդեր համեմատող սխեմաները կիրառելի են նաև ԵԿՏ կոդերի համար:



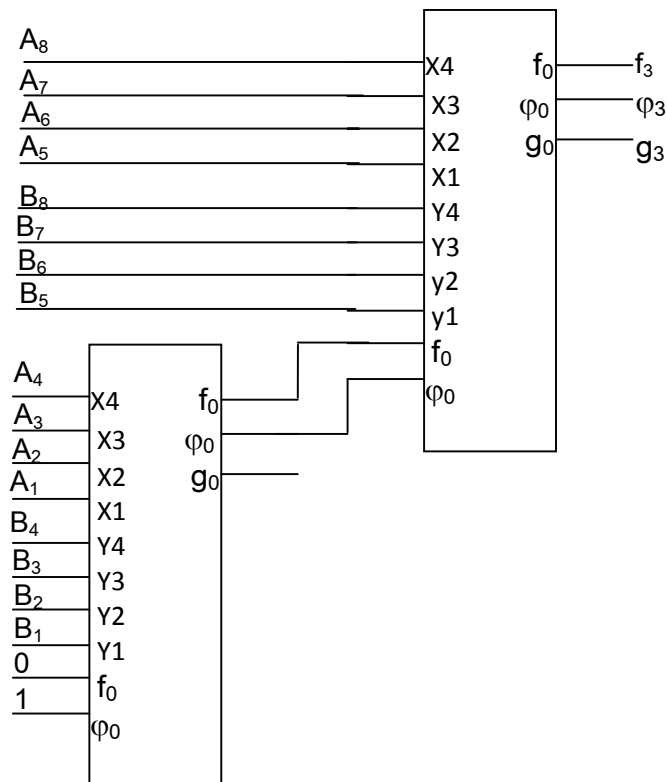
Նկ. 3.32. Եռակարգ թվերի համեմատման սխեման կառուցված միակարգ կոդեր համեմատող սարքերի վրա

Նկատենք, որ երկուական թվերի հավասարության պայմանը կարելի է ստուգել Բացառող-ԿԱՄ տարրերի միջոցով: Քանի որ երկու թվեր հավասար են, եթե հավասար են դրանց բոլոր կարգերը, ապա  $n$ -կարգ թվերի համեմատության համար պետք է օգտագործել  $n$  հատ երկու մուտքերով Բացառող-ԿԱՄ տարրեր, որոնցից յուրաքանչյուրի մուտքերին տրվում են համեմատվող թվերի նույնանուն կարգերը, ինչպես ցույց է տրված նկ. 3.34-ում:

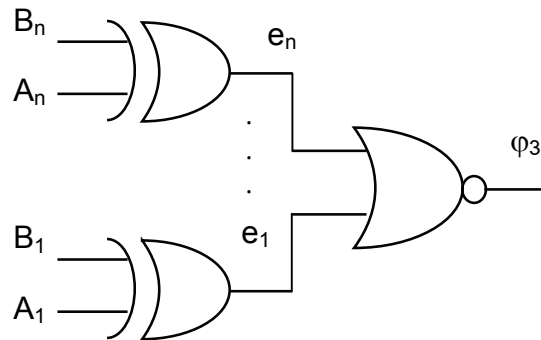
Երկու թվերի հավասարության դեպքում բոլոր Բացառող-ԿԱՄ տարրերի ելքերում կստացվի տրամաբանական 0:  $n$ -ակարգ թվերի հավասարության  $\varphi_n$  ֆունկցիան կստացվի Բացառող-ԿԱՄ տարրերի ելքերը  $n$ -մուտք ԿԱՄ-ՈՉ տարրով միավորման միջոցով.

$$\varphi_n = \overline{e_1 + e_2 + \dots + e_n}: \quad (3.41)$$

Ակնհայտ է, որ  $\varphi_n$ -ը հավասար կլինի 1-ի, եթե  $e_1=e_2=\dots=e_n=0$ :



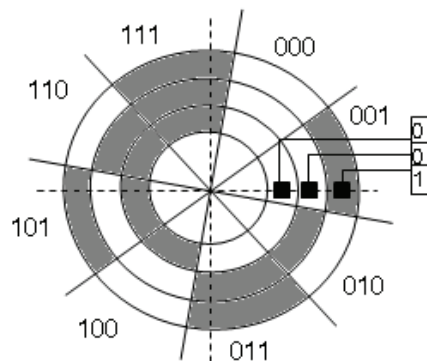
Նկ. 3.33. Ութակարգ կոդերի համեմատման սխեման՝ կառուցված 564MPT2 քառակարգ համեմատող սարքերով



Նկ. 3.34. Երկու երկուական թվերի հավասարությունը ստուգող տրամաբանական շղթա

### 3.12. Գրեյի, ջերմաչափի և “ո-ից մեկ” կոդեր

Շատ էլեկտրամեխանիկական համակարգերում անհրաժեշտ է լինում որոշել տարածության մեջ օբյեկտի դիրքը թվային ազդանշանների տեսքով: Նկ. 3.35—ում ցույց է տրված անկյունային կոորդինատի որոշման համար կիրառվող կոդավորող սարքավորման տեսքը: Կոդավորող սարքը բաղկացած է կոդավորված սեգմենտներով սկավառակից և կոնտակտների համակարգից: Սկավառակի մթնեցրած սեգմենտները միացված են տրամաբանական 1 մակարդակի լարմանը, իսկ լուսավոր սեգմենտները՝ տրամաբանական 0 մակարդակի լարմանը: Կախված սկավառակի դիրքից՝ պտտման անկյունից, կոնտակտային համակարգի միջոցով սկավառակից ստացվում է եռակարգ երկուական կոդի որևէ հավաքածու: Նշենք, որ կոդավորված սկավառակից կոդի ընթերցման համար կոնտակտային համակարգի փոխարեն օգտագործվում են նաև օպտիկական կամ էլեկտրամագնիսական համակարգեր: Օպտիկական համակարգում սկավառակի լուսավոր և մթնեցված սեգմենտներն ընդհատում կամ թափանցում են սկավառակի մեկ կողմում մեկ ուղղով տեղադրված երեք լուսադիոդների լույսային հոսքերը, որոնք ընկալվում են սկավառակի մյուս կողմում տեղադրված ֆոտոդիոդներով: Էլեկտրամագնիսական համակարգում լուսավոր և մթնեցված սեգմենտները համապատասխանում են մագնիսաթափանց և մագնիսաանթափանց նյութերից արված սեգմենտներին: Կոդի ընթերցման համար օգտագործվում են մեկ ուղղով տեղադրված երեք մագնիսական գլխիկներ:



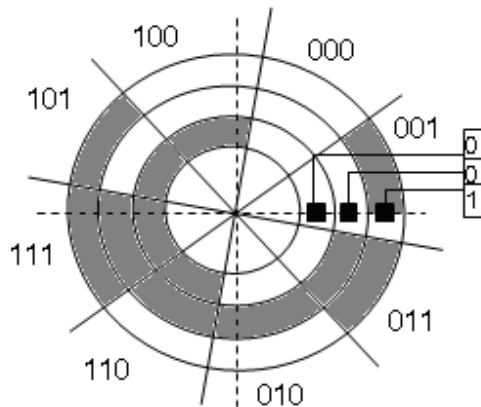
Նկ. 3.35. Եռակարգ երկուական կոդ ձևավորող կոդավորված սկավառակ

Նկ. 3.35-ում պատկերված կոդավորող սարքում, եթե ազդանշանների ընթերցման կոնտակտները գտնվում են երկու հարևան սեկտորների մոտ, հնարավոր է ունենալ ընթերցման մեծ սխալ: Օրինակ, եթե կոնտակտները գտնվում են 001 և 010 սեկտորների սահմանագծում, ցածր կարգի երկու սեգմենտներից յուրաքանչյուրից կարող է ընթերցվել 0 կամ 1: Հետևաբար, հնարավոր է, որ ընթերցվի 001, 010, 000 կամ 011: Սխալի առավելագույն արժեքը կլինի 3: Վատագույն դեպքը կլինի, երբ կոնտակտների դիմաց գտնվի 000-111 սահմանագիծը՝ կարող է ընթերցվել ցանկացած եռաբիթ կոդ: Սխալի առավելագույն արժեքը կլինի 7: Կոդավորող սկավառակից ընթերցման սխալը կարելի է հասցնել նվազագույնի՝ 1 միավորի, եթե սկավառակը կոդավորվի այնպիսի կոդով, որի ցանկացած երկու հաջորդական համակցություններ տարբերվեն միայն մեկ բիթով: Այդպիսի կոդը կոչվում է Գրեյի կամ ցիկլային կոդ: Աղյուսակ 3.12-ում բերված է եռաբիթ Գրեյի կոդը, իսկ նկ. 3.36-ում՝ այդ կոդով կոդավորված սկավառակը: Այդ սկավառակում յուրաքանչյուր երկու սեկտորների սահմանագծում միայն մեկ բիթ է փոխվում: Եթե կոնտակտային համակարգի դիմաց գտնվի երկու սեկտորների սահմանագիծը, կընթերցվի սահմանագծի երկու կողմերում գտնվող սեկտորներից մեկի կոդը, որոնք իրարից տարբերվում են միայն մեկ միավորով:

Գրեյի կոդը հարմար է օգտագործել նաև թվային ավտոմատներում վիճակների կոդավորման համար: Թվային շղթաներում էներգիայի սպառումը համեմատական է շղթայում հանգույցների փոխանջատումների թվին: Իմպուլսների թվի հաշվման սխեմաներում սպառվող էներգիայի նվազեցման տեսանկյունից հարմար է վիճակները կոդավորել Գրեյի կոդով: Եթե հաշվիչի հաջորդական վիճակները կոդավորվեն Գրեյի կոդով, հաշվի յուրաքանչյուր տակտում կփոխանջատվի միայն մեկ տրիգեր:

Աղյուսակ 3.12

Տասական թիվը	Երկուական կոդը	Գրեյի կոդը
	$b_2b_1b_0$	$g_2g_1g_0$
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100



Նկ. 3.36. Գրեյի եռաբիթ կոդով կոդավորված սկավառակ

Գրեյի կոդը կարելի է կառուցել հետևյալ ռեկուրսիվ եղանակով.

- 1) միաբիթ Գրեյի կոդն ունի երկու համակցություն՝ 0 և 1:
- 2)  $n+1$ -բիթ Գրեյի կոդի առաջին  $2^n$  համակցությունները  $n$ -բիթ Գրեյի կոդի համակցություններն են՝ ձախից ավելացված 0-ով:
- 3)  $n+1$ -բիթ Գրեյի կոդի վերջին  $2^n$  համակցությունները  $n$ -բիթ Գրեյի կոդի համակցություններն են՝ գրված հակադարձ հաջորդականությամբ և ձախից ավելացված 1-ով:

Եթե աղյուսակ 3.12.-ի 3 և 4 տողերի միջև տարվի սահմանագիծ, ապա կարելի է տեսնել, որ եռաբիթ Գրեյի կոդի համար իրավացի են վերը նշված 2 և 3 հատկությունները: Նկարագրված եղանակով  $n$ -բիթ Գրեյի կոդ կառուցելու համար անհրաժեշտ է կառուցել  $n$ -ից փոքր բիթերով բոլոր կոդերը:

Գրեյի կոդի կառուցման համար կարելի է սկզբում կառուցել երկուական կոդը, այնուհետև օգտվել երկուական կոդը Գրեյի կոդի ձևափոխման՝ ստորև բերված ալգորիթմից.

- 1)  $n$ -բիթ երկուական և Գրեյի կոդի ամենաձախ բիթին տրվում է  $n-1$ , իսկ ամենաաջին՝ 0 ինդեքս: Գրեյի կոդի ավագ բիթը հավասար է երկուական կոդի ավագ բիթին:
- 2) Գրեյի կոդի  $i$ -րդ բիթը հավասար է 0-ի, եթե երկուական կոդի  $i+1$  և  $i$ -րդ բիթերը հավասար են, հակառակ դեպքում հավասար է 1-ի:

Նկատի ունենալով, որ բիթերի հավասարությունը կարելի է ստուգել մոդուլ 2-ով գումարման միջոցով, այս ալգորիթմը կարելի է ներկայացնել հետևյալ բանաձևով.

$$g_i = \begin{cases} b_i \oplus b_{i+1}, & \text{եթե } i = 0, \dots, n-2; \\ b_{n-1}, & \text{եթե } i = n-1 \end{cases} \quad (3.42)$$

Նկ. 3.37ա-ում ցույց է տրված քառակարգ երկուական կոդը Գրեյի կոդի ձևափոխելի սխեման՝ կառուցված Բացառող-ԿԱՄ տարրերի վրա:

Գրեյի կոդից երկուական կոդին կարելի է անցնել հետևյալ ալգորիթմով. երկուական կոդի  $n-1$ -րդ բիթը հավասար Գրեյի կոդի  $n-1$ -րդ բիթին, երկուական կոդի  $i$ -րդ բիթը հավասար է 0-ի, եթե երկուական կոդի  $i+1$ -րդ բիթը հավասար է Գրեյի կոդի  $i$ -րդ բիթին,

հակառակ դեպքում հավասար է 1-ի: Այս ալգորիթմը կարելի է ներկայացնել հետևյալ բանաձևով.

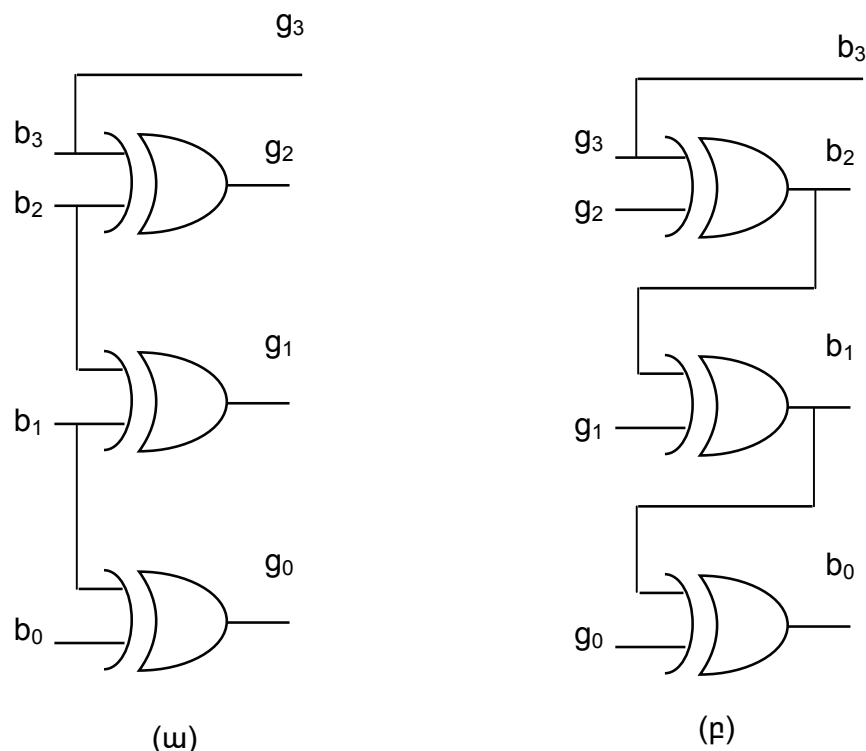
$$b_i = \begin{cases} g_i \oplus b_{i+1}, & \text{եթե } i = 0, \dots, n-2; \\ g_{n-1}, & \text{եթե } i = n-1 \end{cases} \quad (3.43)$$

Նկ. 3.37բ-ում ցույց է տրված քառակարգ Գրեյի կոդը երկուական կոդի ձևափոխիչի սխեման՝ կառուցված Բացառող-ԿԱՄ տարրերի վրա:

Կարելի է նկատել, որ (3.43) բանաձևը կարելի է ներկայացնել նաև հետևյալ տեսքով.

$$b_i = \begin{cases} \sum_{k=i}^{n-1} g_k, & \text{եթե } i = 0, \dots, n-2; \\ g_{n-1}, & \text{եթե } i = n-1 \end{cases} \quad (3.44)$$

Այստեղ գումարը մոդուլ 2-ով է:



Նկ. 3.37 Քառաբիթ երկուական կոդը Գրեյի կոդի (ա) և Գրեյի կոդը երկուականի (բ) ձևափոխիչների սխեմաները

Թվային շղթաներում, ինչպես նաև անալոգային ազդանշանները թվային և թվային անալոգային կերպափոխիչներում, մեծ կիրառություն է գտել ջերմաչափի կոդը: Ուրիշ ջերմաչափի կոդով կոդավորված  $i$ -րդ համակցության  $i$  ցածր բիթերը հավասար են 1-ի, իսկ  $n-i$  ավագ բիթերը՝ 0-ի: Յոթ բիթ ջերմաչափի կոդի համակցությունները ցույց են տրված աղյուսակ 3.13-ում:

Թվային համակարգերում հաճախ օգտագործվող մեկ այլ կոդ է “ո-ից մեկ” (one-hot) կոդը, որի միայն մեկ բիթն է հավասար 1-ի, մնացած բոլոր բիթերը 0 են: Այս



կողը, մասնավորապես, լայն կիրառություն ունի թվային ավտոմատների վիճակների կոդավորման համար: “n-ից մեկ” ութ կարգ կոդի համակցությունները ցույց են տրված աղյուսակ 3.13.-ում: Հաճախ օգտագործվում է նաև “n-ից մեկ” կոդի շրջված` ինվերս տարբերակը, որի միայն մեկ բիթն է 0, իսկ մնացածը` 1:

Աղյուսակ 3.13

Թիվը	Ջերմաչափի կոդ	“n-ից մեկ” կոդ	“n-ից մեկ” ինվերս կոդ
0	0000000	00000001	11111110
1	0000001	00000010	11111101
2	0000011	00000100	11111011
3	0000111	00001000	11110111
4	0001111	00010000	11101111
5	0011111	00100000	11011111
6	0111111	01000000	10111111
7	1111111	10000000	01111111

Թվային համակարգերում օգտագործվում են նաև “n-ից m” կոդեր, որնցում n-աբիթ կոդային համակացություններում m բիթերն ունեն 1 արժեք, (n-m)-ը` 0 արժեք: Այդպիսի կոդի համակացությունների ընդհանուր թիվը կլինի  $\binom{n}{m} = \frac{n!}{m!(n-m)!}$ : Օրինակ, “4-ից 2” կոդը կունենա 6 համակցություն, “10-ից 3” կոդը` 120 համակցություն:

“n-ից m” կոդի մասնավոր դեպքն է 8B10B կոդը, որն օգտագործվում է այն համակարգերում, որտեղ տվյալների փոխանցումն իրագործվում է գիգաբիթ Ethernet ստանդարտով: Այդպիսի համակարգերում սկզբնական տվյալները ներկայացվում են 8-բիթ կոդի 256 կոդային համակցություններով: Փոխանցման համար տվյալները վերակոդավորվում են հիմնականում “10-ից 5” կոդով: “10-ից 5” կոդային համակցությունների կարևոր առանձնահատկությունն այն է, որ դրանք հավասարակշռված են` պարունակում են հավասար թվով 1-եր և 0-ներ, ինչը կարևոր է երկար տարածությունների վրա տվյալների փոխանցման համար (ավելի մանրամասն այդ մասին տես 5-րդ բաժնում):

### 3.13.Աղմկապաշտպանված կոդավորում

Հեռահաղորդման համակարգերում, ինչպես նաև թվային համակարգի հանգույցների միջև, տվյալների փոխանակման ժամանակ հնարավոր են տվյալների աղավաղումներ: Նման աղավաղումների բացասական հետևանքները կանխելու համար օգտագործվում են աղմկապաշտպանված կոդեր, որոնք դասակարգվում են սխալը հայտնաբերող կոդերի և սխալը ճշգրտող կոդերի:

Ցանկացած կարգայնությամբ կոդի մեկ կարգում սխալի հայտնաբերման համար կարելի է օգտագործել “զույգ-կենտ” կոդերը, որոնք ունեն լայն գործնական կիրառություն: Զույգ կոդի հավաքածուներում 1-բիթերի թիվը զույգ է, կենտ կոդում` կենտ: “Զույգ-կենտ” կոդավորման էությունը հետևյալն է. n-աբիթ կոդային հավաքածուին ավելացվում ևս մեկ` ստուգող բիթ, որը հավաքածուին օժտում է պահանջվող

հատկությամբ: Աղյուսակ 3.14-ում բերված են 4-աբիթ (ներառյալ ստուգող բիթը) զույգ և կենտ կողերը: Ակնհայտ է, որ աղմկակայունությունը ձեռք է բերվում կողում ավելացություն մտցնելով՝ 8 հավաքածու կողավորելու համար օգտագործվում է երեքի փոխարեն 4 բիթ:

Տվյալները հաղորդելիս սկզբնական կոդային հավաքածուին ավելացվում է ստուգող բիթը հետևյալ հավասարումներով՝ զույգ կողի դեպքում.

$$b_0 = \sum_{k=1}^n b_k, \quad (3.45)$$

կենտ կողի դեպքում՝

$$b_0 = 1 + \sum_{k=1}^n b_k : \quad (3.46)$$

Գումարումները կատարվում են մոդուլ երկուսով:

Աղյուսակ 3.14

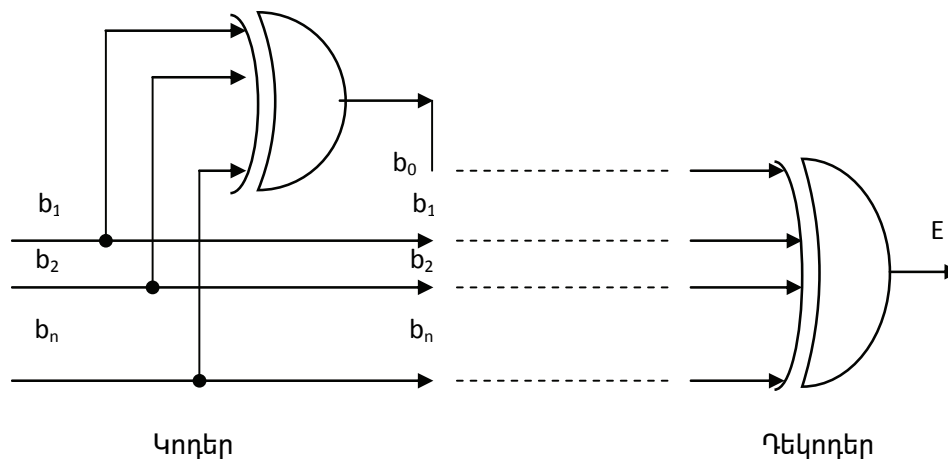
Զույգ կող			Կենտ կող		
Հավաքածուն	սկզբնական բիթերը	ստուգող բիթը	Հավաքածուն	սկզբնական բիթերը	ստուգող բիթը
0	000	0	0	000	1
1	001	1	1	001	1
2	010	1	2	010	0
3	011	0	3	011	1
4	100	1	4	100	0
5	101	0	5	101	0
6	110	0	6	110	1
7	111	1	7	111	0

Ընդունող կողում ստուգվում է կողի զույգ կամ կենտ հայտանիշը հետևյալ հավասարումով.

$$E = \sum_{k=0}^n b_k : \quad (3.47)$$

Ընդունված զույգ կողը համարվում է աղավաղված, եթե  $E=1$ : Ընդունված կենտ կողը համարվում է աղավաղված, եթե  $E=0$ :

Նկ. 3.38-ում բերված է զույգ կողով կոդերի կառուցվածքային սխեման:



Նկ. 3.38. Չույգ կողով կողերի և դեկոդերի կառուցվածքային սխեման

Նկատի ունենալով զույգ կողերի կիրառության տարածվածությունը՝ ԻՍ-եր արտադրողների կողմից թողարկվում են բազմակարգ կողերի զույգության ստուգման միկրոսխեմաներ: Օրինակ, 561 (4000) շարքի K561ՊС միկրոսխեման:

Մի քանի կարգերում միաժամանակ աղավաղումները հայտնաբերելու համար պետք է մտցնել ավելի մեծ թվով լրացուցիչ՝ ստուգող բիթեր:

Սխալի հայտնաբերման և շտկման համար կապահանջվեն ավելի մեծ թվով ստուգող բիթեր: Հետևաբար, սխալը շտկող կողերը օժտված են ավելի մեծ ավելցուկությամբ:

Որպես օրինակ դիտարկենք մեկ կարգում սխալը շտկող Հենմինգի 7-ակարգ կողը, որում 4 բիթերն օգտագործվում են տվյալների կոդավորման համար, իսկ 3 բիթերը՝ ստուգման համար: Սխալն ուղղելու համար պետք է գտնել աղավաղված բիթի համարը և ինվերսել այդ բիթը: Աղավաղված բիթի համարի երկուսական կոդը կոչվում է սխալի հայտնիչ: Աղյուսակ 3.15-ում բերված 7-ակարգ կողի համար սխալի հայտնիչների աղյուսակը: Սխալի բացակայության դեպքում սխալի հայտնիչը պետք է ունենա 000 արժեքը:

Աղյուսակ 3.15

Աղավաղված բիթ	Սխալի հայտնիչ	Աղավաղված բիթ	Սխալի հայտնիչ
1	001	5	101
2	010	6	110
3	011	7	111
4	100		

Ներկայացնենք աղմկապաշտպանված կոդային հավաքածուն  $\{a_7 a_6 a_5 a_4 a_3 a_2 a_1\}$  բիթերով, որտեղ դեռ որոշված չեն, թե որոնք են ստուգող բիթերը: Հենմինգի կոդում հայտնիչի բիթերը հաշվվում են ստուգող հավասարումներից, որոնք իրականացնում են մոդուլ երկուսով գումարումների կողի որոշակի բիթերի հետ: Ստուգող բիթերի արժեքները պետք է որոշվեն այնպես, որ սխալի բացակայության դեպքում հայտնիչի

բոլոր բիթերում ստացվեն 0-ներ: Աղյուսակ 3.15-ից հետևում է, որ եթե հայտնիչի ցածր կարգում ստացվել է 1, ապա աղավաղումը տեղի է ունեցել կոդի 1, 3, 5 կամ 7-րդ կարգում: Հետևաբար առաջին ստուգող հավասարումը կարելի է գրել հետևյալ տեսքով՝  $a_1 \oplus a_3 \oplus a_5 \oplus a_7 = 0$ , ինչը համապատասխանում է այդ կարգերում եզակի սխալի բացակայությանը:

Նույն կերպ, հայտնիչի երկրորդ կարգում կստացվի 1, եթե սխալը տեղի է ունեցել 2, 3, 6, 7 բիթերում, համապատասխան ստուգող հավասարումը կլինի՝  $a_2 \oplus a_3 \oplus a_6 \oplus a_7 = 0$ : Հայտնիչի երրորդ կարգում կստացվի 1, եթե սխալը տեղի է ունեցել 4, 5, 6, 7 բիթերում. երրորդ հավասարումը կլինի՝  $a_4 \oplus a_5 \oplus a_6 \oplus a_7 = 0$ :

Հետևաբար ստուգող հավասարումների համակարգը կլինի.

$$\begin{aligned} a_1 \oplus a_3 \oplus a_5 \oplus a_7 &= 0, \\ a_2 \oplus a_3 \oplus a_6 \oplus a_7 &= 0, \\ a_4 \oplus a_5 \oplus a_6 \oplus a_7 &= 0: \end{aligned} \quad (3.48)$$

Ընդունող կողմում պետք ստուգել այս հավասարումները, եթե դրանցից որևէ մեկի արդյունքը զրոյից տարբեր է, ապա ընդունված կոդը աղավաղված է: Ակնհայտ է, որ հաղորդող կողմում աղմկապաշտպանված կոդավորման ժամանակ պետք է պահպանվեն (3.48) հավասարումները: Ստուգող բիթերը կարելի է ընտրել այնպես, որ դրանք (3.48) հավասարումներից միարժեքորեն արտահայտվեն սկզբնական կոդի բիթերով: (3.48) հավասարումներից կարելի է նկատել, որ որպես ստուգող բիթեր կարելի ընտրել  $a_1$ ,  $a_2$ ,  $a_4$ -ը, որոնց արժեքները կոդավորման ժամանակ կորոշվեն հետևյալ կերպ.

$$\begin{aligned} a_1 &= a_3 \oplus a_5 \oplus a_7, \\ a_2 &= a_3 \oplus a_6 \oplus a_7, \\ a_4 &= a_5 \oplus a_6 \oplus a_7: \end{aligned} \quad (3.49)$$

Այս հավասարումները կանվանենք կոդավորման հավասարումներ:

Դիտարկենք օրինակ, սկզբնական 4-աբիթ կոդային հավաքածուն՝  $\{0\ 1\ 0\ 1\}$ , լրացուցիչ 3 ստուգող բիթերը կորոշվեն (3.49)-ից՝

$$\begin{aligned} a_1 &= 0 \oplus 1 \oplus 1 = 0, \\ a_2 &= 0 \oplus 0 \oplus 1 = 1, \\ a_4 &= 1 \oplus 0 \oplus 1 = 0: \end{aligned}$$

Ավելացնելով այդ բիթերը՝ աղմկապաշտպանված կոդը կլինի՝  $A_{ec} = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\} = \{0\ 1\ 0\ 0\ 1\ 0\ 1\}$ : Ընդունող կողմում այդ հավաքածուն ստուգվում է (3.48) հավասարումներով՝

$$\begin{aligned} 0 \oplus 0 \oplus 1 \oplus 1 &= 0, \\ 1 \oplus 0 \oplus 0 \oplus 1 &= 0, \\ 0 \oplus 1 \oplus 0 \oplus 1 &= 0: \end{aligned}$$

Այսպիսով, սխալի հայտնիչը 000 է, հետևաբար աղաղվաղում չկա՝ ընդունվել է անսխալ կոդը: Սկզբնական հաղորդագությունը վերականգնելու համար ընդունված կոդից պետք է հեռացվեն  $a_1, a_2, a_3$  բիթերը:

Այժմ դիտարկենք դեպք, երբ կապի գծում տեղի է ունեցել հաղորդվող կոդի աղաղվաղում 5-րդ բիթում, այսինքն՝ ընդունվել է  $\{0\ 1\ 0\ 0\ 0\ 01\}$  հավաքածուն: (3.48) հավասարումներով ստուգումից կստացվի՝

$$0 \oplus 0 \oplus 0 \oplus 1 = 1,$$

$$1 \oplus 0 \oplus 0 \oplus 1 = 0,$$

$$0 \oplus 0 \oplus 0 \oplus 1 = 1:$$

Սխալի հայտնիչը կլինի 101, որը ցույց է տալիս, որ աղաղվաղումը տեղի է ունեցել 5-րդ բիթում: Սխալն ուղղելու համար պետք է ինվերսել ընդունված կոդի 5-րդ բիթը: Մեկ այլ օրինակ, ենթադրենք՝ հաղորդվող կոդում աղաղվաղվել է 6-րդ բիթը, և ընդունվել է  $\{0\ 1\ 0\ 0\ 1\ 1\ 1\}$  հավաքածուն: (3.47) հավասարումներով ստուգումից կստացվի՝

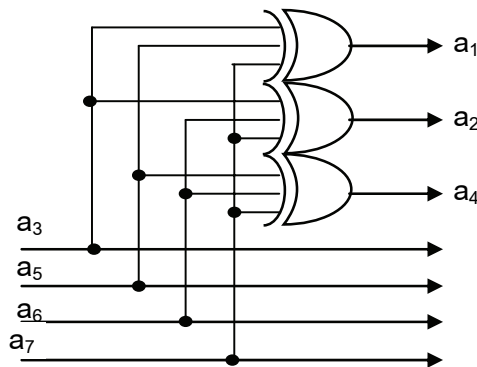
$$0 \oplus 0 \oplus 1 \oplus 1 = 0,$$

$$1 \oplus 0 \oplus 1 \oplus 1 = 1,$$

$$0 \oplus 1 \oplus 1 \oplus 1 = 1:$$

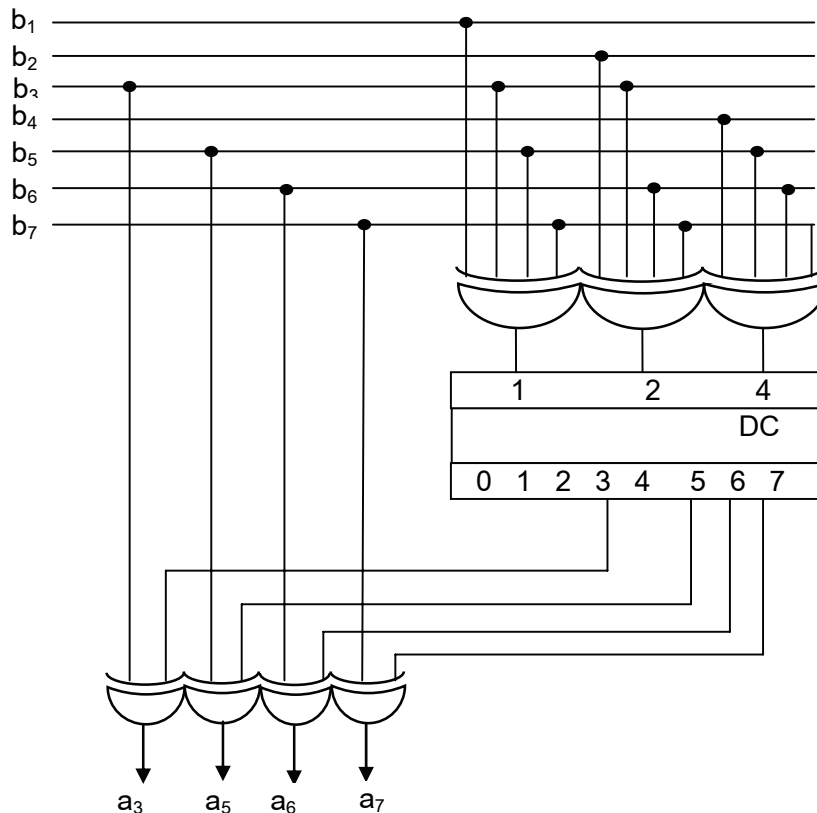
Սխալի հայտնիչը կլինի 110, որը ցույց է տալիս, որ աղաղվաղումը տեղի է ունեցել 6-րդ բիթում:

Հեմինգի (7,4) աղմկակայուն կոդով կոդավորիչն ու վերծանիչը կարելի է կառուցել (3.49), (3.48) հավասարումների հիման վրա: Նկ. 3.39-ում ցույց է տրված կոդավորիչը, իսկ նկ. 3.40-ում՝ վերծանիչը:



Նկ. 3.39. Հեմինգի (7,4) կոդի կոդավորիչ

Նկ.3.40-ում (3.48) հավասարումների ստուգումն իրականացվում է 4-ամուտք Բաքառող-ԿԱՄ տարրերով, որոնց ելքերում ձևավորվում է սխալի հայտնիչը: Սխալի հայտնիչը վերծանվում է վերծանիչով, որի որևէ ելքում 1-ի առկայությունը վկայում է այդ բիթի աղաղված մասին: Աղաղված բիթը վերականգնելու համար այն պետք է ինվերսել, ինչը արվում է 3-ամուտք Բաքառող-ԿԱՄ տարրերով: Նկատենք, որ անհրաժեշտ է վերականգնել միայն տվյալների բիթերը՝  $a_3, a_5, a_6, a_7$ :



Նկ. 3.40. Հենինգի (7,4) կոդի վերծանիչ

### 3.14. Խնդիրներ

**Խնդիր 3.1.** Կատարել անցում մեկ համակարգից մյուսին.

ա)  $1101010_2 = ?_{16}$ , բ)  $10110110_2 = ?_{16}$ , գ)  $1101010_2 = ?_8$ , դ)  $10110110_2 = ?_8$ ,

ե)  $21547_8 = ?_2$ , զ)  $152735_8 = ?_2$ , է)  $7f19_{16} = ?_2$ , ը)  $1111_{16} = ?_2$ ,

թ)  $37517_8 = ?_{16}$ , ժ)  $17436_8 = ?_{16}$ , ի)  $abcd_{16} = ?_8$ , լ)  $3f7d_{16} = ?_8$ :

**Խնդիր 3.2.** Հետևյալ տասական թվերը ձևափոխել մեկ այլ համակարգի.

ա)  $127 = ?_2$ , բ)  $3759 = ?_2$ , գ)  $586 = ?_8$ , դ)  $3789 = ?_8$ , է)  $4568 = ?_{16}$ , զ)  $66852 = ?_{16}$ :

**Խնդիր 3.3.** Հետևյալ թվերը տրված համակարգից ձևափոխել տասականի.

ա)  $1101010_2$ , բ)  $10110110_2$ , գ)  $21547_8$ , դ)  $152735_8$ , է)  $7f19_{16}$ , զ)  $3f7d_{16}$ :

**Խնդիր 3.4.** Հետևյալ տասական թվերը ձևափոխել ութականի և երկուականի.

ա)  $0.5765 = ?_8$ , բ)  $12.307 = ?_8$ , գ)  $0.5765 = ?_2$ , դ)  $12.307 = ?_2$ :

**Խնդիր 3.5.** Հետևյալ թվերը ներկայացնել երկուական-կոդավորված-տասական չափաձևով.

ա)  $127_{10}$ , բ)  $3759_{10}$ , գ)  $576_8$ , դ)  $3766_8$ , է)  $f68_{16}$ , զ)  $8a2_{16}$ , է)  $1101010_2$ , թ)  $10110110_2$ :

**Խնդիր 3.6.** Գումարել առանց նշանի երկուական թվերը՝ ցույց տալով բոլոր փոխանցումները.

ա)  $110101 + 10111$ , բ)  $10110111 + 1001011$ , գ)  $11101011 + 101$ :

**Խնդիր 3.7.** Հանել առանց նշանի երկուական թվերը՝ ցույց տալով բոլոր փոխառությունները.

ա)  $110101+10111$ , բ)  $10110111+1001011$ , գ)  $11101011+101$ :

**Խնդիր 3.8.** Հետևյալ նշանով տասական թվերը ներկայացնել երկուական նշան-արժեք չափաձևով.

ա) -154, բ) +537, գ) -37, դ) -652, ե) +412:

**Խնդիր 3.9.** Որոշել նշան-արժեք չափաձևով հետևյալ երկուական թվերի տասական համարժեքները.

ա) 011010101, բ) 111101101, գ) 100001101, դ) 000101010:

**Խնդիր 3.10.** Որոշել հետևյալ տասական թվերի 10-ի լրացումները.

ա) 57, բ) 678, գ) 3956, դ) 45.37, ե) 0.7675:

**Խնդիր 3.11.** Որոշել հետևյալ տասական թվերի 9-ի լրացումները.

ա) 57, բ) 678, գ) 3956, դ) 45.37, ե) 0.7675:

**Խնդիր 3.12.** Որոշել հետևյալ երկուական թվերի 1-ի լրացումները.

ա) 11010101, բ) 11101101, գ) 10001101, դ) 00101010:

**Խնդիր 3.13.** Որոշել հետևյալ երկուական թվերի 2-ի լրացումները.

ա) 11010101, բ) 11101101, գ) 10001101, դ) 00101010:

**Խնդիր 3.14.** Գումարել հետևյալ նշանով թվերը երկուական 8-բիթ լրացուցիչ կոդի (2-ի լրացում) չափաձևով, ստուգել գերլցման առկայությունը.

ա)  $57+34$ , բ)  $67+(-18)$ , գ)  $39+(-56)$ , դ)  $-45+(-87)$ , ե)  $76+75$ :

**Խնդիր 3.15.** Պարզել, թե հաշվանքի որ համակարգերում է ճիշտ հետևյալ արտահայտություններից յուրաքանչյուրը.

ա)  $1234+5432=6666$ , բ)  $33/3=11$ , գ)  $302/20=12.1$ , դ)  $41/3=13$ ,

ե)  $23+44+14+32=223$ , զ)  $\sqrt{41}=5$

**Խնդիր 3.16.** Կառուցել կիսագումրիչի շղթա՝ օգտագործելով միայն NAND2 տարրեր:

**Խնդիր 3.17.** Կառուցել լրիվ գումրիչի շղթա՝ օգտագործելով միայն NAND2 տարրեր:

**Խնդիր 3.18.** Մուլտիպլեքսորների օգնությամբ կառուցել լրիվ գումարիչի սխեմա:

**Խնդիր 3.19.** Քառաբիթ գումրիչի հիման վրա կառուցել քառաբիթ գումրիչ/հանիչ, որն ունի գործողության ընտրության  $z$  մուտք՝  $z=0$  ընտրում է գումարման, իսկ  $z=1$  հանման գործողությունը:

**Խնդիր 3.20.** Կառուցել 1-բիթ լրիվ գումարիչ ԵՎ, ԿԱՄ և Բացառող-ԿԱՄ տարրերով:

ա) Որոշել այդ գումարիչի առավելագույն հապաղումը, եթե  $t_{\text{and}}=t_{\text{or}}=1$  նվ,  $t_{\text{xor}}=2$  նվ:

բ) Հաշվել այդպիսի գումրիչի հիման վրա կառուցված քառաբիթ գումարիչի առավելագույն հապաղումը:

**Խնդիր 3.21.** Կառուցել համակցական սխեմա, որը գեներացնում է ԵԿՏ թվանշանի 9-ի լրացումը:

**Խնդիր 3.22.** Կառուցել համակցական սխեմա, որը գեներացնում է ԵԿՏ թվանշանի 10-ի լրացումը:

**Խնդիր 3.23.** Կառուցել թվաբանական սարք, որն ունի ԵԿՏ թվանշանների տրման  $a$  և  $b$  քառաբիթ մուտքեր և գործողության ընտրության  $z_1$  և  $z_0$  մուտքեր՝  $z_1z_0=00$  –  $a$ -ին գումարել  $b$ ,  $z_1z_0=01$  –  $a$ -ին գումարել  $b$ -ի 9-ի լրացումը,  $z_1z_0=10$  –  $a$ -ին գումարել  $b$ -ի 10-ի լրացումը,  $z_1z_0=11$  –  $a$ -ին գումարել 1: Կարելի է օգտագործել նախորդ խնդիրների 9-ի և 10-ի լրացումները գեներացնող սխեմաները:

**Խնդիր 3.24.**  $32 \times 6$  ՀՀՄ-ի միջոցով իրականացնել 6-բիթ երկուական կոդը երկկարգ ԵԿՏ կոդի ձևափոխիչ: Ցույց տալ ծրագրավորման աղյուսակը:

**Խնդիր 3.25.** Ինչպիսի՞ նվազագույն ծավալ պետք է ունենա ՀՀՄ-ն, որպեսզի հնարավոր լինի իրականացնել՝ ա) երկու քառաբիթ երկուական թվերի բազմապատկում, բ) 5-աբիթ երկուական թվի քառակուսի բարձրացում, գ) ԵԿՏ թվանշանների գումարում և հանում՝ կախված գործողության ընտրության մուտքի արժեքից:

**Խնդիր 3.26.** Ստանալ  $\overline{\text{OSU}}$ -ի ծրագրավորման աղյուսակը, որն իրականացնում է եռաբիթ երկուական թվերի քառակուսի բարձրացնելու գործողությունը: Նվազարկել անհրաժեշտ կոնյունկցիաների թիվը:

**Խնդիր 3.27.** Կառուցել  $n=11$  մեծամասնական սխեմա գումարիչների միջոցով:

**Խնդիր 3.28.** Քառակարգ գումարիչի հիման վրա կառուցել քառաբիթ  $A$  և  $B$  թվերի համեմատման սարք, որի ելքային ազդանշանը 1 է, երբ  $B > A$ :



## ԵՐԿՐՈՐԴ ԲԱԺԻՆ

### Հաջորդական թվային համակարգեր

#### Գլուխ 4. Հաջորդական տրամաբանական շղթաներ

##### 4.1. Վերջավոր ավտոմատ

Հաջորդական տրամաբանական շղթայի ելքային փոփոխականների արժեքները ժամանակի որևէ պահի կախված են մուտքային փոփոխականների ինչպես տվյալ պահի արժեքներից, այնպես էլ նախորդ պահերի արժեքներից.

$$O^t = F(I^t, I^{t-1}, I^{t-2}, \dots), \quad (4.1)$$

որտեղ  $t$ -ն ընթացիկ ընդհատ ժամանակն է (ավտոմատային ժամանակը),  $t=0, 1, 2, \dots$ ;  $I$ -ն մուտքային փոփոխականն է,  $O$ -ն՝ ելքային փոփոխական: Տրված  $I^t, I^{t-1}, I^{t-2}, \dots$  մուտքային հաջորդականությունը գեներացնում է  $O^t, O^{t-1}, O^{t-2}, \dots$  ելքային հաջորդականություն: Ելքը կարող է որոշվել, եթե առկա են մուտքային փոփոխականի նախորդ արժեքները՝ նախապատմությունը: Այլ կերպ ասած, հաջորդական տրամաբանական շղթան օժտված է հիշողությամբ: Մուտքային փոփոխականի նախորդ պահերի հաջորդականությունը պետք է պահպանվի, որի համար անհրաժեշտ է ունենալ հիշողության տարրեր՝ տրիգերներ: Այսպիսով, հաջորդական տրամաբանական շղթա կառուցելու համար, բացի տրամաբանական տարրերից, պետք է ունենալ նաև տրիգերներ:

Որպեսզի հնարավորություն ստեղծվի որոշել հաջորդական տրամաբանական շղթայի ելքային փոփոխականի արժեքը որևէ պահի, երբ առկա է միայն այդ նույն պահի մուտքային փոփոխականը, շղթայի նկարագրությանն ավելացվում է ներքին վիճակի փոփոխականը: Հաջորդական շղթայի մաթեմատիկական նկարագրությունը տրվում է աբստրակտ ավտոմատի մոդելով: Աբստրակտ ավտոմատը հինգ բաղադրիչներով նկարագրվող հանրահաշվական համակարգ է՝

$$A = (S, I, O, \delta, \lambda), \quad (4.2)$$

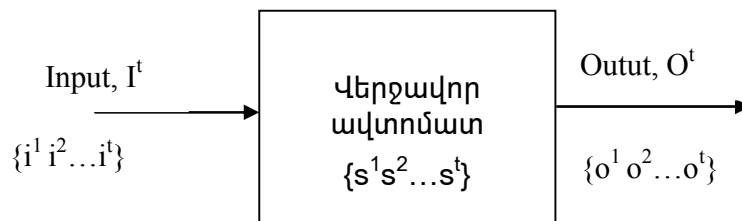
որտեղ -  $S = \{s_1, \dots, s_R\}$  վիճակների բազմությունն է,  $R$ -ը՝ վիճակների քանակն է: Եթե վիճակների թիվը վերջավոր է, ավտոմատը կոչվում է վերջավոր (Finite State Machine, FSM).

- $I = \{i_1, i_2, \dots, i_N\}$  մուտքային սինվոլների (կամ մուտքային փոփոխականի արժեքների) բազմությունն է,  $N$ -ը՝ մուտքային սինվոլների թիվը:
- $O = \{o_1, o_2, \dots, o_M\}$  ելքային սինվոլների (կամ ելքային փոփոխականի արժեքների) բազմությունն է,  $M$ -ը՝ ելքային սինվոլների թիվը:
- $\lambda$ -ն ելքի ֆունկցիան է, որն ավտոմատի ներկա վիճակից ( $S^t$ ) և մուտքային փոփոխականի ընթացիկ արժեքից ( $I^t$ ) որոշում է ավտոմատի ելքի սինվոլը:
- $\delta$ -ն վիճակների անցումների ֆունկցիան է՝

$$S^{t+1} = \delta(S^t, I^t), \quad (4.3)$$

որն ավտոմատի ներկա վիճակից ( $S^t$ ) և մուտքային փոփոխականի ընթացիկ արժեքից որոշվում է ավտոմատի հաջորդ վիճակը՝  $S^{t+1}$ : Ավտոմատի վիճակը  $t=0$  պահին կոչվում է սկզբնական վիճակ՝  $s_0 \in S$ : Եթե ավտոմատի սկզբնական վիճակը տրված է, այդպիսի ավտոմատը կոչվում է սկզբնավորված (initiated):

Տրված մուտքային  $\{i^1 i^2 \dots i^t\}$  հաջորդականությունից ավտոմատը ձևավորում է  $\{o^1 o^2 \dots o^t\}$  ելքային հաջորդականություն, որը կախված է սկզբնական վիճակից և  $\delta$  ու  $\lambda$  ֆունկցիաներից: Այդ ընթացքում ավտոմատն անցնում է  $\{s^1 s^2 \dots s^t\}$  վիճակներով,  $s^1=s_0$  (նկ. 4.1),  $s^2=\delta(s^1, i^1)$ ,  $s^3=\delta(s^2, i^2)$ , ...,  $s^{t+1}=\delta(s^t, i^t)$ :



Նկ. 4.1. Վերջավոր ավտոմատ

Կախված ելքային ֆունկցիայի որոշման եղանակից՝ տարբերում են երկու տիպի ավտոմատ՝ Միլի և Մուրի: Միլի ավտոմատի ելքային փոփոխականը ժամանակի որևէ պահի որոշվում է մուտքային փոփոխականի և վիճակի փոփոխականի արժեքներով այդ նույն պահին՝

$$O^t = \lambda(S^t, I^t): \quad (4.4)$$

Նկ. 4.1-ում ելքային հաջորդականությունը Միլի մոդելի դեպքում կլինի՝  $o^1=\lambda(s^1, i^1)$ ,  $o^2=\lambda(s^2, i^2)$ , ...,  $o^{t+1}=\lambda(s^t, i^t)$ :

Մուրի ավտոմատի ելքային փոփոխականի արժեքը ժամանակի որևէ պահի բացահայտորեն կախված է միայն վիճակի փոփոխականի արժեքից այդ նույն պահին՝

$$O^t = \lambda(S^t): \quad (4.5)$$

Նկ. 4.1-ում ելքային հաջորդականությունը Մուրի մոդելի դեպքում կլինի՝  $o^1=\lambda(s^1)$ ,  $o^2=\lambda(s^2)$ , ...,  $o^{t+1}=\lambda(s^t)$ : Նկատենք, որ Մուրի ավտոմատի ելքային հաջորդականությունը նույնպես որոշվում է մուտքային հաջորդականությունից, սակայն այդ կախվածությունը անբացահայտ է՝ տրվում է վիճակների անցումների միջոցով:

Ավտոմատի այս մոդելները կոչվել են դրանց առաջին հետազոտողների՝ Էդվարդ Մուրի և Ջորջ Միլի անուններով, որոնք այդ մոդելների տարբերությունը հայտնաբերել են 1950-ական թվականներին:

Ցանկացած ավտոմատ կարող է ներկայացվել և՛ Մուրի, և՛ Միլի մոդելներով: Միլի ավտոմատը սովորաբար ունենում է ավելի քիչ թվով վիճակներ, և ելքի արժեքը որոշվում է վիճակի փոփոխությունից ամիջապես հետո: Մուրի մոդելում կարող են անհրաժեշտ լինել ավելի մեծ թվով վիճակներ, իսկ ելքային փոփոխականի արժեքները միշտ առկա են՝ բացի վիճակի փոփոխության պահերից: Ցանկացած ավտոմատի համար գոյություն ունեն Միլի մոդելից Մուրի մոդելի, և հակադարձ անցման համարժեք

ձևափոխություններ:

Գոյություն ունեն ավտոմատների նակարգության մի շարք եղանակներ, որոնցից գործնականում առավել տարածված են գրաֆների և աղյուսակային եղանակները:

Միլի ավտոմատի աշխատանքը տրվում է անցումների աղյուսակով, որում արտապատկերվում են (4.3) (4.4) ֆունկցիաները: Աղյուսակ 4.1-ում ցույց է տրված Միլի ավտոմատի անցումների աղյուսակի օրինակ՝  $S=\{s_1, s_2, s_3\}$ ,  $I=\{i_1, i_2\}$ ,  $O=\{o_1, o_2\}$ : Աղյուսակի “Հաջորդ վիճակ” սյուններում լրացվում են ներկա վիճակին և ներկա մուտքին համապատասխանող հաջորդ վիճակի սիմվոլը, իսկ “Ելք” սյուններում՝ ելքի սիմվոլը: Հաջորդ վիճակի և ելքի սյունների թիվը աղյուսակում հավասար է մուտքի սիմվոլների թվին՝  $N=2$ : Ավտոմատի աշխատանքը լիովին նկարագրվում է անցումների աղյուսակով: Օրինակ՝ եթե ներկա պահին ավտոմատը գտնվում է  $s_1$  վիճակում և մուտքին տրվում է  $i_1$  սիմվոլը, ապա այն անցնում է  $s_2$  վիճակին ( $s_2=\delta(s_1, i_1)$ ), և ելքում ձևավորում է  $o_1$  սիմվոլը ( $o_1=\lambda(s_1, i_1)$ ):

Աղյուսակ 4.1

Ներկա վիճակ ( $S^i$ )	Հաջորդ վիճակ ( $S^{i+1}$ )		Ելք ( $O^i$ )	
	Մուտք՝ $I^i=i_1$	Մուտք՝ $I^i=i_2$	Մուտք՝ $I^i=i_1$	Մուտք՝ $I^i=i_2$
$s_1$	$s_2$	$s_1$	$o_1$	$o_2$
$s_2$	$s_3$	$s_1$	$o_1$	$o_1$
$s_3$	$s_1$	$s_2$	$o_2$	$o_1$

Մուլի ավտոմատում ելքային ազդանշանը կախված է միայն վիճակից, այդ պատճառով “Ելք” սյունակը չի նշվում մուտքային սիմվոլներով (աղյուսակ 4.2):

Աղյուսակ 4.2

Ներկա վիճակ ( $S^i$ )	Հաջորդ վիճակ ( $S^{i+1}$ )		Ելք ( $O^i$ )
	Մուտք՝ $I^i=i_1$	Մուտք՝ $I^i=i_2$	
$s_1$	$s_2$	$s_1$	$o_1$
$s_2$	$s_3$	$s_1$	$o_2$
$s_3$	$s_4$	$s_2$	$o_2$
$s_4$	$s_1$	$s_3$	$o_1$

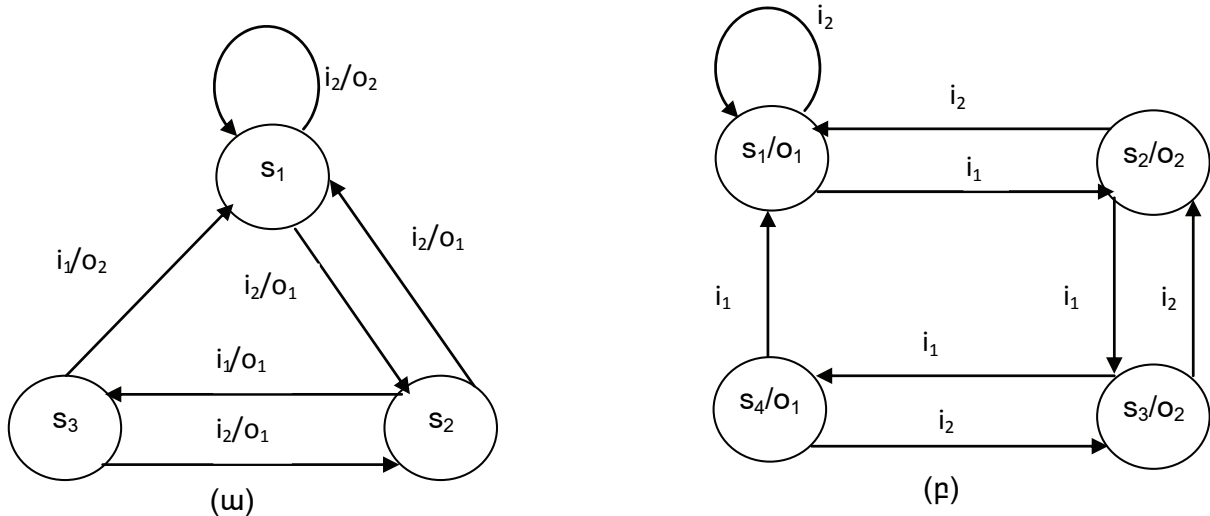
Ընդհանուր դեպքում  $R$  վիճակներով և  $N$  մուտքերով ավտոմատի անցումների աղյուսակն ունի  $R$  տողեր՝ մեկ տող յուրաքանչյուր վիճակի համար: Միլի ավտոմատի դեպքում հաջորդ վիճակի և ելքի սյունների թիվը հավասար է մուտքային սիմվոլների թվին՝  $N$ : Մուլի ավտոմատի դեպքում ելքի համար նախատեսված է միայն մեկ սյուն, անկախ մուտքային սիմվոլների թվից, քանի որ ելքային սիմվոլները միարժեք որոշվում են ներկա վիճակով: Միլի ավտոմատի անցումների աղյուսակում սյունների թիվը կարելի է երկու անգամ կրճատել, եթե հաջորդ վիճակի և ելքի սյունները համատեղվեն, ինչպես ցույց է տրված աղյուսակ 4.3-ում: Այստեղ աղյուսակի վանդակներում գրված են “Հաջորդ վիճակ/Ելք”:

Աղյուսակ 4.3

Ներկա վիճակ ( $S^i$ )	Հաջորդ վիճակ ( $S^{i+1}$ )/ Ելք ( $O^i$ )	
	Մուտք՝ $I^i=i_1$	Մուտք՝ $I^i=i_2$
$s_1$	$s_2/o_1$	$s_1/o_2$
$s_2$	$s_3/o_1$	$s_1/o_1$
$s_3$	$s_1/o_2$	$s_2/o_1$

Ավտոմատային գրաֆում (կոչվում է նաև վիճակների գրաֆ, անցումների գրաֆ) վիճակները ներկայացվում են գրաֆի գագաթներով (պատկերվում են օղակներով), իսկ վիճակների միջև անցումները՝ ուղղորդված աղեղներով, որոնք միացնում են գագաթները:

Նկ. 4.2ա-ում ցույց է տրված աղյուսակ 4.1-ով նկարագրվող Միլի ավտոմատի անցումների գրաֆը: Գրաֆի աղեղները նշվում են երկու սիմվոլներով՝ բաժանված թեք գծով (“/”): Մուտքային սիմվոլը, որի ներգործությամբ ավտոմատը կատարում անցումը, նշվում է առաջինը, իսկ այդ անցման ժամանակ ձևավորվող ելքային սիմվոլը նշվում է թեք գծից հետո՝ “Մուտք/Ելք”: Օրինակ,  $s_1$  վիճակից դեպի  $s_2$  վիճակ ուղղորդված աղեղը նշված է  $i_1/o_1$ : Եթե աղեղն սկսվում և ավարտվում է նույն գագաթում, դա նշանակում է, որ տվյալ մուտքային սիմվոլի դեպքում վիճակի փոփոխություն տեղի չունի: Օրինակ, երբ ավտոմատը գտնվում է  $s_1$  վիճակում և մուտքին տրվում է  $i_2$  սիմվոլը, ավտոմատը մնում է  $s_1$  վիճակում:



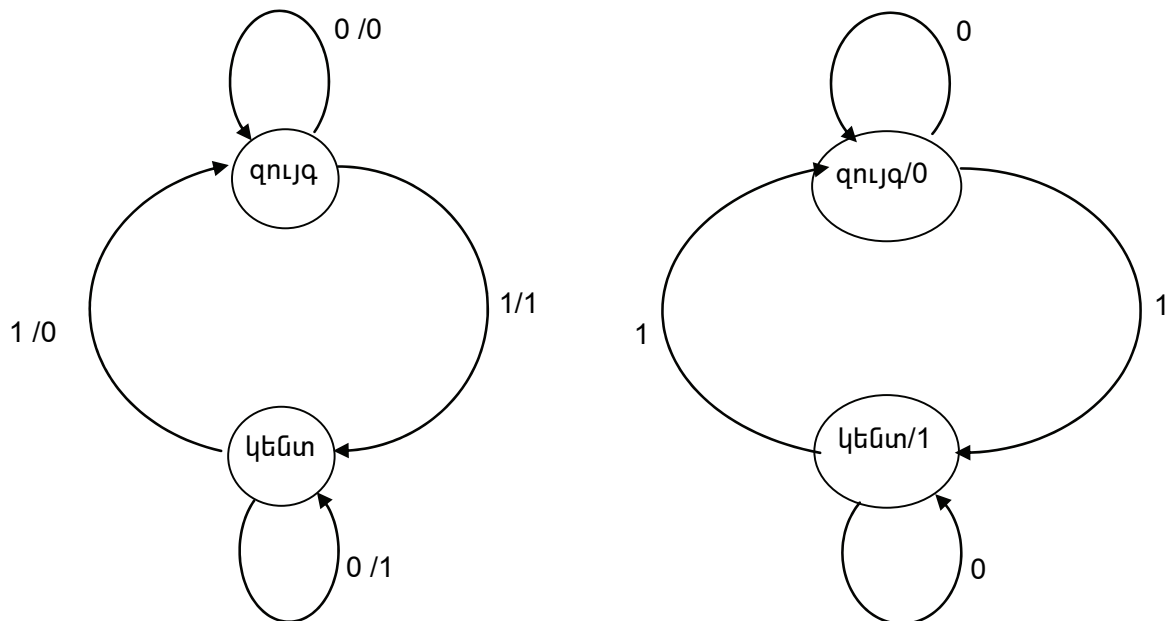
Նկ. 4.2. Ավտոմատի անցումների գրաֆը՝ աղյուսակ 4.1-ով տրված Միլի (ա) և աղյուսակ 4.2-ով տրված Մուրի (բ) ավտոմատների համար

Նկ. 4.2բ-ում ցույց է տրված աղյուսակ 4.2-ով նկարագրվող Մուրի ավտոմատի անցումների գրաֆը: Գրաֆի գագաթներում նշվում է ներկա վիճակը և այդ վիճակում ձևավորվող ելքային սիմվոլը:

Սկզբունքային տարբերությունը չկա անցումների աղյուսակի և գրաֆի միջև, դրանք պարունակում են նույն ինֆորմացիան: Տարբերությունը ներկայացման ձևն է: Անցումների գրաֆը ավելի պատկերավոր է ներկայացնում ավտոմատի աշխատանքը

և ավելի հարմար է այն ստանալ սկզբնական տրամաբանական դատողություններից: Անցումների աղյուսակը հեշտությամբ ստացվում է գրաֆից:

**Օրինակ 4.1.** Դիտարկենք երկուական հաջորդականության “զույգ /կենտ” հատկությունը ստուգող ավտոմատը: Մուտքային սիմվոլների բազմությունը պարունակում է երկու սիմվոլ՝ 0 և 1: Ելքային սիմվոլների բազմությունը նույնպես պարունակում է երկու սիմվոլ՝ 0 և 1: Ավտոմատի ելքում ձևավորվում է 1, եթե մուտքին տրված հաջորդականությունում 1-երի թիվը կենտ է: Իսկ երբ մուտքային հաջորդականությունում 1-երի թիվը զույգ է, ելքում ձևավորվում է 0 սիմվոլ: Ակնհայտ է, որ ելքային փոփոխականի արժեքը ժամանակի որևէ պահի կախված է մուտքային հաջորդականության ողջ նախապատմությունից: Մտցնենք երկու վիճակներ՝ ‘զույգ’ և ‘կենտ’: Եթե մուտքին տրված հաջորդականությունը պարունակում է կենտ թվով 1-եր, ավտոմատը գտնվում է ‘կենտ’ վիճակում, եթե մուտքին տրված հաջորդականությունը պարունակում է զույգ թվով 1-եր, ապա ավտոմատը գտնվում է ‘զույգ’ վիճակում: Ավտոմատի երկու վիճակների միջև անցումները նկարագրվում է նկ. 4.3-ում ցույց տրված գրաֆով: Օրինակ, ‘զույգ’ վիճակից դեպի ‘կենտ’ վիճակ ուղղորդված աղեղը նշված է 1/1: Դա նշանակում է, որ ներկա պահին ավտոմատը ‘զույգ’ վիճակում է և մուտքին տրվում է 1՝ Մուտք=1, որի ազդեցության տակ ավտոմատն անցնում է ‘կենտ’ վիճակ, և ելքում ձևավորվում է 1՝ Ելք=1: Մուտքի ավտոմատի դեպքում աղեղները նշվում են միայն մուտքային սիմվոլներով, իսկ ելքի արժեքը գրված է գրաֆի գագաթի օղակում՝ “վիճակ/ելք” ձևաչափով:



Նկ. 4.3. Երկուական հաջորդականության “զույգ /կենտ” հատկությունը ստուգող ավտոմատի գրաֆը. (ա) Միլի մոդել, (բ) Մուրի մոդել

Ավտոմատի անցումները Միլի և Մուրի մոդելների համար բերված են աղյուսակ 4.3 և 4.4-ում համապատասխանաբար:

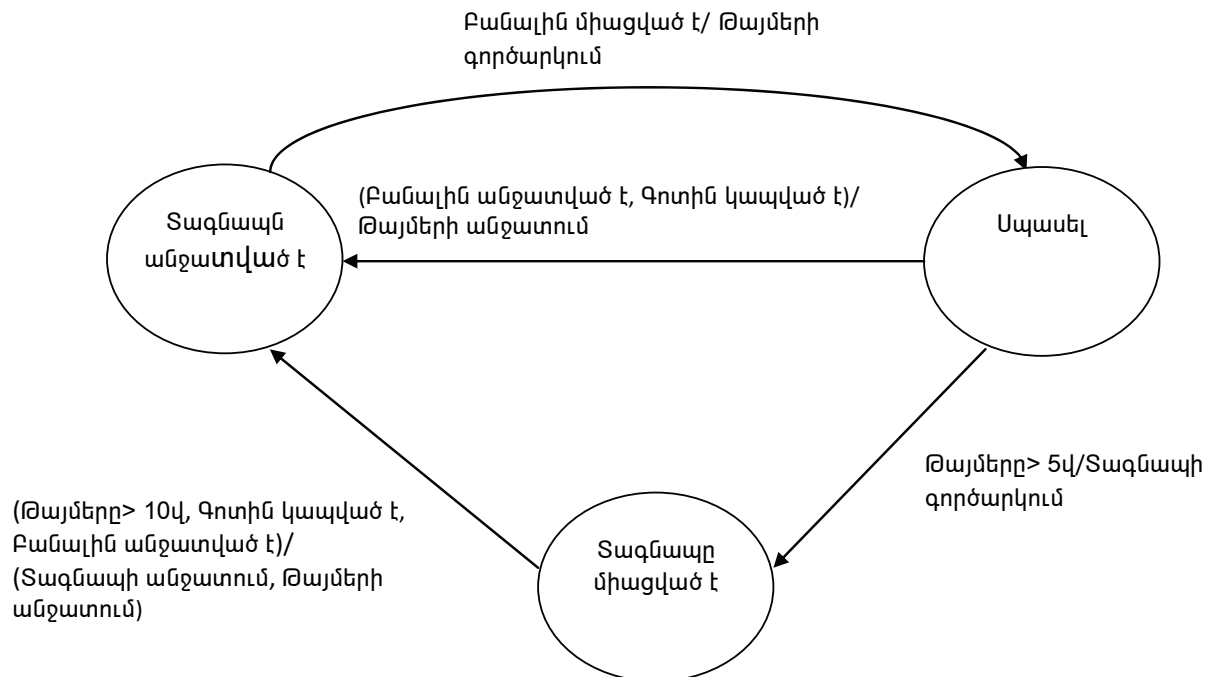
Աղյուսակ 4.3

Ներկա վիճակ	Հաջորդ վիճակ		Ելք	
	Մուտք=0	Մուտք=1	Մուտք=0	Մուտք=1
զույգ	զույգ	կենտ	0	1
կենտ	կենտ	զույգ	1	0

Աղյուսակ 4.4

Ներկա վիճակ	Հաջորդ վիճակ		Ելք
	Մուտք=0	Մուտք=1	
զույգ	զույգ	զույգ	0
կենտ	կենտ	կենտ	1

**Օրինակ 4.2.** Դիտարկենք հետևյալ կառավարող ավտոմատը: Երբ վարորդը միացնում է շարժիչի գործարկման բանալին, բայց չի կապում անվտանգության գոտին 5 վայրկյանի ընթացքում, և եթե այդ ընթացքում վարորդը հետ չի պտտում բանալին, ապա միանում է տագնապի ազդանշանը: Տագնապի ազդանշանը մնում է միացած 5 վայրկյանի ընթացքում: Ժամանակը հսկվում է թայմերի միջոցով: Խնդրի բերված բառացի նկարագրությունից կարելի է ստանալ ավտոմատի անցումների գրաֆը, ինչպես ցույց է տրված նկ. 4.4-ում:



Նկ. 4.4. Ավտանգության գոտու վիճակի հսկման ավտոմատի անցումների գրաֆը

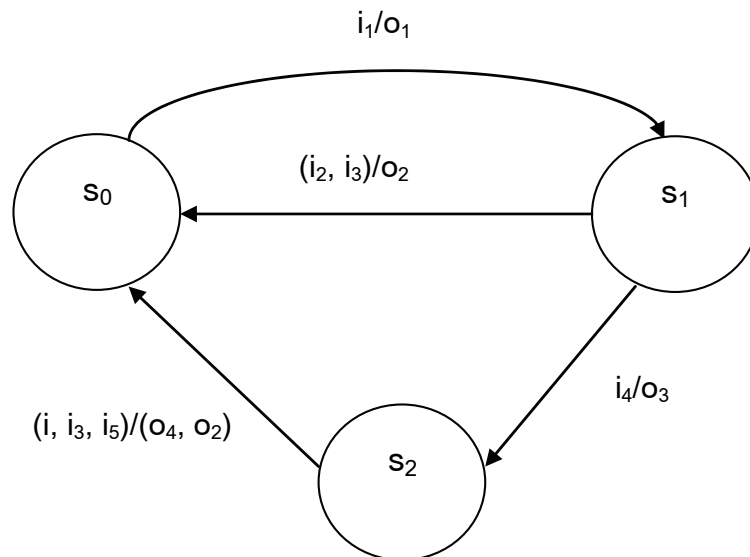
Ավտանգության գոտու վիճակի հսկման ավտոմատին կարելի է վերագրել հետևյալ վիճակները՝  $s_0$ ="Տագնապ անջատված է",  $s_1$ ="Սպասել",  $s_2$ ="Տագնապը միացված է":

Ավտոմատն ունի հինգ մուտքային սիմվոլներ՝  $i_1$ ="Բանալին միացված է",  $i_2$ ="Բանալին անջատված է",  $i_3$ ="Գոտին կապված է",  $i_4$ ="Թայմերը > 5վ",  $i_5$ ="Թայմերը > 10վ":

Ավտոմատն ունի չորս ելքային սինվոլներ՝  $o_1$  = “Թայմերի գործարկում”,  $o_2$  = “Թայմերի անջատում”,  $o_3$  = “Տազնապի գործարկում”,  $o_4$  = “Տազնապի անջատում”:

Ավտոմատը գտնվում է սկզբնական  $s_0$  = “Տազնապն անջատված է” վիճակում այնքան ժամանակ, քանի դեռ վարորդը չի միացրել բանալին: Երբ բանալին միացվում է, ավտոմատը գնում է  $s_1$  = “Սպասել” և թայմերը գործարկվում է՝ սկսում է հաշիվը: Եթե գոտին կապված է կամ 5վ-ի ընթացքում բանալին անջատվում է, ապա ավտոմատը վերադառնում է  $s_0$  = “Տազնապն անջատված է” վիճակ: Հակառակ դեպքում ավտոմատը գնում է  $s_2$  = “Տազնապը միացված է” վիճակ՝ միանում է տազնապի ազդանշանը: Եթե այդ վիճակում բանալին անջատվում է, կամ վարորդը կապում է գոտին, կամ թայմերի հաշիվն անցնում է 10վ-ից, ապա ավտոմատը հետ է վերադառնում  $s_0$  = “Տազնապն անջատված է” վիճակ:

Նկ. 4.5-ում ցույց է տրված ավտոմատի անցումների գրաֆը, որտեղ վիճակները, մուտքերը և ելքերը նշված են համապատասխան սինվոլներով:



Նկ. 4.5. Միլի ավտոմատի անցումների գրաֆը

Նկ. 4.5-ի անցումների գրաֆից կարելի է որոշել անցումների ֆունկցիան ( $\delta$ )՝

$$\begin{aligned}
 s_0 &= s_1 \& (i_2 + i_3) + s_2 \& (i_2 + i_3 + i_5), \\
 s_1 &= (s_0 \& i_1), \\
 s_2 &= s_1 \& i_4
 \end{aligned}
 \tag{4.6}$$

և ելքերի ֆունկցիան ( $\lambda$ )՝

$$\begin{aligned}
 o_1 &= s_0 \& i_1, \\
 o_2 &= s_1 \& (i_2 + i_3) + s_2 \& (i_2 + i_3 + i_5), \\
 o_3 &= s_1 \& i_4, \\
 o_4 &= s_2 \& (i_2 + i_3 + i_5):
 \end{aligned}
 \tag{4.7}$$

Ավտոմատի անցումները ցույց են տրված աղյուսակ 4.5-ում: Աղյուսակում “X” սիմվոլը համապատասխանում է գոյություն չունեցող պայմաններին: Օրինակ, ավտոմատը լինելով  $s_0$  = “Տազնապն անջատված է” վիճակում, թայմերը չի կարող գեներացնել 5վ ժամանակը սպառվել է՝  $i_4$  = “Թայմերը > 5վ” ազդանշան:

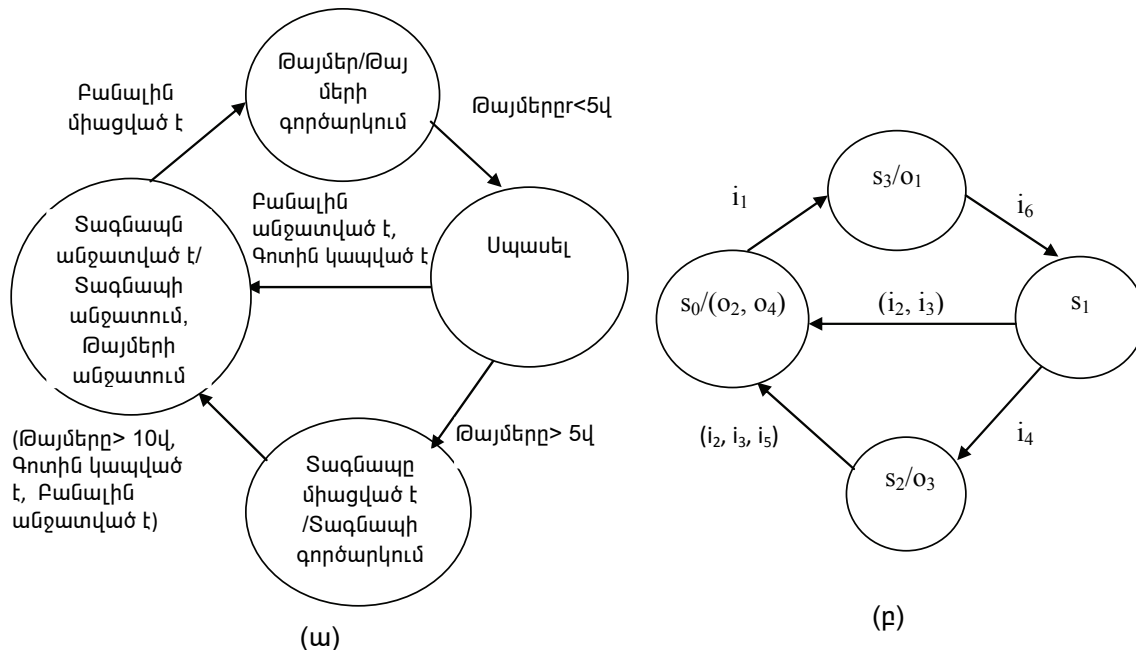
Աղյուսակ 4.5

Ներկա վիճակ	Հաջորդ վիճակ					Ելք				
	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$
$s_0$	$s_1$	X	X	X	X	$o_1$	X	X	X	X
$s_1$	X	$s_0$	$s_0$	$s_2$	X	X	$o_2$	$o_2$	$o_3$	X
$s_2$	X	$s_0$	$s_0$	X	$s_0$	X	$o_4, o_2$	$o_4, o_2$	X	$o_4, o_2$

Այս օրինակի համար Մուրի ավտոմատի անցումների գրաֆը ցույց է տրված նկ. 4.6-ում: Այս գրաֆում ավելացված է ևս մեկ վիճակ՝  $s_3$  = “Թայմեր”, որը հնարավորություն է տալիս ելքային ազդանշանները համապատասխանության մեջ դնել վիճակների հետ՝ անկախ մուտքային ազդանշաններից: Ելքերը նշված են գազաթներում, իսկ աղեղները նշված են միայն մուտքային ազդանշաններով: Միլի ավտոմատի նկատմամբ ավելացվել է նաև մեկ մուտքային ազդանշան՝  $i_6$  = “Թայմերը < 5վ”: Աղյուսակ 4.6-ում ցույց են տրված անցումները:

Աղյուսակ 4.6

Ներկա վիճակ	Հաջորդ վիճակ						Ելք
	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$	$i_6$	
$s_0$	$s_3$	X	X	X	X	X	$o_2, o_4$
$s_1$	X	$s_0$	$s_0$	$s_2$	X	X	X
$s_2$	X	$s_0$	$s_0$	X	$s_0$	X	$o_3$
$s_3$	X	X	X	X	X	$s_1$	$o_1$

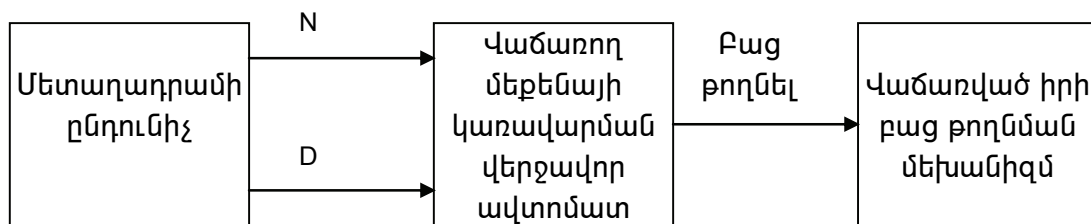


Նկ. 4.6. Օրինակ 4.2-ի Մուրի ավտոմատի անցումների գրաֆը



**Օրինակ 4.3.** Դիտարկենք ևս մեկ պարզ կառավարող ավտոմատի օրինակ՝ վաճառող մեքենայի կառավարման հանգույցը: Կառավարող ավտոմատի աշխատանքի էությունը հետևյալն է. մեքենան բաց է թողում մեկ վաճառվող իր, երբ մետաղադրամներով ստանում է 150 դրամ: Մեքենան ունի մետաղադրամի ընդունման մեկ ճեղք, որով կարելի է ներածել 50 կամ 100 դրամանոց մետաղադրամներ, յուրաքանչյուր անգամ մեկ մետաղադրամ: Մետաղադրամի մեխանիկական ընդունիչը զանազանում է ներմուծված մետաղադրամի տեսակը և հայտնում կառավարող ավտոմատին: Ավտոմատը թույլատրում է բաց թողնել վաճառվող իրի մեկ մուշ: Կատարենք նաև հետևյալ պարզեցումները. ավտոմատը մանրը չի վերադարձնում, եթե գնորդը մտցրել է երկու 100 դրամանոց մետաղադրամներ, ապա կկորցնի 50 դրամ: Այնուհետև, կհամարենք, որ մեքենան պետք է բերվի սկզբնական վիճակ վաճառքի յուրաքանչյուր գործողությունից առաջ: Դա կարող է կատարվել, օրինակ, առանձին մեխանիզմի միջոցով, որով գնորդն ընտրում է վաճառվող իրի տեսակը: Կհամարենք նաև, որ մետաղադրամի ընդունիչը հետ է վերադարձնում 50 և 100 դրամանոցից տարբեր ցանկացած մետաղադրամ:

Նկ. 4.7-ում ցույց է տրված վաճառող մեքենայի բլոկ-սխեման: Կհամարենք, որ մետաղադրամի ընդունիչը ելքում հաստատում է  $N$  ազդանշան, երբ ներածվել է 50 դրամ և հաստատում է  $D$  ազդանշան, երբ ներածվել է 100 դրամ: Ավտոմատը ելքում հաստատում է “Բաց թողնել” ազդանշան, երբ ստացվել է 150 դրամ կամ ավելի:



Նկ. 4.7. Վաճառող մեքենայի բլոկ-սխեման

Խնդրի դրվածքը պարզաբանելուց հետո պետք է բառացի նկարագրությունից անցնել ձևականացված վերացական (աբստրակտ) նկարագրության՝ ավտոմատի գրաֆի կամ անցումների աղյուսակի: Դրա համար հարմար է սկսել հնարավոր մուտքային հաջորդականությունների թվարկումից: Դա կօգնի սահմանել ավտոմատի վիճակները: Այս օրինակում դժվար չէ թվարկել մուտքային բոլոր հնարավոր հաջորդականությունները, որոնց դեպքում պետք է բաց թողնվի մեկ իր.

- երեք հաջորդական 50 դրամանոց մետաղադրամներ՝  $N, N, N$
- երկու 50, հետո մեկ 100 դրամանոց մետաղադրամներ՝  $N, N, D$
- մեկ 50, հետո մեկ 100 դրամանոց մետաղադրամներ՝  $N, D$
- մեկ 100, հետո մեկ 50 դրամանոց մետաղադրամներ՝  $D, N$
- երկու 100 դրամանոց մետաղադրամներ՝  $D, D$

Ավտոմատի ելքն ունի երկու վիճակ (սինվոլ)՝  $O$  = “բաց”, բաց է թողնում մեկ իր և  $C$  = “փակ”, իր բաց չի թողնվում:

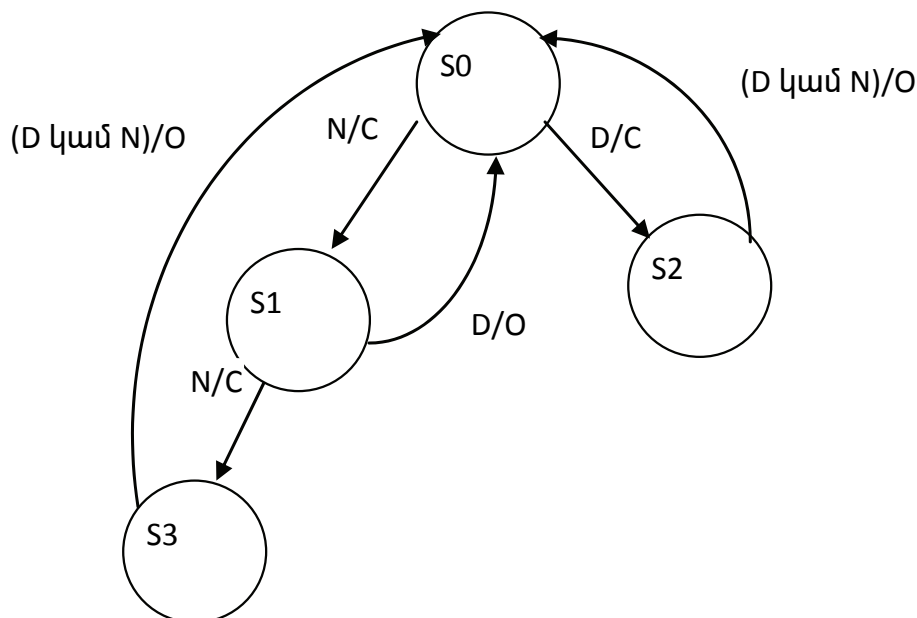
Ավտոմատի աշխատանքը կարելի է ներկայացնել նկ. 4.8-ում ցույց տրված անցումների գրաֆով: Ավտոմատը գտնվում է  $s_0$  վիճակում, երբ ոչ մի մետաղադրամ չի

ներածվել: Երբ ներածվում են մետաղադրամներ, ավտոմատն անցնում է գրաֆի որոշակի վիճակներով՝ գազաթներով, և երբ դրանց գումարը հասնում է 150 դրամի, ելքում ձևավորվում է Օ=»բաց» ազդանշան, և ավտոմատը վերադառնում է  $s_0$  վիճակ: Օրինակ, երբ հաջորդաբար ներածվում են երեք 50 դրամանոցներ, ավտոմատն անցնում է  $S_0$ ,  $S_1$ ,  $S_3$ ,  $S_0$  վիճակներով:

Նկ. 4.8-ում ցույց տրված գրաֆին համապատասխանում է աղյուսակ 4.7-ում ցույց տրված անցումները:

Աղյուսակ 4.7

Ներկա վիճակ	Հաջորդ վիճակ		Ելք	
	Մուտք=N	Մուտք=D	Մուտք=N	Մուտք=D
$S_0$	$S_1$	$S_2$	C	C
$S_1$	$S_3$	$S_0$	C	O
$S_2$	$S_0$	$S_0$	O	O
$S_3$	$S_0$	$S_0$	O	O



Նկ. 4.8. Վաճառող մեքենայի կառավարող ավտոմատի անցումների գրաֆը, Միլի մոդելը

## 4.2. Անցում Մուրի մոդելից Միլի մոդելի

Կարելի է ապացուցել, որ ցանկացած Մուրի մոդելով ավտոմատ կարելի է ձևափոխել Միլի մոդելով ավտոմատի, և ընդհակառակը: Ցույց տանք այդ ձևափոխությունները օրինակների միջոցով, որոնք չեն սահմանափակում կիրառության ընդհանրությունը:

Աղյուսակ 4.8-ում ցույց է տրված Մուրի ավտոմատի անցումների աղյուսակը: Այն Միլի ավտոմատի անցումների աղյուսակի ձևափոխելու համար կօգտագործենք շատ պարզ ընթացակարգ: Նախ՝ նայում ենք ներկա վիճակներին և դրանց համապատասխանող ելքերին, որոնք ցույց են տրված աղյուսակ 4.9-ում: Այնուհետև, կառու-

ցում ենք համարժեք Միլի ավտոմատի անցումների աղյուսակը (աղյուսակ 4.10)՝ հաջորդ վիճակի սյուններում յուրաքանչյուր վիճակի հետ նշելով ելքային սինվոլը ըստ աղյուսակ 4.9-ի:

Աղյուսակ 4.8

Ներկա վիճակ ( $S^t$ )	Հաջորդ վիճակ ( $S^{t+1}$ )		Ելք ( $O^t$ )
	Մուտք՝ $I^t=i_1$	Մուտք՝ $I^t=i_2$	
$S_1$	$S_2$	$S_1$	$O_1$
$S_2$	$S_3$	$S_1$	$O_2$
$S_3$	$S_4$	$S_2$	$O_2$
$S_4$	$S_1$	$S_3$	$O_1$

Աղյուսակ 4.9

Վիճակ ( $S$ )	Ելք ( $O$ )
$S_1$	$O_1$
$S_2$	$O_2$
$S_3$	$O_2$
$S_4$	$O_1$

Աղյուսակ 4.10

Ներկա վիճակ ( $S^t$ )	Հաջորդ վիճակ ( $S^{t+1}$ )/Ելք ( $O^t$ )	
	Մուտք՝ $I^t=i_1$	Մուտք՝ $I^t=i_2$
$S_1$	$S_2/O_2$	$S_1/O_1$
$S_2$	$S_3/O_2$	$S_1/O_1$
$S_3$	$S_4/O_1$	$S_2/O_2$
$S_4$	$S_1/O_1$	$S_3/O_2$

### 4.3. Անցում Միլի մոդելից Մուրի մոդելի

Միլի մոդելից Մուրի մոդելին անցմանը հետևենք աղյուսակ 4.11-ում ցույց տրված օրինակի միջոցով: Միլի մոդելից Մուրի մոդելին անցումն ընդգրկում է ավելի շատ քայլեր, քան Մուրի մոդելից Միլիի անցումը: Դա կապված է այն բանի հետ, որ մեկ որոշակի վիճակի կարող են համապատասխանել մեկից ավելի ելքեր: Օրինակ,  $S$  վիճակի հետ կապված են երկու ելքային սինվոլներ՝  $o_2$  և  $o_1$ , և հնարավոր չէ Մուրի մոդելում տարբեր ելքային սինվոլները կապել մեկ վիճակի հետ:

Աղյուսակ 4.11

Ներկա վիճակ	Հաջորդ վիճակ/Ելք	
	Մուտք՝ $i_1$	Մուտք՝ $i_2$
P	R/ $o_1$	Q/ $o_1$
Q	P/ $o_2$	S/ $o_1$
R	Q/ $o_2$	P/ $o_2$
S	S/ $o_2$	R/ $o_1$

Այս դժվարությունը կարելի է հաղթահարել, եթե մեկից ավելի ելքերի հետ առնչվող վիճակների համար նախատեսենք անհրաժեշտ քանակի լրացուցիչ վիճակներ, որպեսզի մեկ վիճակին առնչվի միայն մեկ ելք: Օրինակ,  $S$  վիճակը վերաորոշ-

վում է իբրև  $S_0$ ՝ կապված  $o_1$  ելքի հետ, և  $S_1$ ՝ կապված  $o_2$  ելքի հետ: Ստորև նկարագրված է այս վիճակների ստացման մեթոդը:

Նախ Միլի ավտոմատի վիճակները և դրանց հետ կապված ելքերը գրվում են նոր աղյուսակում (աղյուսակ 4.12): Աղյուսակ 4.13-ում ելքային սինվոլները գրված են մեկ սյունով: Այս աղյուսակից երևում է, որ Q և S վիճակները պետք վերաորոշվեն, քանի որ դրանք ունեն տարբեր ելքեր: P և R վիճակները պահպանվում են սկզբնական տեսքով: Այժմ կարող ենք կառուցել համարժեք Մուրի ավտոմատի անցումների աղյուսակը՝ օգտվելով ներմուծված նոր վիճակներից՝  $Q_0, Q_1, S_0, S_1$  (աղյուսակ 4.14): Նկատենք, որ ելքի սյունը աղյուսակ 4.14-ում համընկնում է աղյուսակ 4.13-ին հետ:

Աղյուսակ 4.12

Վիճակ	Ելք
P	$o_1$
Q	$o_2, o_1$
R	$o_2$
S	$o_2, o_1$

Աղյուսակ 4.13

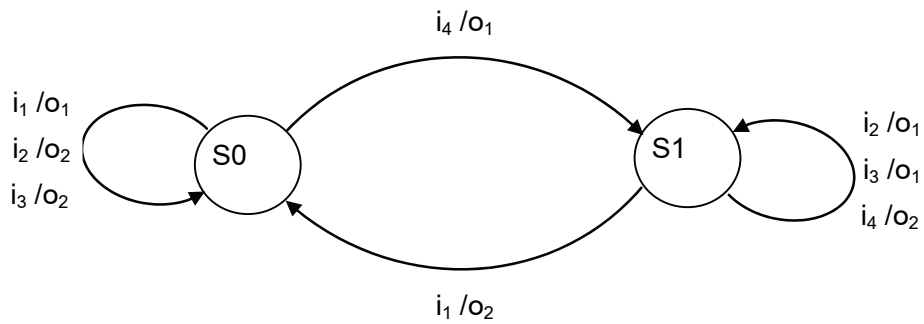
Վիճակ	Ելք	Վերաորոշված վիճակ
P	$o_1$	P
Q	$o_1$ $o_2$	$Q_0$ $Q_1$
R	$o_2$	R
S	$o_1$ $o_2$	$S_0$ $S_1$

Աղյուսակ 4.14

Ներկա վիճակ	Հաջորդ վիճակ		Ելք
	Մուտք $i_1$	Մուտք $i_2$	
P	R	$Q_0$	$o_1$
$Q_0$	P	$S_0$	$o_1$
$Q_1$	P	$S_0$	$o_2$
R	$Q_1$	P	$o_2$
$S_0$	$S_1$	R	$o_1$
$S_1$	$S_1$	R	$o_2$

**Օրինակ 4. 4.** Միլի ավտոմատի գրաֆը ցույց է տրված նկ. 4.9-ում: Կառուցել համարժեք Մուրի ավտոմատի գրաֆը:

Միլի ավտոմատի անցումների աղյուսակը ցույց է տրված աղյուսակ 4.15-ում: Միլի և Մուրի ավտոմատների վիճակների համապատասխանությունը ցույց է տրված աղյուսակ 4.16-ում: Մուրի ավտոմատի համար ներածվել են հետևյալ նոր վիճակները՝  $S_{00}, S_{01}, S_{10}, S_{11}$ : Աղյուսակ 4.17-ում ցույց է տրված Մուրի ավտոմատի անցումների աղյուսակը, իսկ նկ. 4.10-ում՝ գրաֆը:



Նկ. 4.9. Օրինակ 4.4-ի Միլի ավտոմատի գրաֆը

Աղյուսակ 4.15

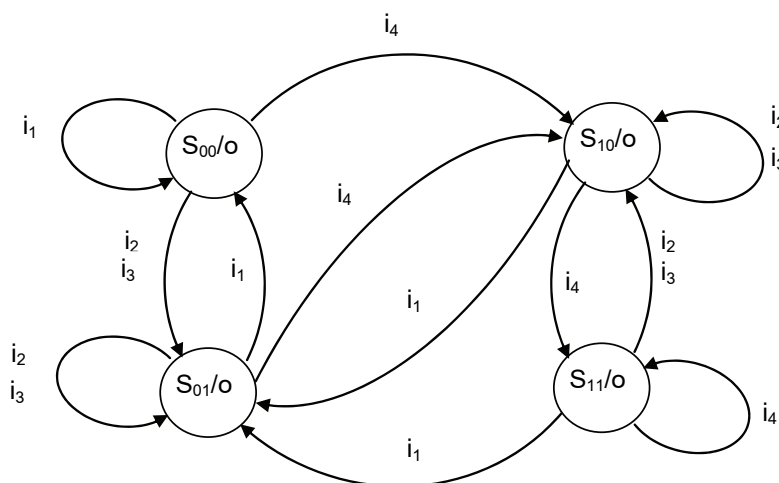
Ներկա վիճակ	Հաջորդ վիճակ/Ելք			
	Մուտք՝ $i_1$	Մուտք՝ $i_2$	Մուտք՝ $i_3$	Մուտք՝ $i_4$
S0	S0/ $o_1$	S0/ $o_2$	S0/ $o_2$	S1/ $o_1$
S1	S0/ $o_2$	S1/ $o_1$	S1/ $o_1$	S1/ $o_2$

Աղյուսակ 4.16

Վիճակ	Ելք	Վերադարձված վիճակ
S0	$o_1$ $o_2$	$S_{00}$ $S_{01}$
S1	$o_1$ $o_2$	$S_{10}$ $S_{11}$

Աղյուսակ 4.17

Ներկա վիճակ	Հաջորդ վիճակ				Ելք
	Մուտք՝ $i_1$	Մուտք՝ $i_2$	Մուտք՝ $i_3$	Մուտք՝ $i_4$	
$S_{00}$	$S_{00}$	$S_{01}$	$S_{01}$	$S_{10}$	$o_1$
$S_{01}$	$S_{00}$	$S_{01}$	$S_{01}$	$S_{10}$	$o_2$
$S_{10}$	$S_{01}$	$S_{10}$	$S_{10}$	$S_{11}$	$o_1$
$S_{11}$	$S_{01}$	$S_{10}$	$S_{10}$	$S_{11}$	$o_2$



Նկ. 4.10. Օրինակ 4.4-ի Մուրի ավտոմատի գրաֆը

#### 4.4. Վիճակների նվազարկում

Ցանկացած նախագծման գործընթաց պետք է նպատակ ունենա նվազարկել վերջնական շղթայի ինքնարժեքը: Ինքնարժեքի նվազարկման երկու ակնհայտ աղբյուրներ են անհրաժեշտ տրամաբանական տարրերի և հիշողության տարրերի (տրիգերների) թվերի նվազարկումը: Քանի որ այս երկու հանգամանքներն ակնառու են, դրանք հանգամանորեն ուսումնասիրվել և հետազոտվել են: Թվային համակարգերի տեսության զգալի մասը նվիրված է հաջորդական տրամաբանական շղթաներում տրամաբանական տարրերի և տրիգերների թվի նվազարկման ալգորիթմներին: Հաջորդական շղթաներում տրիգերների թվի նվազարկումը տեսությունում հայտնի է ավտոմատի վիճակների թվի նվազարկման խնդրի անվանումով: Վիճակների թվի նվազարկումը պետք է իրականացվի՝ պահպանելով անհրաժեշտ մուտք-ելք առնչությունները:

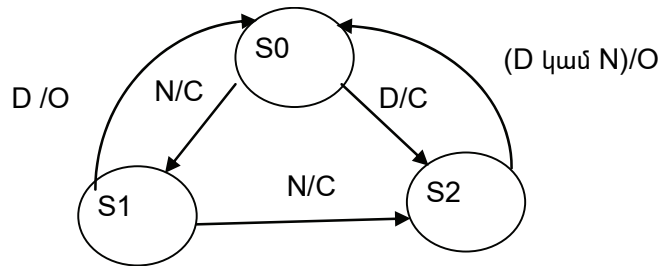
Ավտոմատի վիճակների նվազարկման ալգորիթմները հիմնված են համարժեք վիճակների հայտնաբերման և դրանց կրճատման վրա: Երկու վիճակներ կոչվում են համարժեք, եթե մուտքային յուրաքանչյուր սինվոլի դեպքում դրանք տալիս են միևնույն ելքային սինվոլը և ավտոմատը տանում են կամ միևնույն հաջորդ վիճակ, կամ համարժեք հաջորդ վիճակներ: Երբ երկու վիճակներ համարժեք են, դրանցից մեկը կարելի է հեռացնել՝ առանց փոփոխելու մուտք-ելք առնչությունները:

Վիճակների նվազարկման՝ գործնականում հաճախ օգտագործվող ալգորիթմները հիմնված են անցումների աղյուսակի տողերի համընկման և իմպլիկացիաների աղյուսակի մեթոդների վրա: Տողերի համընկման մեթոդը հարմար է ձեռքով աշխատելու համար, բայց ոչ միշտ է տալիս լավագույն արդյունք: Իմպլիկացիաների աղյուսակի մեթոդը ալգորիթմական է և ավելի բարդ, բայց երաշխավորում է արդյունքի լավագույն տարբերակ:

Որպես օրինակ դիտարկենք աղյուսակ 4.7-ում ցույց տրված վաճառող մեքենայի անցումների աղյուսակը: Աղյուսակ 4.7-ից կարելի է տեսնել, որ վերջին երկու տողերը համարժեք են: Հեռացնելով վերջին տողը և փոխարինելով S3-ը S2-ով՝ կստանանք աղյուսակ 4. 18-ում ցույց տրված նվազարկված վիճակների աղյուսակը: Նվազարկված վիճակներով ավտոմատի անցումների գրաֆը ցույց է տրված նկ. 4.11-ում:

Աղյուսակ 4.18

Ներկա վիճակ	Հաջորդ վիճակ		Ելք	
	Մուտք=N	Մուտք=D	Մուտք=N	Մուտք=D
S0	S1	S2	C	C
S1	S2	S0	C	O
S2	S0	S0	O	O



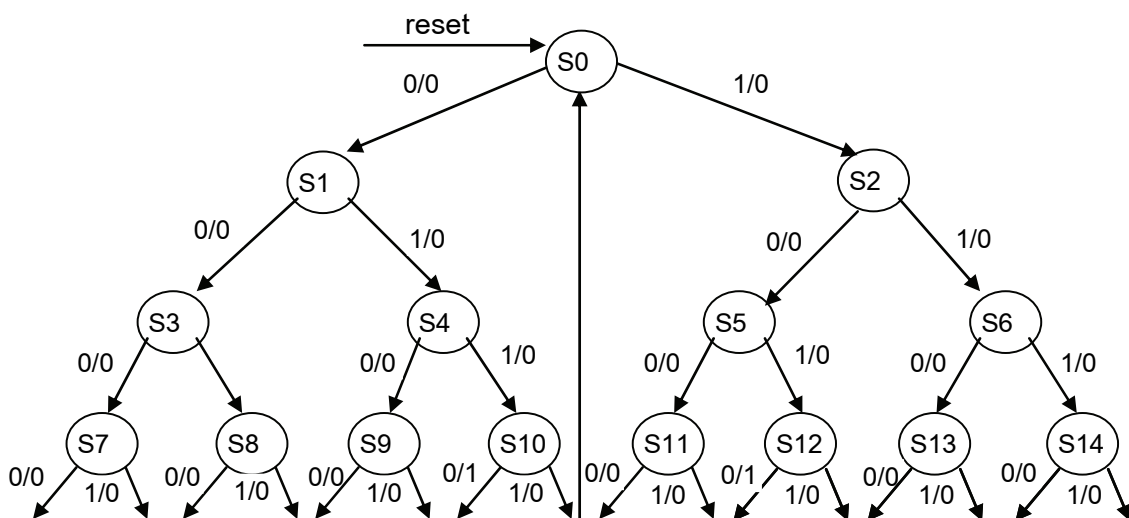
Նկ. 4.11. Վաճառող ավտոմատի նվազարկված անցումների գրաֆը

Անցումների աղյուսակի տողերի համընկնման մեթոդով վիճակների նվազարկման ընթացակարգը կարելի է ներկայացնել հետևյալ քայլերի հաջորդականությամբ.

- կառուցել ավտոմատի անցումների աղյուսակը,
- միևնույն հաջորդ վիճակ և ելք ունեցող տողերը խմբավորել և յուրաքանչյուր խումբ փոխարինել մեկ նոր վիճակով,
- անցումների աղյուսակում հին համարժեք վիճակները փոխարինել նոր վիճակով,
- կրկնել նախորդ երկու քայլերը, մինչև այլևս համարժեք վիճակներ չեն հայտնաբերվում:

**Օրինակ 4.5.** Կառուցել երկուական հաջորդականություն հայտնաբերող ավտոմատ և տողերի համընկնման մեթոդով նվազարկել վիճակները: Ավտոմատը պետք է հայտնաբերի 0110 կամ 1010 քառաբիթ հաջորդականությունները երկուական տվյալների հոսքից: Ավտոմատը վերադարձվում է սկզբնական վիճակ յուրաքանչյուր քառաբիթ հաջորդականությունից հետո:

Ավտոմատն ունի մեկ մուտք՝ X, որին տրվում են 0 կամ 1 սիմվոլներ և մեկ ելք՝ Y, նույնպես երկու սիմվոլներով՝ 1, երբ հայտնաբերվում է 0110 կամ 1010 քառաբիթ հաջորդականություն, 0՝ մյուս դեպքերում: Համապատասխան Միլի ավտոմատի անցումների գրաֆը ցույց է տրված նկ. 4.12-ում, իսկ անցումների աղյուսակը՝ աղյուսակ 4.19-ում:



Նկ. 4.12. Օրինակ 4.5-ի Միլի ավտոմատի գրաֆը

Աղյուսակ 4.19

Մուտքային հաջորդականություն	Ներկա վիճակ	Հաջորդ վիճակ		Ելք, Y	
		X=0	X=1	X=0	X=1
Սկզբնական վիճակի կարգում՝ reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S6	0	0
00	S3	S7	S8	0	0
01	S4	S9	S10	0	0
10	S5	S11	S12	0	0
11	S6	S13	S14	0	0
000	S7	S0	S0	0	0
001	S8	S0	S0	0	0
010	S9	S0	S0	0	0
011	S10	S0	S0	1	0
100	S11	S0	S0	0	0
101	S12	S0	S0	1	0
110	S13	S0	S0	0	0
111	S14	S0	S0	0	0

Համեմատելով աղյուսակ 4.19-ի տողերը՝ կարելի է տեսնել, որ S10-ը համար-  
ժեք է S12-ին: Աղյուսակի S10 և S12 տողերը փոխարինվում են մեկ նոր՝ տողով (աղյու-  
սակ 4.20), և շարունակվում է նվազարկման գործընթացը:

Աղյուսակ 4.20

Մուտքային հաջորդականություն	Ներկա վիճակ	Հաջորդ վիճակ		Ելք, Y	
		X=0	X=1	X=0	X=1
Սկզբնական վիճակի կարգում՝ reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S6	0	0
00	S3	S7	S8	0	0
01	S4	S9	S10*	0	0
10	S5	S11	S10*	0	0
11	S6	S13	S14	0	0
000	S7	S0	S0	0	0
001	S8	S0	S0	0	0
010	S9	S0	S0	0	0
011/101	S10*	S0	S0	1	0
100	S11	S0	S0	0	0
110	S13	S0	S0	0	0
111	S14	S0	S0	0	0

S7, S8, S9, S11, S13, S14 վիճակները համարժեք են, դրանք կփոխարինվեն  
S7\* նոր վիճակով (աղյուսակ 4.21): Այստեղ նորից կարելի է նկատել, որ S3 և S6 վի-  
ճակները համարժեք են, դրանք կփոխարինվեն S6\* նոր վիճակով: S4 և S5 վիճակ-  
ները նույնպես համարժեք են և փոխարինվում են S4\*-ով (աղյուսակ 4.22):



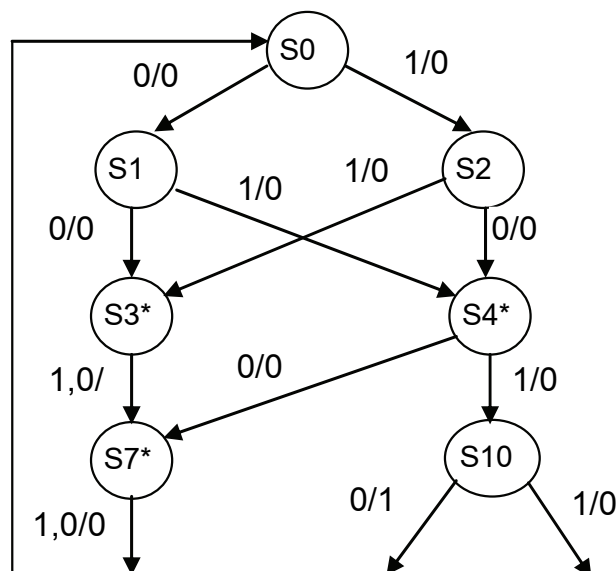
Աղյուսակ 4.21

Մուտքային հաջորդականություն	Ներկա վիճակ	Հաջորդ վիճակ		Ելք, Y	
		X=0	X=1	X=0	X=1
Սկզբնական վիճակի կարգում` reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S6	0	0
00	S3	S7*	S7*	0	0
01	S4	S7*	S10*	0	0
10	S5	S7*	S10*	0	0
11	S6	S7*	S7*	0	0
000/001/010/100/110/111	S7*	S0	S0	0	0
011/101	S10*	S0	S0	1	0

Աղյուսակ 4.22

Մուտքային հաջորդականություն	Ներկա վիճակ	Հաջորդ վիճակ		Ելք, Y	
		X=0	X=1	X=0	X=1
Սկզբնական վիճակի կարգում` reset	S0	S1	S2	0	0
0	S1	S3*	S4*	0	0
1	S2	S4*	S3*	0	0
00/11	S3*	S7*	S7*	0	0
01/10	S4*	S7*	S10*	0	0
000/001/010/100/110/111	S7*	S0	S0	0	0
011/101	S10*	S0	S0	1	0

Աղյուսակ 4.22-ում այլևս համարժեք վիճակներ չկան: Հետևաբար այն նվազարկված վիճակներով ավտոմատի անցումների աղյուսակն է: Համապատասխան գրաֆը ցույց է տրված նկ. 4.13-ում

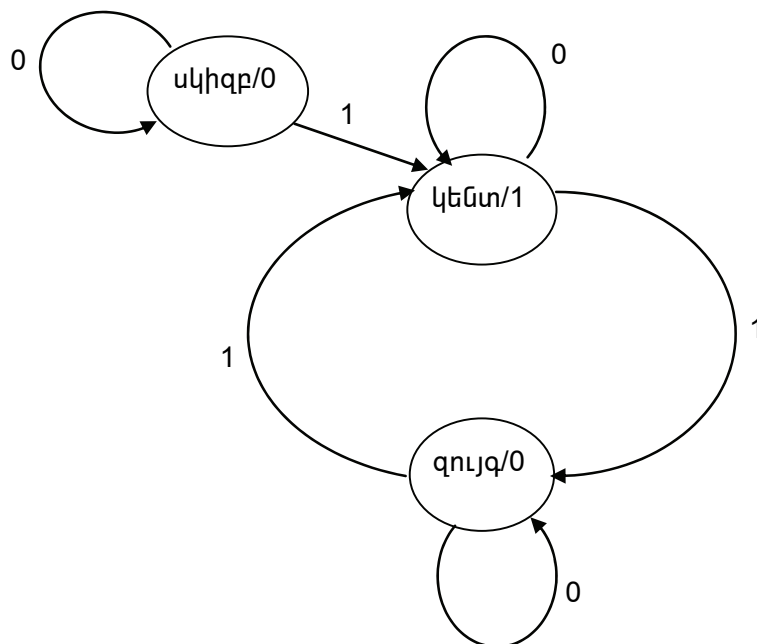


Նկ. 4.13. Օրինակ 4.5-ի նվազարկված վիճակներով Միլի ավտոմատի գրաֆը

#### 4.5. Ավտոմատի վիճակների նվազարկման իմպլիկացիաների քարտերի մեթոդ

Տողերի համընկման մեթոդն ունի որոշակի սահմանափակումներ. նույն ներկա վիճակ և հաջորդ վիճակ ունեցող տողերը ներկայացնում են միավորման որոշակի խնդիրներ: Օրինակ, երկուական հաջորդականությունում զույգության հայտանիշն ստուգող ավտոմատի անցումների աղյուսակից (աղյուսակ 4.23) չի հետևում, որ Սկիզբ և Ջույգ վիճակները համարժեք են: Սակայն, ինչպես կտեսնենք հետագայում, դրանք համարժեք են:

Իմպլիկացիաների (ենթադրությունների, հետևությունների) քարտերի մեթոդը լուծում է այս խնդիրը: Հետևենք իմպլիկացիաների քարտերի մեթոդին օրինակների միջոցով:



Նկ. 4.14. “Սկիզբ” սկզբնական վիճակով երկուական հաջորդականության զույգության հայտանիշը ստուգող Մուրի ավտոմատի անցումների գրաֆը

Աղյուսակ 4.23

Ներկա վիճակ	Հաջորդ վիճակ		Ելք
	X=0	X=1	
սկիզբ	սկիզբ	կենտ	0
կենտ	կենտ	զույգ	1
զույգ	զույգ	կենտ	0

Կան դեպքեր, երբ երկու վիճակներ չունեն համընկնող հաջորդ վիճակներ, բայց չնայած դրան՝ դրանք հանգեցնում են համարժեք հաջորդ վիճակների: Դիտարկենք աղյուսակ 4.24-ում ներկայացված անցումների աղյուսակը: S0 և S1 վիճակներն ունեն նույն ելքերը: Դրանց հաջորդ վիճակներն են S2 և S3 x=0 մուտքի դեպքում և S1 և S0 x=1 մուտքի դեպքում: Եթե կարողանանք ցույց տալ, որ (S2,S3) վիճակները համարժեք են, ապա դրանից կհետևի, որ (S0,S1) վիճակները նույնպես համարժեք են, քանի որ

դրանք կունենան միևնույն կամ համարժեք հաջորդ վիճակներ: Այսինքն (S0,S1) վիճակների համարժեքությունը ենթադրում է, որ (S2,S3)-ը համարժեք են: Երբ այս հարաբերակցությունը գոյություն ունի, ասում են, որ (S0,S1)-ը ենթադրում է (S2,S3): Նույն ձևով, աղյուսակ 4.24-ի վերջին երկու տողերից կարելի է եզրակացնել, որ (S2,S3)-ը ենթադրում է, որ (S0,S1) համարժեք վիճակներ են: Վիճակների համարժեքության հայտանիշը հետևյալն է՝ եթե (S0,S1)-ը ենթադրում է (S2,S3) և (S2,S3)-ը ենթադրում է (S0,S1), ապա վիճակների այս երկու զույգերն էլ համարժեք են՝ S0-ն համարժեք է S1-ին և S2-ը համարժեք է S3-ին: Որպես հետևանք, աղյուսակ 4.24-ի չորս տողերը կարող են բերվել երկու տողերի՝ S0 ու S1 վիճակները միավորելով մեկ վիճակի և S2 ու S3 վիճակները՝ երկրորդ վիճակի:

Աղյուսակ 4.24

Ներկա վիճակ	Հաջորդ վիճակ		Ելք	
	X=0	X=1	X=0	X=1
S0	S2	S1	0	1
S1	S3	S0	0	1
S2	S0	S3	1	0
S3	S1	S3	1	0

Ավտոմատի բոլոր վիճակների զույգ առ զույգ ստուգումը հնարավոր համարժեքության հայտնաբերման համար կարելի է իրագործել համակարգված եղանակով՝ իմպլիկացիաների քարտի միջոցով: Իմպլիկացիաների քարտը կազմված է վանդակներից՝ վիճակների յուրաքանչյուր զույգին տրամադրվում է մեկ վանդակ:

Համարժեք վիճակների որոնումն իրագործվում է հետևյալ ընթացակարգով.

– կառուցել իմպլիկացիաների քարտ՝ մեկ վանդակ վիճակների յուրաքանչյուր զույգի համար,

– եթե որևէ երկու վիճակների ելքերը տարբերվում են, ապա դրանք համարժեք չլինել չեն կարող, և համապատասխան վանդակը նշվում է “X”-ով, հակառակ դեպքում վանդակում լրացվում են հաջորդ վիճակի զույգերը բոլոր մուտքային համակցությունների համար,

– նախորդ քայլում նկարագրված ստուգումները իրականացնել քարտի բոլոր վանդակների համար,

– վանդակի յուրաքանչյուր հաջորդ վիճակի զույգի համար ստուգել համարժեքությունը, եթե դրանց հաջորդ վիճակները համարժեք չեն՝ նշված են “X”-ով, այդ վիճակները նույնպես համարժեք չեն կարող լինել՝ այդ վանդակը նույնպես նշվում “X”-ով,

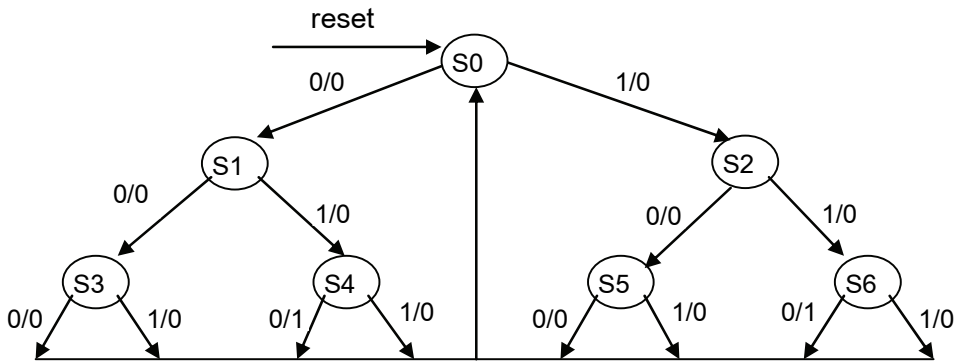
– նախորդ քայլի ստուգումները կրկնել այնքան, քանի դեռ դրանք բերում է քարտի վիճակի փոփոխության: Եթե քարտում այլևս ոչինչ չի փոփոխվում, ուրեմն համարժեք վիճակների որոնումն ավարտված է,

– քարտի “X”-ով չնշված վանդակները համապատասխանում են համարժեք վիճակների:

**Օրինակ 4.6.** Կառուցել երկուական հաջորդականություններ հայտնաբերող ավտոմատ և իմպլիկացիաների քարտի մեթոդով նվազարկել վիճակները: Ավտոմատը պետք է

հայտնաբերի 010 կամ 110 եռաբիթ հաջորդականությունները երկուական տվյալների հոսքից: Ավտոմատը վերադարձվում է սկզբնական վիճակ յուրաքանչյուր եռաբիթ հաջորդականությունից հետո:

Ավտոմատն ունի մեկ մուտք՝ X, որին տրվում են 0 կամ 1 սինվոլներ և մեկ ելք՝ Y, նույնպես երկու սինվոլներով՝ 1, երբ հայտնաբերվում է 010 կամ 110 հաջորդականություն, 0՝ մյուս դեպքերում: Համապատասխան Միլի ավտոմատի անցումների գրաֆը ցույց է տրված նկ. 4.15-ում, իսկ անցումները՝ աղյուսակ 4.25-ում:



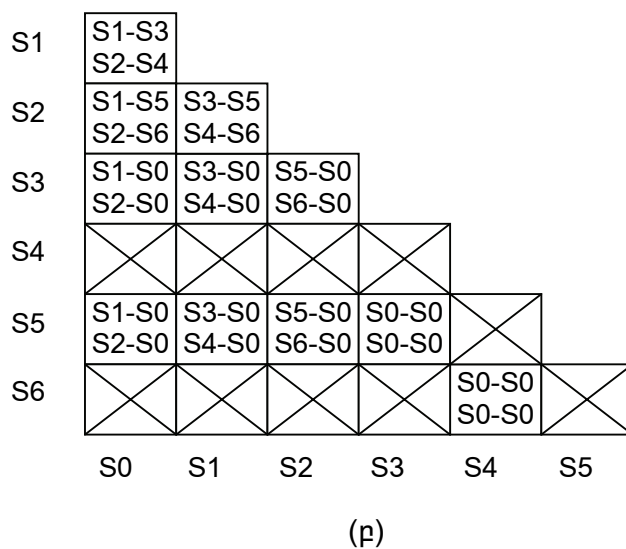
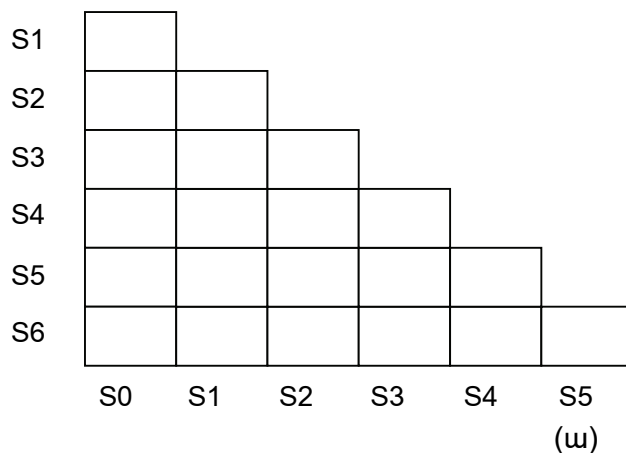
Նկ. 4.15. Օրինակ 4.6-ի Միլի ավտոմատի գրաֆը

Աղյուսակ 4.25

Մուտքային հաջորդականություն	Ներկա վիճակ	Հաջորդ վիճակ		Ելք, Y	
		X=0	X=1	X=0	X=1
reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S6	0	0
00	S3	S0	S0	0	0
01	S4	S0	S0	1	0
10	S5	S0	S0	0	0
11	S6	S0	S0	1	0

Թվով յոթ վիճակների դեպքում չլրացված իմպլիկացիաների քարտը ցույց է տրված նկ. 4.16ա-ում: Ձախից ուղղահայաց ուղղությամբ նշված են բոլոր վիճակները, բացի առաջինից: Ներքևում հորիզոնական ուղղությամբ նշված են բոլոր վիճակները, բացի վերջինից: Արդյունքում ստացվել է մի պատկեր, որը ցույց է տալիս վիճակների բոլոր հնարավոր զույգերի համակցությունները, որոցից յուրաքանչյուրին համապատասխանում է մեկ վանդակ՝ ուղղահայաց և հորիզոնական նշված վիճակների հատման վայրում:

Առաջին նշումից հետո ստացված իմպլիկացիաների քարտը ցույց է տրված նկ. 4.16բ-ում: Երկու վիճակներ, որոնց ելքերը տարբեր են, համարժեք չեն և դրանց համապատասխանող վանդակը նշված է խաչով (X): Հավասար ելքերով վիճակներով որոշվող վանդակների առաջին սյունում նշվում է հորիզոնական ուղղությամբ գրված մերկա վիճակի հաջորդ վիճակը, իսկ երկրորդ սյունում՝ “-” նշանով բաժանված, նշվում է ուղղահայաց ուղղությամբ գրված մերկա վիճակի հաջորդ վիճակը: Օրինակ,



Ներկա վիճակ	Հաջորդ վիճակ		Ելք, Y	
	X=0	X=1	X=0	X=1
S0	S1	S2	0	0
S1	S3	S4	0	0
S2	S5	S6	0	0
S3	S0	S0	0	0
S4	S0	S0	1	0
S5	S0	S0	0	0
S6	S0	S0	1	0

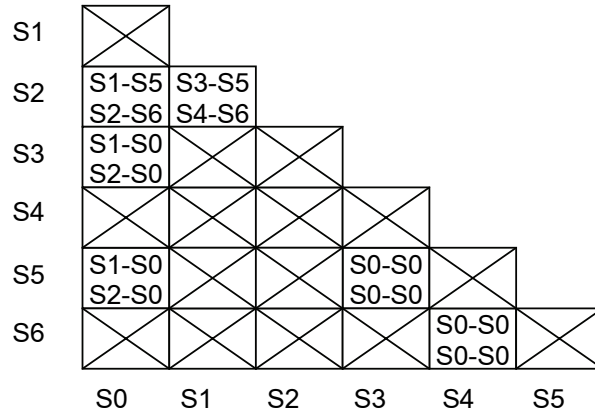
(գ)

Նկ. 4. 16. (ա) չլրացված իմպլիկացիաների քարտը յոթ վիճակների դեպքում, (բ) իմպլիկացիաների քարտը առաջին նշումից հետո, (գ) վիճակների աղյուսակը (ցույց է տրված հարմարության համար)

դիտարկենք հորիզոնական ուղղությամբ S1 և ուղղահայաց ուղղությամբ S2 վիճակներով որոշվող վանդակը: S1 ներկա վիճակի հաջորդ վիճակներ են S3 (երբ  $x=0$ ) և S4 (երբ  $x=1$ ), դրանք գրված են (S1; S2) վանդակի առաջին սյունում: S2 ներկա վիճակի հաջորդ վիճակներ են S4 (երբ  $x=0$ ) և S6 (երբ  $x=1$ ), դրանք գրված են (S1; S2) վանդակի երկրորդ սյունում: Նույն եղանակով լրացվում են քարտի մնացած բոլոր վանդակները: Նկատենք, որ (S4, S6) վիճակները համարվում են անկախ որևէ այլ զույգի համարժեքության ենթադրությունից, քանի որ դրանք ունեն նույն հաջորդ վիճակները՝ (S0, S0): Նույն ձևով համարվում են նաև (S3, S5) վիճակները: Որոշ վանդակներ պետք է հետազոտվեն հաջորդ քայլերում, որպեսզի պարզվի դրանք որոշող վիճակների համարժեքությունը:

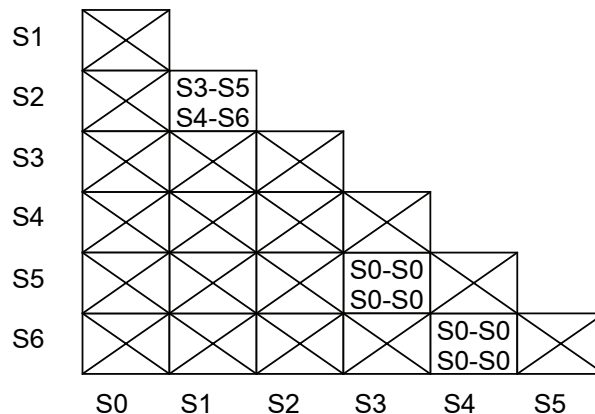
Հաջորդ քայլում հաջորդաբար պետք է ստուգել քարտի վանդակները, որպեսզի գտնվեն այն վանդակները, որոնք պետք է նշվեն խաչով: Որևէ վանդակ նշվում է խաչով, եթե այն պարունակում է թեկուզ մեկ ենթադրվող զույգ, որի վիճակները համարվում են

չեն: Օրինակ, S0 և S1-ով որոշվող վանդակը նշվում է խաչով (նկ. 4.17), քանի որ այդ վանդակում գրված (S2-S4) զույգով որոշվող վանդակը խաչ է պարունակում:



Նկ. 4.17. Ինպլիկացիաների քարտը երկրորդ քայլից հետո

Քարտի վանդակների այս ստուգումները պետք է կրկնվեն, քանի դեռ նոր վանդակներ են նշվում խաչերով (նկ. 4.18): Վերջնական քարտում խաչով չնշված վանդակները որոշում են համարժեք վիճակները: Այս օրինակում համարժեք վիճակներն են՝ S1=S2, S3=S5, S4=S6. Ինպլիկացիաների քարտի վերջնական տեսքը ցույց է տրված նկ. 4.18-ում:



Նկ. 4.18. Ինպլիկացիաների քարտի վերջնական տեսքը

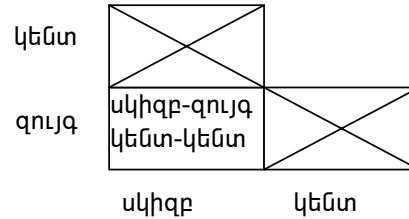
Հետ գնալով սկզբնական անցումների աղյուսակին (աղյուսակ 4.25), փոխարինելով համարժեք վիճակները՝ S1=S2; S3=S5; S4=S6, կստանանք նվազարկված վիճակներով ավտոմատի անցումների աղյուսակը (աղյուսակ 4.26):

Աղյուսակ 4.26

Ներկա վիճակ	Հաջորդ վիճակ		Ելք, Y	
	X=0	X=1	X=0	X=1
S0	S1	S1	0	0
S1	S3	S4	0	0
S3	S0	S0	0	0
S4	S0	S0	1	0

Իմպլիկացիաների քարտերի եղանակը ալգորիթմական է. այն աշխատում է բոլոր դեպքերում: Նույն ներկա և հաջորդ վիճակ ունեցող տողերը, որոնք տողերի համեմատման մեթոդում չեն ստուգվում համարժեք վիճակների հայտնաբերման համար, իմպլիկացիաների քարտերի եղանակում որևէ դժվարություն չի ներկայացնում: Օրինակ, կիրառելով իմպլիկացիաների քարտը երկուական հաջորդականությունում զույգության հայտանիշն ստուգող ավտոմատի անցումների աղյուսակի (աղյուսակ 4.23) նկատմամբ (նկ.4.19) կարելի է տեսնել, որ «Սկիզբ» և «Ձույգ» վիճակները համարժեք են:

Ներկա վիճակ	Հաջորդ վիճակ		Ելք
	X=0	X=1	
սկիզբ	սկիզբ	կենտ	0
կենտ	կենտ	զույգ	1
զույգ	զույգ	կենտ	0

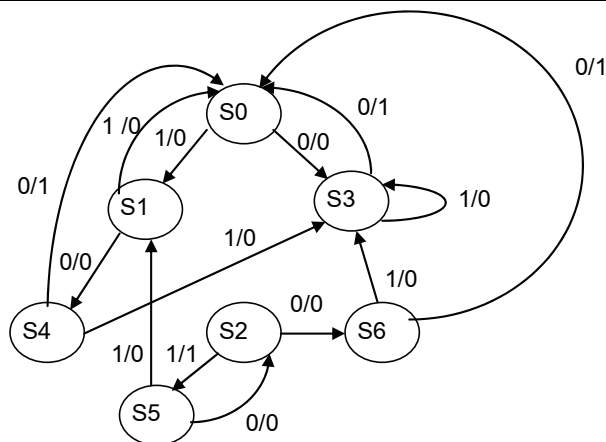


Նկ. 4.19. «Սկիզբ» սկզբնական վիճակով երկուական հաջորդականության զույգության հայտանիշը ստուգող Մուրի ավտոմատի անցումների աղյուսակը և իմպլիկացիաների քարտը

Դիտարկենք իմպլիկացիաների քարտերի մեթոդի կիրառության ևս մեկ օրինակ: **Օրինակ 4.7.** Դիտարկենք աղյուսակ 4.27-ով և նկ. 4.20-ի գրաֆով տրված ավտոմատը: Կարելի է տեսնել, որ S3-ը սպասող վիճակ է՝ աղյուսակում S3 տողում x=1 դեպքում ներկա և հաջորդ վիճակները նույնն են (S3):

Աղյուսակ 4.27

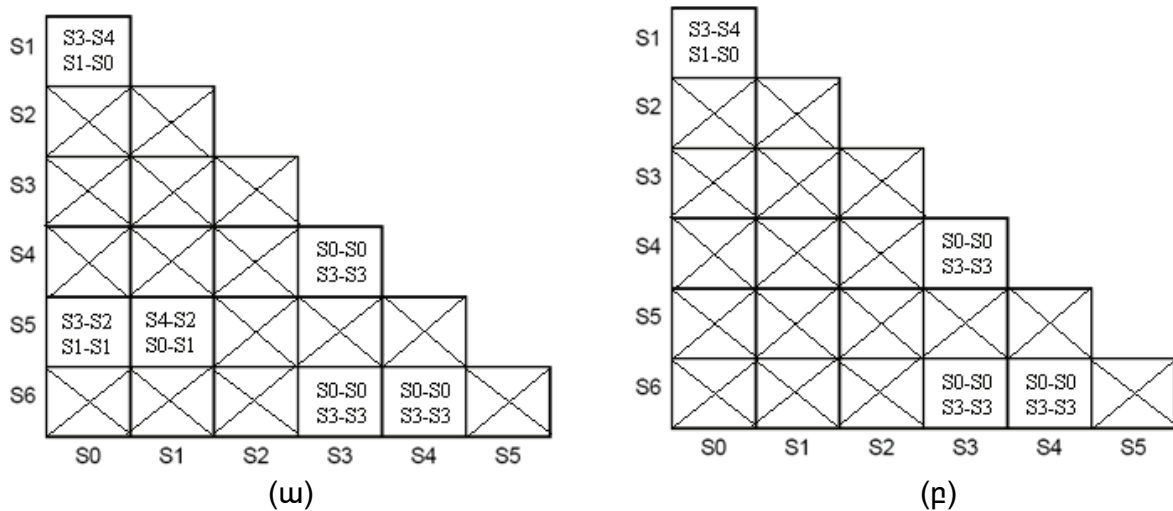
Ներկա վիճակ	Հաջորդ վիճակ		Ելք, Y	
	X=0	X=1	X=0	X=1
S0	S3	S1	0	0
S1	S4	S0	0	0
S2	S6	S5	0	1
S3	S0	S3	1	0
S4	S0	S3	1	0
S5	S2	S1	0	0
S6	S0	S3	1	0



Նկ. 4.20. Օրինակ 4.7-ի Միլի ավտոմատի սկզբնական գրաֆը

Նկ. 4.21-ում ցույց է տրված իմպլիկացիաների քարտը առաջին անցումից հետո (ա) և երկրորդ անցումից հետո՝ վերջնական քարտը (բ):

Առաջին անցման ժամանակ քարտը նշվում է ըստ անցումների աղյուսակի՝ նույն ելքերն ունեցող որոշվող վանդակներում գրվում են հաջորդ վիճակները, տարբեր ելքեր ունեցող վիճակներով որոշվող վանդակները նշվում են խաչով: Երկրորդ քայլում հաջորդաբար անցնելով չխաչված վանդակներով՝ ստուգվում է՝ արդյոք վանդակը պետք է խաչվի: Օրինակ, S0 և S5-ով որոշվող վանդակը նկ. 4.21բ-ում խաչվում է, քանի որ նրանում նկ. 4.19ա-ում գրված (S3-S2) զույգով որոշվող վանդակը խաչված է: Վերջնական քարտում չխաչված վանդակները որոշում են համարժեք վիճակների զույգերը՝ (S0, S1), (S3, S4), (S3, S6), (S4, S6):



Նկ. 4.21. Օրինակ 4.7-ի իմպլիկացիաների քարտը առաջին անցումից հետո (ա) և երկրորդ անցումից հետո՝ վերջնական քարտը (բ)

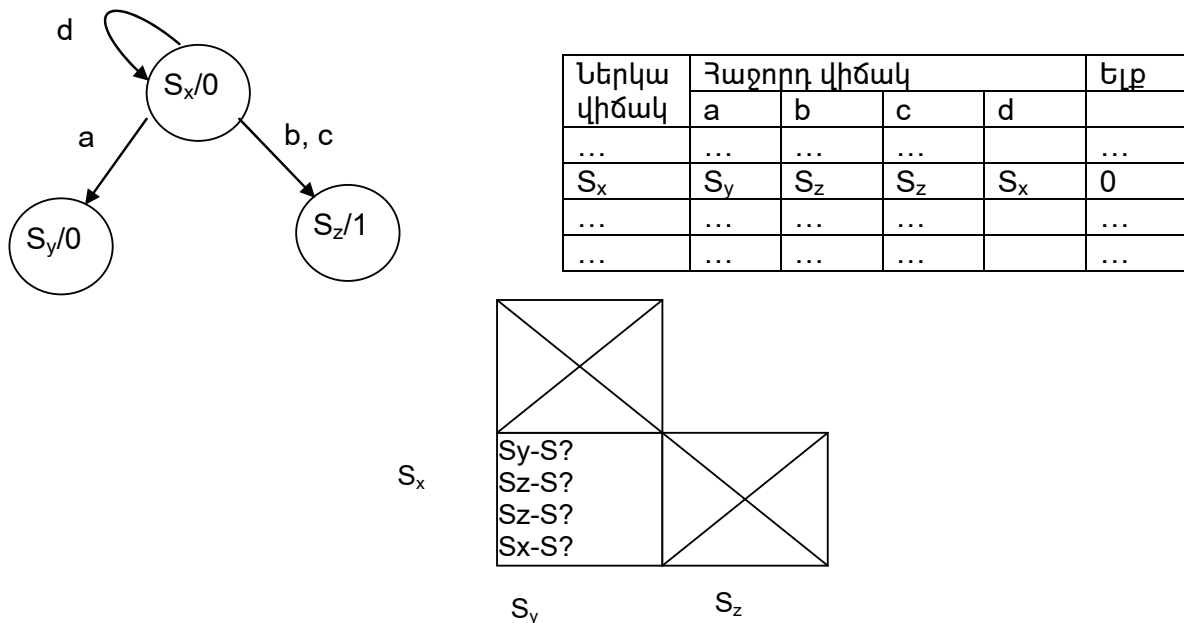
Այժմ կարող ենք վիճակները միավորել ավելի մեծ թվով համարժեք վիճակներով խմբերով: Վերջին երեք խմբերը կարող են միավորվել երեք համարժեք վիճակներով մեկ խմբում՝ (S3, S4, S6), քանի որ այս վիճակներից յուրաքանչյուրը համարժեք է մնացած երկուսին: Համարժեք վիճակների վերջնական խմբերը կլինեն՝ (S0, S1), (S2), (S3, S4, S6), (S5): Հետևաբար յոթ վիճակներով անցումների՝ աղյուսակ 4.27-ը կարող է բերվել ընդամենը չորս վիճակներով անցումների՝ աղյուսակ 4.28-ին: Նվազարկված աղյուսակում S1-ը փոխարինվել է S0-ով, իսկ S4, S6 վիճակները՝ S3-ով:

Աղյուսակ 4.28

Ներկա վիճակ	Հաջորդ վիճակ		Ելք, Y	
	X=0	X=1	X=0	X=1
S0	S3	S0	0	0
S2	S3	S5	0	1
S3	S0	S3	1	0
S5	S2	S0	0	0



Բազմաթիվ մուտքային սիմվոլներով ավտոմատի վիճակների իմպլիկացիաների քարտի վանդակները պարունակում են այնքան հաջորդ վիճակների զույգեր, որքան մուտքային սիմվոլներ ունի ավտոմատը: Այսինքն, եթե ավտոմատն ունի  $N$  մուտքային սիմվոլներ, իմպլիկացիաների քարտի վանդակները կպարունակեն  $N$  վիճակների զույգեր: Նկ. 4.22-ում ցույց են տրված բազմաթիվ մուտքային սիմվոլներով ավտոմատի անցումների գրաֆի, անցումների աղյուսակի և իմպլիկացիաների քարտի հատվածներ:



Նկ. 4.22. Բազմաթիվ մուտքային սիմվոլներով ավտոմատի գրաֆի, անցումների աղյուսակի և իմպլիկացիաների քարտի հատվածներ

#### 4.6. Կառուցվածքային ավտոմատ. վիճակների, մուտքերի և ելքերի կոդավորում

Աբստրակտ ավտոմատը հաջորդական գործողության տրամաբանական սարքավորման մաթեմատիկական նկարագրությունն է: Ավտոմատի ֆիզիկական իրականացումը հիմնվում է թվային շղթաների վրա, որոնց մուտքային և ելքային ազդանշանները երկուական են: Հիշողությամբ ավտոմատում պետք է ունենալ նաև հիշողության տարրեր, որոնք պետք է հիշեն ավտոմատի ընթացիկ վիճակը: Հաջորդական տրամաբանական շղթաներում օգտագործվող հիշողության տարրերը տրիգերներն են, որոնք ունակ են պահպանել մեկ բիթ ինֆորմացիա: Տրիգերն ունի երկու վիճակներ, որոնք նշվում են 0 և 1: Տրիգերը սովորաբար ունի երկու ելքեր՝ մեկը նորմալ, որի արժեքը համընկնում է վիճակի հետ, իսկ մյուսը՝ ժխտված: Այսպիսով, (4.2) աբստրակտ ավտոմատից կառուցվածքային ավտոմատի (ֆիզիկական իրականացման) անցնելու համար պետք է կոդավորել մուտքային սիմվոլների  $I = \{i_1, i_2, \dots, i_N\}$  բազմությունը, ելքային սիմվոլների  $O = \{o_1, o_2, \dots, o_M\}$  բազմությունը և վիճակների  $S = \{s_0, s_1, \dots, s_R\}$  բազմությունը երկուական փոփոխականներով:

R վիճակների կոդավորման համար պետք է ունենալ

$$r = \log_2 R \quad (4.8)$$

երկուական փոփոխականներ՝  $Q_1, \dots, Q_r$ :

N մուտքային սինվոլների կոդավորման համար պետք է ունենալ

$$n = \log_2 N \quad (4.9)$$

երկուական փոփոխականներ՝  $x_1, \dots, x_n$ .

M ելքային սինվոլների կոդավորման համար պետք է ունենալ

$$m = \log_2 M \quad (4.10)$$

երկուական փոփոխականներ՝  $y_1, \dots, y_m$ .

Երկուական կոդավորված մուտքերով, ելքերով և վիճակներով ավտոմատի անցումների և ելքերի ֆունկցիաները ներկայացվում են հավասարումների հետևյալ համակարգերով՝

$$y_i = f_i(Q_r, Q_{r-1}, \dots, Q_1; x_n, x_{n-1}, \dots, x_1), \quad i=1, 2, \dots, r, \quad (4.11)$$

$$Q_j^+ = \varphi_j(Q_r, Q_{r-1}, \dots, Q_1; x_n, x_{n-1}, \dots, x_1), \quad j=1, 2, \dots, m, \quad (4.12)$$

որտեղ  $f_i$  և  $\varphi_j$  բուլյան ֆունկցիաներ են՝ կախված վիճակների և մուտքերի երկուական փոփոխականներից,  $Q$ -ն ներկա վիճակն է,  $Q^+$ -ը հաջորդ վիճակն է: (4.11)-ը ելքերի ֆունկցիաների համակարգն է, (4.12)-ը վիճակների հավասարումների համակարգն է: Վիճակների ընթացիկ արժեքները պետք է պահպանվեն հիշողության տարրերում, որպեսզի դրանք առկա լինեն (4.12)-ով հաջորդ վիճակի հաշվման ժամանակ:

Ելքային ֆունկցիաների (4.11) համակարգը համապատասխանում է Միլի ավտոմատի մոդելին: Մուրի մոդելի դեպքում ելքային փոփոխականները ֆունկցիա են միայն վիճակի փոփոխականներից և տրվում են հետևյալ համակարգով՝

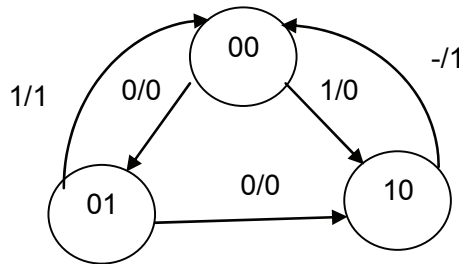
$$y_i = f_i(Q_r, Q_{r-1}, \dots, Q_1), \quad i=1, 2, \dots, r: \quad (4.13)$$

**Օրինակ 4.8.** Կոդավորել աղյուսակ 4.18-ով տրված վաճառող մեքենայի կառավարող ավտոմատը:

Սկզբում կոդավորենք վիճակները՝  $S_0, S_1, S_2$ : Երեք վիճակների կոդավորման համար անհրաժեշտ է ունենալ երկու փոփոխական՝  $Q_1, Q_2$ : Վիճակներին կոդերը վերագրվում են հետևյալ կարգով  $S_0=00, S_1=01, S_2=01$ : Այստեղ առկա է ևս մեկ հավաքածու՝ 11, որն ավելցուկային է՝ այստեղ չի օգտագործվում: Մուտքային  $N$  և  $D$  երկու սինվոլները կարող են կոդավորվել մեկ երկուական փոփոխականով՝  $x$ : Մուտքային փոփոխականին անհրաժեշտ արժեքները վերագրվում են հատկապես կարգով՝ “N”  $\rightarrow x=0$ ,  $D \rightarrow x=1$ : Ելքային  $O$  և  $C$  երկու սինվոլները կոդավորվում են մեկ երկուական փոփոխականով՝  $y$ , “O”  $\rightarrow y=1$ , “C”  $\rightarrow y=0$ : Կոդավորված փոփոխականներով կառուցվածքային ավտոմատի անցումների աղյուսակը ցույց է տրված աղյուսակ 4.29-ում, իսկ գրաֆը՝ նկ. 4.23-ում: Գրաֆում 10 վիճակից 00 վիճակ տանող աղեղը նշված է -/1, որը նշանակում է, որ այդ անցումը տեղի ունի մուտքային  $x$  փոփոխականի ցանկացած արժեքի դեպքում:

Աղյուսակ 4.29

Ներկա վիճակ $Q_2Q_1$	Հաջորդ վիճակ, $Q_2^+Q_1^+$		Ելք, $y$	
	$x=0$	$x=1$	$x=0$	$x=1$
00	01	10	0	0
01	10	00	0	1
10	00	00	1	1



Նկ. 4.23. Վաճառող ավտոմատի կողավորված փոփոխականներով գրաֆը

Անցումների աղյուսակում վիճակի փոփոխականների համակցություններում  $Q_1$  փոփոխականը նշված է ցածր կարգում, իսկ  $Q_2$ -ը՝ բարձր կարգում:  $Q_2Q_1=11$  հավաքածուն ավելցուկային է, որից ոչինչ կախված չէ (a don't care condition): Վիճակի փոփոխականները իրականացվում են տրիգերների միջոցով՝ դրանք տրիգերների նորմալ ելքերն են: Օրինակ, երբ ավտոմատը 00 վիճակից անցնում է 01 վիճակին,  $Q_2$  ելքով տրիգերի ելքը մնում է 0 վիճակում, իսկ  $Q_1$  ելքով տրիգերի ելքը 0-ից անցնում է 1 վիճակ:

Ելքային և վիճակի բուլյան ֆունկցիաների բանաձևերը կարելի է հեշտությամբ ստանալ անցումների աղյուսակից՝ ճիշտ նույն եղանակով, ինչպես դա արվում է Կառնոյի քարտից: Դիտարկելով  $y$ -ը իբրև բուլյան ֆունկցիա ներկա վիճակի ( $Q_2Q_1$ ) և մուտքի ( $x$ ) փոփոխականներից, ելքային ֆունկցիան կարելի է գրել հետևյալ ձևով՝

$$y = \overline{Q_2}Q_1x + Q_2\overline{Q_1}\bar{x} + Q_2\overline{Q_1}x = \overline{Q_2}Q_1x + Q_2\overline{Q_1}:$$

“Հաջորդ վիճակ” սյուններում գրված են ( $Q_2^+Q_1^+$ ) գույգի արժեքները: Սկզբում ընտրելով  $Q_2^+$ -ի արժեքները, այնուհետև՝  $Q_1^+$ -ինը, կարող ենք դուրս բերել հետևյալ հավասարումները՝

$$Q_2^+ = \overline{Q_2}Q_1\bar{x} + \overline{Q_2}\overline{Q_1}x,$$

$$Q_1^+ = \overline{Q_2}\overline{Q_1}\bar{x}:$$

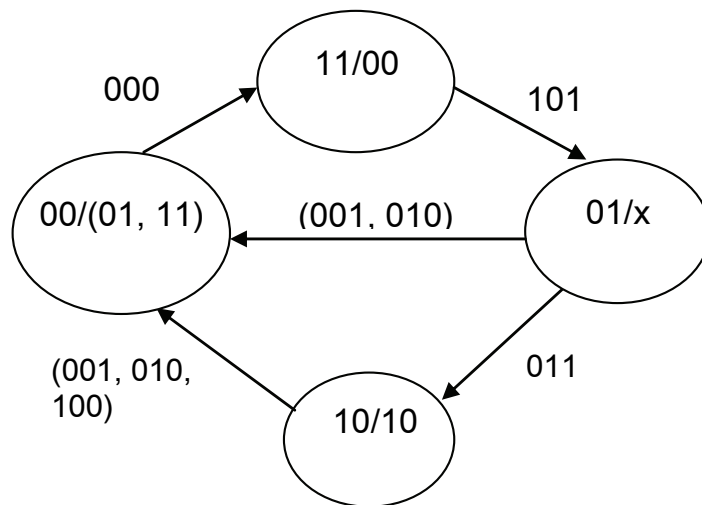
**Օրինակ 4.9.** Կողավորել աղյուսակ 4.6-ով տրված ավտոմոբիլի ամրագոտիների վիճակի հսկման Մուրի ավտոմատը:

$s_0, s_1, s_2, s_3$  չորս վիճակների կողավորման համար անհրաժեշտ է ունենալ երկու փոփոխական՝  $Q_1, Q_2$ : Վիճակներին կոդերը վերագրվում են հետևյալ կարգով՝  $s_0=00, s_1=01, s_2=01, s_3=11$ : Մուտքային  $i_1, \dots, i_6$  վեց սինվոլները կարող են կողավորվել երեք երկուական փոփոխականով՝  $x_1, x_2, x_3$ : Մուտքային փոփոխականին անհրաժեշտ արժեքները վերագրվում են հետևյալ կարգով՝  $i_1=000, i_2=001, i_3=010, i_4=011, i_5=100, i_6=101$ : Ելքային  $o_1, o_2, o_3, o_4$  չորս սինվոլները կողավորվում են երկու երկուա-

կան փոփոխականով՝  $y_1, y_2$   $o_1=00, o_2=01, o_3=10, o_4=11$ : Կոդավորված փոփոխականներով կառուցվածքային ավտոմատի անցումների աղյուսակը ցույց է տրված աղյուսակ 4.30-ում, իսկ գրաֆը՝ նկ. 4.24-ում: Աղյուսակում  $x$ -ով նշված են գոյություն չունեցող անցումները:

Աղյուսակ 4.30

Ներկա վիճակը	Հաջորդ վիճակը						Ելք
	000	001	010	011	100	101	
00	11	x	x	x	x	x	01, 11
01	x	00	00	10	x	x	x
10	x	00	00	x	00	x	10
11	x	x	x	x	x	01	00



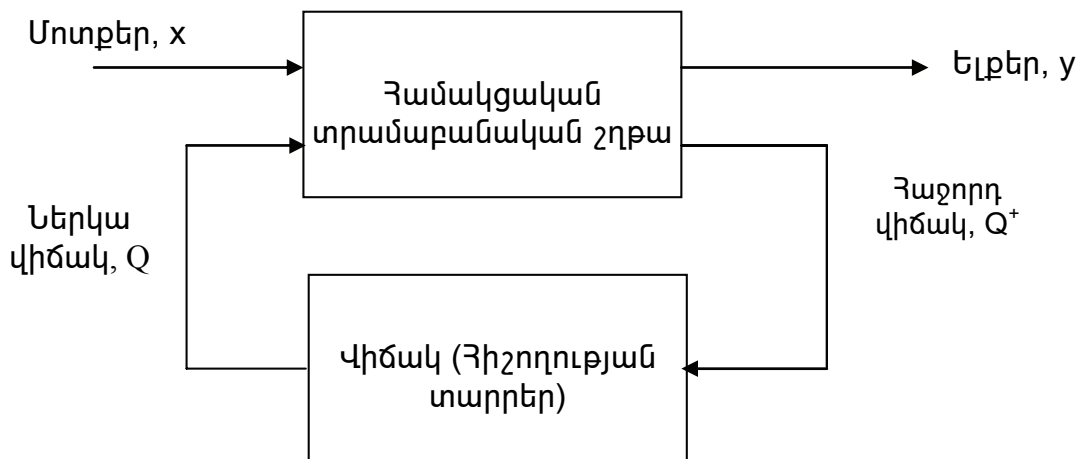
Նկ. 4.24. Ավտոմոբիլի անվտանգության գոտիների վիճակի հսկման ավտոմատի կոդավորված փոփոխականներով գրաֆը

#### 4.7. Հաջորդական տրամաբանական շղթաներ

Հաջորդական տրամաբանական շղթայի ընդհանրացված բլոկ-սխեման ցույց է տրված նկ. 4.25-ում: Այն կազմված է համակցական շղթայից, որին միացված հիշողության տարրերը կազմում են հետադարձ կապի օղակ: Համակցական շղթան իրականացնում է (4.11) և (4.11) ֆունկցիաները: Հիշողության տարրերը տրիգերներ են, որոնք պահպանում են ընթացիկ վիճակի փոփոխականները: Ժամանակի տրված պահին տրիգերներում պահպանվող երկուսական տվյալները՝ տրիգերների ելքերը, որոշում են ավտոմատի վիճակը:

Հաջորդական գործողության թվային շղթաները բաժանվում են երկու տիպերի՝ սինքրոն և ասինքրոն: Դրանք դասակարգվում են ըստ ազդանշանների ժամանակային համաձայնեցման: Սինքրոն թվային շղթայի վարքագիծը որոշվում է ազդանշանի արժեքներով ժամանակի որոշակի ընդհատ պահերին: Ասինքրոն թվային շղթայի վարքագիծը որոշվում է մուտքայի ազդանշանների փոփոխություններով և կարող է փոխվել ժամանակի ցանկացած պահին:

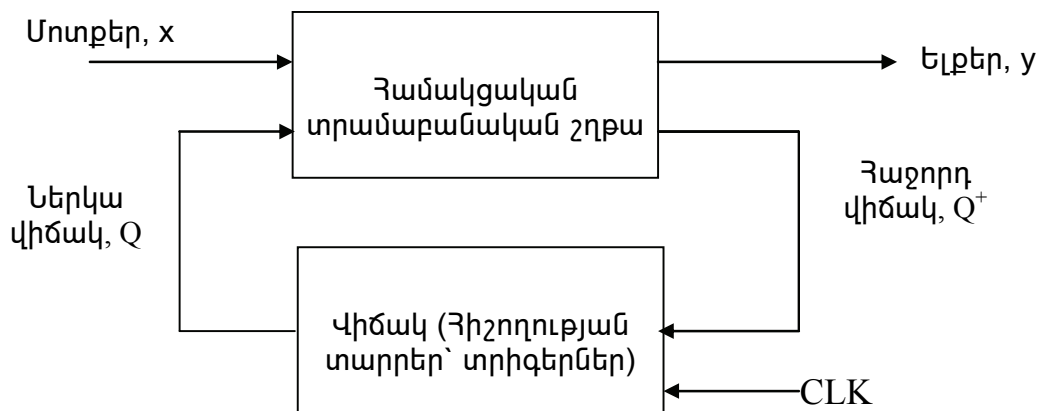
Ասինքրոն թվային շղթաներում՝ որպես հիշողության տարրեր, օգտագործվում են ասինքրոն տրիգերներ (սևեռիչներ), որոնք ասինքրոն հապաղման շղթաներ են: Որևէ հապաղման շղթայի հիշողության հատկությունը պայմանավորված է այն հանգամանքով, որ նրանով ազդանշանը տարածվում է որոշակի վերջավոր ժամանակում: Գործնականում տրամաբանական տարրերի հապաղման ժամանակները կարող են բավարար լինել անհրաժեշտ հապաղումը ձևավորելու համար: Այդ դեպքում հատուկ հապաղման տարրերի անհրաժեշտություն չկա: Տրամաբանական տարրերի վրա կառուցված ասինքրոն շղթայում (նկ. 4.25) հիշողության տարրերը բաղկացած են տրամաբանական տարրերից: Հետևաբար, ասինքրոն թվային շղթան կարելի է ներկայացնել հետադարձ կապերով համակցական շղթայի տեսքով: Տրամաբանական տարրերի միջև հետադարձ կապի պատճառով ասինքրոն շղթայի աշխատանքը կարող է դառնալ անկայուն: Անկայունության խնդիրը առաջացնում է նախագծման շատ բարդություններ:



Նկ. 4.25. Ասինքրոն հաջորդական թվային համակարգ

Սինքրոն թվային համակարգը (նկ. 4.26), ըստ սահմանման, պետք է օգտագործի տակտային ազդանշաններ, որոնք ազդում են հիշողության տարրերի վրա միայն ժամանակի ընդհատ պահերի: Գործնականում սինքրոն համակարգերում գործողությունների համաժամանակեցումը (սինքրոնացումը) իրականացվում է պարբերական տակտային իմպուլսների միջոցով՝ CLK, որոնք գեներացվում են հատուկ իմպուլսների գեներատորի միջոցով: Հիշողության տարրերի վրա դրանց մուտքային ազդանշանները ներգործում են միայն տակտային իմպուլսի առկայության դեպքում: Հետևաբար, համակարգի վիճակի փոփոխությունը ըստ (4.12) հավասարման տեղի ունի միայն տակտային իմպուլսի կիրառման պահին:

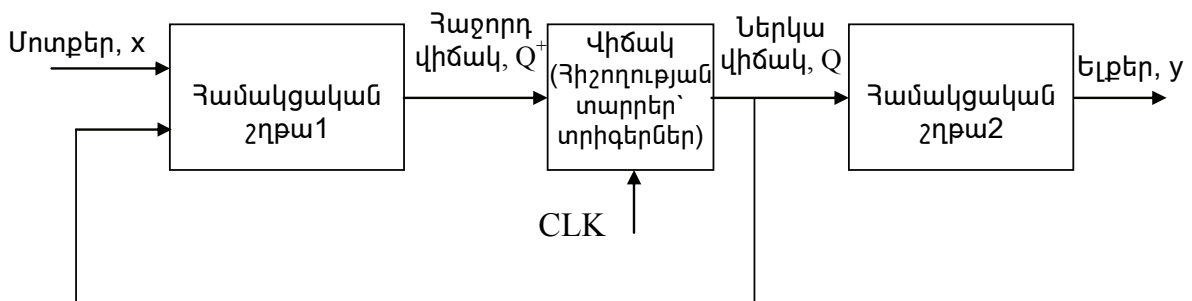
Սինքրոն թվային համակարգերը, որոնցում հիշողության տարրերի (տակտավորվող տրիգերների) մուտքերում օգտագործվում են տակտային իմպուլսներ, կոչվում են տակտավորվող թվային համակարգեր: Դրանք անկայունություն չեն ցուցաբերում, և դրանց աշխատանքը կարելի է բաժանել անկախ ընդհատ քայլերի: Հետագայում կուսումնասիրվեն միայն տակտավորվող թվային համակարգեր:



Նկ. 4.26. Սինթրոն թվային համակարգի բլոկ-սխեման (Միլի ավտոմատ)

Նկ. 4.26-ի սինթրոն թվային համակարգում համակցական տրամաբանական շղթան ձևավորում է հաջորդ վիճակի փոփոխականներն ու ելքային փոփոխականները՝ օգտագործելով ներկա վիճակի և մուտքային փոփոխականները: Սա համապատասխանում է Միլի ավտոմատի մոդելին:

Մուրի ավտոմատի դեպքում (Նկ. 4.27) հաջորդ վիճակի փոփոխականները ձևավորվում են ներկա վիճակի և մուտքային փոփոխականներից (համակցական շղթա1), իսկ ելքային փոփոխականները որոշվում են միայն ներկա վիճակի փոփոխականներով (համակցական շղթա2): Կարելի է տեսնել, որ Մուրի ավտոմատում (Նկ. 4.27) ելքերը մեկ տակտով ավելի ուշ են ձևավորվում, քան Միլի ավտոմատում (Նկ. 4.26):



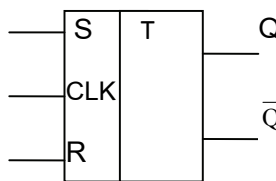
Նկ. 4.27. Սինթրոն թվային համակարգի բլոկ-սխեման (Մուրի ավտոմատ)

#### 4.8. Տրիգերների հիմնական տիպերը

Տակտավորվող թվային համակարգերում օգտագործվող հիշողության տարրերը սևեռիչներն են և տրիգերները: Այս տարրերը երկուսական բջիջներ են, որոնք ունեն երկու վիճակներ՝ 0 և 1: Տրիգերը կարող է պահպանել վիճակը անորոշ երկար ժամանակ (քանի դեռ սնման լարումը միացված է), մինչև որ տակտային իմպուլսի տրման պահին մուտքային ազդանշանի ազդեցությամբ փոխի վիճակը: Տրիգերների տարբեր տիպեր միմյանցից տարբերվում են մուտքային ազդանշանների թվով և տրիգերի վիճակի վրա դրանց ներգործության տրամաբանությամբ: Տրիգերը (սևեռիչը) տակտավորվում է տակտային ազդանշանի ակտիվ մակարդակով, որը կարող է լինել իմպուլսի ցածր լարման մակարդակը կամ բարձր լարման մակարդակը:

Տակտավորվող RS տրիգերի (սևեռիչի) գրաֆիկական սիմվոլը անցումների աղյուսակը և հաջորդ վիճակի ֆունկցիայի Կառնոյի քարտը ցույց են տրված նկ. 4.28-ում: Տրիգերի վիճակը մնում է անփոփոխ այնքան ժամանակ, քանի դեռ տակտային ազդանշանն ունի լարման ցածր մակարդակ՝ CLK=0, անկախ S (set, 1 վիճակի կարգում) և R (reset, 0 վիճակի կարգում) մուտքերի արժեքներից: Երբ CLK=1, S և R մուտքերի ազդանշաններն ազդում են տրիգերի ներքին շղթաների վրա և կարող են փոխել վիճակը: Տրիգերը կարգվում է 1 վիճակ, երբ CLK=1 և S=1, R=0: Տրիգերը դրվում է 0 վիճակ, երբ CLK=1 և S=0, R=1: Տրիգերը վիճակը չի փոխում, երբ CLK=1 և S=0, R=0: Իսկ երբ S=1 և R=1, տակտային իմպուլսի ակտիվ մակարդակի՝ CLK=1 տրման պահին, տրիգերի երկու ելքերն էլ գնում են 1 վիճակ: Երբ տակտային իմպուլսն անցնում է CLK=0 արժեքին, տրիգերի ելքերի վիճակները դառնում են անորոշ: Այդ պատճառով R=S=1 մուտքային հավաքածուն համարվում է արգելված: RS տրիգերի աշխատանքը աղյուսակային տեսքով ցույց է տրված նկ. 4.28բ-ում՝ R և S սյուներում նշված են մուտքային ազդանշանների բոլոր հնարավոր համակցությունները, Q-ն ներկա վիճակն է,  $Q^{t+1}$ -ը հաջորդ վիճակն է տակտային իմպուլսի CLK=1 մակարդակի կիրառումից հետո: Տրիգերի անցումների ֆունկցիայի բանաձևը կարելի դուրս գրել  $Q^{t+1}$ -ի Կառնոյի քարտից (նկ. 4.28գ).

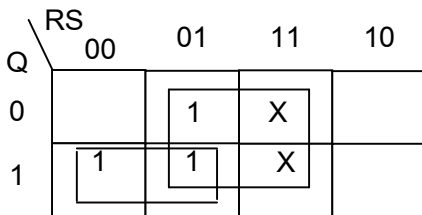
$$Q^+ = S + \bar{R}Q \quad : (4.14)$$



(ա)

R	S	Q	$Q^{t+1}$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	x
1	1	1	x

(բ)



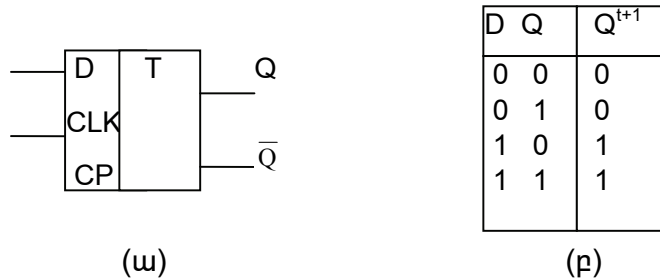
(գ)

Նկ. 4.28. Տակտավորվող RS տրիգեր (սևեռիչ). ա- գրաֆիկական սիմվոլը, բ- անցումների աղյուսակը, գ- հաջորդ վիճակի ֆունկցիայի Կարնոյի քարտը

D տրիգերի (սևեռիչի) գրաֆիկական սիմվոլը և անցումների աղյուսակը ցույց են տրված նկ. 4.29-ում: Անցումների աղյուսակից կարելի է տեսնել, որ CLK=1 մակարդակի կիրառումից հետո տրիգերի հաջորդ վիճակը համընկնում է D մուտքի արժեքի հետ՝ անկախ ներկա վիճակի արժեքից:

Անցումների հավասարումը կարելի է ստանալ անցումների աղյուսակից՝

$$Q^+ = D: \quad (4.15)$$



Նկ. 4.29. Տակտավորվող D տրիգեր (սևեռիչ). ա- գրաֆիկական սիմվոլը, բ- անցումների աղյուսակը

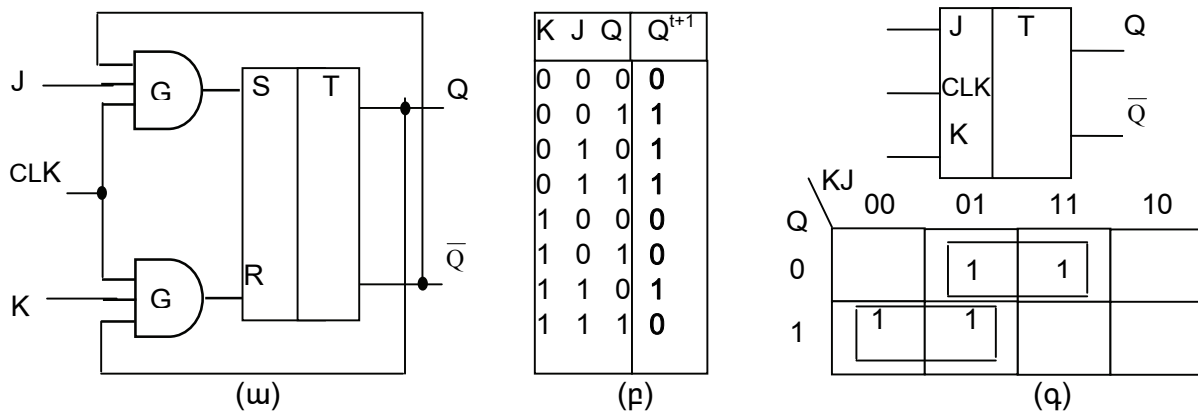
JK տրիգերը կարելի համարել RS տրիգերի կատարելագործված տարբերակն այնքանով, որ այստեղ որոշված են բոլոր վիճակները: J և K մուտքերը տրիգերը կարգում են համապատասխանաբար 1 և 0 վիճակ (ինչպես S և R մուտքերը): Երբ J և K մուտքերը միաժամանակ 1 են, ապա CLK=1 մակարդակի կիրառումից հետո տրիգերի հաջորդ վիճակը փոխանջատվում է նախորդի վիճակի ժխտմանը. երբ Q=1, ապա  $Q^{t+1}=0$ , և ընդհակառակը: Տակտավորվող JK տրիգերի կառուցվածքը ցույց է տրված նկ.4.30ա-ում: Տրիգերի Q ելքը, K մուտքը և տակտային CLK մուտքը միացված են ՅԵՎ տարրի մուտքերին, որի ելքը միացված է RS տրիգերի R մուտքին: Տրիգերն անցնում է 0 վիճակ, եթե մինչև տակտային իմպուլսի կիրառելն այն եղել էր 1 վիճակում: Նույն ձևով  $\bar{Q}$  ելքը, J մուտքը և տակտային CLK մուտքը միացված են ՅԵՎ տարրի մուտքերին, որի ելքը միացված է RS տրիգերի S մուտքին. տրիգերն անցնում է 1 վիճակ, եթե մինչև տակտային իմպուլսի կիրառելն այն եղել էր 0 վիճակում: Ինչպես երևում է նկ. 4.30բ-ում ցույց տրված անցումների աղյուսակից, JK տրիգերի աշխատանքը համընկնում է RS տրիգերին հետ, բացառությամբ J=K=1 համակցության դեպքի: Երբ J=K=1, CLK=1 իմպուլսն անցնում է ՅԵՎ տարրից միայն մեկով՝ այն ՅԵՎ տարրով, որի մուտքերից մեկը միացված է տրիգերի այն մուտքին, որը տվյալ պահին 1 է: Եթե Q=1, ապա ներքևի ՅԵՎ տարրի մուտքին հետադարձ կապով տրվում է Q=1 և ելքը շրջվում է  $Q^{t+1}=0$  վիճակ: Եթե  $\bar{Q}=1$ , ապա վերևի ՅԵՎ տարրի ելքը դառնում է 1 և տրիգերը Q=0 վիճակից անցնում է  $Q^{t+1}=1$  վիճակ: JK տրիգերի անցումների հավասարումը կարելի է դուրս գրել նկ. 4.30գ-ի Կառնոյի քարտից՝

$$Q^+ = \bar{Q}J + \bar{K}Q: \quad (4.16)$$

Նկատենք, որ երբ J=K=1 և CLK ազդանշանը մնում է 1 վիճակում, ապա հետադարձ կապերի պատճառով ելքերն անընդհատ կփոփոխվեն, այսինքն՝ տրիգերի աշխատանքն անկայուն է: Որպեսզի կանխվեն ելքերի անընդհատ փոխանջատումները, տակտային ազդանշանը պետք է ունենա տրիգերի շղթայով ազդանշանի տարածման հապաղումից կարճ տևողություն: Սա անցանկալի սահմանափակում է, քանի որ շղթայի աշխատանքի կայունությունը կախված է տակտային ազդանշանի տևողությունից: Այդ պատճառով, սովորաբար, JK տրիգերը չի կառուցվում նկ. 4.30ա-ի սխեմայով: Տակտային իմպուլսի տևողությանը ներկայացվող սահմանափակումը կարելի է հանել, եթե տրիգերը փոխանջատվի ոչ թե տակտային իմպուլսի մակարդակով, այլ



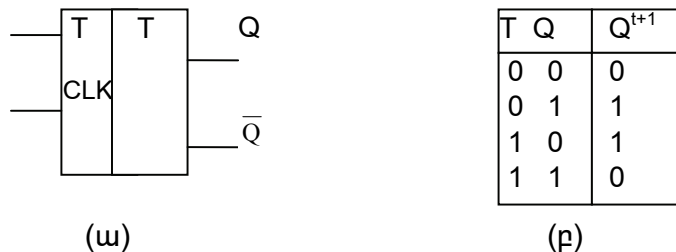
ճակատով, որը կքննարկվի հետագայում:



Նկ. 4.30. Տակտավորվող JK տրիգեր. ա- գրաֆիկական սիմվոլը, բ- անցումների աղյուսակը, գ-հաջորդ վիճակի ֆունկցիայի կարմույի քարտը

T տրիգերը JK տրիգերի մեկ մուտքով տարբերակն է: T տրիգերն ստացվում է JK տրիգերից՝ J և K մուտքերը միմյանց միացնելով: Անկախ տրիգերի ներկա վիճակից՝ տրիգերը փոխում է վիճակը (շրջվում է), երբ  $T=1$ : T տրիգերի գրաֆիկական սիմվոլը և անցումների աղյուսակը ցույց են տրված նկ. 4.31-ում: Տրիգերի անցումների հավասարումը ստացվում է անցումների աղյուսակից՝

$$Q^+ = \bar{Q}T + \bar{T}Q: \quad (4.17)$$



Նկ. 4.31. Տակտավորվող T տրիգեր. ա- գրաֆիկական սիմվոլը, բ- անցումների աղյուսակը

#### 4.9. Տակտավորվող տրիգերներով թվային համակարգի աշխատանքի կայունությունը

Ինչպես երևում է նկ. 4.28ա-ի սխեմայից, JK տրիգերում առկա են հետադարձ կապեր: Այս հետադարձ կապերը ստեղծում են անկայունություն՝ ելքերը փոփոխվում են, երբ տակտային ազդանշանը մնում է անփոփոխ՝  $CLK=1$  մակարդակի վրա: Տակտավորման այս թերությունը կարելի է կանխարգելել, եթե տրիգերի մուտքից ելք հապաղման չափը դարձվի տակտային իմպուլսի տևողությունից ավելի երկար: Այս դեպքում տրիգերի ելքերի փոփոխության պահին տակտային իմպուլսն արդեն վերադարձած կլինի  $CLK=0$  ոչ ակտիվ մակարդակին: Կիրառելով այս սկզբունքը նկ. 4.26-ում ներկայացված հաջորդական թվային համակարգի ընդհանուր կառուցվածքի նկատմամբ՝ կարելի է ստանալ տակտավորվող թվային համակարգի կայուն աշխատանքի պայմանը՝

$$t_{pd} > t_{pl}, \quad (4.18)$$

որտեղ  $t_{pd}$ -ն համակցական շղթայում ազդանշանի տարածման հապաղումն է,  $t_{pl}$ -ն՝ տակտային իմպուլսի ակտիվ մակարդակի տևողությունը:

Բազմաթիվ հիշողության տարրերով թվային համակարգի նկատմամբ (4.18) պայմանի կիրառումը պահանջում է, որ՝

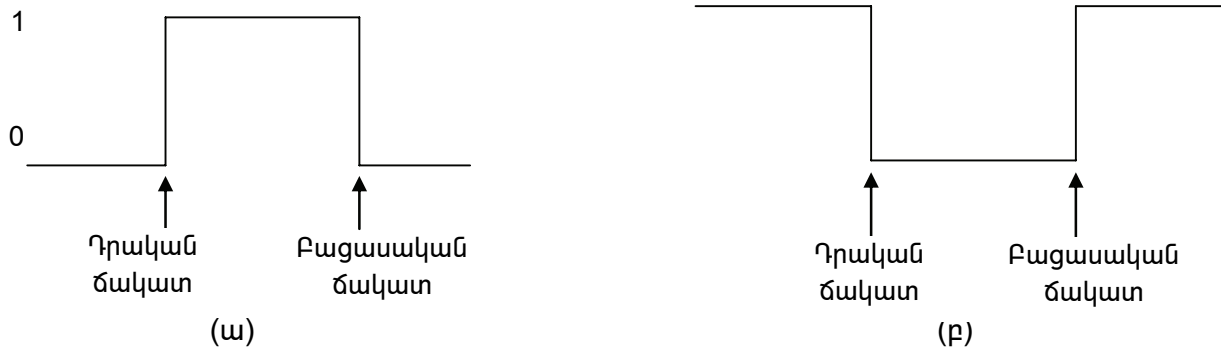
- բոլոր հիշողության տարրերի՝ տրիգերների, հուսալի փոխանջատման համար տակտային իմպուլսի պահանջվող տևողությունը պետք է  $t_{pl}$ -ից մեծ չլինի,

- թվային համակարգի համակցական շղթայի բոլոր մուտքից ելքի ուղիները պետք է ունենան  $t_{pd}$ -ից ոչ փոքր հապաղում: Այսինքն՝  $t_{pd}$ -ն ամենակարճ ուղու հապաղումն է:

Թվային համակարգի աշխատանքի կայունության ապահովման նկարագրված պայմանը կոչվում է “կարճ” համաժամանակեցման պայման՝ այն պահանջում է կարճ տակտավորման իմպուլսներ: Այս եղանակի հիմնական թերությունն այն է, որ տակտային իմպուլսի պահանջվող տևողությունը կախված է համակցական շղթայի պարամետրերից: Երբ (4.18) պայմանը խախտվում է, ասում են, որ համակարգում առկա է մրցավազք՝ տրիգերների մուտքերին հաջորդ վիճակի ազդանշանները հասնում են ավելի շուտ, քան տակտային ազդանշանը կանցներ պասիվ մակարդակի:

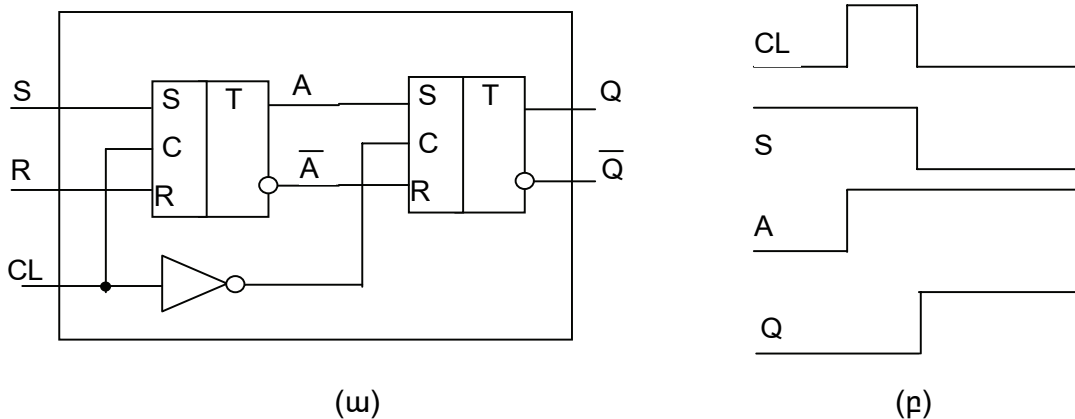
Հետադարձ կապերի առկայության դեպքում թվային համակարգի ավելի կայուն համաժամանակեցման կարելի է հասնել, եթե տրիգերների վիճակների փոփոխությունները դարձվեն զգայուն տակտային իմպուլսի ոչ թե մակարդակի, այլ ճակատի նկատմամբ: Տակտային իմպուլսը կարող է լինել ակտիվ ‘1’ մակարդակով՝ դրական իմպուլս կամ ակտիվ ‘0’ մակարդակով՝ բացասական իմպուլս: Դրական իմպուլսի դեպքում  $CLK=1$  իմպուլսի առկայության ժամանակ,  $CLK=0$  իմպուլսների միջև դադարի տևողություններում: Իմպուլսի մակարդակը փոխվում է՝ կատարելով 0-ից 1 կամ 1-ից 0 անցում: Ինչպես ցույց է տրված նկ. 4.32-ում, 0-ից 1 անցումը սահմանվում է որպես դրական անցում կամ դրական ճակատ, 1-ից 0 անցումը՝ բացասական անցում կամ բացասական ճակատ:

Տակտային ազդանշանի մակարդակի նկատմամբ զգայուն հիշողության տարրը (տրիգերը) կոչվում է նաև սևեռիչ (latch), իսկ տակտային ազդանշանի ճակատի նկատմամբ զգայուն տարրը ուղղակի կոչվում է տրիգեր (flip-flop, FF): Հայտնի են ճակատով տակտավորվող տրիգերների կառուցման շատ տարբերակներ: Ներկայումս թվային համակարգերում հիմնականում օգտագործվում է երկաստիճան կամ տեր-ժառա տրիգեր (master-slave, MS):



Նկ. 4.32. Տակտային իմպուլսի անցումների սահմանումները. (ա) դրական իմպուլս (բ) բացասական իմպուլս

Տեր-ծառա երկաստիճան տրիգերը բաղկացած է երկու առանձին տրիգերներից: RS տեր-ծառա տրիգերի տրամաբանական կառուցվածքը ցույց է տրված նկ. 4.33-ում: Այն բաղկացած է տեր տրիգերից, ծառա տրիգերից և մեկ շրջիչից: Երբ  $CLK = 0$ , շրջիչի ելքը հավասար է 1-ի, որը կիրառված է ծառա տրիգերի տակտային մուտքին: Հետևաբար, ծառա տրիգերի ելքերը որոշվում են նրա մուտքերով՝  $Q=A$ ,  $\bar{Q}=\bar{A}$ : Իսկ տեր տրիգերի աշխատանքն արգելված է  $CLK=0$  ազդանշանով: Երբ  $CLK = 1$ , արտաքինից տրվող մուտքային R և S ազդանշաններն ազդում են տեր տրիգերի A և  $\bar{A}$  ելքերի վրա: Իսկ ծառա տրիգերի մուտքերն արգելված են շրջիչի ելքի 0 ազդանշանով: Երբ տակտային ազդանշանը վերադառնում է 0 մակարդակ, տեր տրիգերը նորից մեկուսացվում է, և արտաքին մուտքերը չեն կարող ազդել տրիգերի վրա: Ծառա տրիգերը անցնում է նույն վիճակը, ինչ տեր տրիգերը:



Նկ. 4.33. Տեր-ծառա RS տրիգեր: (ա) տրամաբանական սխեման, (բ) ժամանակային դիագրամները

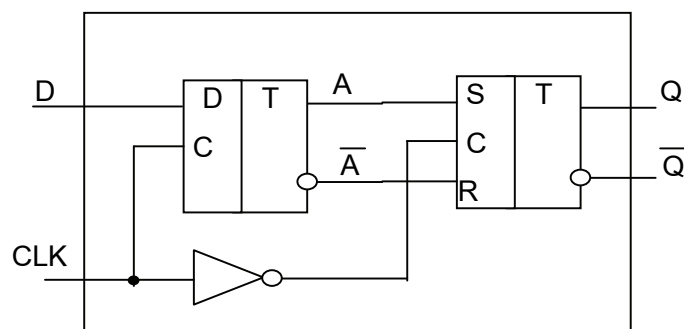
Նկ. 4.33բ-ում ներկայացված ժամանակային դիագրամները ցույց են տալիս տեր-ծառա տրիգերում տեղի ունեցող պատահարների հաջորդականությունը: Ենթադրենք մինչև տակտային իմպուլսի գալը տրիգերը գտնվում է 0 վիճակում՝  $A=0$  և  $Q=0$ : Մուտքային ազդանշաններն են  $S=1$ ,  $R=0$ : Հաջորդ տակտային իմպուլսը պետք է տրիգերը դնի  $Q=1$  վիճակ: Տակտային իմպուլսի 0-ից 1 անցման ժամանակ տեր տրիգերը գործում

է, և նրա վիճակը փոխվում է  $A=1$ : Ծառա աստիճանի տրիգերի վիճակը չի փոխվի՝ նրա տակտային մուտքին կիրառված է  $\overline{CLK} = 0$ : Քանի որ տեր տրիգերը ներքին շղթա է, ուստի նրա ելքերի փոփոխություններն աննկատ են տեր-ծառա տրիգերի արտաքին  $Q$  և  $\bar{Q}$  ելքերում: Երբ տակտային իմպուլսը վերադառնում է 0 մակարդակի, տեր տրիգերի վիճակը փոխանցվում է ծառա աստիճանին՝ փոփոխելով արտաքին ելքերը՝  $Q=1$ ,  $\bar{Q} = 0$ : Նկատենք, որ արտաքին  $S$  և  $R$  մուտքերը կարող են փոխվել այն ժամանակահատվածում, երբ  $CLK=0$ , քանի որ այդ ժամանակահատվածում տեր տրիգերի մուտքերը մեկուսացված են:

Այժմ վերադառնանք նկ. 4.26-ի հաջորդական թվային շղթային և համարենք, որ հիշողության տարրերը տեր-ծառա տրիգերներ են: Տակտային ազդանշանի  $CLK=0$  փուլում փոփոխվում են տրիգերների ելքերը՝ ներկա վիճակի  $Q$  փոփոխականները, ինչպես նաև համակցական շղթայի ելքային ազդանշանները՝  $Q^{t+1}$ , որոնք կիրառվում են տրիգերների արտաքին մուտքերին: Սակայն այդ փուլում տրիգերների մուտքերը մեկուսացված են, և արտաքին ազդանշանները տրիգերների վրա չեն ազդում:  $CLK=1$  փուլում  $Q$  փոփոխականները փոխվել չեն կարող՝ դրանք ծառա աստիճանի ելքերն են: Իսկ տեր աստիճանի տրիգերները փոխում են վիճակները, բայց այդ փոփոխությունները տեր-ծառա տրիգերի ելքում չեն երևում: Հետևաբար, համակարգի հետադարձ կապերը չեն կարող բերել դրա աշխատանքի անկայունության և վիճակների մրցավազքի:

Նկարագրված RS տեր-ծառա տրիգերի ելքում ազդանշանները փոխվում են, երբ տակտային ազդանշանն անցնում է 1-ից 0, այսինքն՝ այս տրիգերը տակտավորվում է տակտային ազդանշանի բացասական ճակատով: Սակայն շատ տեր-ծառա տրիգերներ տակտավորվում են տակտային ազդանշանի դրական ճակատով: Դրա համար պետք է շրջիչ ունենալ արտաքինից տրվող  $CLK$  ազդանշանի և տեր աստիճանի տրիգերի տակտային մուտքի միջև: Այս դեպքում տակտային իմպուլսի բացասական ճակատն ազդում է տեր տրիգերի, իսկ դրական ճակատը՝ ծառա տրիգերի վրա:

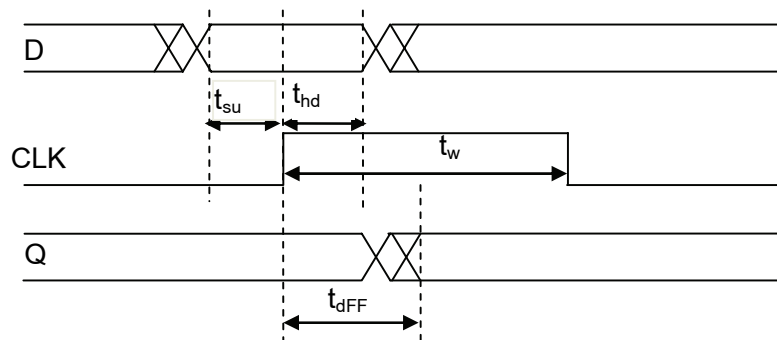
Տեր-ծառա կառուցվածքը կիրառելի է բոլոր տիպի տրիգերների համար: Նկ. 4.34-ում ցույց է տրված տեր-ծառա D տրիգերի կառուցվածքը: Նշենք, որ ներկայումս թվային համակարգերում մեծամասամբ օգտագործվում են տեր-ծառա D տրիգերներ:



Նկ. 4.34. Տեր-ծառա D տրիգերի կառուցվածքը

#### 4.10. Տակտավորվող հաջորդական շղթաների անխափան աշխատանքի պայմանները

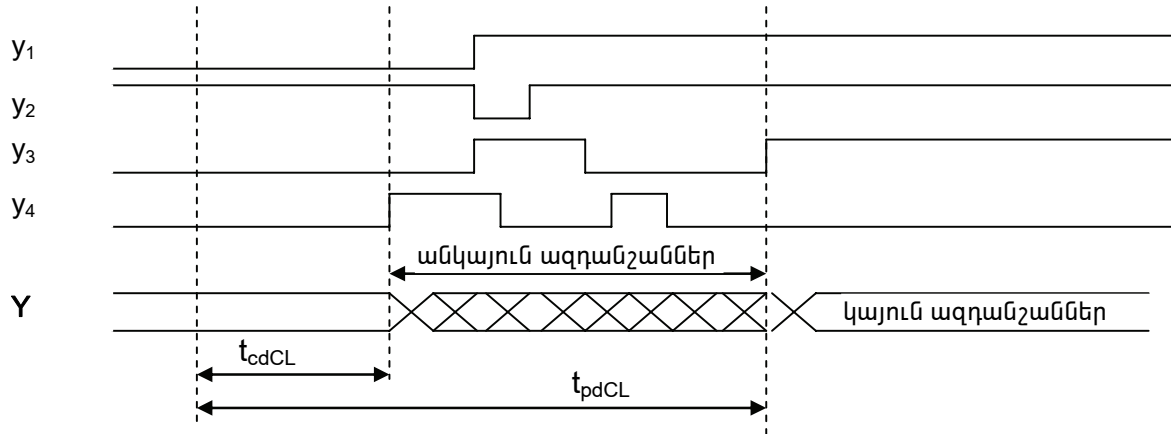
Հաջորդական թվային համակարգերի անխափան և հուսալի աշխատանքի համար անհրաժեշտ է պահպանել տրիգերների համաժամանակեցման որոշակի պայմաններ: Տրիգերի աշխատանքը ժամանակի մեջ բնութագրվում է հետևյալ ժամանակային պարամետրերով (տե՛ս նկ. 4.35)՝ տեղակայման ժամանակ (setup time)՝  $t_{su}$ , պահման ժամանակ (hold time)՝  $t_{hd}$ , տակտային իմպուլսի թույլատրելի նվազագույն տևողություն՝  $t_w$ , հապաղման ժամանակ՝  $t_{dFF}$  կամ  $t_{c2q}$ : Հապաղման ժամանակը տակտային իմպուլսի ճակատի պահից մինչև ելքային նոր վիճակի հաստատման պահն ընկած ժամանակահատվածն է (CLK-ից Q հապաղում՝  $t_{c2q}$ ): Տեղակայման ժամանակն այն ժամանակահատվածն է, որի ընթացքում մուտքային D տվյալը պետք է մնա անփոփոխ մինչև տակտային իմպուլսի ակտիվ ճակատի կիրառումը: Պահման ժամանակն այն ժամանակահատվածն է, որի ընթացքում մուտքային D տվյալը պետք է մնա անփոփոխ տակտային իմպուլսի ակտիվ ճակատի կիրառման պահից հետո: Նվազագույն թույլատրելի տևողությամբ տակտային իմպուլսը պետք է կարողանա հուսալիորեն շրջել տրիգերը:



Նկ. 4.35. Տրիգերի ժամանակային պարամետրերի սահմանումները

Համակցական շղթայով ազդանշանը տարածվում է որոշակի հապաղումով (տե՛ս բաժին 2.5): Նկ. 4.36-ում ցույց են տրված համակցական շղթայի չորս տարբեր ելքերում՝  $y_1$ ,  $y_2$ ,  $y_3$ ,  $y_4$ , ազդանշանների տարածման ժամանակային դիագրամները: Մուտքային ազդանշանների փոփոխության պահից սկսած՝ որոշակի ժամանակահատված հետո միայն ելքային ազդանշաններն սկսում են փոփոխվել: Այդ ժամանակահատվածը՝  $t_{cdCL}$ , կոչվում է համակցական տրամաբանական շղթայի սկզբնական հապաղում: Մուտքային ազդանշանների փոփոխության պահից մինչև ելքային ազդանշանների նոր արժեքների հաստատման պահն ընկած ժամանակահատվածը կոչվում է համակցական տրամաբանական շղթայի տարածման հապաղում՝  $t_{pdCL}$ : Թվային համակարգերում համակցական շղթայի հապաղումները հարմար է ցույց տալ մեկ ժամանակային դիագրամի՝ շղթայի ելքային ազդանշանների  $Y = [y_1, y_2, y_3, y_4]$  վեկտորի ժամանակային դիագրամի վրա, ինչպես ցույց է տրված նկ. 4.36-ում:  $t_{cdCL} < t < t_{pdCL}$  ժամանակահատվածում համակցական շղթայի ելքերն անկայուն են և պիտանի չեն

տրիգերներում սևեռելու համար: Հաջորդական թվային համակարգի անսխալ աշխատանքի համար հաջորդ վիճակի ազդանշանները պետք է տակտային իմպուլսով սևեռվեն տրիգերներում, երբ դրանք արդեն կայուն են:

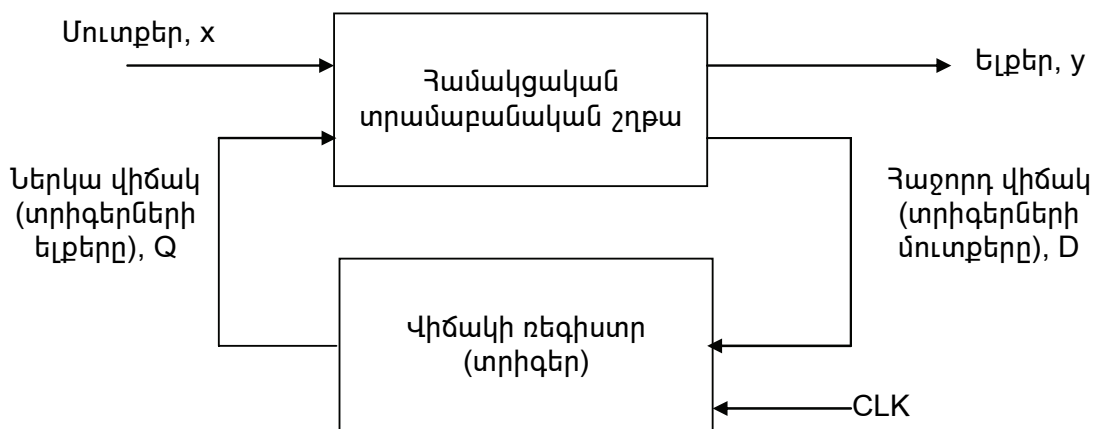


Նկ. 4.36. Ազդանշանների տարածումը համակցական տրամաբանական շղթայով՝ սկզբնական  $t_{cdCL}$  և տարածման հապաղում՝  $t_{pdCL}$

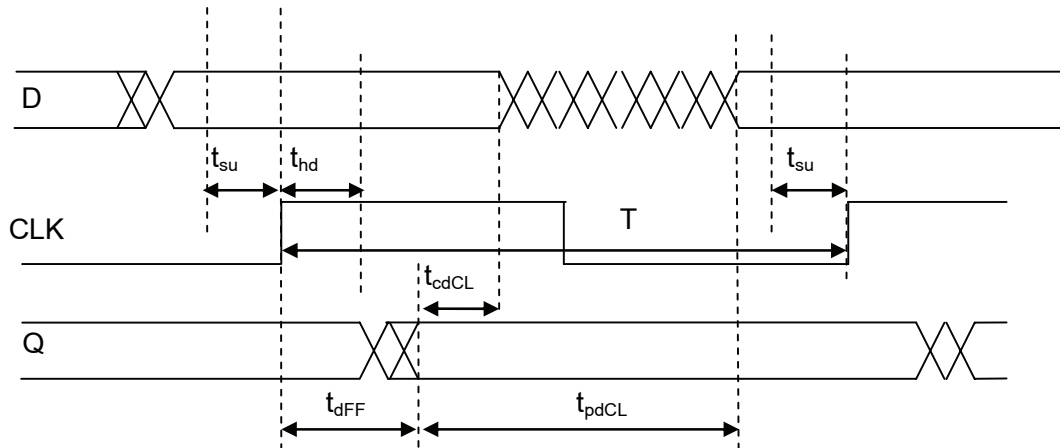
Բացի համակցական շղթայով ազդանշանների հապաղմանը ներկայացվող սահմանափակումներից, հաջորդական թվային համակարգի անսխալ աշխատանքի համար կան նաև այլ ժամանակային սահմանափակումներ:

Նկ. 4.37-ում ցույց է տրված հաջորդական թվային համակարգի ընդհանուր կառուցվածքը, իսկ նկ. 4.38-ում՝ համապատասխան ժամանակային դիագրամները:

Երբ վիճակի ռեգիստրի տրիգերներին կիրառվում է տակտային իմպուլս, դրանց ելքերը հաստատվում են նոր Q վիճակում  $t_{dFF}$  հապաղումից հետո, այնուհետև, Q ազդանշանները տարածվում են համակցական շղթայով, և  $t_{pdCL}$  ժամանակահատված հետո նոր տվյալները հաստատվում են համակցական շղթայի D ելքերում (տրիգերների մուտքերում): Տրիգերի անսխալ աշխատանքի համար անհրաժեշտ է, որ նրա D մուտքում ազդանշանը հաստատվի տակտային ազդանշանի կիրառման պահից  $t_{su}$  ժամանակահատված առաջ:



Նկ. 4.37. Հաջորդական թվային համակարգի կառուցվածքային սխեման



Նկ. 4.38. Հաջորդական թվային համակարգի ժամանակային դիագրամները

Այսպիսով, տակտային իմպուլսների պարբերությունը պետք է բավարարի հետևյալ պայմանին՝

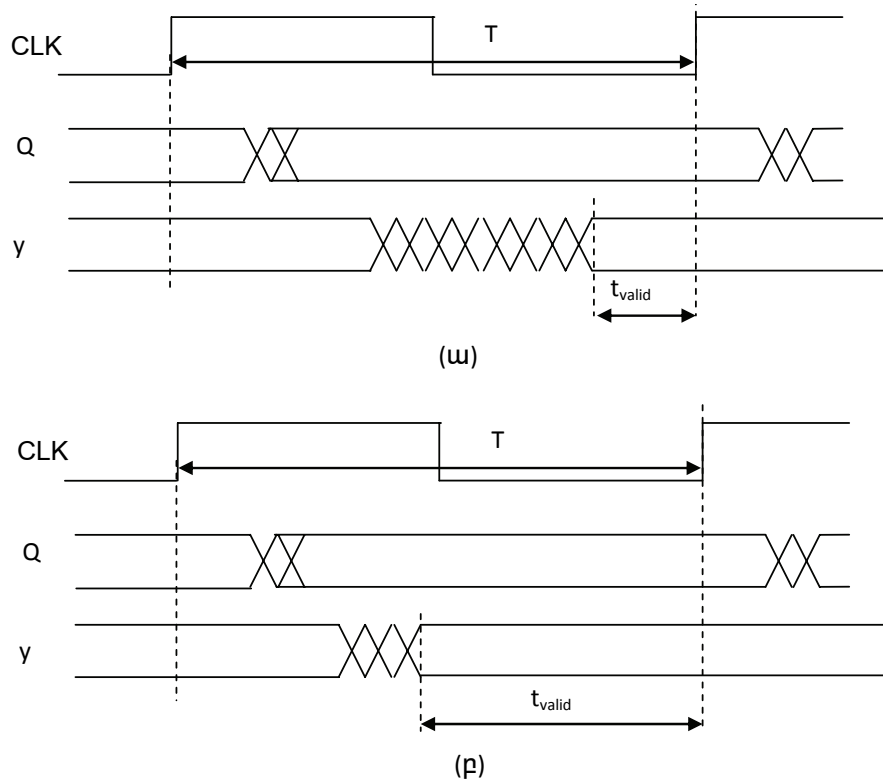
$$T > t_{dFF} + t_{pdCL} + t_{su} \quad (4.19)$$

Թվային համակարգի անսխալ աշխատանքի մյուս պայմանը պայմանավորված է տրիգերների պահման ժամանակով՝  $t_h$ , տրիգերի մուտքային D ազդանշանը պետք է մնա անփոփոխ  $t_h$  ժամանակահատվածում տակտային իմպուլսի ակտիվ ճակատի կիրառման պահից հետո: Տրիգերի D մուտքում տվյալներն սկսում են փոփոխվել տակտային իմպուլսի ակտիվ ճակատի կիրառման պահից  $t_{dFF} + t_{cdCL}$  ժամանակահատված հետո: Հետևաբար, անսխալ աշխատանքի համար պետք է բավարարվի հետևյալ պայմանը՝

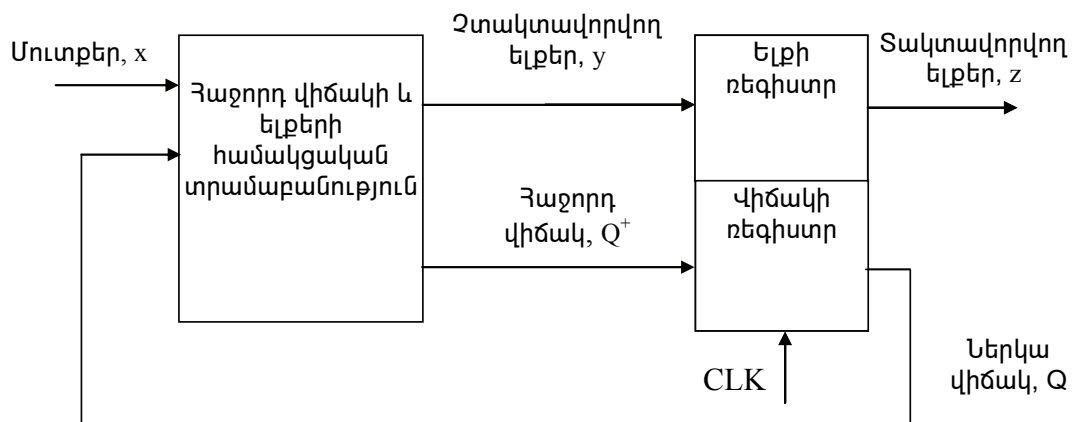
$$t_h < t_{dFF} + t_{cdCL} \quad (4.20)$$

Սովորաբար, մեկ հաջորդական թվային շղթայի ելքերը ծառայում են որպես մուտքեր մեկ այլ հաջորդական շղթայի համար: Որպեսզի ապահովվի անսխալ աշխատանքի (4.19) պայմանը բարձր հաճախականությունների համար, երբ ազդանշանները փոխանցվում են երկու հաջորդական շղթաների միջև, համակցական տրամաբանական շղթայով ազդանշանի տարածման հապաղումը պետք է լինի հնարավորինս կարճ: Դա նշանակում է, որ համակցական շղթայի ելքերը պետք է կայուն լինեն տակտի պարբերությամ մեծ մասում: Նկ. 4.39-ում ցույց են տրված Մուրի (նկ. 4.27) և Միլի (նկ. 4.26) ավտոմատների ժամանակային դիագրամները: Սովորաբար, Մուրի ավտոմատում “Համակցական շղթա2”-ը (նկ. 4.27) շատ ավելի պարզ է, քան “Համակցական շղթա”-ն Միլի ավտոմատում (նկ. 4.26): Հետևաբար, ելքային ազդանշանների հաստատման ժամանակը Մուրի ավտոմատում ավելի կարճ կլինի, քան Միլի ավտոմատում՝ կայուն տվյալները Մուրի ավտոմատում առկա են ավելի երկար ժամանակահատվածում: Սա նշանակալից առավելություն է արագագործ թվային շղթաների կառուցման խնդրում:

Որպեսզի Միլի ավտոմատում ելքային կայուն տվյալները առկա լինեն պարբերության ավելի մեծ մասում, համակցական շղթայի ելքային ազդանշանները կարելի է տակտային ազդանշանով սևեռել ելքի ռեգիստրում, ինչպես ցույց է տրված նկ. 4.40-ում: Այս դեպքում ելքերը կայուն են ամբողջ պարբերության ընթացքում, բացի ելքային ռեգիստրի տրիգերների  $t_{dFF}$  հապաղման ժամանակից (տե՛ս նկ. 4.41): Սակայն հարկ է նշել, որ ելքային ռեգիստրի ներկայությունը ներմուծում է մեկ պարբերության չափով լրացուցիչ ուշացում:

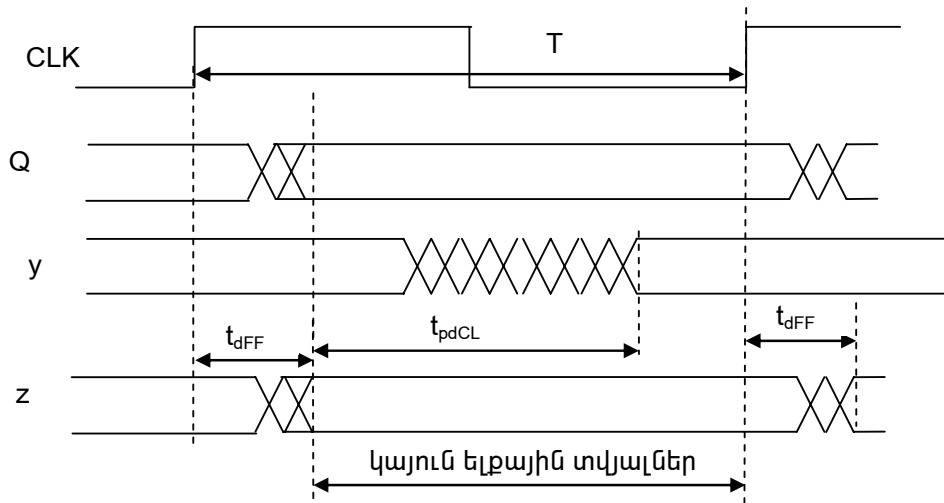


Նկ. 4.39. Միլի (ա) և Մուրի (բ) մոդելով կառուցված հաջորդական շղթաների ժամանակային դիագրամները: Համեմատե՛ք կայուն տվյալների առկայության ժամանակահատվածները



Նկ. 4.40. Տակտավորվող ելքերով Միլի ավտոմատի կառուցվածքային սխեման





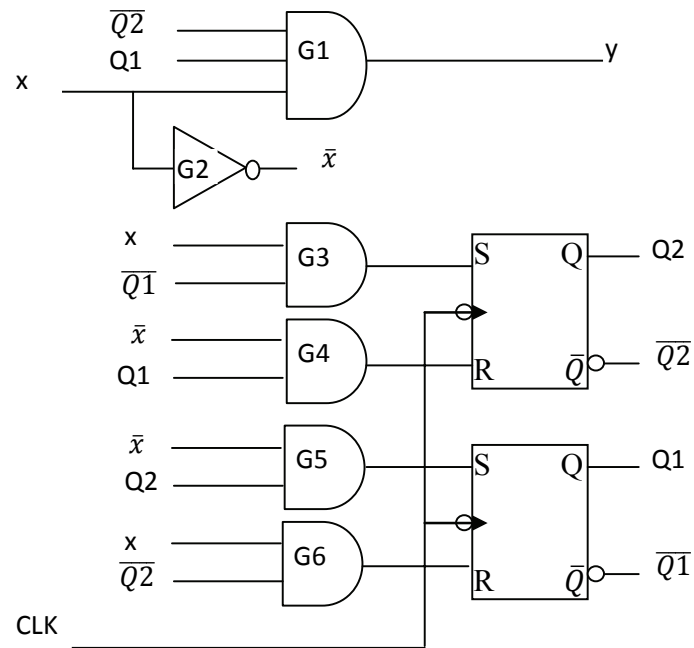
Նկ. 4.41. Տակտավորվող ելքերով Միլի ավտոմատի ժամանակային դիագրամները

#### 4.11. Թվային հաջորդական շղթաների վերլուծություն

Հաջորդական շղթայի վերլուծությունը ներառում է ելքերի և վիճակների փոփոխությունների հաջորդականությունների որոշում՝ տրված մուտքային հաջորդականությունների համար աղյուսակների, գրաֆների կամ ժամանակային դիագրամների միջոցով: Հնարավոր է նաև արտածել բանաձևեր, որոնք նկարագրում են հաջորդական շղթայի վարքագիծը:

Տրամաբանական շղթան ճանաչվում է որպես տակտավորվող հաջորդական շղթա, եթե այն պարունակում է տակտավորվող տրիգերներ: Տրիգերները կարող են լինել ցանկացած տիպի, իսկ տրամաբանական շղթան կարող է պարունակել կամ չպարունակել տրամաբանական տարրեր: Այս բաժնում կքննարկենք հաջորդական թվային շղթայի օրինակ և կստանանք նրա վարքագծի նկարագրությունը տարբեր եղանակներով:

Թվային հաջորդական շղթայի տրամաբանական սխեմայի օրինակը ցույց է տրված նկ. 4.42-ում: Այն ունի մեկ մուտքային փոփոխական՝ x, մեկ ելքային փոփոխական՝ y, տակտավորվող երկու RS տրիգերներ՝ նշված Q1 և Q2: Սխեմայում տրիգերների ելքերից տրամաբանական տարրերի մուտքերին գնացող կապերը բացահայտ ցույց չեն տրված՝ սխեմայի պարզության համար: Հաղորդալարերով միացումների փոխարեն կապերը տրամաբանական տարրերի մուտքերում նշված են սիմվոլներով: Օրինակ, G1 տարրի մուտքը Q1-ով նշելը նշանակում է, որ Q1 տրիգերի նորմալ ելքը միացված է G1-ի մուտքին: Նշենք, որ տրիգերների գրաֆիկական սիմվոլներից հետևում է, որ դրանք տակտավորվում են տակտային իմպուլսի բացասական ճակատով: Պարզության համար կանտեսենք տրիգերներում և տրամաբանական տարրերում ազդանշանների տարածման հապաղումները: Սակայն, պետք է հիշել, որ սա իդեալական դեպք է, իրական շղթաներում այդ հապաղումները շատ կարևոր են և կարող են առաջացնել համակարգի աշխատանքի խափանում՝ հատկապես բարձր հաճախականությունների տիրույթում:



Նկ. 4.42. Հաջորդական թվային համակարգի տրամաբանական սխեմայի օրինակ

Սկզբում տրամաբանական սխեմայից ստանանք ելքային  $y$  ազդանշանի և տրիգերների վիճակի հավասարումները՝

$$y = xQ_1\overline{Q_2}, \quad (4.21)$$

$$R_1 = x\overline{Q_2}, \quad (4.22)$$

$$S_1 = \bar{x}Q_2, \quad (4.23)$$

$$R_2 = \bar{x}Q_1, \quad (4.24)$$

$$S_2 = x\overline{Q_1} : \quad (4.25)$$

Այնուհետև, փոխարինելով  $R$ -ը և  $S$ -ը RS տրիգերի անցումների (4.14) հավասարման մեջ (4.22) – (4.25) արտահայտություններով ( $R_1$ ,  $S_1$ ,  $R_2$ ,  $S_2$ -ի համար), կստանանք հաջորդական շղթայի վիճակների հավասարումները՝

$$Q_1^+ = S_1 + \overline{R_1}Q_1 = \bar{x}Q_2 + \overline{x\overline{Q_2}}Q_1, \quad (4.26)$$

$$Q_2^+ = S_2 + \overline{R_2}Q_2 = x\overline{Q_1} + \overline{\bar{x}Q_1}Q_2 : \quad (4.27)$$

Հաջորդական շղթայի անցումների աղյուսակը կարելի է ստանալ կամ ուղղակի տրամաբանական սխեմայից, կամ (4.21) և (4.26)-(4.27) հավասարումներից: Անցումների աղյուսակը ցույց է տրված աղյուսակ 4.32-ում: Քննարկելով նկ. 4.42-ի տրամաբանական տարրերից ոչ մեկի ելքը 1 չէ, հետևաբար, տրիգերները չեն փոխում վիճակը: Երբ  $Q_2Q_1=00$  և  $x=1$ ,  $G3$  տարրի ելքում կստացվի 1, որը տրվում է  $Q_2$  տրիգերի  $S$  մուտքին, միաժամանակ  $G6$ -ի ելքում կծնավորվի տրամաբանական 1, որը տրվում է

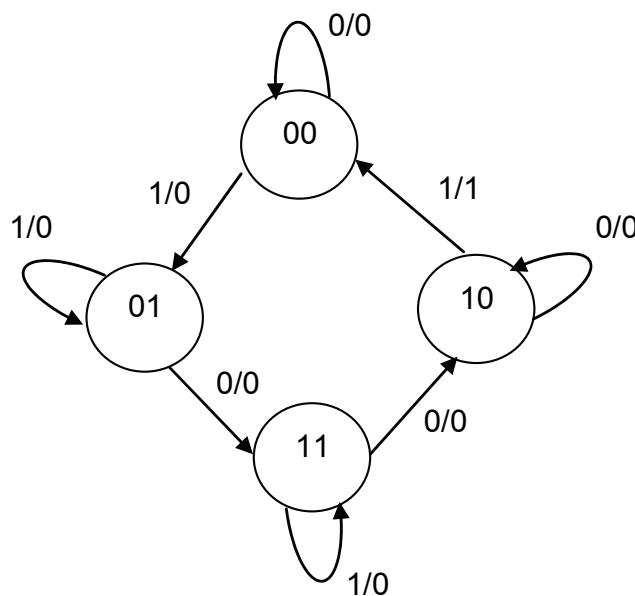
Q1-ի R մուտքին: Հաջորդ տակտային իմպուլսը Q1-ը կդնի 0, Q2՝ 1 վիճակ: Ուստի հաջորդ վիճակը կլինի 10: Նույն եղանակով կարելի է ստանալ հաջորդական շղթայի հաջորդ վիճակը՝ սկսելով մնացած երեք սկզբնական վիճակներից:

Աղյուսակ 4.32

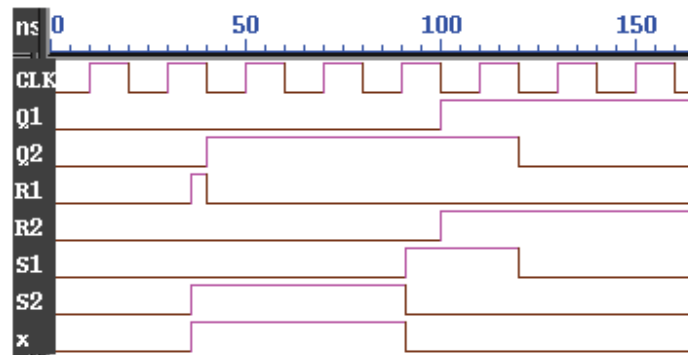
Ներկա վիճակ $Q_2Q_1$	Հաջորդ վիճակ, $Q_2^+Q_1^+$		Ելք, y	
	x=0	x=1	x=0	x=1
00	00	10	0	0
01	01	00	0	1
10	11	10	0	0
11	01	11	0	0

Հաջորդական շղթայի անցումների գրաֆը կարելի է ստանալ անցումների աղյուսակից: Այն ցույց է տրված նկ. 4.43-ում:

Օգտվելով անցումների աղյուսակից, գրաֆից կամ հավասարումներից՝ կարելի է ստանալ շղթայի աշխատանքի նկարագրության ժամանակային դիագրամները: Նկ. 4.44-ում ցույց են տրված տակտային CLK ազդանշանի, մուտքային x ազդանշանի, տրիգերների R1, S1, R2, S2 մուտքերի և Q1, Q2 մուտքերի ժամանակային դիագրամները, որոնք ստացվել են Verilog նմանակումից՝ համարելով, որ աշխատանքն սկսվում է 00 վիճակից: Տակտային ազդանշանի առաջին բացասական ճակատի տրման պահին ( $t=20$  նվ)  $x=0$ ,  $R1=S1=R2=S2=0$ , շղթան վիճակը չի փոխում՝ մնում է 00 վիճակում: Երկրորդ տակտային ազդանշանի բացասական ճակատի տրման պահին ( $t=40$  նվ)  $x=1$ ,  $R1=1$ ,  $S1=0$ ,  $R2=0$ ,  $S2=1$  և շղթան 00 վիճակից անցնում է 10 վիճակ, որը կարելի է ստանալ վիճակների հավասարումներից կամ անցումների աղյուսակից: Նույն կերպ կարելի է հետևել և վերլուծել շղթայի վարքագիծը տակտային ազդանշանի բոլոր հաջորդ բացասական ճակատների համար:



Նկ. 4.43. Հաջորդական շղթայի (նկ. 4.40) անցումների գրաֆը:



Նկ. 4.44. Վերլուծվող թվային շղթայի ազդանշանների ժամանակային դիագրամները

#### 4.12. Տրիգերների բնութագրիչ աղյուսակներ

Տրիգերի անցումների աղյուսակով որոշվում է նրա տրամաբանական հատկությունները և լրիվ բնութագրվում նրա աշխատանքը: Անցումների աղյուսակներն օգտակար են՝ հաջորդական թվային համակարգերի աշխատանքի վերլուծության համար: Թվային համակարգերի նախագծման դեպքում տրված են տրիգերների պահանջվող անցումները, պետք է որոշել տրիգերների մուտքային ազդանշանները, որոնք կարող են ապահովել այդ անցումները: Այդ նպատակով պետք է տրիգերի համար կառուցել աղյուսակ, որում բոլոր հնարավոր անցումների համար թվարկվում են մուտքային ազդանշանների համապատասխան հավաքածուները: Այդպիսի աղյուսակը կոչվում է տրիգերի բնութագրիչ աղյուսակ:

Աղյուսակ 4.33-ում բերված են գործնականում հաճախ օգտագործվող չորս տիպի տրիգերների բնութագրիչ աղյուսակները: Յուրաքանչյուր աղյուսակ բաղկացած է վիճակի փոփոխության երկու սյուներից՝  $Q$  և  $Q^+$ , և մեկական սյուն յուրաքանչյուր մուտքի համար: Կան ներկա  $Q$  վիճակից հաջորդ  $Q^+$  վիճակին անցման չորս հավաքածուներ: Յուրաքանչյուր անցման համար անհրաժեշտ մուտքային արժեքները որոշվում են անցումների աղյուսակում պարունակվող ինֆորմացիայից:  $X$ -ով նշվում են ոչ էական պայմանները՝ դրանց 1 կամ 0 լինելը տվյալ անցման վրա չեն ազդում:

Աղյուսակ 4.33

RS տրիգեր				D տրիգեր			JK տրիգեր				T տրիգեր		
Q	$Q^+$	R	S	Q	$Q^+$	D	Q	$Q^+$	J	K	Q	$Q^+$	T
0	0	x	0	0	0	0	0	0	0	x	0	0	0
0	1	0	1	0	1	1	0	1	1	x	0	1	1
1	0	1	0	1	0	0	1	0	x	1	1	0	1
1	1	0	x	1	1	1	1	1	x	0	1	1	0

RS տրիգերի բնութագրիչ աղյուսակում 0-ից 0 անցման համար գրված է  $R=x$ ,  $S=0$ : Նկ. 4.28բ-ում ցույց տրված անցումների աղյուսակից հետևում է, որ հաջորդ վիճակը 0 լինելու համար  $S$  մուտքը պետք է 0 լինի, և եթե  $R = 0$ , նշանակում է վիճակի պահպանում, իսկ եթե  $R=1$ , նշանակում է անցում 0 վիճակի: Երկու դեպքում էլ ունենք, որ հաջորդ վիճակը պետք է 0 լինի, երկու դեպքում էլ վիճակը չպետք է փոխվի:

Այսպիսով, անկախ R-ի արժեքից, տրիգերը մնում է 0 վիճակում: Այդ պատճառով աղյուսակ 4.33-ում 0-ից 0 անցման համար R-ը նշված է X-ով: Եթե տրիգերը 0 ներկա վիճակից պետք է անցնի 1 հաջորդ վիճակ, ապա անցումների աղյուսակից կարելի է որոշել, որ դա հնարավոր է միայն  $S=1$  և  $R=0$  պայմանի դեպքում: Ուստի բնութագրիչ աղյուսակում  $QQ+=01$  հավաքածուի համար նշված է  $RS=01$ : Եթե տրիգերը 1 ներկա վիճակից պետք է անցնի 0 հաջորդ վիճակ, անցումների աղյուսակից կարելի է որոշել, որ դա հնարավոր է միայն  $S=0$  և  $R=1$  պայմանի դեպքում: Ուստի բնութագրիչ աղյուսակում  $QQ+=10$  հավաքածուի համար նշված է  $RS=10$ : Վերջին  $QQ+=11$  հավաքածուի համար անցումների աղյուսակից կարելի է տեսնել, որ R-ը պետք է 0 լինի, իսկ S-ը կարող է լինել 0 կամ 1, այսինքն S-ի արժեքը կարևոր չէ: Բնութագրիչ աղյուսակում նշվում է  $RS=0X$ :

D տրիգերի անցումների աղյուսակից (նկ. 4.29բ) կարելի է տեսնել, որ հաջորդ վիճակը միշտ հավասար է D մուտքին՝ անկախ ներկա վիճակի արժեքից: Հետևաբար, D-ն պետք է լինի 0, եթե  $Q^+=0$ , և D-ն պետք է լինի 1, եթե  $Q^+=1$ :

JK տրիգերի անցումների աղյուսակից (նկ. 4.30բ) կարելի է տեսնել, որ 0 վիճակը պահպանելու համար ( $QQ+=00$ ) J մուտքը պետք է 0 լինի, իսկ K մուտքը կարող է լինել 0 կամ 1: Նույն կերպ, 1 վիճակը պահպանելու համար ( $QQ+=11$ ) K մուտքը պետք է 0 լինի, իսկ J մուտքը կարող է լինել 0 կամ 1: Եթե տրիգերը պետք է անցնի 0-ից 1 ( $QQ+=01$ ), J մուտքը պետք է 1 լինի, իսկ K մուտքը կարող է լինել 0 կամ 1: Եթե  $K=0$ ,  $J=1$  պայմանը տրիգերը դնում է պահանջվող 1 վիճակ, իսկ եթե  $K=1$ ,  $J=1$  ազդանշանը շրջում է տրիգերի վիճակը՝ շրջում է պահանջվող 1 վիճակ: Հետևաբար, բնութագրիչ աղյուսակում K-ն նշվում է X-ով: Եթե տրիգերը պետք է 0-ից անցնի 1 վիճակ ( $QQ+=10$ ), K մուտքը պետք է 1 լինի, իսկ J մուտքը կարող է լինել 0 կամ 1: Եթե  $J=0$ ,  $K=1$  պայմանը տրիգերը դնում է պահանջվող 0 վիճակ, իսկ եթե  $J=1$ ,  $K=1$  ազդանշանը շրջում է տրիգերի վիճակը՝ դնում է պահանջվող 0 վիճակ, հետևաբար, բնութագրիչ աղյուսակում J-ն նշվում է X-ով:

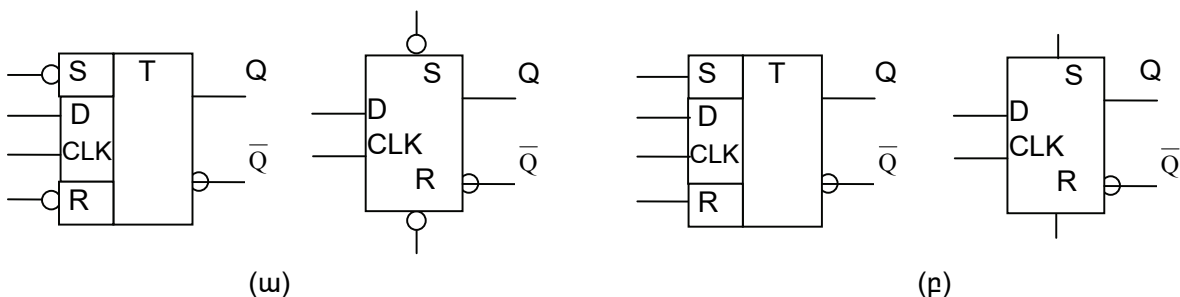
Նկ. 4.30բ-ում ցույց տրված T տրիգերի անցումների աղյուսակից կարելի է տեսնել, որ երբ  $T=1$ , տրիգերի ելքը կատարում է անցում հակադիր վիճակ, իսկ երբ  $T=0$ , տրիգերը վիճակը չի փոխում, հետևաբար, բնութագրիչ աղյուսակում  $QQ+=00$  և  $QQ+=11$  անցումների համար նշվում է  $T=0$ , իսկ  $QQ+=01$  և  $QQ+=10$  անցումների համար՝ նշվում է  $T=1$ :

#### 4.13. Հաջորդական թվային համակարգի սկզբնական վիճակի կարգման ասինքրոն մուտքեր

Հաճախ հաջորդական թվային համակարգի պահանջվող վարքագիծն ապահովելու համար այն պետք է նախապես, մինչև տակտավորումն սկսելը, կարգել անհրաժեշտ սկզբնական վիճակ: Պահանջվող սկզբնական վիճակ կարգելու համար պետք է վիճակի ռեգիստրի տրիգերները դնել որոշակի վիճակում: Պետք է հիշել, որ սնման լարումը միացնելուց հետո տրիգերները հայտնվում են անհայտ վիճակներում: Այդ նպատակով հաճախ տրիգերներում նախատեսվում են սկզբնական վիճակի կարգման

ասինքրոն S և R մուտքեր: Այդ մուտքերը փոխում են տրիգերի վիճակը առանց տակտային ազդանշանի առկայության:

Ասինքրոն S և R մուտքերով տրիգերների գրաֆիկական սիմվոլները ցույց են տրված նկ. 4.45-ում: Ակտիվ ցածր մակարդակով ասինքրոն մուտքերի գծերը նշվում են բացասման օղակով: Նորմալ աշխատանքային պայմաններում ասինքրոն մուտքերը պետք է պահվեն պասիվ վիճակում: Եթե որևէ ասինքրոն մուտք ակտիվ վիճակում է, ապա տրիգերն այլևս չի փոխի վիճակը տակտավորման միջոցով՝ ասում են ասինքրոն մուտքերը սինքրոն մուտքերից ավելի ուժեղ են: Օրինակ, եթե նկ. 4.45ա-ի տրիգերին տրվել է  $R=0$ , տրիգերը կմնա 0 վիճակում, անկախ D և CLK ազդանշանների վիճակներից: Պարտադիր չէ, որ տրիգերն ունենա և՛ S, և՛ R մուտքեր, այն կարող է ունենալ դրանցից միայն մեկը:



Նկ. 4.45. R և S ասինքրոն մուտքերով D տրիգերներ՝ (ա) ակտիվ ցածր մակարդակով, (բ) ակտիվ բարձր մակարդակով

Տրիգերի աշխատանքը կարելի է նկարագրել ֆունկցիաների աղյուսակով: Աղյուսակ 4.34-ում ցույց է տրված աստիվ ցածր մակարդակի ասինքրոն մուտքերով տրիգերի ֆունկցիաների աղյուսակը: X-ով նշված են ոչ էական պայմանները: Նորմալ տակտավորվող աշխատանքի համար պետք է S և R մուտքերը պահել 1 վիճակում:

Աղյուսակ 4.34

Մուտքեր				Ելք
R	S	CLK	D	Q
0	1	X	X	0
1	0	X	X	1
1	1	↑	0	0
1	1	↑	1	1

#### 4.14. Թվային ավտոմատների (հաջորդական թվային շղթաների) նախագծում

Տակտավորվող թվային շղթաների նախագծումը (կամ սինթեզը) սկսվում է խնդրի բառացի նկարագրությունից և ավարտվում է հաջորդական տրամաբանական շղթայի սխեմայով կամ բուլյան ֆունկցիաների համակարգով, որից կարող է կառուցվել կամ ծրագրավորվել (երբ օգտագործվում է Ծրագրավորվող Տրամաբանական Սարքեր՝ ԾՏՍ) տրամաբանական շղթայի սխեման: Ի տարբերություն համակցական շղթաների, որոնք լրիվ նկարագրվում են իսկության աղյուսակներով, հաջորդական շղթայի նկարագրության համար պահանջվում է նաև վիճակի անցումների աղյուսակ

կամ այլ համարժեք ներկայացում՝ անցումների գրաֆ կամ հավասարումներ:

Սինքրոն հաջորդական շղթան կառուցվում է տրիգերներից և տրամաբանական տարրերից: Շղթայի նախագծումը ներառում է տրիգերների ընտրությունը և համակցական շղթայի կառուցվածքի սինթեզը, այնպես, որ տրիգերներից և համակցական շղթայից բաղկացած համակարգի վարքագիծը կբավարարի խնդրի սկզբնական պահանջներին: Անհրաժեշտ տրիգերների թիվը որոշվում է ավտոմատի վիճակների թվից՝ համաձայն (4.8) բանաձևի: Համակցական շղթան սինթեզվում է անցումների աղյուսակից ստորև նկարագրված եղանակով: Գործնականում, երբ տրիգերների տիպը և քանակը որոշված են, հաջորդական շղթայի նախագծումը բերվում է համակցական շղթայի սինթեզի խնդրին:

Նախագծման ընթացակարգը բաղկացած է հետևյալ քայլերից.

1. Տալ նախագծվող համակարգի բառացի նկարագրությունը և ներկայացվող պահանջները: Նկարագրության համար կարելի է օգտագործել բլոկ-սխեմաներ, գրաֆներ, ժամանակային դիագրամներ և այլ հարմար միջոցներ ու տվյալներ:
2. Համակարգի առկա նկարագրությունից և տվյալներից ստանալ անցումների աղյուսակը:
3. Վիճակների թիվը կարելի է կրճատել՝ օգտվելով վիճակների նվազարկման եղանակներից:
4. Որոշել տրիգերների անհրաժեշտ թիվը ըստ (4.8) բանաձևի և կոդավորել 2 և 3 քայլերում ստացված անցումների աղյուսակի վիճակները: Կոդավորել մուտքերի և ելքերի սիմվոլները:
5. Ընտրել օգտագործվող տրիգերների տիպերը:
6. Համակարգի անցումների աղյուսակից և տրիգերների բնութագրիչ աղյուսակներից դուրս բերել տրիգերների մուտքերի հավասարումները՝ գրգռման ֆունկցիաները: Ելքերի աղյուսակից ստանալ համակարգի ելքերի ֆունկցիաների բանաձևերը:
7. Օգտվելով բուլյան ֆունկցիաների նվազարկման եղանակներից՝ ստանալ տրիգերների մուտքերի և համակարգի ելքերի ֆունկցիաների նվազարկված բանաձևերը:
8. Քայլ 7-ում ստացված բանաձևերի հիման վրա կառուցել համակարգի տրամաբանական սխեման:

Նախագծվող համակարգի աշխատանքի բառացի նկարագրությունը և նախագծմանը ներկայացվող պահանջների ձևակերպումը կոնկրետ դեպքին վերաբերող խնդիր են և ձևականացման ենթակա չեն: Նախագծողը պետք է օգտագործի իր փորձը և հմտությունները, որպեսզի իրագործի խնդրի պահանջներին բավարարող ճշգրիտ նկարագրություն: Բառացի նկարագրությունը կարող է շատ դեպքերում թերի կամ ոչ ճշգրիտ լինել, հետևաբար այս փուլում նկարագրության ճշգրտության ստուգումը չափազանց կարևոր է: Այնուամենայնիվ, եթե նկարագրությունն արդեն կազմված է, և անցումների աղյուսակն ստացվել է, ապա կարելի է սինթեզի հետագա ընթացքը իրականացնել ձևականացված ընթացակարգով: Ներկայումս բոլոր այդ ձևականացված

ընթացակարգերը իրագործվում են էլեկտրոնային Նախագծման Ավտոմատացման (ԷՆԱ, Electronic Design Automation, EDA) ծրագրային գործիքներով: Այդպիսի ծրագրային գործիքներ են տրամաբանական սինթեզի գործիքները, օրինակ, Սինոփսիս (Synopsys) ընկերության “Design Compiler” և Քեյդընս (Cadence) ընկերության “Build Gates” գործիքները: Ավտոմատացված սինթեզի համար նախագծվող համակարգը պետք է նկարագրված լինի սխեմաների նկարագրության լեզուներով (ՄՆԼ, HDL). ՄՆԼ նկարագրությամբ համակարգի վարքագիծը կարելի է նմանակել ծրագրային գործիքներով, ինչը թույլ է տալիս ստուգել համակարգի աշխատանքի ճշգրտությունը և կատարել անհրաժեշտ ուղղումներ նախագծման վաղ փուլում՝ նախքան նախագծման բոլոր հետագա փուլերով անցնելը:

Այնուամենայնիվ, նախագծման քայլերի կատարումը ձեռքով հնարավորություն է տալիս լավագույն պատկերացում ունենալ նախագծման ընթացակարգի մասին և ամուրի հիմք է ավտոմատացված նախագծման հսկման և ստուգման համար:

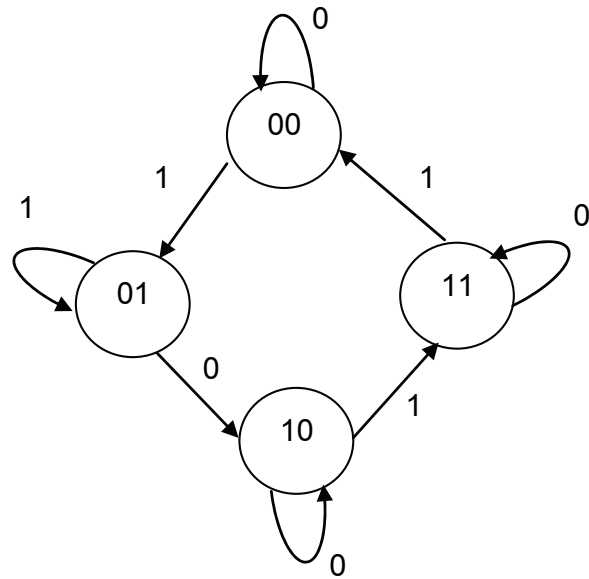
Նախագծման ընթացակարգի քայլերի մեծ մասն արդեն քննարկվել են նախորդ բաժիններում: Նոր, դեռևս չքննարկված, խնդիր է տրիգերների մուտքերի ֆունկցիաների ստացումը (քայլ 6): Դա իրականացվում է թվային համակարգի բնութագրիչ աղյուսակի միջոցով: Համակարգի բնութագրիչ աղյուսակը նման է առանձին տրիգերների բնութագրիչ աղյուսակներին, բացառությամբ, որ մուտքերի պայմանները թելադրվում են անցումների աղյուսակի ներկա վիճակի և հաջորդ վիճակի սյուներից:

Համակարգի բնութագրիչ աղյուսակների ստացումը և այդ աղյուսակներից տրիգերների մուտքերի ֆունկցիաների նվազարկված բանաձևերի ստացումը հարմար է ցուցադրել օրինակներով: Անհրաժեշտ է նախագծել հաջորդական թվային շղթա, որի անցումների աղյուսակը և գրաֆը ցույց են տրված աղյուսակ 4.35-ում և նկ 4.46-ում: Կօդազործները JK տիպի տրիգերներ:

Աղյուսակ 4.35

Ներկա վիճակ $Q_2Q_1$	Հաջորդ վիճակ, $Q_2^+Q_1^+$		Ելք, $y$	
	$x=0$	$x=1$	$x=0$	$x=1$
00	00	01	0	0
01	10	01	0	0
10	10	11	0	1
11	11	00	0	1





Նկ. 4.46. Նախագծվող հաջորդական համակարգի անցումների գրաֆը

Տակտավորվող հաջորդական շղթայի ընդհանուր կառուցվածքը ցույց է տրված նկ. 4.37-ում (Միլի մոդել): Հիշողության տարրերի՝ տրիգերների ընտրությունից հետո մնում է սինթեզել համակցական շղթան: Այդ համակցական շղթայի համար մուտքեր են ծառայում շղթայի  $x$  մուտքերը և տրիգերների ներկա վիճակի  $Q$  փոփոխականները, իսկ ելքերը համակարգի  $y$  ելքերը և տրիգերների մուտքերի փոփոխականներն են:

Այժմ հետևենք համակարգի բնութագրիչ աղյուսակի ստացմանը և համակցական շղթայի սինթեզի ընթացակարգին:

Նախագծվող համակարգի բնութագրիչ աղյուսակը պարունակում է նույն ինֆորմացիան, ինչ անցումների աղյուսակը՝ այն անցումների աղյուսակի ձևափոխված տեսքն է, ինչպես ցույց է տրված աղյուսակ 4.36-ում: Այս աղյուսակն իր տեսքով մնան է իսկության աղյուսակի՝ մուտքային փոփոխականները և ներկա վիճակի փոփոխականները ծառայում են համակարգի (նկ. 4.37) համակցական շղթայի մուտքեր, իսկ տրիգերների մուտքային փոփոխականները և շղթայի ելքերը ծառայում են համակցական շղթայի ելքեր: Աղյուսակ 4.36-ում յուրաքանչյուր ներկա վիճակի և մուտքային ազդանշանի համապատասխանող հաջորդ վիճակը արտագրված է աղյուսակ 4.35-ից: Տրիգերների մուտքերի ֆունկցիաների արժեքները որոշվում են պահանջվող անցման իրականացման պայմանից և օգտագործվող տրիգերի տիպից: Քանի որ այս օրինակում օգտագործվում են JK տրիգերներ, պետք է ունենալ  $Q1$  և  $Q2$  տրիգերների JK մուտքերի սյուներ (նշանակված են  $J1$ ,  $K1$ ,  $J2$ ,  $K2$ ): JK տրիգերի բնութագրիչ աղյուսակը ցույց է տրված աղյուսակ 4.33-ում: Աղյուսակ 4.36-ում տրիգերների մուտքերի ֆունկցիաների սյուները լրացվում են հետևյալ կերպ, օրինակ, աղյուսակ 4.36-ի առաջին տողում ունենք  $Q2Q1=00$ -ից  $Q2^+Q1^+=00$  անցում՝  $Q2$  տրիգերը 0 ներկա վիճակից անցնում է 0 հաջորդ վիճակի (նույնը նաև  $Q1$ -ի համար): Աղյուսակ 4.33-ից որոշում ենք, որ JK տրիգերի 0-ից 0 անցման համար անհրաժեշտ մուտքերն են՝  $J=0$  և  $K=X$ : Հետևաբար

այդ 0 և X արժեքները լրացվում են աղյուսակ 4.36- առաջին տողում J2-ի և K2-ի տակ համապատասխանաբար: Քանի որ առաջին տողում Q1-ը նույնպես կատարում է 0-ից 0 անցում, ապա 0 և X են լրացվում նաև J1-ի և K1-ի տակ: Աղյուսակ 4.36-ի երկրորդ տողում ունենք Q2-ի անցում 0-ից 1: Աղյուսակ 4.33-ից որոշում ենք, որ այդ անցման համար պետք է՝ J=1 և K=X: Այդ արժեքները լրացվում են աղյուսակ 4.36-ի երկրորդ տողում J2-ի և K2-ի տակ: Երկրորդ տողում ունենք, որ Q1-ն անցնում է 1-ից 0: Աղյուսակ 4.33-ից որոշում ենք անհարժեշտ մուտքերը՝ J=X և K=1, որոնք լրացվում են J1-ի և K1-ի տակ: Այս ձևով պետք է անցնել աղյուսակ 4.36-ի բոլոր տողերով:

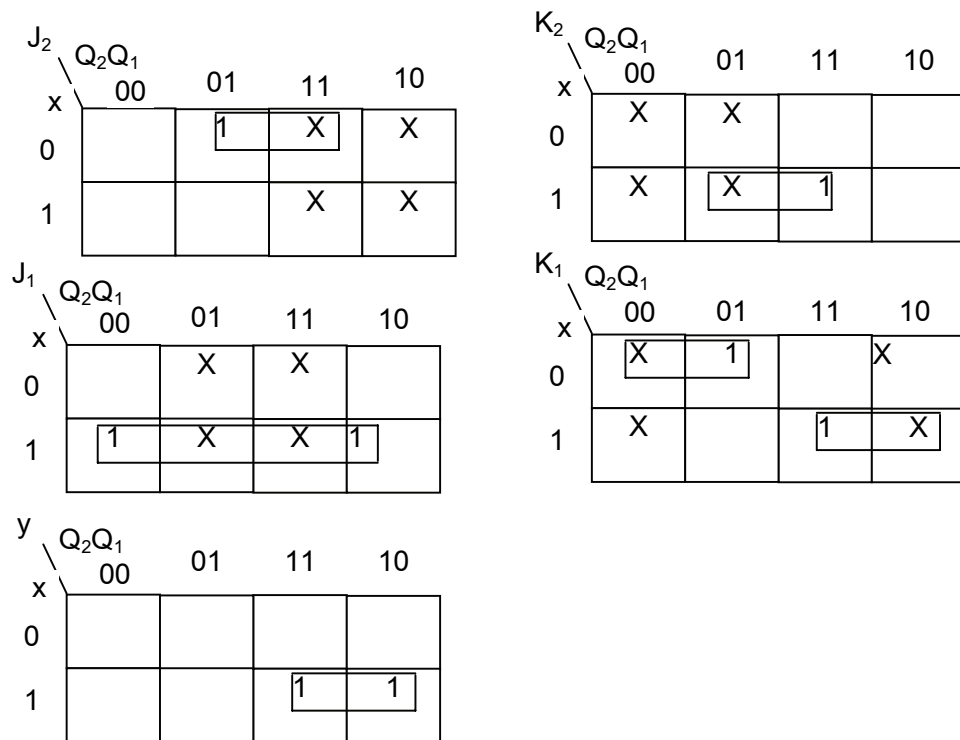
Աղյուսակ 4.36

Համակցական շղթայի մուտքեր			Հաջորդ վիճակ		Համակցական շղթայի ելքեր				
Մուտք	Ներկա վիճակ				Տրիգերների մուտքեր				Ելք
x	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>2</sub> <sup>+</sup>	Q <sub>1</sub> <sup>+</sup>	J <sub>2</sub>	K <sub>2</sub>	J <sub>1</sub>	K <sub>1</sub>	y
0	0	0	0	0	0	X	0	X	0
0	0	1	1	0	1	X	X	1	0
0	1	0	1	0	X	0	0	X	0
0	1	1	1	1	X	0	X	0	0
1	0	0	0	1	0	X	1	X	0
1	0	1	0	1	0	X	X	0	0
1	1	0	1	1	X	0	1	X	1
1	1	1	0	0	X	1	X	1	1

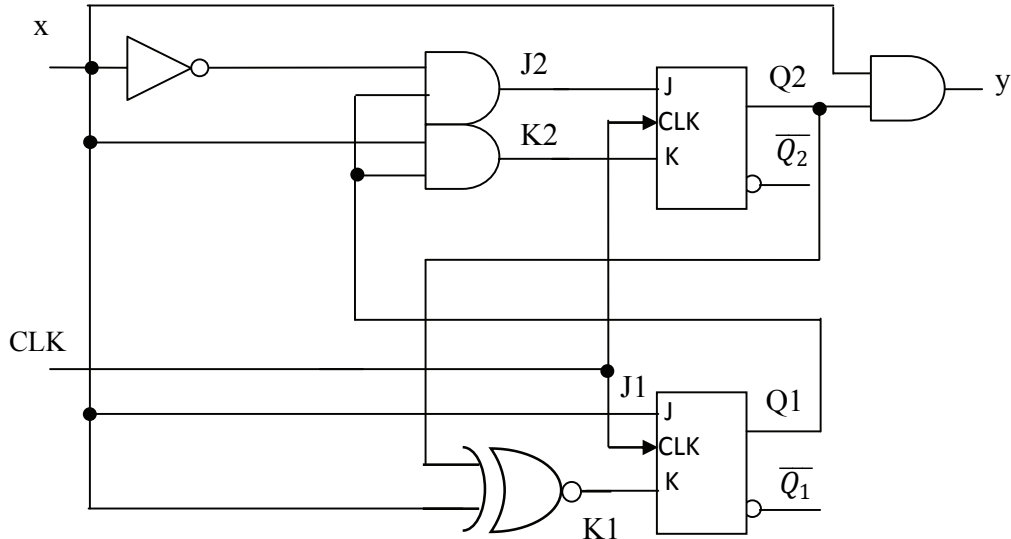
Աղյուսակ 4.36-ում ունենք համակարգի համակցական շղթան նկարագրող բոլոր բուլյան ֆունկցիաները: Այդ ֆունկցիաներն են՝ J<sub>2</sub>, K<sub>2</sub>, J<sub>1</sub>, K<sub>1</sub>, y, որոնք կախված են x, Q<sub>2</sub>, Q<sub>1</sub> փոփոխականներից: Տեղափոխելով J<sub>2</sub>, K<sub>2</sub>, J<sub>1</sub>, K<sub>1</sub>, y ֆունկցիաների արժեքները Կառնոյի քարտերի վանդակներում (նկ. 4.47), նվազարկումից հետո կստանանք նվազարկված ֆունկցիաների բանաձևերը.

$$J_2 = \bar{x}Q_1, K_2 = xQ_1, J_1 = x, K_1 = \bar{x}\bar{Q}_2 + xQ_2 = \bar{x} \oplus Q_2, y = xQ_2:$$

Այս բանձների հիման վրա կառուցվում է նախագծված հաջորդական համակարգի տրամաբանական սխեման, որը ցույց է տրված նկ. 4.48-ում: Այն բաղկացած է երկու տրիգերներից, որոնք կազմում են համակարգի վիճակի ռեգիստրը, երեք 2ԵՎ տարրերից, մեկ շրջիչից և մեկ Բացառող-ԿԱՄ-ՈՉ տարրից:



Նկ. 4.47. Նախագծվող հաջորդական շղթայում տրիգերների մուտքերի ֆունկցիաների և շղթայի ելքի ֆունկցիայի Կառնոյի քարտերը



Նկ. 4.48. Նախագծված հաջորդական համակարգի տրամաբանական սխեման

Որոշակի փորձ ձեռք բերելուց հետո կարելի է կրճատել տրիգերների մուտքերի ֆունկցիաների ստացման համար անհրաժեշտ քայլերի թիվը: Օրինակ, այդ ֆունկցիաների Կառնոյի քարտերը (նկ. 4.47) կարելի է լրացնել միանգամից անցումների աղ. 4.35-ից՝ առանց կառուցելու համակարգի բնութագրիչ աղ. 4.36-ը: Դա կատարվում է՝ անցնելով անցումների աղյուսակի բոլոր ներկա վիճակներով՝ որոշելով տրի-

գերների անցումները: Այնուհետև, այդ անցումների համար տրիգերների բնութագրիչ աղյուսակից որոշվում են համապատասխան մուտքերի արժեքները՝ 0, 1, կամ X, որոնք լրացվում են Կառնոյի քարտի համապատասխան վանդակներում: Օրինակ, աղ. 4.35-ի առաջին տողից երևում է, որ  $x=0$  դեպքում ունենք  $Q2Q1=00$  ներկա վիճակից անցում  $Q2^+Q1^+=00$  հաջորդ վիճակին: Այսինքն,  $Q1$  տրիգերն 0-ից անցնում է 0 վիճակ: JK տրիգերի բնութագրիչ աղ. 4.43-ից որոշվում է՝  $J=0$  և  $K=X$ : Այդ արժեքները լրացվում են նկ. 4.47-ի  $J1$  և  $K1$  քարտերի ( $x, Q2Q1$ )=000 վանդակը: Նկատենք, որ D տրիգերների դեպքում անցումը անցումների աղյուսակից Կառնոյի քարտերի շատ ավելի պարզ է, քանի որ տրիգերի մուտքի փոփոխականի անհրաժեշտ արժեքը համընկնում է հաջորդ վիճակի արժեքի հետ: Օրինակ, եթե անցումների աղյուսակի առաջին տողից որոշվում է, որ  $x=1$  դեպքում  $Q2Q1=00$  ներկա վիճակից կատարվում է անցում  $Q2^+Q1^+=01$  հաջորդ վիճակին, ապա D2-ի և D1-ի Կառնոյի քարտերի ( $x, Q2Q1$ )=100 վանդակներում պետք է լրացվեն 0 և 1 արժեքներ, համապատասխանաբար:

$n$  արտաքին մուտքերով,  $m$  ելքերով,  $r$  տրիգերներով հաջորդական շղթայի բնութագրիչ աղյուսակը կունենա համակցական շղթայի մուտքերի  $r+n$  սյուներ, հաջորդ վիճակի փոփոխականների  $r$  սյուներ և համակցական շղթայի  $m+kr$  ելքեր՝ տրիգերների մուտքայի փոփոխականների  $kr$  սյուներ, որտեղ  $k$ -ն տրիգերի մուտքերի թիվն է, և ելքերի  $m$  սյուներ: Ընդհանուր առմամբ աղյուսակը կունենա  $2^{r+n+kr+m}$  սյուներ և  $2^{r+n}$  տողեր:

#### 4.15. Ավելցուկային վիճակներով հաջորդական շղթաների սինթեզ

$r$  տրիգերներ պարունակող հաջորդական շղթան ունի  $2^r$  վիճակներ: Կան դեպքեր, երբ հաջորդական շղթայում կարող են օգտագործվել վիճակների առավելագույն թվից քիչ վիճակներ՝  $R < 2^r$ : Չօգտագործվող ( $2^r - R$ ) ավելցուկային վիճակներն անցումների աղյուսակում ցույց չեն տրվում: Տրիգերների մուտքերի ֆունկցիաների և համակարգի ելքի ֆունկցիաների նվազարկման ժամանակ այդ չօգտագործվող վիճակները կարելի է ընդունել որպես ոչ էական պայմաններ: Մանրամասները կցուցադրվեն հետևյալ օրինակով:

**Օրինակ 4.10.** Նախագծել աղ. 4.29-ով տրված վաճառող մեքենայի կառավարող ավտոմատը:

Ինչպես կարելի է տեսնել աղ. 4.29-ից, երկու վիճակի փոփոխականների չորս հնարավոր համակցություններից միայն երեքն են օգտագործվում՝ 11-ն ավելցուկային է: Հաջորդական շղթայում կօգտագործենք D տրիգերներ:

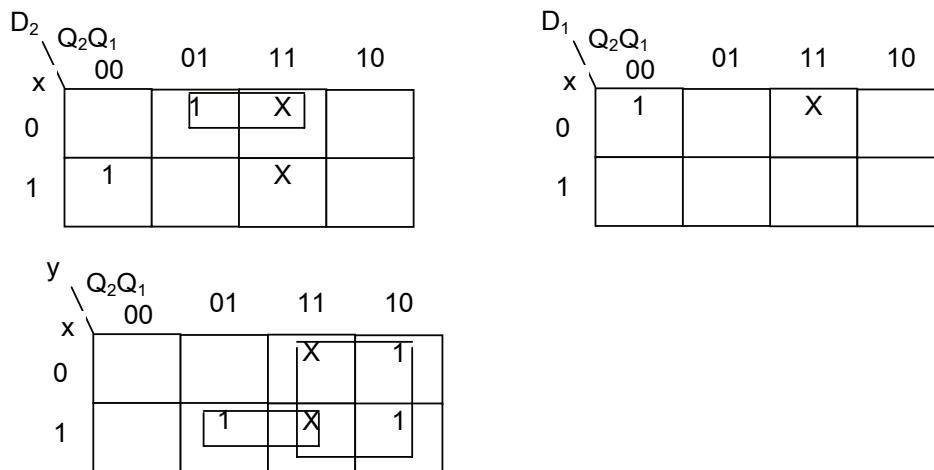
Սկզբում ստանաք նախագծվող հաջորդական շղթայի բնութագրիչ աղյուսակը (4.37): Ուշադրություն դարձրեք, որ D տրիգերի դեպքում բնութագրիչ աղյուսակի հաջորդ վիճակի և տրիգերների մուտքերի փոփոխականների սյուները համընկնում են: Չօգտագործվող 11 վիճակը բացակայում է այդ աղյուսակում:  $x=0$  կամ  $x=1$  մուտքերի դեպքում կունենանք 011 և 111 արգելված միներմներ: Նկ. 4.49-ի Կառնոյի քարտերում այդ միներմներին համապատասխանող վանդակները նշված են X-ով: Նվազարկված  $f D_2, D_1$  և  $y$  ֆունկցիաների բանաձևերը կունենան հետևյալ տեսքերը.

$$D_2 = \bar{x}Q_1 + x\bar{Q}_2\bar{Q}_1, D_1 = \bar{x}\bar{Q}_2\bar{Q}_1, y = Q_2 + xQ_1:$$

Աղյուսակ 4.37

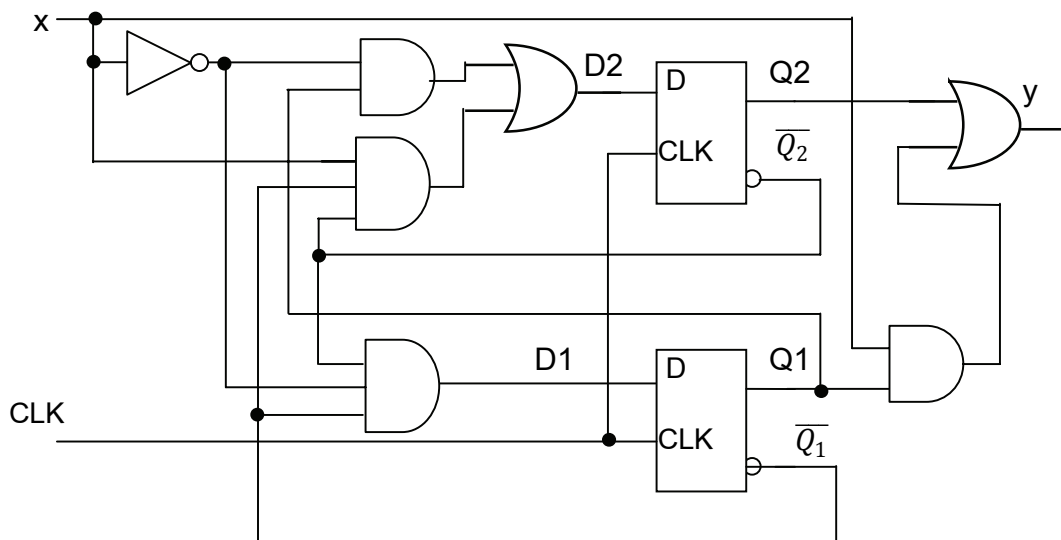
Մուտք	Ներկա վիճակ		Հաջորդ վիճակ		Տրիգերների մուտքեր		Ելք
x	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>2</sub> <sup>+</sup>	Q <sub>1</sub> <sup>+</sup>	D <sub>2</sub>	D <sub>1</sub>	y
0	0	0	0	1	0	1	0
0	0	1	1	0	1	0	0
0	1	0	0	0	0	0	1
1	0	0	1	0	1	0	0
1	0	1	0	0	0	0	1
1	1	0	0	0	0	0	1

Նկ. 4.50-ում ցույց է տրված նախագծված կառավարող ավտոմատի տրամբանական սխեման:



Նկ. 4.49. Օրինակ 4.10-ի կառավարող ավտոմատի ելքի ֆունկցիայի և տրիգերների մուտքերի ֆունկցիաները

Մի կարևոր հանգամանք, որը պետք է քննարկվի հաջորդական շղթաների նախագծման ժամանակ, դա շղթան հայտնի սկզբնական վիճակում կարգելն է: Երբ թվային համակարգը միացվում է սնման աղբյուրին, հայտնի չէ, թե որ վիճակում այն կհայտնվի: Սովորաբար, թվային համակարգն ունի սկզբնական վիճակի կարգման ընդհանուր մուտք, որը, մինչև համակարգի տակտավորվող աշխատանքն սկսելը, ասինքրոն եղանակով բոլոր տրիգերները կարգում է նախագծով նախատեսված հայտնի սկզբնական վիճակ: Շատ դեպքերում այդ սկզբնական վիճակը համապատասխանում է բոլոր տրիգերների 0 վիճակին, չի բացառվում, որ որոշ տրիգերներ կարգվեն 1 վիճակ: Բայց այն դեպքերում, երբ շղթան ունի ավելցուկային արգելված վիճակներ, և հաջորդական շղթան չի դրվում հայտնի սկզբնական վիճակում, կամ ավելի վատ, երբ աղմուկների պատճառով հնարավոր է, որ հայտնվի այդ ավելցուկային վիճակներից որևէ մեկում, պետք շղթան նախագծել այնպես, որ այն այդ ավելցուկային վիճակներից ինքնուրույն անցնի որևէ թույլատրելի վիճակի և շարունակի նորմալ աշխատանքը: Հակառակ դեպքում հաջորդական շղթան կարող է կատարել անվերջ անցումներ արգելված վիճակներով:



Նկ. 4.50. Օրինակ 4.10-ում նախագծված կառավարող ավտոմատի տրամբանական սխեման

#### 4.16. Վիճակների նշանակում

Երբ համակարգում առկա են ավելցուկային վիճակներ՝  $P < 2^r$ , որտեղ  $P$ -ն համակարգի վիճակների թիվն է,  $r$ -ը տրիգերների թիվը, սկզբունքորեն  $2^r$  հնարավոր վիճակներից կարելի է ընտրել ցանկացած  $P$ -ն: Սակայն հաջորդական թվային համակարգի համակցական շղթայի բարդությունը, հետևաբար նաև համակարգի ինքնաբերական կախված է օգտագործվող վիճակների ենթաբազմության ընտրությունից: Ավելին, եթե նույնիսկ ավելցուկային վիճակներ չկան՝  $P=2^r$ , այս դեպքում նույնպես համակցական շղթայի բարդությունը կախված է վիճակներին վերագրվող երկուական կոդերի հաջորդականությունից: Վիճակների նշանակման ընթացակարգը միտված է վիճակներին վերագրել երկուական կոդի այնպիսի հաջորդականություն, որի դեպքում հաջորդական համակարգի համակցական շղթան, որը կառավարում է տրիգերների մուտքերը, կլինի պարզագույնը: Նման ընթացակարգի կիրառությունը հատկապես արդյունավետ է այն դեպքերում, երբ համակարգը տեսանելի է միայն նրա ելք-մուտք գծերից: Այդպիսի համակարգում վիճակների փոփոխության հաջորդականությունը կարևոր չէ այնքանով, քանի դեռ այն ելքում ձևավորում է ազդանշանների պահանջվող հաջորդականությունները: Նկատենք, որ դա կիրառելի չէ այն դեպքերի համար, որոնցում համակարգի ելքերը ուղղակի վերցվում են տրիգերների ելքերից, որոնց վիճակների փոփոխությունների հաջորդականությունները նախապես տրված են՝ օրինակ, ինպուլսների հաշվիչներում:

**Օրինակ 4.11.** Դիտարկենք աղյուսակ 4.38-ով տրված վերջավոր ավտոմատը: Վիճակների  $(S_0, S_1, S_2, S_3)=(00, 01, 10, 11)$  կոդավորման արդյունքում կստանանանք աղ. 4.39-ով տրված անցումների աղյուսակը:

Աղյուսակ 4.38

Մուտք	Ներկա վիճակ	Հաջորդ վիճակ	Ելք
x	S	S <sup>+</sup>	y
0	S0	S0	0
0	S1	S1	0
0	S2	S2	0
0	S3	S3	1
1	S0	S1	0
1	S1	S2	0
1	S2	S3	0
1	S3	S3	1

Աղյուսակ 4.39

Մուտք	Ներկա վիճակ		Հաջորդ վիճակ		Ելք
x	Q2	Q1	Q2 <sup>+</sup>	Q1 <sup>+</sup>	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	1	1	1

D տրիգերների մուտքերի ֆունկցիաների և համակարգի ելքային ֆունկցիայի նվազարկված բանաձևերը հետևյալներն են (հիշենք, որ  $Q^+ = D$ )`

$$D_2 = Q_2 + xQ_1,$$

$$D_1 = xQ_2 + x\overline{Q_1} + \overline{x}Q_1,$$

$$y = Q_2Q_1:$$

Այս բանաձևերով նկարագրվող համակցական շղթայի երկու մակարդակով իրականացման համար անհրաժեշտ են 6 հատ երկմուտք տրամաբանական տարրեր, մեկ եռամուտք տարր և մեկ շրջիչ (x-ը ժխտելու համար):

Այժմ դիտարկենք վիճակների կոդավորման մեկ այլ տարբերակ՝ (S0, S1, S2, S3)=(10, 00, 11, 01), որի դեպքում կունենանք աղ.4.40-ում ցույց տրված անցումների աղյուսակը:

Աղյուսակ 4.40

Մուտք	Ներկա վիճակ		Հաջորդ վիճակ		Ելք
x	Q2	Q1	Q2 <sup>+</sup>	Q1 <sup>+</sup>	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	1	1	0
1	0	1	0	1	1
1	1	0	0	0	0
1	1	1	0	1	0

Այս դեպքում  $D$  տրիգերների մուտքերի ֆունկցիաների և համակարգի ելքային ֆունկցիայի նվազարկված բանաձևերը կունենան հետևյալ տեսքը՝

$$D_2 = \bar{x}Q_2 + x\bar{Q}_2\bar{Q}_1,$$

$$D_1 = xQ_2 + Q_1,$$

$$y = \bar{Q}_2Q_1:$$

Այս բանաձևերով նկարագրվող համակցական շղթայի երկու մակարդակով իրականացման համար անհրաժեշտ են 5 հատ երկմուտք տրամաբանական տարրեր, մեկ եռամուտք տարր և մեկ շրջիչ: Այսպիսով, համակցական շղթայի բարդությունը, համակարգի միևնույն մուտք-ելք կախվածությունների դեպքում, կախված է վիճակների կոդավորումից:

Տարբեր ընթացակարգեր են առաջարկված վիճակների հնարավոր հաջորդականություններից մեկի՝ օպտիմալի, ընտրության համար: Վիճակի կոդերի  $r$  բիթերի դեպքում  $P$  վիճակների հաջորդականության ընտրության համար կան  $2^{r-1}! / (2^r - P)!r!$  հնարավոր տարբերակներ ( $P \leq 2^r$ ): Նույնիսկ ոչ մեծ  $r$ -ի դեպքում այդ տարբերակների թիվը չափազանց մեծ է: Վիճակների օպտիմալ հաջորդականությունը պետք է բերի նվազագույն բարդության համակցական շղթայի: Սակայն դեռևս գոյություն չունի վիճակների նշանակման այնպիսի ալգորիթմ, որը կերաշխավորի վիճակների հաջորդականության օպտիմալ ընտրությունը:

Գոյություն ունեն վիճակների նշանակման մի շարք մոտեցումներ.

- հաջորդական՝ վիճակները թվարկվում են (կոդավորվում են) անցումների աղյուսակում դրանց հայտնվելու հաջորդականությամբ:
- պատահական՝ վիճակներին վերագրվող կոդերն ընտրվում են պատահական հաջորդականությամբ:
- մեկը-տաք՝ վիճակի կոդի բիթերի թիվը հավասար է վիճակների թվին: Յուրաքանչյուր վիճակի կոդում միայն մեկ բիթն է 1 (տաք):
- էվրիստիկ՝ ելնելով որոշակի հատկություններից, կանխագուշակումից և փորձից: Չի երաշխավորում օպտիմալ լուծում, բայց շատ դեպքերում բերում է զգալի պարզեցման:

Թվարկված մոտեցումների մի մասն իրականացվել է ալգորիթմների տեսքով ավտոմատացված նախագծման ծրագրային գործիքներում: Քննարկենք դրանցից մի քանիսը:

**Վիճակների նշանակման «մեկը-տաք» եղանակը** պարզ է և հեշտ կոդավորվող ու ստուգվող: Օրինակ, չորս վիճակների «մեկը-տաք» կոդավորումը կլինի հետևյալը՝

$S_0=0001$ ;

$S_1=0010$ ;

$S_2=0100$ ;

$S_3=1000$ .

Կարելի է տեսնել, որ տվյալ տրիգերի մուտքի ֆունկցիայում մասնակցում են միայն նախորդ վիճակի բիթերը (հիշենք, որ  $Q^+=D$ ): Հետևաբար, տրիգերների մուտքային ֆունկցիաները ձևավորող համակցական շղթան կլինի պարզ: Սա «մեկը-տաք»



կողավորման հիմնական առավելությունն է: Օրինակ, աղյուսակ 4.41-ով տրված անցումների աղյուսակից կստացվեն տրիգերների մուտքերի հետևյալ ֆունկցիաները՝

Աղյուսակ 4.41

Ներկա վիճակ				Հաջորդ վիճակ			
Q4	Q3	Q2	Q1	Q4 <sup>+</sup>	Q3 <sup>+</sup>	Q2 <sup>+</sup>	Q1 <sup>+</sup>
0	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	0
1	0	0	0	0	0	0	1

$$D1 = Q4\overline{Q3}\overline{Q2}\overline{Q1},$$

$$D2 = \overline{Q4}\overline{Q3}\overline{Q2}Q1,$$

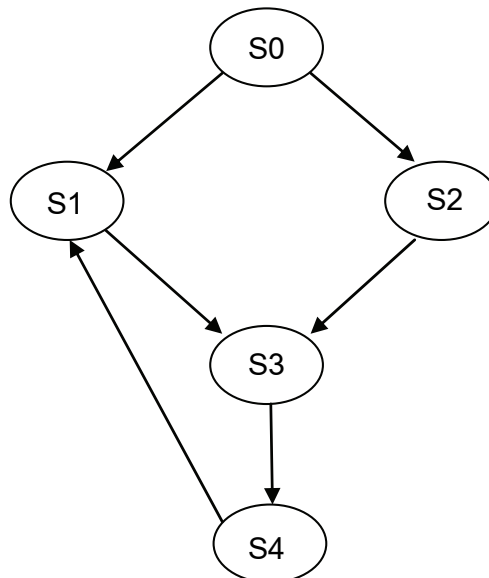
$$D3 = \overline{Q4}\overline{Q3}Q2\overline{Q1},$$

$$D4 = \overline{Q4}Q3\overline{Q2}\overline{Q1}:$$

«Մեկը-տաք» կողավորումը վիճակների կողավորման հիմնական եղանակն է ծրագրավորվող տրամաբանական ԻՍ-ով (ԾՏՍ, ԿԾՓԶ) հաջորդական համակարգերի կառուցման դեպքում: Այս եղանակի թերությունն այն է, որ այն պահանջում է մեծ թվով տրիգերներ՝ տրիգերների թիվը հավասար է վիճակների թվին: Օրինակ, 64 վիճակներ երկուական կոդով կողավորելու համար անհրաժեշտ է ունենալ 6 տրիգեր, իսկ «մեկը-տաք» կողավորման դեպքում՝ 64 տրիգեր:

Դիտարկենք մի քանի էվրիստիկ մոտեցումներ:

**Նվազագույն թվով բիթերի փոփոխության չափանիշ:** Լավագույն է համարվում վիճակների կոդերի այնպիսի նշանակումը, որի դեպքում վիճակների բոլոր անցումների վրա վիճակի բիթերի փոփոխությունների թիվը նվազագույնն է: Դիտարկենք նկ. 4.51-ում ցույց տրված անցումների գրաֆը:



Նկ. 4.51. Ավտոմատի անցումների գրաֆը

Աղյուսակ 4.42-ում ցույց են տրված վիճակների կոդավորման երկու տարբերակ:

Աղյուսակ 4.42

Վիճակների կոդավորում1				Վիճակների կոդավորում2			
State	Q3	Q2	Q1	State	Q3	Q2	Q1
S0	0	0	0	S0	0	0	0
S1	1	0	1	S1	0	0	1
S2	1	1	1	S2	0	1	0
S3	0	1	0	S3	0	1	1
S4	0	1	1	S4	1	1	1

Վիճակների կոդավորման այդ երկու տարբերակներում անցումների բիթերի փոփոխությունների թվերը ներկայացված են աղյուսակ 4.43-ում: Օրինակ,  $S0(000) \rightarrow S1(101)$  անցման ժամանակ կոդավորման առաջին տարբերակի դեպքում փոփոխվում է վիճակի երկու բիթ:

Աղյուսակ 4.43

Անցումը	կոդավորում 1	կոդավորում 2
	բիթի փոփոխություններ	բիթի փոփոխություններ
S0 to S1	2	1
S0 to S2	3	1
S1 to S3	3	1
S2 to S3	2	1
S3 to S4	1	1
S4 to S1	2	2
Բիթի ընդհանուր փոփոխությունները	13	7

Որպեսզի վիճակի բիթերի փոփոխությունների թիվը փոքր լինի, պետք է ներկա վիճակի և հաջորդ վիճակի կոդերի հեռավորությունը լինի նվազագույնը: Դրա համար վիճակների կոդերը պետք է ընտրել այնպես, որ Կառնոյի քարտում ներկա և հաջորդ վիճակները դասավորված լինեն հարևան վանդակներում (ինչքանով որ դա հնարավոր է): Կոդավորման երկու տարբերակներին համապատասխանող Կառնոյի քարտերը ցույց են տրված նկ. 4.52-ում:

		$Q_2Q_1$			
		00	01	11	10
$Q_3$	0	S0		S4	S3
	1		S1	S2	

		$Q_2Q_1$			
		00	01	11	10
$Q_3$	0	S0	S1	S3	S2
	1			S4	

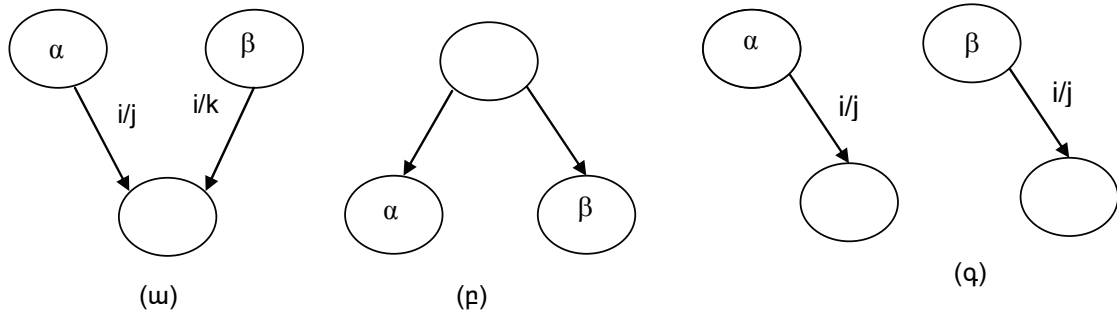
Նկ. 4.52. Կոդավորման երկու տարբերակներին համապատասխանող Կառնոյի քարտերը

Նկատենք, որ նվազագույն կոդային հեռավորությամբ վիճակների կոդավորումը օպտիմալ լուծման չի հանգեցնում:

**Էվրիստիկ մոտեցում՝ հիմնված մուտք-ելք վարքագծի և անցումների թվի վրա:**  
Այս մոտեցումը նույնպես հիմնված է նվազագույն հեռավորությամբ կոդավորման վրա:

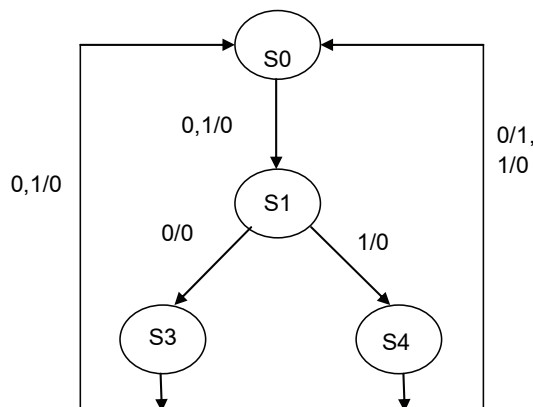
Վիճակները խմբավորվում են տարբեր գերակայության դասերով՝

- ամենաբարձր գերակայության դաս. վիճակները կիսում են միևնույն հաջորդ վիճակը (նկ. 4.53ա),
- միջին գերակայության դաս. վիճակները կիսում են ընդհանուր նախորդ վիճակ (նկ. 4.53բ),
- ամենացածր գերակայության դաս. վիճակներն ունեն ընդհանուր ելքի վարքագիծ (նկ. 4.53գ):



Նկ. 4.53. Վիճակների նշանակման գերակայությունների դասեր՝ (ա) ամենաբարձր գերակայության դաս, (բ) միջին գերակայության դաս, (գ) ամենացածր գերակայության դաս

**Օրինակ 4.12.** Նկ. 4.54-ում ցույց է տրված եռաբիթ (010, 110) հաջորդականությունների հայտնաբերման ավտոմատի (տե՛ս Օրինակ 4.6) անցումների գրաֆը, որը կառուցված է անցումների նվազարկված աղյուսակից (աղյուսակ 4.26): S3, S4 վիճակները կազմում են առավելագույն գերակայության խումբ, քանի որ դրանք ունեն ընդհանուր հաջորդ վիճակ՝  $x=0$  և  $x=1$  մուտքերի դեպքում: Դրանք նաև միջին գերակայության խումբ են կազմում, քանի որ ունեն ընդհանուր նախորդ վիճակ՝ S1: Ամենացածր գերակայության խմբեր են կազմում (S0, S1, S3) վիճակները, որոնք ունեն ընդհանուր մուտք/ելք՝ 0/0 և (S0, S1, S3, S4) վիճակները, որոնք ունեն ընդհանուր մուտք/ելք՝ 1/0: Նկ. 4.55ա,բ քարտերում ամենաբարձր գերակայության (S3, S4) վիճակները տեղադրված են հարևան վանդակներում: Վիճակների այդ երկու նշանակումները գրեթե ոչինչով չեն տարբերվում:



Նկ. 4.53. Եռաբիթ հաջորդականությունների հայտնաբերման ավտոմատի անցումների գրաֆը (օրինակ 4.12)

		$Q_1$	
		0	1
$Q_2$	0	S0	S1
	1	S3	S4

(ա)

		$Q_1$	
		0	1
$Q_2$	0	S0	S3
	1	S1	S4

(բ)

Նկ. 4.55. Վիճակների նշանակումը ըստ գերակայության դասերի

**Օրինակ 4.13.** Քառաբիթ հաջորդականությունների հայտնաբերման ավտոմատի անցումների գրաֆը ցույց է տրված նկ. 4.13-ում (օրինակ 4.5). Ամենաբարձր գերակայության խմբերն են՝ (S3', S4') և (S7', S10'): Միջին գերակայության խմբերն են՝ (S1, S2), (S3', S4') x2 (դրանք ունեն երկու ընդհանուր նախորդ վիճակ), (S7', S10'). Ամենացածր գերակայության խմբերն են՝ (S0, S1, S2, S3', S4', S7'), որոնք ունեն 0/0 մուտք/ելք և (S0, S1, S2, S3', S4', S7'), որոնք ունեն 1/0 մուտք/ելք: Վիճակների նշանակման քայլերը ցույց են տրված նկ. 4.56-ում:

		$Q_2Q_1$			
		00	01	11	10
$Q_3$	0	S0			
	1				

Քայլ 1

		$Q_2Q_1$			
		00	01	11	10
$Q_3$	0	S0		S3'	
	1			S4'	

Քայլ 2

		$Q_2Q_1$			
		00	01	11	10
$Q_3$	0	S0		S3'	S7'
	1			S4'	S10'

Քայլ 3

		$Q_2Q_1$			
		00	01	11	10
$Q_3$	0	S0	S1	S3'	S7'
	1		S2	S4'	S10'

Step

Նկ. 4.56. Վիճակների նշանակման քայլերը՝ ներկայացված է երկու տարբերակ:

Սկզբում նշանակվում է S0-ն, S0=000, համարելով, որ S0-ն սկզբնական վիճակն է, որը պետք է հաստատվի ասինքրոն սկզբնական վիճակի կարգման ազդանշանի կիրառումից հետո: Այնուհետև տեղադրվում են ամենաբարձր գերակայության խմբերի վիճակները: Արդյունքում, (S3', S4'), (S7', S10') և (S1, S2) վիճակները տեղադրված են հարևան վանդակներում:

Երկու տարբերակով վիճակների նշանակումները ցույց են տրված աղյուսակներ 4.44-ում և 4.45-ում: Հաջորդ վիճակի քարտերը ցույց են տրված նկ. 4.57-ում և 4.58-ում: Կողավորման առաջին տարբերակը ցուցաբերում է ավելի լավ ստանձնման հնարավորություններ հաջորդ վիճակի քարտերում: Այսպիսով, մուտք-ելք վարքագծի և անցումների թվի վրա հիմնված էվրիստիկ ալգորիթմը չի զուգամիտում մեկ օպտիմալ լուծման: Այս մոտեցումով պետք է ստանալ բոլոր հնարավոր կողավորումները և դրանցից ընտրել լավագույնը:

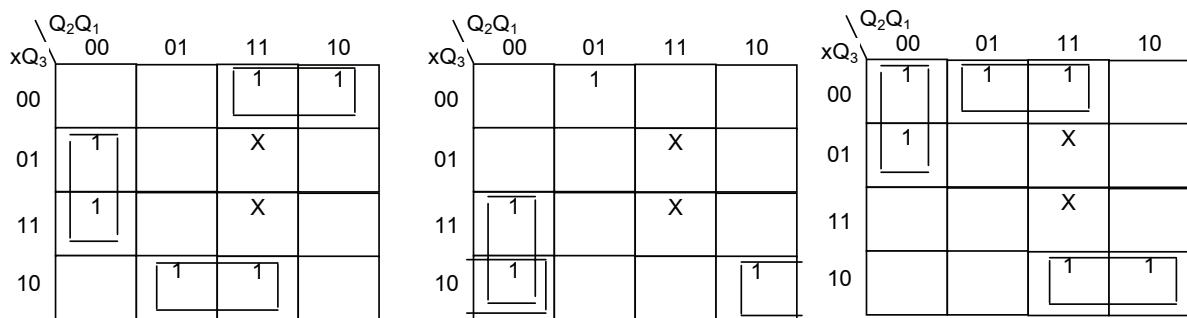
Աղյուսակ 4.44

Մուտք	Ներկա վիճակ (Տարբերակ 1)				Հաջորդ վիճակ (Տարբերակ 1)				Ելք
x	S	Q3	Q2	Q1	S <sup>+</sup>	Q3 <sup>+</sup>	Q2 <sup>+</sup>	Q1 <sup>+</sup>	y
0	S0	0	0	0	S1	0	0	1	0
0	S1	0	0	1	S3'	0	1	1	0
0	S2	1	0	1	S4'	1	1	1	0
0	S3'	0	1	1	S7'	0	1	0	0
0	S4'	1	1	1	S7'	0	1	0	0
0	S7'	0	1	0	S0	0	0	0	0
0	S10'	1	1	0	S0	0	0	0	1
1	S0	0	0	0	S2	1	0	1	0
1	S1	0	0	1	S4'	1	1	1	0
1	S2	1	0	1	S3'	0	1	1	0
1	S3'	0	1	1	S7'	0	1	0	0
1	S4'	1	1	1	S10'	1	1	0	0
1	S7'	0	1	0	S0	0	0	0	0
1	S10'	1	1	0	S0	0	0	0	0

Նկ. 4.57. Հաջորդ վիճակի քարտերը վիճակների նշանակման տարբերակ1-ի համար

Աղյուսակ 4.45

Մուտք	Ներկա վիճակ (Տարբերակ 2)				Հաջորդ վիճակ (Տարբերակ 2)				Ելք
x	S	Q3	Q2	Q1	S <sup>+</sup>	Q3 <sup>+</sup>	Q2 <sup>+</sup>	Q1 <sup>+</sup>	y
0	S0	0	0	0	S1	0	0	1	0
0	S1	0	0	1	S3'	0	1	1	0
0	S2	0	1	0	S4'	1	0	0	0
0	S3'	0	1	1	S7'	1	0	1	0
0	S4'	1	0	0	S7'	1	0	1	0
0	S7'	1	0	1	S0	0	0	0	0
0	S10'	1	1	0	S0	0	0	0	1
1	S0	0	0	0	S2	0	1	0	0
1	S1	0	0	1	S4'	1	0	0	0
1	S2	0	1	0	S3'	0	1	1	0
1	S3'	0	1	1	S7'	1	0	1	0
1	S4'	1	0	0	S10'	1	1	0	0
1	S7'	1	0	1	S0	0	0	0	0
1	S10'	1	1	0	S0	0	0	0	0

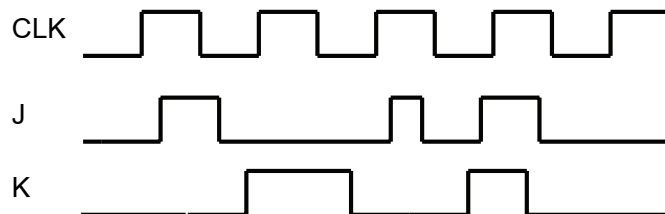


Նկ. 4.58. Հաջորդ վիճակի քարտերը վիճակների նշանակման տարբերակ 2-ի համար

#### 4.17. Խնդիրներ

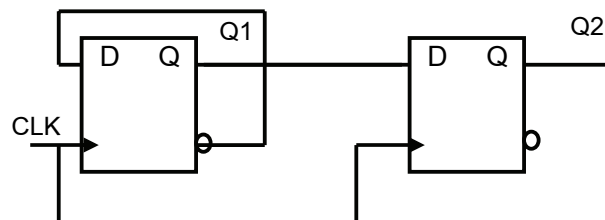
**Խնդիր 4.1.** Կառուցել Մուրի ավտոմատի անցումների գրաֆը, որն ունի երկու մուտք՝  $a$ ,  $b$  և մեկ ելք՝  $y$ : Երբ տրվում է  $a=1$  ազդանշան, ելքը հաստատուն է և հավասար 0-ի: Երբ  $a=0$ ,  $y$ -ը շարունակում է մնալ 0 այնքան ժամանակ, քանի դեռ երկու հաջորդական տակտերում  $b$  ազդանշանի արժեքը փոխվում է: Երբ երկու հաջորդական տակտերում  $b$  ազդանշանի արժեքը մնում է հաստատուն՝ 1 կամ 0,  $y$  ազդանշանը ընդունում 1 արժեք և շարունակում է մնալ 1, քանի դեռ չի տրվել  $a=0$ :

**Խնդիր 4.2.** Նկ.4.59-ում ցույց է տրված դրական ճակատով տակտավորվող տեր-ծառա JK տրիգերի մուտքային ազդանշանների ժամանակային դիագրամները: Համարելով, որ տրիգերը նախապես գտվում էր 0 վիճակում, ցույց տալ տրիգերի երկու աստիճանների ելքային ազդանշանների ժամանակային դիագրամները:



Նկ. 4.59. Խնդիր 4.2-ին վերաբերող ժամանակային դիագրամներ

**Խնդիր 4.3.** Ինչ վիճակում կհայտնվի նկ. 4.60-ում ցույց տրված ավտոմատը տակտային չորս իմպուլս կիրառելուց հետո, եթե սկզբնական վիճակն է  $Q1Q2=1x$ :



Նկ. 4.60. Խնդիր 4.3-ին վերաբերող շղթան

**Խնդիր 4.4.** Կառուցել դրական ճակատով տակտավորվող տեր-ծառա D-տրիգեր, որն ունի նաև ասինքրոն  $\bar{R}$  և  $\bar{S}$  մուտքեր՝ օգտագործելով միայն ԵՎ-ՈՉ տարրեր: Տրիգերի մուտքերն են՝  $D$ ,  $CLK$ ,  $\bar{R}$ ,  $\bar{S}$ , իսկ ելքը՝  $Q$ : Համարակալել ԵՎ-ՈՉ տարրերը և պատաս -

խանել հետևյալ հարցերին.

(ա) Երբ  $CLK=0$ ,  $D=1$  և  $\bar{R} = \bar{S}=1$ , որոշել յուրաքանչյուր ԵՎ-ՈՉ տարրի ելքի վիճակը:

Համարել, որ տրիգերը նախապես գտնվում էր 0 վիճակում:

(բ) Կրկնել (ա) հարցի պահանջները  $CLK=1$  պահից հետո:

(գ) Երբ  $CLK=0$ ,  $D=1$ ,  $\bar{R}=0$ ,  $\bar{S}=1$ , որոշել յուրաքանչյուր ԵՎ-ՈՉ տարրի ելքի վիճակը:

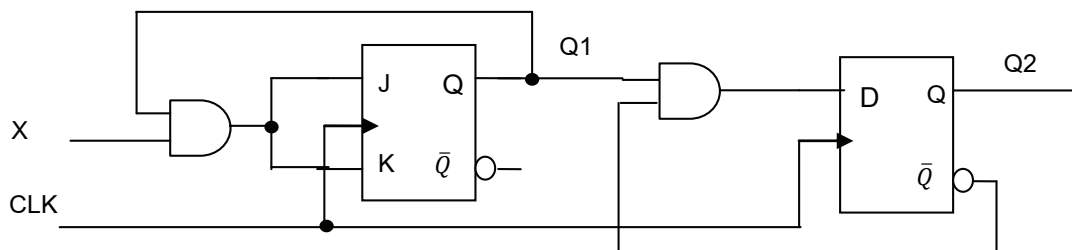
(դ) Կրկնել (գ) հարցի պահանջները  $CLK=1$  պահից հետո:

**Խնդիր 4.5.** MN-տրիգերն աշխատում է հետևյալ կերպ. երբ  $M = 1$ , հաջորդ վիճակը հավասար է ընթացիկի բացասմանը, երբ  $M = 0$ , հաջորդ վիճակը հավասար է N-ի արժեքին:

ա) Կառուցել MN-տրիգերի սխեման D-տրիգերի և մուլտիպլեքսորի հիման վրա, դուրս բերել տրիգերի իսկության աղյուսակը:

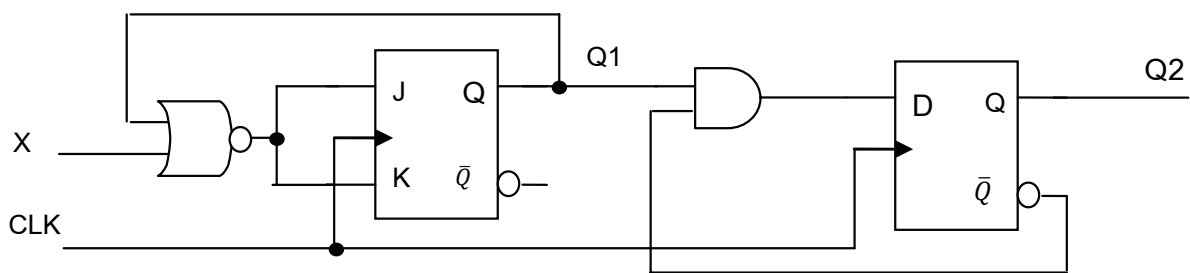
բ) Կառուցել MN-տրիգերի սխեման JK-տրիգերի հիման վրա՝ ավելացնելով տրամաբանական տարրեր:

**Խնդիր 4.6.** Կառուցել նկ. 4.61-ում ցույց տրված հաջորդական շղթայի անցումների աղյուսակը:



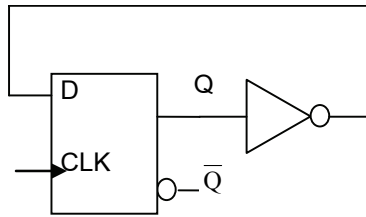
Նկ. 4.61. Խնդիր 4.6-ին վերաբերող շղթան

**Խնդիր 4.7.** Կառուցել նկ. 4.62-ում ցույց տրված հաջորդական շղթայի անցումների աղյուսակը:



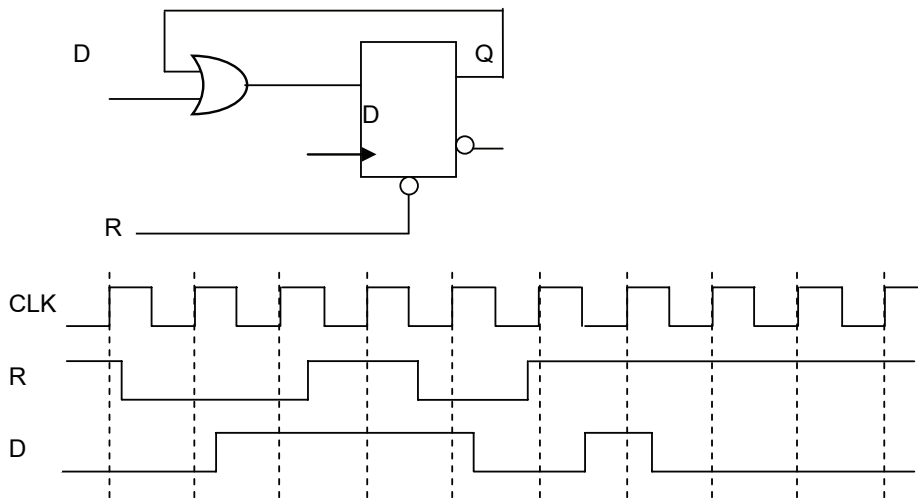
Նկ. 4.62. Խնդիր 4.7-ին վերաբերող շղթան

**Խնդիր 4.8.** Նկ. 4.63-ում ցույց է տրված հաճախականության բաժանիչի սխեման: Որոշել տակտային իմպուլսների նվազագույն պարբերությունը, եթե  $t_{su}=200\mu\text{վ}$ ,  $t_{hd}=-150\mu\text{վ}$ ,  $t_{c2q}=500\mu\text{վ}$ ,  $t_{pinv}=60\mu\text{վ}$ :



Նկ. 4.63. Խնդիր 4.8-ին վերաբերող շղթան

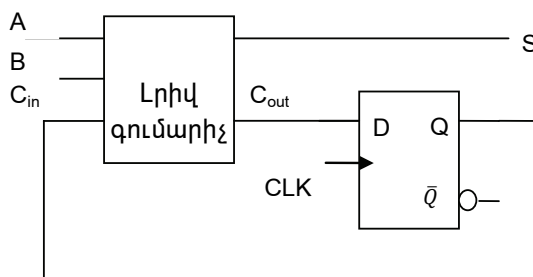
**Խնդիր 4.9.** Նկ. 4.64-ում ցույց տրված ժամանակային դիագրամներին ավելացնել տրիգերի ելքի ժամանակային դիագրամը, համարելով, որ սկզբնական վիճակը 0 է:



Նկ. 4.64. Խնդիր 4.9-ին վերաբերող շղթան և ժամանակային դիագրամները

Որոշել տակտային իմպուլսների նվազագույն պարբերությունը, եթե  $t_{su}=100\mu s$ ,  $t_{hd}=-50\mu s$ ,  $t_{c2q}=100\mu s$ ,  $t_{por}=150\mu s$ :

**Խնդիր 4.10.** Նկ. 4.65-ում պատկերված բազմաբիթ երկուսական թվերի գումարման հաջորդական գումարիչի սխեման ունի երկու մուտք՝ A, B և մեկ ելք՝ S: Սխեման բաղկացած է D-տրիգերից և լրիվ գումարիչից: Լրացնել անցումների աղյուսակը:

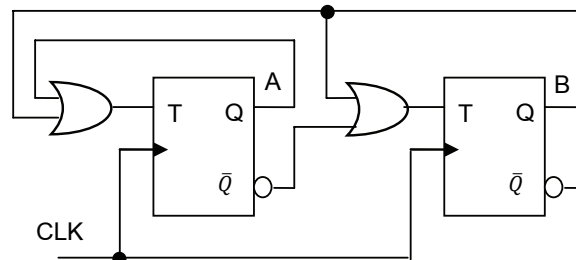


A	B	$C_{in}$	$C_{out}$	S
0	0	0		
0	1			
1	0			
1	1			
0	0			
0	1			
1	0			
1	1			
0	0			

Նկ. 4.65. Խնդիր 4.10-ին վերաբերող շղթան և անցումների աղյուսակը (չլրացրած)



**Խնդիր 4.11.** Ստանալ նկ. 4.66-ում ցույց տրված հաջորդական շղթայի վիճակների աղյուսակը: Ի՞նչ ֆունկցիա է իրականացնում այս շղթան:



Նկ. 4.66. Խնդիր 4.11-ին վերաբերող շղթան

**Խնդիր 4.12.** Հաջորդական շղթան բաղկացած է չորս տրիգերից՝ A,B,C,D և ունի մեկ մուտք՝ x: Այն նկարագրվում է հետևյալ վիճակի հավասարումներով.

$$A^+ = (C\bar{D} + \bar{C}D)x + (CD + \bar{C}\bar{D})\bar{x}, B^+ = A, C^+ = B, D^+ = C:$$

ա) ցույց տալ վիճակների հաջորդականությունը՝ սկսած ABCD=0001 –ից, երբ x=1:

բ) ցույց տալ վիճակների հաջորդականությունը՝ սկսած ABCD=0000 –ից, երբ x=0:

**Խնդիր 4.13.** Նվազարկել աղյուսակ 4.46-ով տրված ավտոմատի վիճակների թիվը:

Աղյուսակ 4.46

Ներկա վիճակ	Հաջորդ վիճակ		Ելք	
	X=0	X=1	X=0	X=1
A	F	B	0	0
B	D	C	0	0
C	F	E	0	0
D	G	A	1	0
E	D	C	0	0
F	F	B	1	1
G	G	H	0	1
H	G	A	1	0

**Խնդիր 4.14.** Հաջորդական շղթան բաղկացած է երկու տրիգերից՝ A, B և ունի երկու մուտք՝ x, y: Տրիգերների մուտքի ֆունկցիաները և շղթայի ելքի ֆունկցիան ունեն հետևյալ տեսքը՝

$$J_A = xB + \bar{y}\bar{B}, K_A = x\bar{y}\bar{B},$$

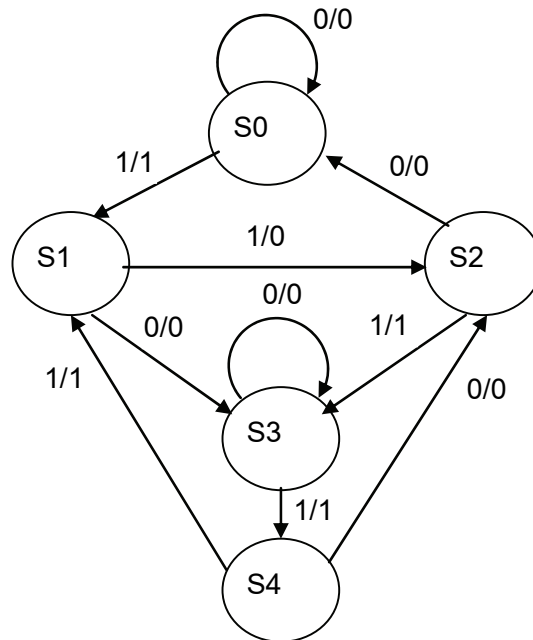
$$J_B = x\bar{A}, K_B = x\bar{y} + A,$$

$$z = xyA + \bar{x}\bar{y}B:$$

Ստանալ վիճակների հավասարումը, վիճակների գրաֆը, վիճակների աղյուսակը և կառուցել տրամաբանական սխեման:

**Խնդիր 4.15.** Հաջորդական շղթան ունի մեկ մուտք և մեկ ելք: Վիճակների գրաֆը ցույց է տրված նկ. 4.67-ում: Նախագծել հաջորդական ավտոմատը՝

- ա) T տրիգերների վրա:  
բ) RS տրիգերների վրա:  
գ) JK տրիգերների վրա:



Նկ. 4.67. Խնդիր 4.15-ին վերաբերող շղթան

**Խնդիր 4.16.** Նախագծել ավտոմատ JK տրիգերների վրա, որը նկարագրվում է հետևյալ վիճակի հավասարումներով.

$$A^+ = xAB + y\bar{A}C + xy,$$

$$B^+ = xAC + \bar{y}B\bar{C},$$

$$C^+ = \bar{x}B + yA\bar{B}:$$

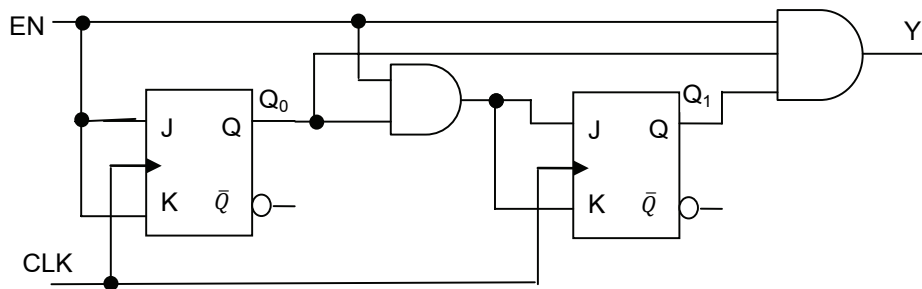
**Խնդիր 4.17.** Նախագծել ավտոմատ, որը նկարագրվում է աղյուսակ 4.47-ում ցույց տրված վիճակների աղյուսակով: Օգտագործել D տրիգերներ և տրամաբանական տարրեր:

Աղյուսակ 4.47

Ներկա վիճակ	Հաջորդ վիճակ	
	X=0	X=1
00	00	01
01	10	01
10	10	11
11	10	01

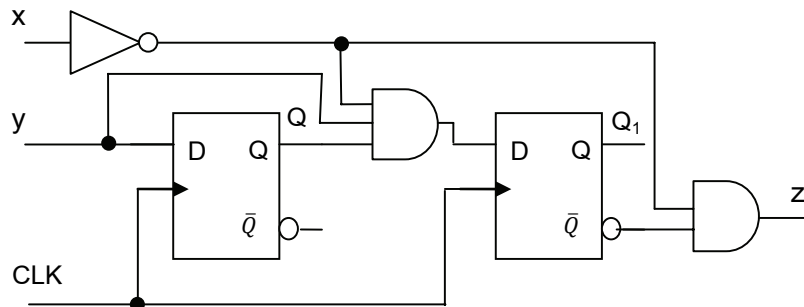
**Խնդիր 4.18.** Ստանալ նկ. 4.68–ում ցույց տրված սինթրոն ավտոմատի տրիգերների մուտքերի հավասարումները և ավտոմատի անցումների գրաֆն ու աղյուսակը: Կառու-

ցել ժամանակային դիագրամները տասը տակտի համար, համարելով սկզբնական վիճակը՝  $Q_1Q_0=00$ , իսկ  $EN=1$  ամբողջ ընթացքում:



Նկ. 4.68. Խնդիր 4.15-ին վերաբերող շղթան

**Խնդիր 4.19.** Ստանալ նկ. 4.69-ում ցույց տրված սինթրոն ավտոմատի տրիգերների մուտքերի հավասարումները և ավտոմատի անցումների գրաֆն ու աղյուսակը:



Նկ. 4.69. Խնդիր 4.19-ին վերաբերող շղթան

**Խնդիր 4.20.** Կառուցել սինթրոն ավտոմատ D տրիգերների վրա, ըստ ստորև բերված անցումների աղ. 4.48-ի:

Աղյուսակ 4.48

Ներկա վիճակ	Հաջորդ վիճակ		Ելք
	X=0	X=1	
A	B	D	0
B	C	B	0
C	B	A	1
D	B	C	0

## **Գլուխ 5. Տիպային հաջորդական թվային հանգույցներ**

Տակտավորվող հաջորդական շղթաների որոշ տիպային ֆունկցիաներ հաճախ են հանդիպում տարբեր նշանակության թվային համակարգերում: Այդպիսի համապիտանի թվային հանգույցներից են ռեգիստրները, իմպուլսների հաշվիչները և հիշողությունները: Այդ ֆունկցիաներն իրականացնող թվային շղթաները, սովորաբար, թողարկվում են միջին կամ բարձր ինտեգրացման աստիճանի ԻՍ-երի տեքով:

Ռեգիստրը տրիգերների խումբ է, որն ունակ է պահպանել երկուական տվյալներ: *r*-բիթ ռեգիստրը բաղկացած է *r* տրիգերներից և կարող է պահպանել *r* երկուական կարգերով բառ: Բացի տրիգերներից, ռեգիստրը կարող է նաև ունենալ լրացուցիչ համակցական շղթաներ, որոնք հնարավորություն են տալիս տվյալների պահպանման հետ միաժամանակ կատարել գործողություններ այդ տվյալներով: Տրիգերները պահպանում են տվյալները, իսկ համակցական շղթաները կառավարում են նոր տվյալների գրանցման ֆունկցիաները:

Իմպուլսների հաշվիչը նույնպես կարելի է ներկայացնել իբրև ռեգիստր՝ տրիգերների խումբ: Սակայն հաշվիչը պետք անցնի վիճակների նախանշված հաջորդականության՝ տակտային իմպուլսների ազդեցության տակ: Չնայած, որ հաշվիչները նույնպես ռեգիստրներ են, հաշվի առնելով թվային համակարգերում դրանցով իրականացվող հաջորդական ֆունկցիայի կարևորությունը, հաշվիչները դասկարգվում են որպես առանձին խմբի տիպային շղթաներ:

Հիշասարքը հիշողության տարրերի հավաքածու է, որը կարող է պահպանել մեծ ծավալի երկուական տվյալներ: Կախված հիշասարքով իրականացվող ֆունկցիաներից՝ դրանք դասակարգվում են միայն ընթերցվող հիշողությունների (ՄԸՀ, Read-Only Memory, ROM) և ընթերցվող-գրանցվող հիշողությունների (ԸԳՀ, Random Access Memory, RAM): ՄԸՀ պահպանում է տվյալները և պահանջված պահի դրանք փոխանցում էլք: ԸԳՀ-ն կարող է ինչպես էլք փոխանցել պահպանվող տվյալները, այնպես էլ ընդունել՝ գրանցել հիշողության տարրերում նոր տվյալներ:

Ռեգիստրների, հաշվիչների և հիշասարքերի կիրառությունը թույլ է տալիս զգալի պարզեցնել բարդ թվային համակարգերի կառուցման խնդիրը:

### **5.1. Իմպուլսների թվի հաշվիչներ**

Հաշվիչները թվային ավտոմատներ են, որոնք հաշվում են մուտքին տրված իմպուլսների թիվը և պահպանում են այդ թվի կոդը: Հաշվիչի հաշվային մուտքին կիրառված տակտային ազդանշանները ստիպում են հաշվիչին անցնել վիճակների նախապես տրված հաջորդականության: Վիճակների այդ փոփոխությունը ցիկլիկ կրկնվում է: Եթե հաշվիչի վիճակների թիվը հավասար է *M*-ի, ապա *M* տակտային ազդանշաններից հետո, հաշվիչն անցնելով բոլոր վիճակներով, կվերադառնա սկզբնական վիճակին: Վերադարձը սկզբնական վիճակին վկայում է գերլցման մասին: Հաշվիչի վիճակների թիվը կոչվում է նաև վերահաշվման գործակից: Հաշվիչի վիճակներից մեկը համարվում է սկզբնական: Սովորաբար դա համապատասխանում է բոլոր տրի-

գերների 0 վիճակին: Եթե սկզբնական վիճակում գտնվող հաշվիչի մուտքին կիրառվի  $N$  տակտային իմպուլսների հաջորդականություն, ապա  $N > M$  դեպքում հաշվիչի վերջնական վիճակի կողը կլինի  $N$ -ը  $M$ -ի վրա բաժանման մնացորդը: Այդ պատճառով հաճախ  $M$  վերահաշվման գործակցով հաշվիչը կոչվում է մոդուլ  $M$ -ով հաշվիչ: Օրինակ, եթե  $M=10$  և աշխատանքն սկսվում է 0 սկզբնական վիճակից, ապա  $N=27$  իմպուլսներից հետո հաշվիչը կհայտնվի  $N \bmod M = \text{մնացորդ}(N/M) = \text{մնացորդ}(27/10) = 7$  վիճակում: Յուրաքանչյուր  $M=10$  իմպուլսից հետո հաշվիչն անցնում 0 վիճակով: Եթե հաշվիչի աշխատանքն սկսվում է կամայական  $S_0$  վիճակից, ապա  $N$  իմպուլսներից հետո հաշվիչի վերջնական վիճակի կողը կլինի  $((S_0 + N)/M)$  բաժանման մնացորդը: Օրինակ,  $M=12$  հաշվիչն աշխատանքն սկսելով  $S_0=6$  վիճակից՝ 135 իմպուլս հետո կհայտնվի մնացորդ  $((135+6)/12)=9$  վիճակում:

Հաշվիչները կարող են լինել՝

- սինքրոն. բոլոր տրիգերները տակտավորվում են միևնույն ազդանշանով,
- ասինքրոն (կամ բաբախող). հաջորդ տրիգերը տակտավորվում է նախորդի ելքային ազդանշանով,
- երկուական. վերահաշվման գործակիցը երկուսի ամբողջ թիվ աստիճան է՝  $M=2^r$ ,  $r$ -ը տրիգերների թիվն է,
- ոչ երկուական.  $M < 2^r$ ,
- փոփոխական կամ ծրագրավորվող վերահաշվման գործակցով,
- գումարող. հաջորդ վիճակի կողը նախորդից մեկով մեծ է՝  $S_{n+1} = S_n + 1$ ,
- հանող. հաջորդ վիճակի կողը նախորդից մեկով փոքր է՝  $S_{n+1} = S_n - 1$ ,
- դարձափոխելի. հաշվի ուղղությունը կարելի է կառավարել  $UD$  ազդանշանով՝  $UD=1$  դեպքում հաշվիչը գումարող է,  $UD=0$  դեպքում՝ հանող:

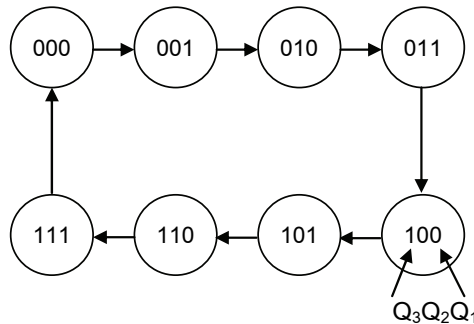
Հաշվիչը կարող է ունենալ նաև սկզբնական 0 վիճակի կարգման  $R$  մուտք, կամ կամայական  $SI$  սկզբնական վիճակի կարգման մուտքեր  $S_1, \dots, S_r$ :

Հաշվիչները հաճախ օգտագործվում են որպես հաճախականության բաժանիչներ: Այդ դեպքում որպես բաժանիչի ելք վերցնում են վերջին տրիգերի ելքը: Եթե մուտքային տակտային ազդանշանի հաճախականությունը  $F_{in}$  է, ապա ելքային ազդանշանինը կլինի՝  $F_{out} = F_{in}/M$ :

Հաշվիչների հիմնական պարամետրերն են վերահաշվման գործակիցը և արագագործությունը: Արագագործությունը գնահատվում է թույլատրող ունակությամբ՝  $t_{min}$  և ելքային կողի նոր արժեքի հաստատման ժամանակով՝  $t_{st}$ : Թույլատրող ունակությունը որոշվում է երկու տակտային ազդանշանների միջև ընկած այն ամենափոքր ժամանակով, որի դեպքում դեռևս հաշվման խափանում չի նկատվում: Հաճախ ավելի հարմար է օգտվել հակադարձ մեծությունից՝ հաշվման առավելագույն հաճախականությունից՝  $F = 1/t_{min}$ : Հաստատման ժամանակը՝  $t_{st}$  հավասար է տակտային ազդանշանի փոփոխության պահի և հաշվիչի նոր վիճակին անցման ավարտման պահի միջև ընկած ժամանակահատվածին:

## 5.2. Երկուական հաշվիչներ

Որպես երկուական հաշվիչի կառուցման օրինակ դիտարկենք  $M=2^3=8$  վերահաշվման գործակցով հաշվիչը: Հաշվիչի անցումների գրաֆը ցույց է տրված նկ. 5.1-ում: Ինչպես կարելի է տեսնել գրաֆի գագաթներում գրված եռաբիթ վիճակներից, տրիգերների ելքերը հաջորդաբար անցնում են երկուական կոդի արժեքներով՝ 000-ից անցնելով 111-ին: Գրաֆի աղեղները նշված չեն մուտք/ելք արժեքներով: Վիճակների փոփոխությունները տեղի են ունենում միայն տակտային ազդանշանների կիրառման պահերին: Եթե տակտային իմպուլս չի կիրառվում հաշվիչը պահպանում է ներկա վիճակը: Այդ պատճառով տակտային CLK ազդանշանը բացահայտ չի նշվում գրաֆի աղեղների վրա: Հաշվիչի միակ մուտքը տակտային ազդանշանն է, իսկ ելքերը համընկնում են տրիգերների ելքերի հետ, և դրանց նշումը աղեղների վրա իմաստ չունի: Հաշվիչի հաջորդ վիճակն ամբողջությամբ որոշվում է ներկա վիճակից:



Նկ. 5.1.  $M=8$  վերահաշվման գործակցով երկուական հաշվիչի անցումների գրաֆը

$M$  վերահաշվման գործակցով երկուական հաշվիչը կառուցվում է  $r=\log_2 M$  տրիգերների միջոցով: Հաշվիչի սինթեզը կատարվում է ավտոմատների սինթեզի ալգորիթմով: Հաշվիչը կարելի է կառուցել ցանկացած տիպի տրիգերների միջոցով, սակայն, ավելի հարմար են  $T$  և  $JK$  տրիգերները: Աղյուսակ 5.1-ում ցույց են տրված  $M=8$  հաշվիչի վիճակների անցումները՝ լրացված են անցումների գրաֆից, իսկ տրիգերների մուտքի ֆունկցիաները՝  $T$  տրիգերի բնութագրիչ աղ. 4.43-ից: Տրիգերների մուտքի ֆունկցիաների փոփոխականները ավտոմատի ներկա վիճակի փոփոխականներն են: Նկ. 5.2-ում ցույց են տրված տրիգերների մուտքի ֆունկցիաների Կառնոյի քարտերը: Այդ քարտերից ստացված նվազարկված բանաձևերն ունեն հետևյալ տեսքը՝

Աղյուսակ 5.1

Ներկա վիճակ			Հաջորդ վիճակ			Տրիգերների մուտքեր		
Q3	Q2	Q1	Q3	Q2	Q1	T3	T2	T1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

$$T_3 = Q_2 Q_1, \quad (5.1)$$

$$T_2 = Q_1, \quad (5.2)$$

$$T_1 = 1: \quad (5.3)$$

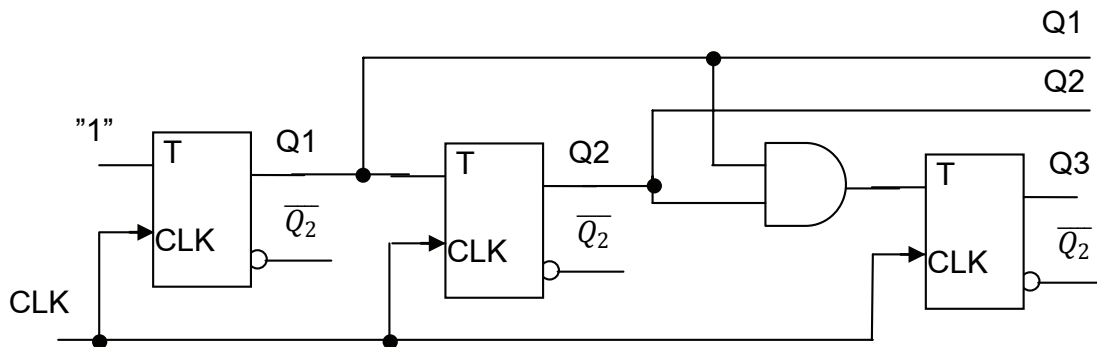
$T_3$	$Q_2 Q_1$	00	01	11	10
0	Q			1	
1	Q			1	

$T_2$	$Q_2 Q_1$	00	01	11	10
0	Q		1	1	
1	Q		1	1	

$T_1$	$Q_2 Q_1$	00	01	11	10
0	Q	1	1	1	1
1	Q	1	1	1	1

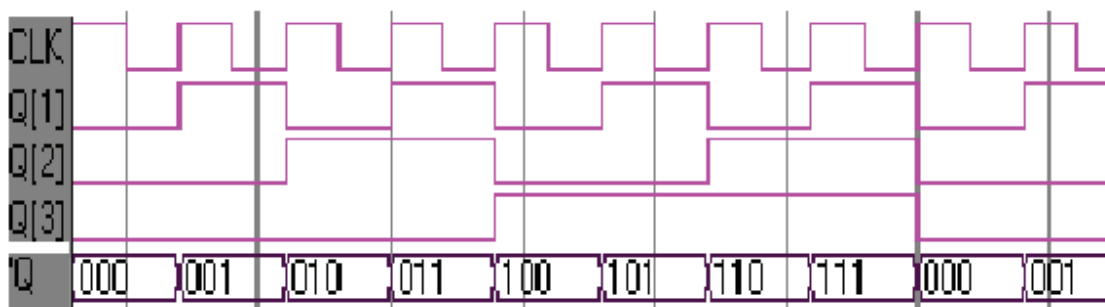
Նկ. 5.2.  $M=8$  հաշվիչի կառուցիկ քարտերը

Նկ. 5.3-ում բերված է T տրիգերների վրա իրականացված հաշվիչի սխեման, որը կառուցված է ըստ (5.1)- (5.3) բանաձևերի:



Նկ. 5.3.  $M=8$  երկուական հաշվիչի սխեման

Հաշվիչի ժամանակային դիագրամները ցույց են տրված նկ. 5.4-ում: Հաշվիչի վիճակի փոփոխությունները տեղի են ունենում տակտային CLK ազդանշանի դրական ճակատի ազդեցությամբ:  $Q_1$  տրիգերը վիճակը փոխում է յուրաքանչյուր տակտային իմպուլսով, քանի որ նրա T մուտքին տրված է հաստատուն տրամաբանական 1:

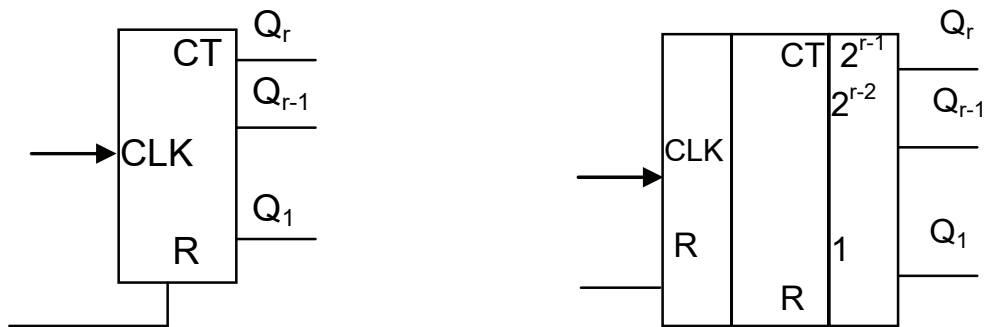


Նկ. 5.4.  $M=8$  երկուական հաշվիչի ժամանակային դիագրամները

$Q_2$  տրիգերը վիճակը փոխում է, երբ CLK-ի դրական ճակատի տրման պահին  $T_2=Q_1=1$ : Իսկ  $Q_3$ -ը շրջվում է, երբ CLK-ի դրական ճակատի տրման պահին

$T_3=Q_2 \& Q_1=1$ , այսինքն, երբ  $Q_2$  և  $Q_1$ -ը միաժամանակ հավասար են 1-ի: Ժամանակային դիագրամներից կարելի է տեսնել, որ  $Q_1$ -ի վիճակի փոփոխությունների պարբերությունը երկու անգամ մեծ է  $CLK$  իմպուլսների պարբերությունից՝  $T_1=2T_{CLK}$ ,  $Q_2$ -ինը՝ չորս անգամ՝  $T_2=4T_{CLK}$ , և  $T_3=8T_{CLK}$ :  $Q_3, Q_2, Q_1$  ազդանշանների հաճախականությունները կլինեն՝  $F_1=F_{CLK}/2$ ,  $F_2=F_{CLK}/4$ ,  $F_3=F_{CLK}/8$ : Հետևաբար, հաշվիչը կարող է օգտագործվել նաև որպես հաճախականության բաժանիչ, որը բաժանում է տակտային զենեքատորի  $F_{CLK}$  հաճախականությունը  $2^i$  բաժանման գործակիցներով,  $i \leq r$ :

Նկ 5.4-ում ցույց են տրվել երկուական հաշվիչի գրաֆիկական սինվոլների երկու տարբերակ: Հաճախ հաշվիչ ունի սկզբնական 0 վիճակ ասինքրոն կարգման  $R$  մուտք, որը հաշվիչի բոլոր տրիգերները դնում է 0 վիճակ:



Նկ. 5.5. Երկուական հաշվիչի գրաֆիկական սինվոլները

Նկատենք, որ նկ. 5.1-ում ցույց տրված անցումների գրաֆը նկարագրում է անցումները գումարող հաշվիչում: Եթե գրաֆում փոխենք աղեղների ուղղությունը, կստացվի հանող հաշվիչի գրաֆը: Կատարելով հանող հաշվիչի սինթեզը ճիշտ նույն եղանակով, ինչ գումարողի դեպքում՝ կստանանք հանող հաշվիչի տրիգերների մուտքերի հավասարումները.

$$T_3 = \bar{Q}_2 \bar{Q}_1, \quad (5.4)$$

$$T_2 = \bar{Q}_1, \quad (5.5)$$

$$T_1 = 1: \quad (5.6)$$

Ընդհանուր դեպքում  $r$  տրիգերներով  $M=2^r$  վերահաշվման գործակցով հաշվիչի համար կարելի է ստանալ.

$$T_i = \&_{j=1}^i Q_j, i = 2, 3, \dots, r, \quad (5.7)$$

$$T_1 = 1 \quad (5.8)$$

գումարող հաշվիչի համար, և

$$T_i = \&_{j=1}^i \bar{Q}_j, i = 2, 3, \dots, r, \quad (5.9)$$

$$T_1 = 1 \quad (5.10)$$

հանող հաշվիչի դեպքում:



(5.7)- (5.8) բանաձևերը կարելի ձևափոխել հետևյալ տեսքի.

$$T_i = Q_{i-1} T_{i-1}, i = 2, 3, \dots, r, \quad (5.11)$$

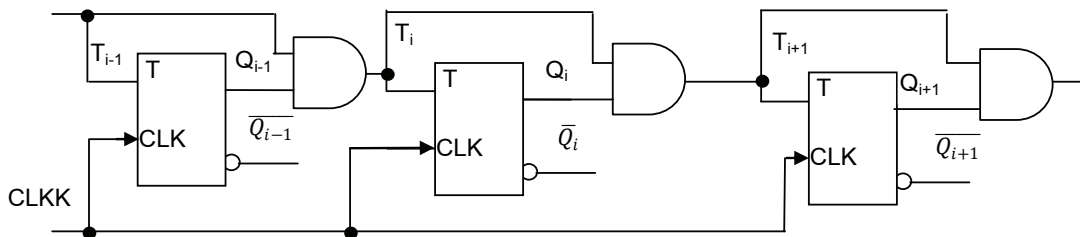
$$T_1 = 1: \quad (5.12)$$

Նույն ձևով (5.9)- (5.10) բանաձևերից կարելի ստանալ.

$$T_i = \bar{Q}_{i-1} T_{i-1}, i = 2, 3, \dots, r, \quad (5.13)$$

$$T_1 = 1: \quad (5.14)$$

Նկ. 5.6-ում ցույց է տրված (5.11)-(5.12) բանաձևերի հիման վրա կառուցված հաշվիչի մեկ հատված:

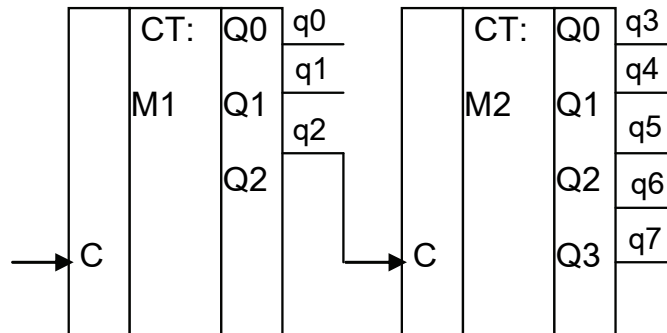


Նկ. 5.6.  $M=2^r$  վերահաշվման գործակցով հաշվիչի մեկ հատված

Դժվար չէ նկատել, որ (5.11)-(5.12) բանաձևերի հիման վրա կառուցված հաշվիչի սխեման ավելի պարզ տեսք կունենա, քան (5.7)-(5.8) բանաձևերի հիման վրա կառուցվածը: Նկ. 5.6-ի սխեման ունի կարգավորված կառուցվածք. յուրաքանչյուր բջիջ ներառում է մեկ տրիգեր և մեկ 2ԵՎ տարր: (5.7) բանաձևով կառուցվող սխեմայում յուրաքանչյուր բջիջ կներառի մեկ տրիգեր և մեկ  $i$  մուտքերով ԵՎ տարր, մեծ  $i$ -երի դեպքում սխեման կբարդանա: Սակայն կարգավորված կառուցվածքով սխեման կունենա ավելի ցածր արագագործություն, քանի որ վերջին տրիգերի մուտքի ազդանշանը կծնավորվի բոլոր նախորդ տրիգերների մուտքերի ազդանշանների հաջորդաբար ձևավորումից հետո:

### 5.3. Հաշվիչների սխեմաների ընդարձակում

Վերահաշվման գործակցի մեծացմանը զուգընթաց բարդանում է հաշվիչի սխեման: Հաճախ հարմար է մեծ վերահաշվման գործակցով հաշվիչի իրականացումը մի քանի հաշվիչների կասկադային միացմամբ: Նկ. 5.7-ում ցույց է տրված երկու հաշվիչների հաջորդական կամ կասկադային միացման սխեման, առաջին հաշվիչի բարձր կարգի տրիգերի ելքը միացվում է հաջորդ հաշվիչի տակտային մուտքին: Արդյունաբար հաշվիչի վերահաշվման գործակիցը հավասար է առանձին հաշվիչների վերահաշվման գործակիցների արտադրյալին՝  $M=M_1 M_2$ : Արդյունաբար հաշվիչի վիճակի փոփոխականներն են՝  $q_0, q_1, \dots, q_7$ :



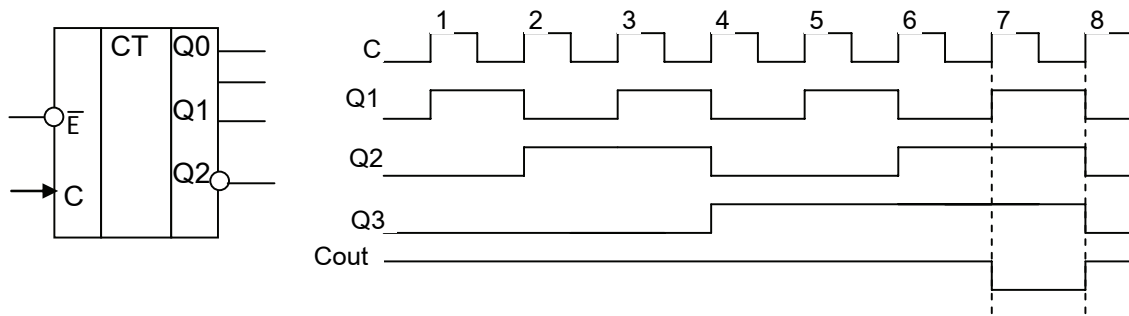
Նկ. 5.7. Հաշվիչների կասկադային (հաջորդական, ասինքրոն) միացում

Անհրաժեշտ է նկատի ունենալ, որ չնայած հաշվիչներից յուրաքանչյուրը կարող է լինել սինքրոն ավտոմատ, արդյունաբար հաշվիչը ասինքրոն է, քանի որ դրա առանձին բաղադրիչներ տակտավորվում են տարբեր տակտային ազդանշաններով՝ առաջինը մուտքային տակտային իմպուլսներով, իսկ երկրորդը՝ առաջին հաշվիչի Q2 ելքային իմպուլսներով։ Ասինքրոն աշխատանքի պատճառով մեկ վիճակից մյուսին անցման ժամանակ հաշվիչի ելքային կոդի բիթերը փոխանջատվում են ժամանակի տարբեր պահերի։ Դա էական թերություն չէ, եթե հաշվիչը կիրառվում է իբրև հաճախականության բաժանիչ կամ աշխատում է ցածր հաճախականային թվային համակարգերում, որտեղ առանձին բիթերի փոխանջատումների միջև հապաղումները կարելի է անտեսել։ Բայց այս երևույթը կարող է ունենալ լուրջ բացասական հետևանքներ բարձր արագության թվային համակարգերում։ Օրինակ, եթե արագագործ համակարգում հաշվիչի - ելքային բիթերը հասցեավորում են հիշողության, ապա առանձին բիթերի փոխանջատումների միջև հապաղումների պատճառով հիշողությունում կընտրվի նպատակայինից տարբեր հասցեով բջիջ։

Ընդհանուր դեպքում, կանայական թվով հաշվիչների հաջորդական միացման դեպքում արդյունաբար հաշվիչի վերահաշվման գործակիցը կորոշվի.

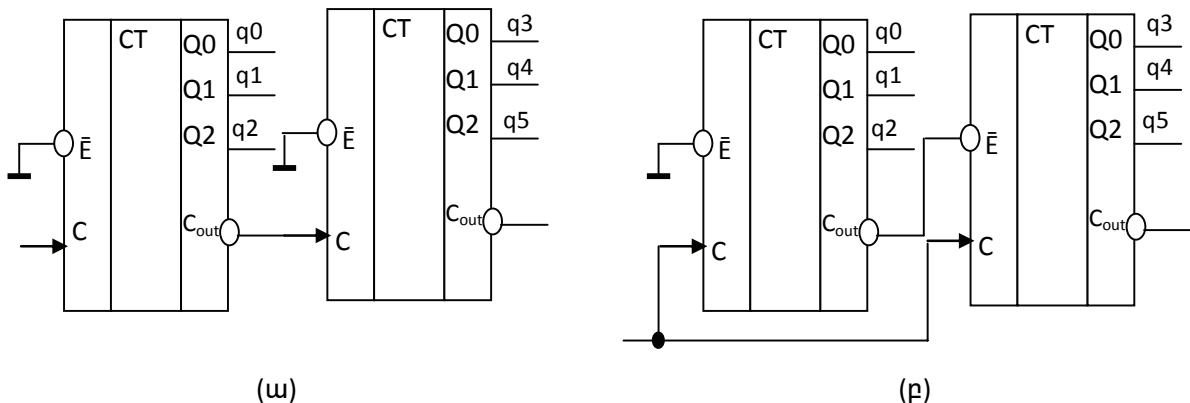
$$M = M_1 M_2 \cdots M_n : \quad (5.15)$$

Հաշվիչի սխեմայի ընդարձակելիության համար հաճախ հաշվիչում նախատեսված է լինում գերլցման ազդանշան՝  $C_{out}$ , որն ընդունում է ակտիվ մակարդակ, երբ հաշվիչը հայտնվում է վերջին վիճակում, մնում է այդ մակարդակին տակտային ազդանշանի միայն մեկ պարբերության ընթացքում և հետո է փոխանջատվում պասիվ մակարդակի, երբ հաշվիչը վերջին վիճակից անցնում է նորից առաջին վիճակին։ Միաժամանակ հաշվիչի ընդարձակման հարմարավետության համար այն օժտված է լինում հաշվի թույլտվության մուտքով՝ E: Նկ. 5.8-ում ցույց է տրված ընդարձակվող եռակարգ հաշվիչի գրաֆիկական սիմվոլը և ժամանակային դիագրամները, երբ  $\bar{E} = 0$ : Գերլցման ազդանշանի ակտիվ մակարդակը 0-ն է։ Սովորաբար E-ն և  $C_{out}$  -ը ունեն նույն ակտիվ մակարդակը, որը հնարավորություն է տալիս նախորդ հաշվիչի  $C_{out}$  ելքը ուղղակի միացնել հաջորդ հաշվիչի E կամ C մուտքին։



Նկ. 5.8. Ընդարձակելի հաշվիչի գրաֆիկական սինվոլը և ժամանակային դիագրամները

Նկ. 5.9–ում ցույց են տրված հաշվիչների ասինքրոն (ա) և սինքրոն (բ) ընդարձակման տարբերակները: Ինչպես կարելի է տեսնել սինքրոն տարբերակում երկու հաշվիչներն էլ տակտավորվում են արտաքինից տրվող տակտային ազդանշանով, նախորդ հաշվիչի  $C_{out}$  ելքի ազդանշանը ծառայում է հաշվի թույլտվություն հաջորդ հաշվիչի համար: Չնայած որ տակտային իմպուլսներն անընդհատ կիրառվում են երկրորդ հաշիչին, այն հաշվում է միայն այն տակտերում, երբ  $C_{out}=0$ , այսինքն, նախորդ հաշվիչի յուրաքանչյուր լրիվ ցիկլից հետո միայն մեկ անգամ: Երկրորդ հաշվիչը հաշվում է նախորդի գերլցումների (ցիկլերի) թիվը: Սինքրոն ընդարձակումը նախընտրելի է վերը քննարկված պատճառով:



Նկ. 5.9. Հաշվիչի սխեմայի ասինքրոն (ա) և սինքրոն (բ) ընդարձակում

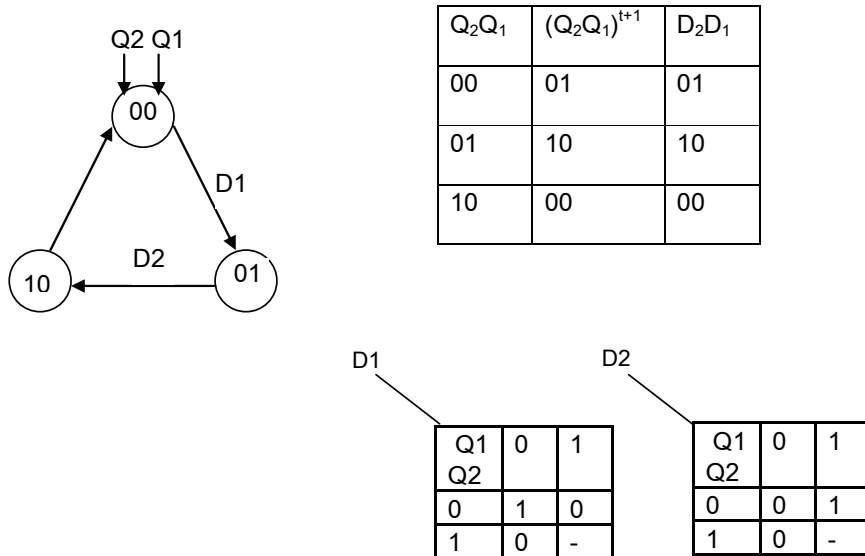
#### 5.4. Ոչ երկուական հաշվիչներ

Այս հաշվիչներում ( $M < 2^R$ ) հնարավոր  $2^R$  վիճակներից օգտագործվում է միայն  $M$ -ը: Սկզբունքորեն  $2^R$  վիճակներից կարող է ընտրվել ցանկացած  $M$ -ը, սակայն այդ ընտրությունից կախված է հաշվիչի իրականացման բարդությունը:

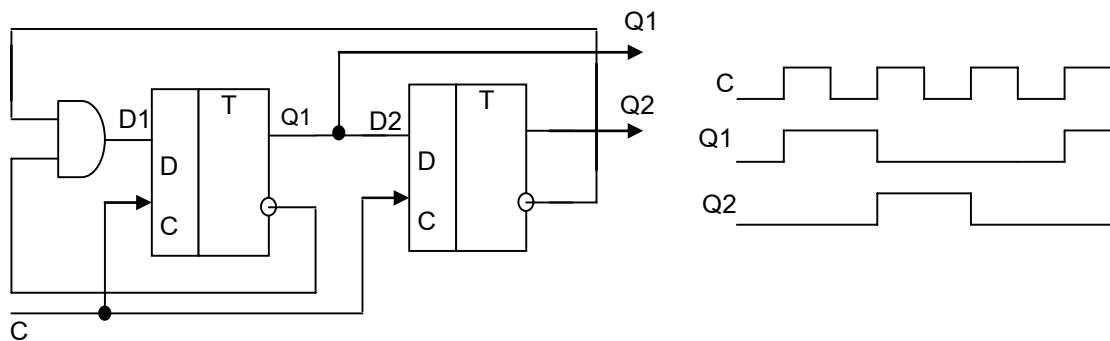
**Օրինակ 5.1.**  $M=3$  վերահաշվման գործակցով հաշվիչը կարելի է կառուցել 2 տրիգերների վրա, 4 հնարավոր վիճակներից օգտագործվում է 3-ը: Վերցնենք առաջին երեքը՝ 00, 01, 10: Հաշվիչի անցումների գրաֆն ու աղյուսակը և D տրիգերների մուտքային ֆունկցիաների Կառնոյի քարտերը ցույց են տրված նկ. 5.10 –ում: Կառնոյի քարտերից կստացվեն.

$$\begin{aligned} D_1 &= \overline{Q_1} \& \overline{Q_2} \\ D_2 &= Q_1 \end{aligned} \quad (5.16)$$

Հաշվիչի սխեման և ժամանակային դիագրամները ցույց են տրված նկ. 5.11–ում: Սխեմայից կան (5.16) հավասարումներից հեշտ է նկատել, որ եթե հաշվիչը միացման պահին հայտնվի չօգտագործվող 11 վիճակում, ապա հաջորդ տակտում կհայտնվի 10 վիճակում, այսինքն՝ ինքնուրույն դուրս կգա չօգտագործվող վիճակից:



Նկ. 5.10.  $M=3$  վերահաշվման գործակցով հաշվիչի անցումների գրաֆը, անցումների աղյուսակը և  $D$  տրիգերների մուտքային ֆունկցիաների Կառնոյի քարտերը

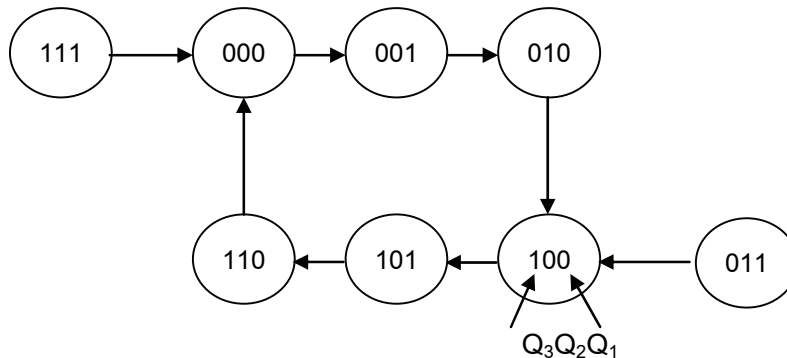


Նկ. 5.11.  $M=3$  վերահաշվման գործակցով հաշվիչի սխեման  $D$  տրիգերների վրա և ժամանակային դիագրամները

Ոչ երկուական հաշվիչում կան  $2^R - M$  չօգտագործված վիճակներ: Հաշվիչի սնման լարումը միացնելիս հնարավոր է, որ հաշվիչը հայտնվի այդ չօգտագործված վիճակներից որևէ մեկում: Հետևաբար, պետք է քննարկել՝ արդյոք հաշվիչն ինքնուրույն կարող է անցնել ցանկացած չօգտագործված վիճակից որևէ օգտագործվող վիճակ:

### Աղյուսակ 5.2

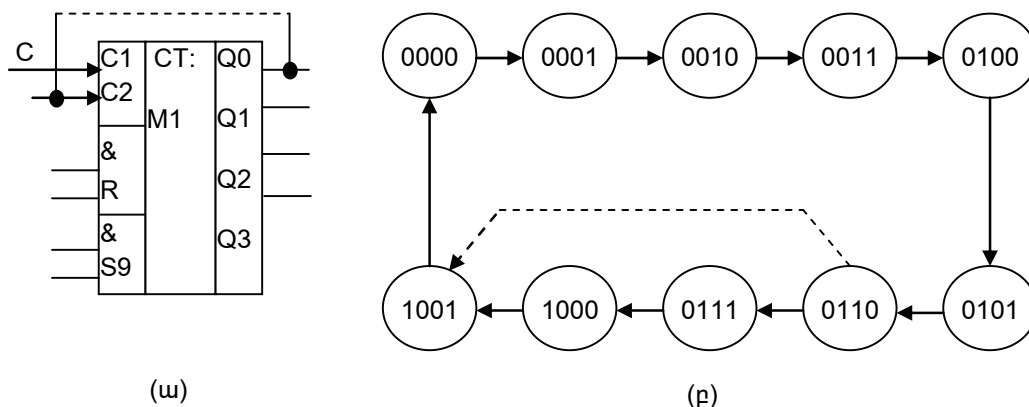
237



Նկ. 5.13.  $M=6$  վերահաշվման գործակցով հաշվիչի անցումների գրաֆը

#### 5.4. Երկուական-տասական հաշվիչներ ( $M=10$ )

Ոչ երկուական հաշվիչների մեջ առանձնահատուկ նշանակություն ունեն երկուական-տասական հաշվիչները ( $M=10$ ), որոնք մուտքին կիրառված տակտային իմպուլսների թիվը ներկայացնում են երկուական-տասական կոդով: Տարբեր արտադրողների կողմից թողարկվում են երկուական-տասական հաշվիչների ԻՍ-եր: Դրանցից առաջինը 7490-ն է (K155IE2), որի գրաֆիկական սիմվոլը և անցումների գրաֆը ցույց է տրված նկ. 5.14-ում: 7490-ը բաղկացած է մոդուլ 2-ով ( $M=2$ ) և մոդուլ 5-ով ( $M=5$ ) հաշվիչներից, որոնց հաջորդական միացումից ստացվում է երկուական-տասական հաշվիչ (այդ միացումը նկ. 5.14-ում ցույց է տրված կետագծով): Հաշվիչն ունի R և S9 ասինքրոն մուտքեր՝ 0 (0000) և 9 (1001) վիճակներում հաշվիչը կարգելու համար: Հաշվիչի վիճակների փոփոխությունը կատարվում է նկ. 5.14բ-ում ցույց տրված գրաֆի համաձայն: Օգտագործելով R և S9 ասինքրոն մուտքերը՝ կարելի է 7490 հաշվիչի հիման վրա ստանալ ցանկացած  $2 \leq M \leq 10$  վերահաշվման գործակցով հաշվիչ: Օրինակ,  $M=7$  ստանալու համար կարելի է Q2 և Q3 ելքերը միացնել S9 ասինքրոն մուտքերին, այդ դեպքում հաշվիչը 0110 վիճակից կանցնի միանգամից 1001 վիճակին: Նկ. 5.14բ-ում այդ անցումը ցույց է տրված կետագծով:



Նկ. 5.14. 7490 հաշվիչը (ա) և նրա անցումների գրաֆը (բ)

4017 (564IE8) միկրոսխեման, բացի երկուական-տասական հաշվիչից, պարունակում է նաև  $4 \times 10$  վերծանիչ, որը թույլ է տալիս ելքերում ազդանշանները ներկայաց -

Ունի 10-ից մեկը դիրքային կոդով:

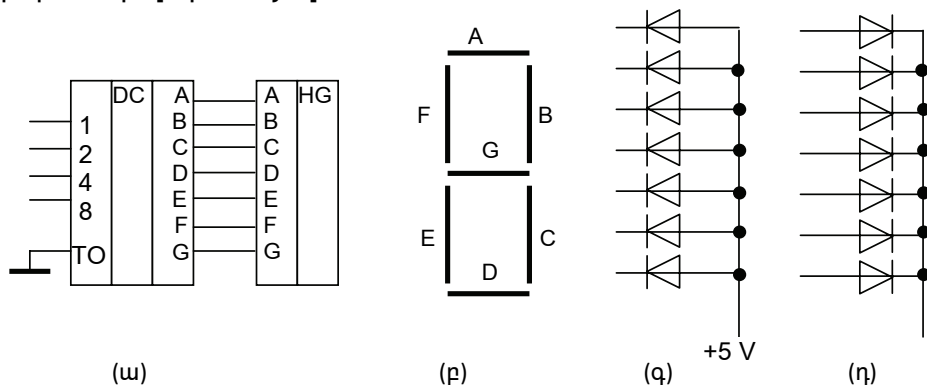
Երկուական-տասական կոդով ներկայացված տեղեկատվությունը ավելի ընկալելի է օգտագործողի համար. հաշվի արդյունքը կարելի է միանգամից ցուցադրել տասական թվանշաններով: Հաշվի արդյունքի ստացումը երկուական-տասական կոդով և դրա հետագա ցուցադրումը տասական թվանշաններով առավել տարածված է թվային չափիչ սարքերում:

Երկուական-տասական կոդը թվային արտացոլիչի կոդի ձևափոխումը բերված է աղյուսակ 5.3-ում:

Աղյուսակ 5.3

$x_3x_2x_1x_0$	A	B	C	D	E	F	G	$x_3x_2x_1x_0$	A	B	C	D	E	F	G
0000	1	1	1	1	1	1	0	1000	1	1	1	1	1	1	1
0001	0	1	1	0	0	0	0	1001	1	1	1	1	0	1	1
0010	1	1	0	1	1	0	1	1010	x	x	x	x	x	x	x
0011	1	1	1	1	0	0	1	1011	x	x	x	x	x	x	x
0100	0	1	1	0	0	1	1	1100	x	x	x	x	x	x	x
0101	1	0	1	1	0	1	1	1101	x	x	x	x	x	x	x
0110	1	0	1	1	1	1	1	1110	x	x	x	x	x	x	x
0111	1	1	0	0	1	0	1	1111	x	x	x	x	x	x	x

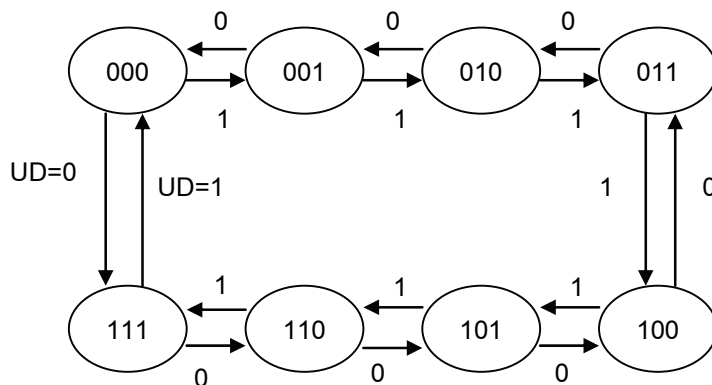
Նկ. 5.15-ում ցույց է տրված երկուական-տասական կոդը յոթ սեգմենտներով թվային ցուցասարքի կոդի ձևափոխիչի և ցուցասարքի միացման սխեման: Ցուցասարքի սեգմենտները կարող են լինել լուսադիոդներ կամ հեղուկ բյուրեղի սեգմենտներ: Լուսադիոդները սովորաբար ցուցասարքում միացվում են ընդհանուր անոդով կամ ընդհանուր կատոդով սխեմայով:



Նկ. 5.15. Թվային արտացոլիչի միացման սխեման

## 5.5. Դարձափոխելի հաշվիչներ

Դարձափոխելի հաշվիչներն ունեն մեկ լրացուցիչ կառավարող մուտք, որին տրվում է հաշվման ուղղությունը կառավարող UD տրամաբանական ազդանշանը: Կհամարենք, որ UD=1-ի դեպքում հաշվիչը աշխատում է գումարման ռեժիմում, իսկ UD=0-ի դեպքում՝ հանման ռեժիմում: Նկ. 5.16-ում ցույց է տրված եռակարգ դարձափոխելի հաշվիչի անցումների գրաֆը, որի հիման վրա կարելի է սինթեզել հաշվիչը: Դ տրիգերների դեպքում մուտքերի ֆունկցիաների նվազագույն ԴՆՁ-երը սինթեզվող հաշվիչում ունեն հետևյալ տեսքը.



Նկ. 5.16. Դարձափոխելի հաշվիչի անցումների գրաֆը՝ UD=1 գումարման ռեժիմ, UD=0 հանման ռեժիմ

$$T_1 = 1,$$

$$T_2 = UD \& Q_1 + \overline{UD} \& \overline{Q_1}, \quad (5.18)$$

$$T_3 = UD \& Q_2 + \overline{UD} \& \overline{Q_2}:$$

Այս բանաձևերը կարելի է ստանալ նաև (5.1)-(5.3)-ի և (5.4)-(5.6)-ի միավորումից: Ընդհանուր դեպքում, կամայական  $n$  կարգերով դարձափոխելի հաշվիչի համար կարելի է գրել՝

$$T_1 = 1,$$

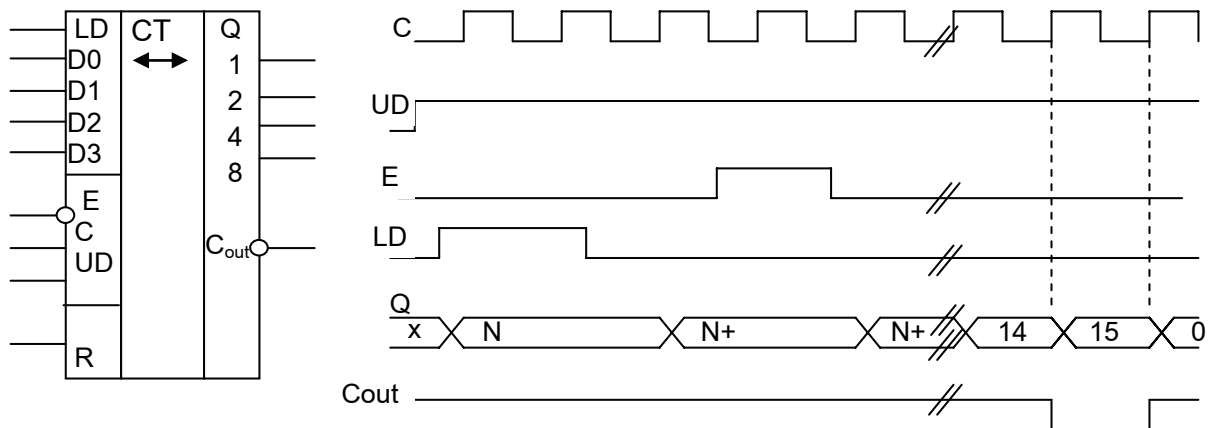
$$T_i = UD \&_{j=1}^{i-1} Q_j + \overline{UD} \&_{j=1}^{i-1} \overline{Q_j} : \quad (5.19)$$

ԿՄՕԿ միկրոսխեմային քառակարգ 4510 (K561IE11) (նկ. 5.17) և 4029 (K561IE14) դարձափոխելի հաշվիչները, բացի հաշվման ուղղության ընտրության մուտքից, ունեն նաև կամայական սկզբնական վիճակի քառակարգ երկուական կոդի տրման՝ D0, D1, D2, D3 և այդ կոդի գրանցման կառավարման LD մուտքեր: K561IE11 հաշվիչն ունի M=16 վերահաշվման գործակից, իսկ K561IE14 հաշվիչը կարող է հաշվել երկուական և երկուական-տասական կոդով՝ համապատասխանաբար M=16 ու M=10 վերահաշվման գործակիցներով: Տակտային C, հաշվի թույլտվության E մուտքերը հաշվիչի վրա ներգործություն չունեն սկզբնական վիճակի ասինքրոն բեռնավոր-

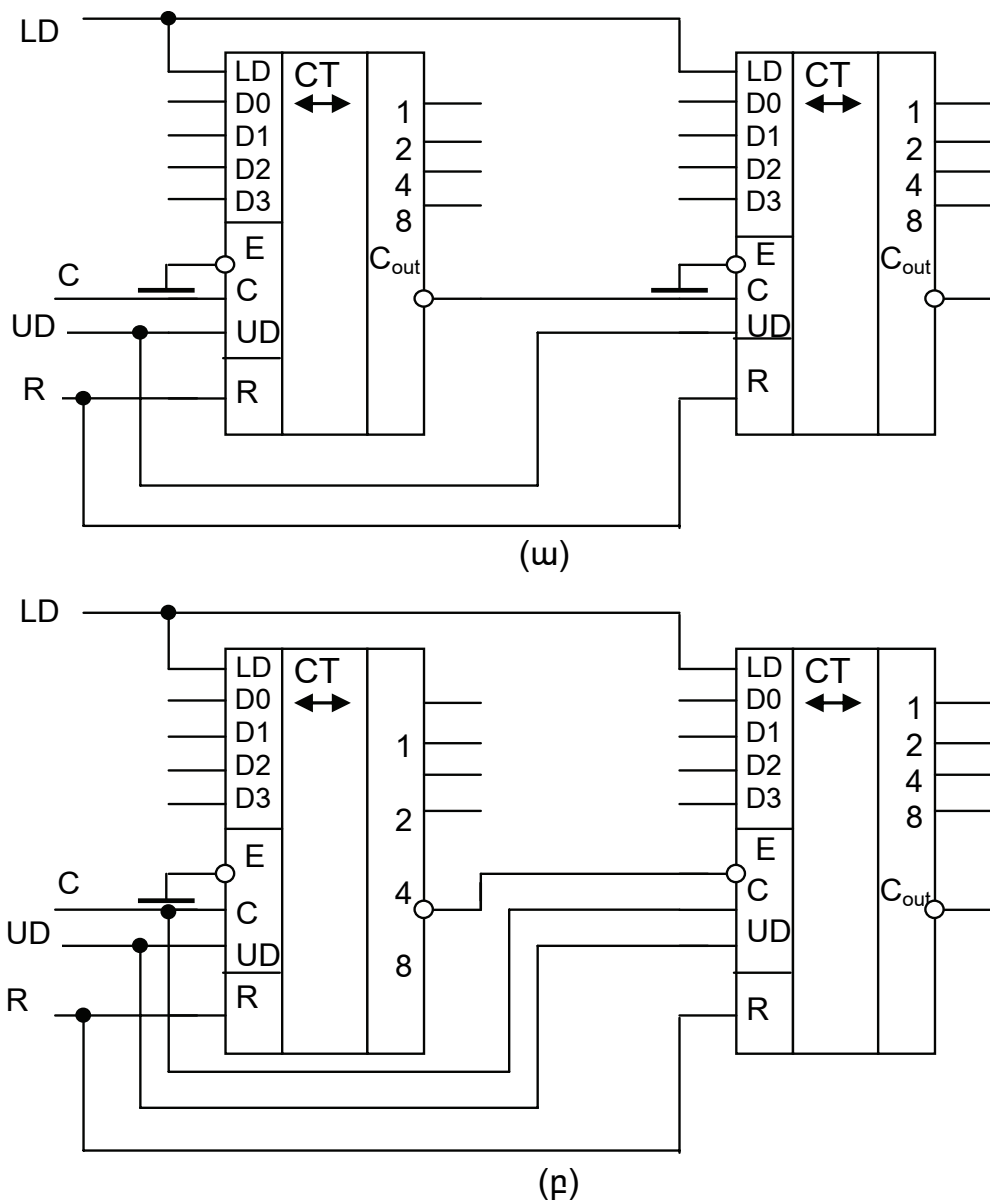


ման ժամանակ՝ երբ  $LD=1$ : Հաշվի ուղղության ընտրության ազդանշանի  $UD=1$  արժեքի դեպքում հաշվին աշխատում է գումարման, իսկ  $UD=0$  դեպքում՝ հանման ռեժիմում: Հաշվին աշխատում է, եթե  $E$  մուտքն ակտիվ է, տվյալ դեպքում՝ 0 է: Նկ. 5.17-ում ցույց է տրված K561IE11 հաշվիչի գրաֆիկական սիմվոլը և ժամանակային դիագրամները: Գումարման ռեժիմում  $C$  մուտքին կիրառված տակտային իմպուլսների թիվը գումարվում է  $D0, D1, D2, D3$  մուտքերից գրանցված սկզբնական թվին: Եթե հաշվիչում նախապես գրանցվել է  $N$  թիվը, ապա գումարման ռեժիմում  $(16-N)$  իմպուլս հաշվելուց հետո հաշվիչը կգա 0 վիճակ՝ վերահաշվման գործակիցը կլինի  $(16-N)$ : Հանման ռեժիմում վերահաշվման գործակիցը կլինի ուղղակի  $N$ : Որպեսզի հաջորդ ցիկլերում պահպանվի նույն վերահաշվման գործակիցը, անհրաժեշտ է յուրաքանչյուր գերլցման ժամանակ նորից բեռնավորել սկզբնական վիճակը: Դա կարելի է անել  $C_{out}$  ազդանշանով՝ այն պետք է անցկացնել շրջիչով և միացնել  $LD$  մուտքին:

Նկ. 5.18-ում ցույց է տրված K561IE11 հաշվիչի ափսեքրոն (ա) և սինքրոն (բ) ընդարձակման սխեմաները: Ափսեքրոն ընդարձակման դեպքում երկրորդ հաշվիչը տակտավորվում է նախորդի գերլցման  $C_{out}$  ազդանշանով, սինքրոն ընդարձակման դեպքում երկու հաշվիչն էլ տակտավորվում են միևնույն  $C$  տակտային ազդանշանով: Եթե յուրաքանչյուր հաշվիչի գերլցման  $C_{out}$  ելքը միացրած է իր  $LD$  մուտքին, ընդարձակված հաշվիչի վերահաշվման գործակիցը կլինի  $(16-N_1)(16-N_2)$  գումարման և  $N_1 \cdot N_2$  հանման դեպքում, որտեղ  $N_1$  և  $N_2$ -ը առաջին և երկրորդ հաշվիչների սկզբնական վիճակի կոդերն են: Եթե երկու հաշվիչների  $LD$  մուտքերը միացվեն երկրորդ հաշվիչի ելքին, ապա վերահաշվման գործակիցը կլինի  $(256-N)$  գումարման և  $N$  հանման դեպքում, որտեղ  $N$ -ը երկու հաշվիչների սկզբնական վիճակի կարգման ութ մուտքերին տրված 8-բիթ թիվն է՝ ցածր 4 բիթերը առաջին հաշվիչինը, իսկ բարձր 4 բիթերը երկրորդինը:



Նկ. 5.17. K561IE11 հաշվիչի գրաֆիկական սիմվոլը և ժամանակային դիագրամները



Նկ. 5.18. K561ME11 հաշվիչի ասինքրոն (ա) և սինքրոն (բ) ընդարձակման սխեմաները

## 5.6. Հաճախականության թվային սինթեզատորներ

Տակտավորվող թվային համակարգերում կայուն հաճախականությամբ տակտային իմպուլսների գեներացման համար օգտագործվում են քվարցային գեներատորներ: Շատ դեպքերում թվային համակարգերում անհրաժեշտ է ունենալ մի շարք կայուն հաճախականություններ: Դրանցից յուրաքանչյուրի համար առանձին քվարցային գեներատորի օգտագործումը տնտեսապես հարմար չէ: Առաջ է գալիս մեկ առաջնային գեներատորի հաճախականությունից անհրաժեշտ թվով հաճախականությունների ձևավորման՝ սինթեզի անհրաժեշտությունը:

Պարզագույն սինթեզատորը հաճախականության բաժանիչ է:

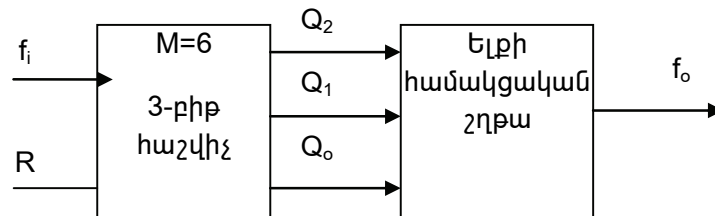
Հաճախության բաժանիչի ելքում իմպուլսների հաճախականությունը որոշվում է՝

$$f_o = \frac{f_i}{M}, \quad (5.20)$$

որտեղ՝  $f_i$  և  $f_o$ -ն մուտքային և ելքային հաճախականություններն են,  $M$ -ը բաժանման գործակիցն է: Հաշվիչների սխեմաներով կառուցվող թվային բաժանիչներում  $M > 1$  և ամբողջ թիվ է, հետևաբար,  $f_o < f_i$ : Հաճախության բաժանիչի ելքը համընկնում է հաշվիչի բարձր կարգի տրիգերի ելքի հետ:

Շատ դեպքերում անհրաժեշտ է լինում կառուցել հաճախականության բաժանիչ, որի ելքային իմպուլսների տևողությունը համընկնում է մուտքային իմպուլսների պարբերության հետ՝ անկախ բաժանման գործակից կամ ելքային իմպուլսների պարբերությունից: Այս դեպքում բաժանիչի ելքային իմպուլսները կարելի է ձևավորել հաշվիչի վիճակի փոփոխականներից՝ համակցական շղթայի միջոցով:

Դիտարկենք օրինակ: Կառուցել  $M=6$  բաժանման գործակցով հաճախականության բաժանիչ, որի ելքային իմպուլսների տևողությունը հավասար է մուտքային իմպուլսների պարբերությանը: Այսպիսի բաժանիչ կառուցելու համար պետք է օգտագործել  $M=6$  վիճակներով հաշվիչ կառուցված 3 տրիգերների վրա և համակցական շղթա (նկ. 5.19), որը ելքում կձևավորի տրամաբանական 1, երբ հաշվիչը հայտնվի  $Q_2Q_1Q_0 = 101_2 = 5_{10}$  վիճակում:  $M=6$  հաշվիչի անցումների աղյուսակն ու սխեման ցույց են տրված աղյուսակ 5.2-ում և նկ. 5.12-ում: Ելքային իմպուլսի ձևավորման համակցական շղթայի իսկության աղյուսակը ցույց է տրված աղյուսակ 5.4-ում:

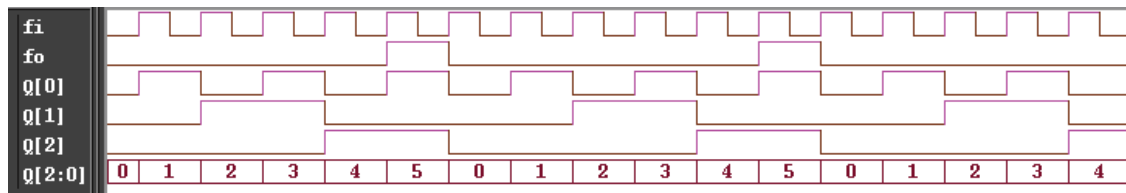


Նկ. 5.19.  $M=6$  բաժանման գործակցով հաճախականության բաժանիչ

Աղյուսակ 5.4

$Q_2$	$Q_1$	$Q_0$	$f_o$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Նկ. 5.20-ում ցույց են տրված  $M=6$  հաճախականության բաժանիչի ժամանակային դիագրամները:



Նկ. 5.20.  $M=6$  հաճախականության բաժանիչի ժամանակային դիագրամները

Թվային եղանակով կարելի է սինթեզել միայն այնպիսի  $f_1$  և  $f_2$  հաճախականություններ, որոնց հարաբերությունը ռացիոնալ կոտորակ է՝

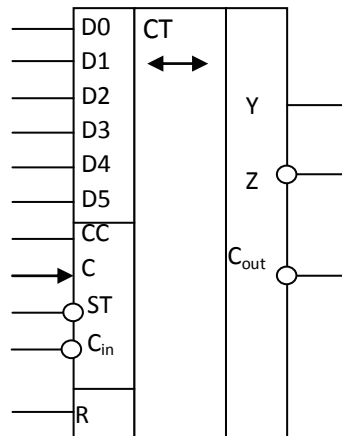
$$\frac{f_1}{f_2} = \frac{M_1}{M_2}, \quad (5.21)$$

որտեղ  $M_1$  և  $M_2$ -ը դրական ամբողջ թվեր են:

Եթե  $M_2=1$ , ապա  $f_2$  հաճախականությունը ձևավորվում է  $f_1$ -ից  $M_1$  բաժանման գործակցով հաճախականության բաժանիչի միջոցով՝  $M_1$  վերահաշվման գործակցով հաշվիչի միջոցով: Եթե  $M_2 \neq 1$ , ապա պետք է օգտագործել ավելի բարդ սխեմաներ, այսպես կոչված, հաճախականության սինթեզատորներ:

Աշխատանքի տրամաբանության տեսանկյունից հաճախականության թվային սինթեզատորները կարելի է դասել երկու խմբի՝ սինթեզատորներ, որոնք իրականացվում են կոտորակային, փոփոխական բաժանման գործակցով հաշվիչների վրա, և սինթեզատորներ, որոնք կառուցվում են կուտակող գումարիչների հիման վրա:

Հաճախականության սինթեզատորի պարզ օրինակ է SN7497N (K155IE8) միկրոսխեման, որը պարունակում է 6-աբիթ սինքրոն հաշվիչ և տրամաբանական շղթաներ, որոնք հնարվորություն են տալիս ստանալ կոտորակային բաժանման գործակիցներ:



Նկ. 5.21. SN7497N հաճախականության սինթեզատորի գրաֆիկական սիմվոլը

Հաշվիչի վերահաշվման գործակիցը 64 է: 64 տակտային իմպուլսներից հետո  $C_{out}$  ելքում ձևավորվում է հաշվիչի գերլցման մեկ բացասական իմպուլս, որի տևողությունը հավասար է տակտային ազդանշանի պարբերությանը:

Ներքին տրամաբանական շղթաների միջոցով հաշվիչի 6 տրիգերների ելքային ազդանշաններից հաշվիչի Z ելքում ձևավորվում է մուտքային տակտային իմպուլսի

տևողությամբ իմպուլսների հաջորդականություն: Այդ հաջորդականությունում իմպուլսների թիվը կառավարվում է  $D_5, \dots, D_0$  մուտքերից տրված կոդի արժեքով: Յուրաքանչյուր  $n=64$  մուտքային իմպուլսների գործման ցիկլում միկրոսխեմայի  $Z$  ելքում ստացված իմպուլսների թիվը որոշվում է հետևյալ բանաձևով.

$$M = 2^5 D_5 + 2^4 D_4 + 2^3 D_3 + 2^2 D_2 + 2^1 D_1 + 2^0 D_0 : \quad (5.22)$$

Հետևաբար,  $C$  տակտային մուտքից  $Z$  ելք հաճախականության բաժանման գործակիցը կլինի.

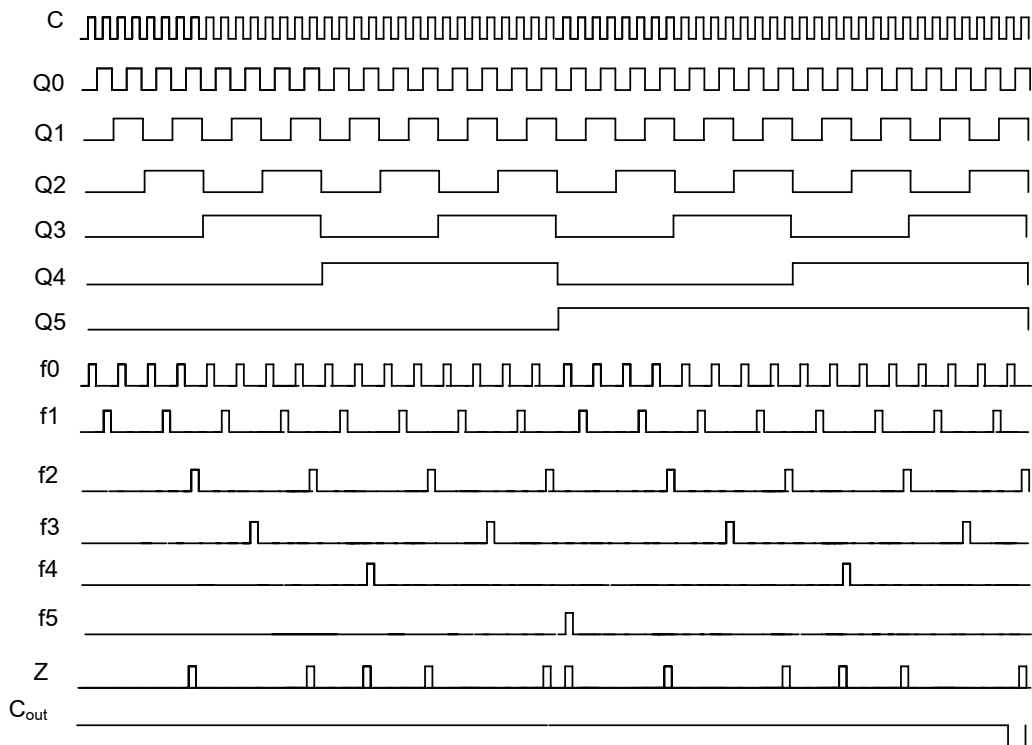
$$\frac{f_{out}}{f_{in}} = \frac{M}{64} : \quad (5.23)$$

Նկ. 5.22-ում ցույց են տրված SN7497N սինթեզատորի ժամանակային դիագրամները: ԻՍ-ի տրամաբանական շղթաները երկուական հաշվիչի տրիգերների  $Q_5, \dots, Q_0$  ելքերից ձևավորում են այդ ելքերի հաճախականությամբ, բայց մուտքային իմպուլսների տևողությամբ իմպուլսների հաջորդականություններ.

$$f_0 = C \& \overline{Q_0}; f_1 = C \& \overline{Q_1} Q_0; f_2 = C \& \overline{Q_2} Q_1 Q_0; f_3 = C \& \overline{Q_3} Q_2 Q_1 Q_0, \\ f_4 = C \& \overline{Q_4} Q_3 Q_2 Q_1 Q_0; f_5 = C \& \overline{Q_5} Q_4 Q_3 Q_2 Q_1 Q_0 : \quad (5.24)$$

Այդ հաջորդականությունների համադրումից՝ ըստ  $D_5, \dots, D_0$  մուտքերից տրվող կոդի,  $Z$  ելքում ձևավորվում է պահանջվող հաճախականությամբ հաջորդականություն՝

$$Z = \sum_{i=0}^5 D_{5-i} f_i ST \quad (5.25)$$



Նկ. 5.22. SN7497N հաճախականության սինթեզատորի ժամանակային դիագրամները

Օրինակ, նկ. 5.22-ում ցույց տրված դեպքում  $D_5 \dots D_0 = 001011$  և  $Z$  ելքում ձևավորվում է

$$Z = f_5 + f_4 + f_2$$

հաջորդականություն, որում  $n=64$  մուտքային իմպուլսների գործման ցիկլում առկա են  $2^3 D_3 + 2^1 D_1 + 2^0 D_0 = 8 + 2 + 1 = 11$  իմպուլսներ:

Յուրաքանչյուր ցիկլում ԻՍ-ի  $C_{out}$  ելքում ձևավորվում է մուտքային իմպուլսների պարբերությանը հավասար տևողությամբ մեկ բացասական իմպուլս:

$$C_{out} = C_{in} \& Q_5 Q_4 Q_3 Q_2 Q_1 Q_0: \quad (5.26)$$

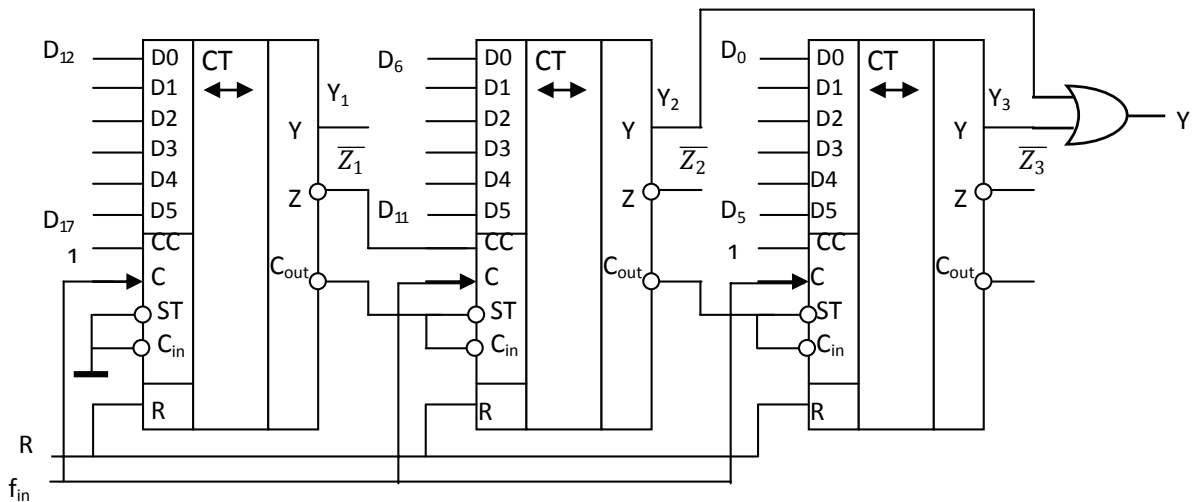
SN7497N ԻՍ-ն ունի ևս մեկ ելք՝

$$Y = \overline{CC} \overline{Z}, \quad (5.27)$$

որն ընձեռնում է սինթեզատորի սխեմայի ընդարձակման լրացուցիչ հնարավորություններ: Նկ. 5.23-ում ցույց է տրված երեք ԻՍ-երի կասկադային միացման օրինակ: Այդ միացումից կարելի է տեսնել, որ՝

$$Y_2 = \overline{CC_2} \overline{Z_2} = \overline{Z_1} \overline{Z_2} = Z_1 + Z_2,$$

$$Y = Y_2 + Y_3 = Z_1 + Z_2 + Z_3:$$



Նկ. 5.23. SN7497N ԻՍ-ի վրա հիմնված սինթեզատորի սխեմայի ընդարձակում

Հետևաբար,  $Y$  ելքում ձևավորվող ազդանշանի հաճախականությունը կլինի

$$\frac{f_{out}}{f_{in}} = \frac{M}{2^{18}},$$

որտեղ՝

$$M = \sum_{i=0}^{17} D_i 2^i$$

Եթե սխեմայի ելքը միացվի  $M_1$  վերահաշվման գործակցով հաշվիչի մուտքի, ապա այդ հաշվիչի ելքային իմպուլսների հաճախականությունը կլինի՝

$$\frac{f_{out}}{f_{in}} = \frac{M}{M_1 2^{18}} : \quad (5.28)$$

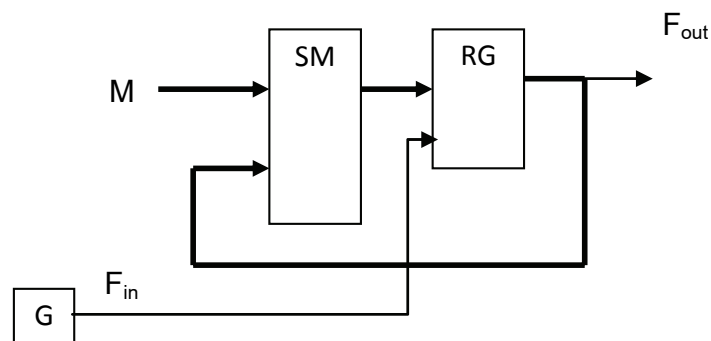
Այս եղանակով կառուցված հաճախականության սինթեզատորի թերությունն այն է, որ ելքային իմպուլսների հաջորդականությունը ժամանակային առանցքի վրա ունի անհավասարաչափ բաշխվածություն, երբ  $M$ -ը երկուսի ամբողջ աստիճան չէ։ Սինթեզատորի ելքում ձևավորվում է  $M$ -ով առաջադրվող միջին հաճախականությունը։ Իրականում ելքային հաճախականության ակնթարթային արժեքը հաստատուն չէ։ Օրինակ, նկ.5.22-ում ելքային  $Z$  հաջորդականության միջին հաճախականությունը  $f_{in} \cdot (6/64)$  է, բայց հաճախականության (կամ պարբերության) ակնթարթային արժեքը հաստատուն չէ։

Այս թերությունից զերծ է կուտակող գումարիչի վրա կառուցված սինթեզատորը (նկ.5.24)։ Կուտակող գումարիչը ռեգիստր է, որի պարունակությունը յուրաքանչյուր տակտում ավելանում է որոշակի հաստատուն  $M$  արժեքով։ Յուրաքանչյուր տակտում  $SM$  գումարիչի մուտքերին տրվում են ռեգիստրի նախորդ պարունակությունը և  $M$ -ը։

$$RG^+ = RG + M : \quad (5.29)$$

Ենթադրենք՝ քառակարգ ռեգիստրի սկզբնական պարունակությունը 0000 է, իսկ  $M=0001$ ։ Կուտակիչի պարունակությունը յուրաքանչյուր տակտում աճում է 0001 արժեքով։  $2^4=16$  տակտերից հետո կուտակիչի պարունակությունը վերադառնում է սկզբնական վիճակի արժեքին, որից հետո ցիկլը կրկնվում է։

Ինչպես և հաշվիչի դեպքում, այստեղ նույնպես ռեգիստրի պարունակությունը ժամանակի ընթացքում գծայնորեն աճում է, և այդ աճը կախված է  $M$  հաստատունի արժեքից։



Նկ.5.24. Հաճախականության սինթեզատոր կուտակող գումարիչի հիմքի վրա

Իրոք, եթե գումարի աճը հավասար է մեկի, ինչպես բերված օրինակում, ապա կուտակող գումարիչի վարքը ոչ մի բանով չի տարբերվի երկուական հաշվիչի վարքից։ Բայց, եթե գումարի աճը, օրինակ, հավասար լինի երկուսի՝  $M=0010$ , ապա կուտակիչի կողը կփոփոխվի երկու անգամ ավելի արագ։ Կուտակիչի պարունակությունը վերադառնում է սկզբնական վիճակի արժեքին  $2^4/2=8$  տակտերից հետո։ Գեներացվող ազդանշանի հաճախությունը այդ դեպքում կլինի երկու անգամ մեծ։

N կարգանի կուտակող գումարիչի դեպքում սինթեզատորի ելքային հաճախությունը կլինի՝

$$f_{out} = M \cdot f_{in} / 2^N : \quad (5.30)$$

Այս հավասարումը հայտնի է, որպես սինթեզատորի վերալարման հավասարում։ Այս հավասարումից կարելի է որոշել պահանջվող ելքային հաճախականության ձևավորման համար անհրաժեշտ մուտքային M թիվը՝

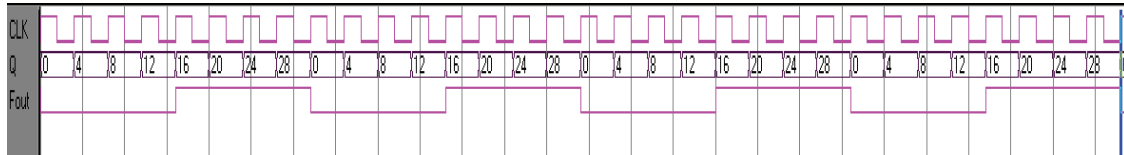
$$M = f_{out} 2^N / f_{in} : \quad (5.31)$$

Ելքային հաճախականության ձևավորման ճշգրտությունը կախված է կուտակիչի կարգայնությունից՝ N: Հաճախության վճռունակությունը (երկու հաճախականությունների նվազագույն տարբերությունը) որոշվում է հետևյալ կերպ՝

$$\Delta f = f_{in} / 2^N : \quad (5.32)$$

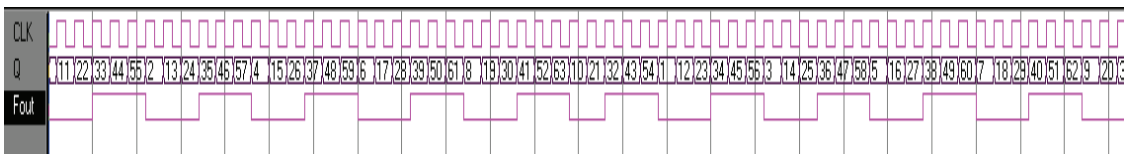
Օրինակ, եթե կուտակող գումարիչը  $N=32$  կարգանի է, իսկ տակտային հաճախությունը 50 ՄՀց է, ապա հաճախության վճռունակությունը կստացվի մոտ 0.01 Հց։

Նկ. 5.25-ում ցույց են տրված 5-ակարգ կուտակող գումարիչով հաճախականության սինթեզատորի մուտքային CLK տակտային իմպուլսների հաջորդականությունը, կուտակիչ ռեգիստրի Q վիճակը և ելքային իմպուլսների  $F_{out}$  հաջորդականությունը  $M=4$  դեպքում։ Այդ ժամանակային դիագրամներից երևում է, որ ելքային հաճախականությունը  $(4/2^5)F_{CLK}$  է, մուտքային 32 իմպուլսին համապատասխանում է ելքային 4 իմպուլս։



Նկ. 5.25. 5-կարգ կուտակող գումարիչով հաճախականության սինթեզատորի ժամանակային դիագրամները (ստացվել են Verilog նմանակումից)

Նկ. 5.26-ում ցույց են տրված 6-ակարգ կուտակող գումարիչով հաճախականության սինթեզատորի ժամանակային դիագրամները  $M=11$  դեպքում։ Ըստ այդ ժամանակային դիագրամների, ելքային հաճախականությունը  $(11/2^6)F_{CLK}$  է, մուտքային 64 իմպուլսին համապատասխանում է ելքային 11 իմպուլս։



Նկ. 5.26. 6-կարգ կուտակող գումարիչով հաճախականության սինթեզատորի ժամանակային դիագրամները (ստացվել են Verilog նմանակումից)

Նկատենք, որ  $N=2^6=64$  սինթեզատոր է ծառայում նաև վերևում քննարկված SN7497N ԻՍ-ը։ Նկ. 5.22-ում բերված ժամանակային դիագրամներում ցույց է տրված ելքային Z հաջորդականությունը  $M=11$  դեպքում, որն ըստ միջինացված հաճախակա-



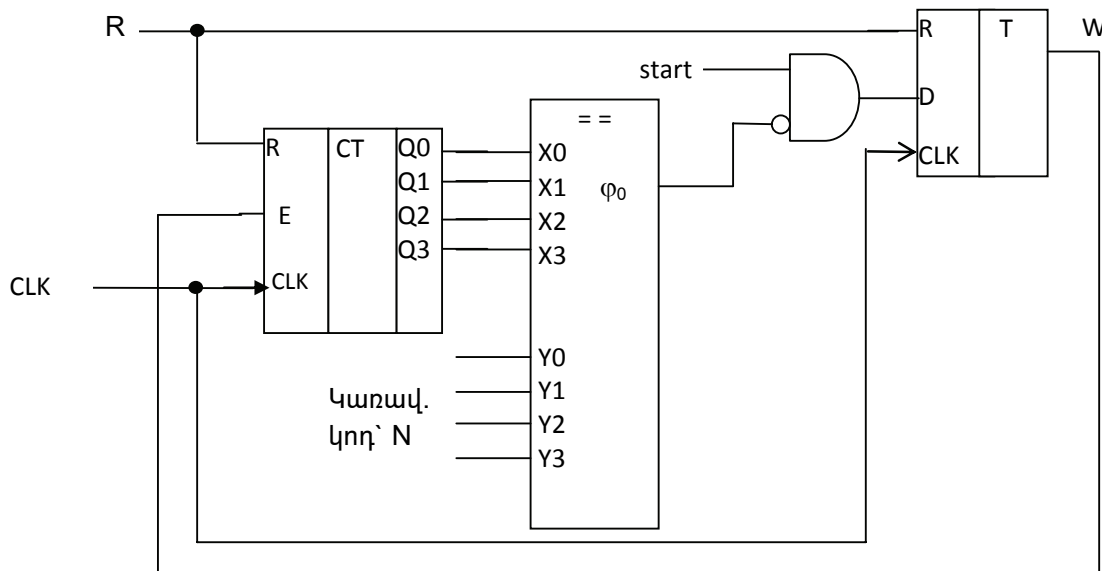
նության համարժեք է նկ. 5.26-ում ցույց տրված  $F_{out}$  հաջորդականությունը: Սակայն, ինչպես կարելի է տեսնել,  $F_{out}$  հաջորդականությունն ունի ավելի կայուն ակնթարթային պարբերություն, քան  $Z$  հաջորդականությունը՝  $F_{out}$  իմպուլսներն ավելի հավասարաչափ են բաշխված ժամանակային առանցքի վրա: Մեծ  $N$ -երի դեպքում կուտակող գումարիչով հաճախականության սինթեզատորի ելքային հաջորդականությունը գործնականում հավասարաչափ է բաշխված ժամանակային առանցքի վրա  $M$ -ի ցանկացած արժեքի դեպքում:

Կուտակող գումարիչով հաճախականության սինթեզատորներն ապահովում են ելքային հաջորդականության ավելի բարձր որակ, քան հաշվիչների վրա կառուցված սինթեզատորները: Այդ պատճառով սինթեզատորները կառուցվում են հիմնականում կուտակող գումարիչի սկզբունքով:

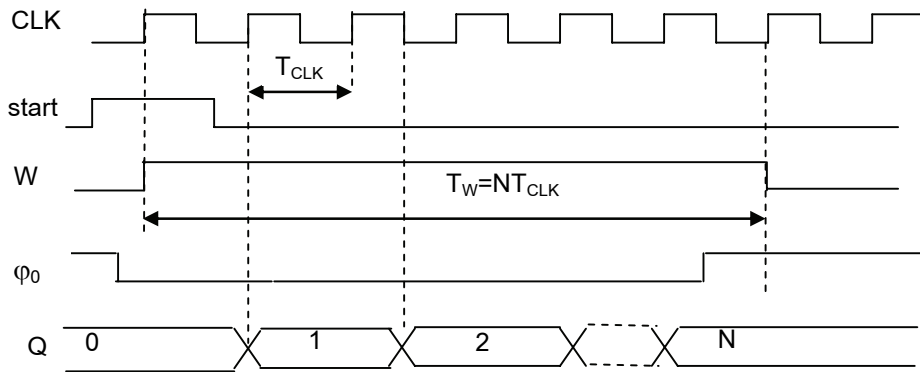
### 5.7. Ժամանակահատվածային սխեմաներ

Թվային համակարգերում հաճախ անհրաժեշտ է լինում ձևավորել իմպուլսային ազդանշաններ, որոնց տևողությունները տրվում են երկուական կոդով: Նման ձևափոխություններ իրականացնող սխեմաները կոչվում են կոդով կառավարվող ժամանակահատվածների գեներատորներ կամ կոդ-ժամանակահատված կերպափոխիչներ:

Կոդով կառավարվող ժամանակահատվածների գեներատորների կառուցման մեկ տարբերակ ցույց է տրված նկ. 5.27, 28-ում: Գեներատորի հիմնական հանգույցը երկուական հաշվիչն է, որը հաշվում է պահանջվող քանակությամբ տակտային իմպուլսներ: Ելքային ազդանշանը ստացվում է կառավարող տրիգերի ելքում՝ ձևավորվող ժամանակահատվածում ելքային ազդանշանը գտնվում 1 վիճակում:



Նկ. 5.27. Կոդով կառավարվող ժամանակահատվածների գեներատոր

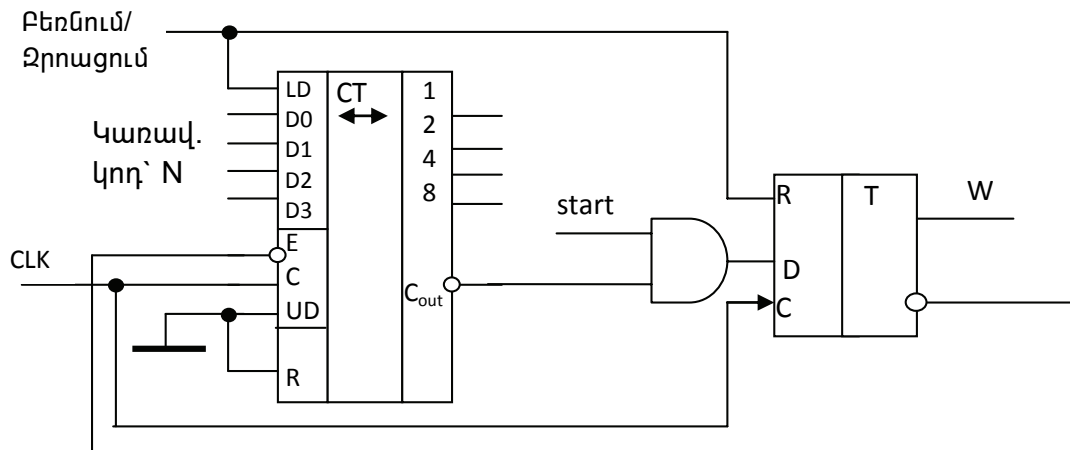


Նկ. 5.28. Կոդով կառավարվող ժամանակահատվածների գեներատորի ժամանակային դիագրամները

Աշխատանքի սկզբում հաշվիչը և կառավարող տրիգերը դրվում են 0 վիճակ: Կոդի կոմպարատորի մուտքերին տրվում է ձևավորվող  $T_W$  ժամանակահատվածի  $N$  կոդը: Աշխատանքն սկսվում է “start” ազդանշանով, որը տրիգերը հաջորդ տակտային իմպուլսով դնում է 1 վիճակ, իսկ տրիգերի ելքի  $W=1$  ազդանշանով թույլատրվում է հաշվիչի աշխատանքը:  $W$ -ն 1 վիճակ կարգելուց հետո հաջորդ տակտային իմպուլսից սկսած հաշվիչը հաշվում է: Այն պահին, երբ հաշվիչի ելքային  $Q$  կոդը հավասարվում է տրված  $N$  թվին, կոմպարատորի ելքն ընդունում է 1 արժեք, իսկ շրջված մուտքով 2ԵՎ տարրի ելքում ձևավոնվում է 0, որից հետո եկող հաջորդ տակտային իմպուլսով տրիգերը  $D$  մուտքից դրվում է 0 վիճակ: Տրիգերի 1 կարգելու և 0 իջեցնելու պահերը սինքրոնացված են տակտային ազդանշանի հետ, հետևաբար  $W=1$  ազդանշանի տևողության ճշգրիտ արժեքը կլինի՝  $T_W = N T_{CLK}$ :

Թվային կառավարումով ժամանակահատվածների ձևավորիչի մեկ այլ կառուցվածք ցույց է տրված նկ. 5.29, 30-ում: Այստեղ հաշվիչը  $UD=0$  ազդանշանով դրված է հանման ռեժիմ, իսկ  $R$  մուտքը չի գործում՝  $R=0$ : Նախապես “բեռնում/զրոյացում” ազդանշանով կառավարող կոդի արժեքը բեռնվում է հաշվիչ և միաժամանակ տրիգերը դրվում է 0 վիճակ: “start” ազդանշանի կիրառման պահից հետո եկող հաջորդ տակտային ազդանշանով տրիգերը դրվում է 1 վիճակ, որին հաջորդող տակտային իմպուլսից սկսած հաշվիչի պարունակությունը նվազեցվում է, մինչև որ այն դառնա 0:

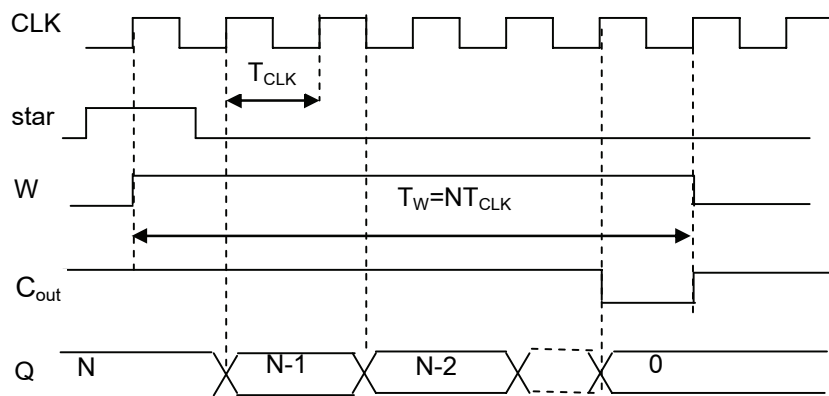
Հաշվիչը 0 վիճակին անցնելիս  $C_{out}$  ելքում ձևավորում է մեկ տակտ տևողությամբ 0 ազդանշան, որով դադարեցվում է հաշվիչի աշխատանքը: Այստեղ նույնպես տրիգերի 1 կարգելու և 0 իջեցնելու պահերը սինքրոնացված են տակտային ազդանշանի հետ, հետևաբար  $W=1$  ազդանշանի տևողության ճշգրիտ արժեքը կլինի՝  $T_W = N T_{CLK}$ :



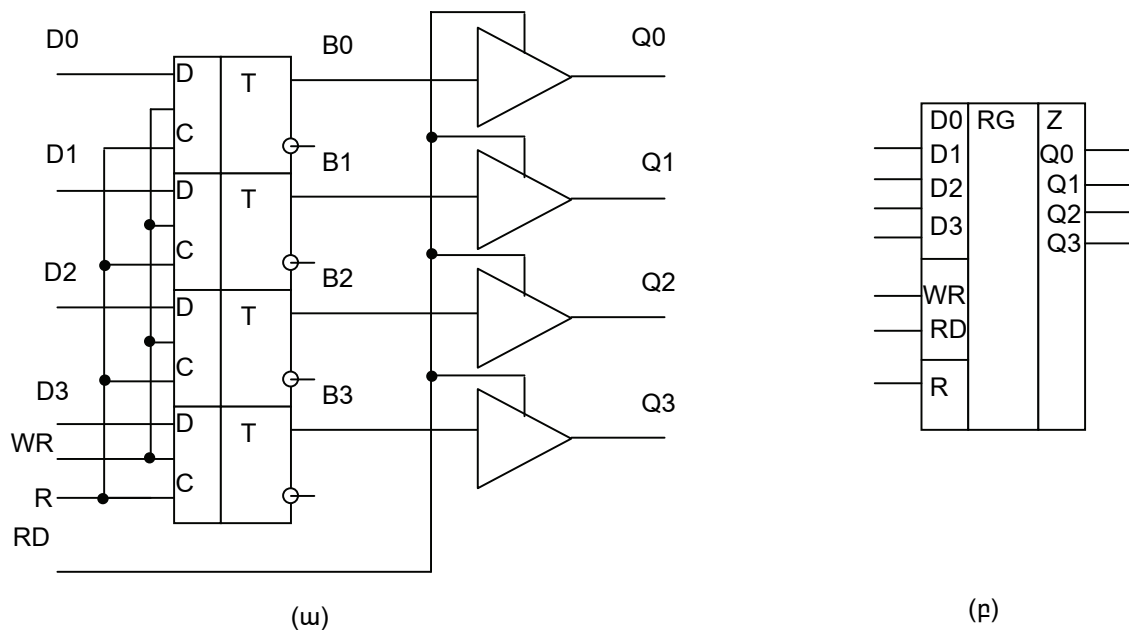
Նկ. 5.29. Թվային կառավարումով ժամանակահատվածների գեներատոր

### 5.8. Ձուգահեռ ռեգիստրներ

Ձուգահեռ կամ հիշողության ռեգիստրներն օգտագործվում են բազմակարգ երկուական թվերի կարճատև պահպանման համար: Դրանք տրիգերների հավաքածուներ են: Ռեգիստրի յուրաքանչյուր տրիգերում պահպանվում է  $n$ -կարգ թվի մեկ բիթը: Ռեգիստրի բոլոր տրիգերներն ունեն մեկ ընդհանուր գրանցման կառավարման  $WR$  մուտք և տվյալների առանձին մուտքեր՝  $D_0, D_1, \dots, D_{n-1}$ : Ռեգիստրի ելքերը կարող են լինել միշտ ակտիվ կամ կարող են լինել եռավիճակ բուֆերացմամբ, այսինքն՝ ռեգիստրում պահպանվող տվյալների ընթերցման համար պետք է ունենալ  $RD$  կառավարող ազդանշան, որով ելքի եռավիճակ բուֆերները կբերվեն ակտիվ վիճակ: Քառաբիթ ռեգիստրի կառուցվածքը ցույց է տրված Նկ. 5.31-ում:



Նկ. 5.30. Ժամանակահատվածների գեներատորի ժամանակային դիագրամները



Նկ. 5.31. Քառակարգ զուգահեռ ռեգիստրի սխեման  $D$  տրիգերների վրա (ա) և գրաֆիկական սիմվոլը (բ)

Գրանցման ռեժիմում  $D_0, D_1, \dots, D_{n-1}$  երկուսական թիվը զուգահեռ գրանցվում է բոլոր կարգերով  $D$  տրիգերներ՝  $C$  տակտային մուտքերին տրվող  $WR=1$  ազդանշանով, երբ զրոյացման ազդանշանը պասիվ է՝  $R=0$ :  $D$  տրիգերի անցումների հավասարումից ռեգիստրի համար կունենանք, որ գրանցման ռեժիմում

$$B_i^{t+1} = D_i^t, \quad i = 0, 1, \dots, n-1, \quad (5.33)$$

որտեղ  $B_0, B_1, \dots, B_{n-1}$  ներքին վիճակի փոփոխականներն են: Տվյալների գրանցման հաճախորդ  $B_i$  ներքին վիճակի փոփոխականներն են: Տվյալների գրանցման հաճախորդ  $B_i$  ներքին վիճակի փոփոխականներն են: Տվյալների գրանցման հաճախորդ  $B_i$  ներքին վիճակի փոփոխականներն են:

Թվի զուգահեռ ընթերցումը ռեգիստրից կատարվում է  $RD=1$  ազդանշանով, երբ  $R=0, WR=0$ .

$$Q_i^t = B_i^t, \quad i = 0, 1, \dots, n-1: \quad (5.34)$$

Երբ  $RD=0$ , ռեգիստրի  $Q_i$  ելքերը գտնվում են բարձր դիմադրության վիճակում:

Երբ  $R=0, WR=0, RD=0$ , առկա է ինֆորմացիայի պահպանման ռեժիմը՝

$$Q_i^{t+1} = Q_i^t, \quad i = 0, 1, \dots, n-1: \quad (5.35)$$

Սովորաբար զուգահեռ ռեգիստրները թողարկվում են 8-աբիթ կամ ավելի բարձր կարգաթվով: Ցածր կարգաթվով զուգահեռ ռեգիստր կարելի է կառուցել  $D$  տրիգերների ԻՍ-երի վրա, որոնք սովորաբար պարունակում են 4-ական տրիգեր:

### 5.9. Ընթերցվող-գրանցվող հիշասարքեր (ԸԳՀ)

Մեծ ծավալի տվյալներ պահպանելու համար օգտագործվում են ԸԳՀ-ներ՝ տվյալների գրանցման և ընթերցման հասցեային կազմակերպմամբ: Դա հնարավորություն է տալիս սահմանափակ թվով ելուստներով միկրոսխեմայում դիմել մեծ թվով հիշողության ռեգիստրների: Ի տարբերություն այլ տիպի հիշասարքերի, ԸԳՀ-ներում յուրաքանչյուր ռեգիստր կարող է ընտրվել անհատապես գրանցման/ընթերցման համար: Այդ պատճառով դրանք կոչվում են նաև կամայական ընտրությամբ հիշասարքեր (ԿԸՀ, Random Access Memory, RAM): ԿԸՀ-ները կառուցվում են երկու սկզբունքով՝ ստատիկ ԿԸՀ և դինամիկ ԿԸՀ: Ստատիկ ԿԸՀ-ում (ՍԿԸՀ, SRAM) հիշողության տարրական բջիջը ստատիկ տրիգեր է, իսկ դինամիկ ԿԸՀ-ում (ԴԿԸՀ, DRAM) հիշողության տարրական բջիջը ունակություն է, որում տրամաբանական 1 և 0 մակարդակներին համապատասխանում են ունակության լիցքավորված և լիցքաթափված վիճակները: Ունակությունից լիցքի ցրման պատճառով տրամաբանական մակարդակները ժամանակի ընթացքում աղավաղվում են: Այդ պատճառով դինամիկ հիշասարքում տվյալները պետք է պարբերաբար թարմացվեն՝ բջիջներից տվյալների ընթերցման և գրանցման ցիկլերի միջոցով: Չնայած այս հանգամանքին՝ դինամիկ հիշասարքերն ունեն շատ մեծ կիրառություն հաշվողական համակարգերում, քանի որ դրանք ապահովում են պահպանվող տվյալների ավելի մեծ խտություն՝ ստատիկ հիշողությունների համեմատ: Ցանկացած դեպքում ԿԸՀ-ն կարելի է պատկերացնել իբրև հիշողության ռեգիստրների բազմություն (ինչպես ռեգիստրների ֆայլը), որում յուրաքանչյուր ռեգիստր ունի իր անհատական հասցեն: Հիշողության ռեգիստրի  $n_w$  կարգաթիվը կոչվում է պահպանվող բառի երկարություն:  $n_a$  հասցեային մուտքերի դեպքում հիշասարքը բաղկացած է  $N=2^{n_a}$  ռեգիստրներից, հետևաբար հիշասարքի տեղեկատվական ունակությունը կլինի  $C = N \times n_w = 2^{n_a} \times n_w$  բիթ: Սովորաբար հիշողության ԻՍ-երում բառի երկարությունը կարող է լինել 4, 8 կամ 16 բիթ: Օրինակ,  $n_w = 8$  բիթ (1 բայթ) երկարությամբ բառի և  $n_a = 12$  հասցեային մուտքերի դեպքում հիշասարքի ունակությունը կլինի  $2^{12} \times 8 = 4096 \times 8$  բիթ = 4096 բայթ (ընդունված է ասել 4 կբայթ):

Հաճախ թվային համակարգում պահանջվող հիշողության ծավալը հնարավոր չէ ապահովել մեկ ԻՍ-ով, և անհրաժեշտ է լինում դա անել մի քանի ԻՍ-երի միջոցով, այսինքն՝ ընդարձակել հիշողության տարածքը:

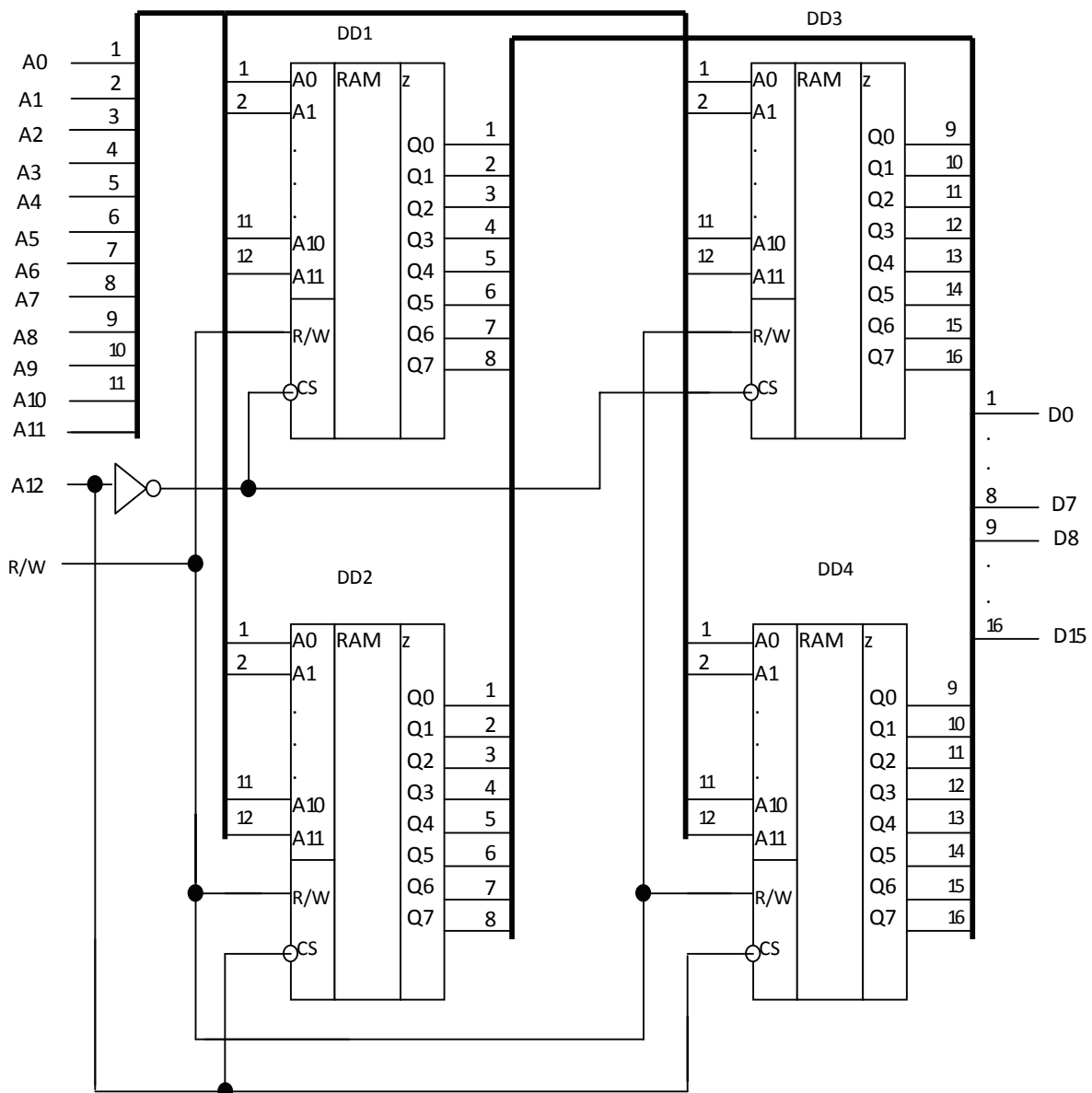
Նկ. 5.32-ում ցույց է տրված  $8096 \times 16$  բիթ ծավալով ԿԸՀ-ի սխեմա՝ կառուցված  $4096 \times 8$  բիթ ունակությամբ հիշողության ԻՍ-երի վրա:

Յուրաքանչյուր ԻՍ ունի 12 հասցեային մուտք՝  $A_0, A_1, \dots, A_{11}$ , տվյալների 8 մուտք/ելք ելուստ՝  $D_0, D_1, \dots, D_7$ , ընթերցման/գրանցման կառավարման  $R/W$  մուտք և ԻՍ-ի ընտրության  $CS$  (Chip Select) մուտք: ԻՍ-ից տվյալների ընթերցման համար անհրաժեշտ է հաստատել  $R/W=1$  և ակտիվացնել  $CS=0$  ազդանշան: Գրանցման համար անհրաժեշտ է հաստատել  $R/W=0$  և  $CS=0$ : Երբ ԻՍ-ին տրվում է  $CS=1$  ազդանշան, արգելվում են գրանցման և ընթերցման գործողությունները, իսկ  $D_0, D_1, \dots, D_7$  մուտք-/ելք գծերը բերվում են բարձր դիմադրության վիճակ: Այս հատկությունը թույլ է տալիս ԻՍ-երի համարժեք ելքերը միացնել միմյանց ամիջական կապերով՝ առանց միջանկ-

յալ տրամաբանական տարրերի, եթե այդ ԻՍ-երի CS ազդանշանները միաժամանակ չեն կարող ակտիվ լինել:

Հիշողության հասցեային տարածքի 0-ից 4095 հասցեների բջիջներն ընտրվում են DD1 և DD3 ԻՍ-երից, երբ  $A12=0$ : Այդ ԻՍ-երում պահվող յուրաքանչյուր բառի 8 ցածր բիթերը՝ D0, D1, ..., D7 պահվում են DD1-ի, իսկ 8 բարձր բիթերը՝ D8, D1, ..., D15 պահվում են DD3-ի համանուն հասցեով բջիջներում: Հիշողության բարձր՝ 4096-ից 8095 հասցեների բջիջներն ընտրվում են DD2 և DD4 ԻՍ-երից, երբ  $A12=1$ :

Հիշողության հասցեային տարածքի ընդարձակման համար (հիշողության ընդարձակում ուղղահայաց ուղղությամբ) օգտագործվում է ԻՍ-երի CS մուտքը: Եթե հիշողություն համակարգի անհրաժեշտ հասցեների թիվը  $2^n$  է, իսկ հիշողության ԻՍ-ինը՝  $2^{n1}$ , ապա պետք է օգտագործել  $2^n / 2^{n1} = 2^{n-n1}$  ԻՍ-եր: Համակարգի ցածր  $A_0, \dots, A_{n1-1}$  հասցային գծերը միացվում են ամիջապես բոլոր ԻՍ-երի հասցեային մուտքերին, իսկ բարձր՝  $A_{n1}, \dots, A_n$  գծերը տրվում են հասցեների արտաքին վերձանիչին, որը բարձր բիթերից ձևավորում է ԻՍ-երի ընտրության թվով  $2^{n-n1}$  CS ազդանշանները: Հիշողությունում պահվող բառի կարգայնության ընդարձակման (հորիզոնական ընդարձակում) համար բառի բիթերի առանձին խմբեր պահվում են տարբեր ԻՍ-երում, որոնք ընտրվում են նույն CS ազդանշանով:  $m$ -բիթ բառը  $m_1$ -բիթ բառի կարգայնությամբ ԻՍ-երում պահպանելու համար անհրաժեշտ է օգտագործել թվով  $[m/m_1]$  ԻՍ-եր: Հետևաբար  $C_1 = 2^{n1} \times m_1$  բիթ ունակությամբ ԻՍ-երի վրա  $C = 2^n \times m$  բիթ անհրաժեշտ ծավալով հիշողության համակարգ կառուցելու համար պետք է օգտագործել  $2^{n-n1} \times [m/m_1]$  ԻՍ-եր:



Նկ. 5.32. 8096 x 16 բիթ ծավալով ԿԸՀ-ի սխեմա՝ կառուցված 4096 x 8 բիթ ունակությամբ հիշողության ԻՍ-երի միջոցով

### 5.10. Ռեգիստրների ֆայլ

Տվյալների մշակման համակարգերում հարմար է միջանկյալ տվյալները պահպանել ռեգիստրներում: Դրա համար կապահանջվի մեծ թվով ռեգիստրներ: Օգտագործվող ԻՍ-երի թվի նվազեցման և բարձր արագագործությունն ապահովելու համար նպատակահարմար է օգտագործել մեկ ԻՍ-ում ինտեգրված ռեգիստրների հավաքածու, որը կոչվում է ռեգիստրների ֆայլ:

Նկ. 5.33-ում ցույց է տրված չորս ռեգիստրից բաղկացած՝ RG0, RG1, RG2, RG3 ռեգիստրների ֆայլի կառուցվածքային սխեման: Տվյալների ընդհանուր  $DI = \{DI_0, DI_1,$

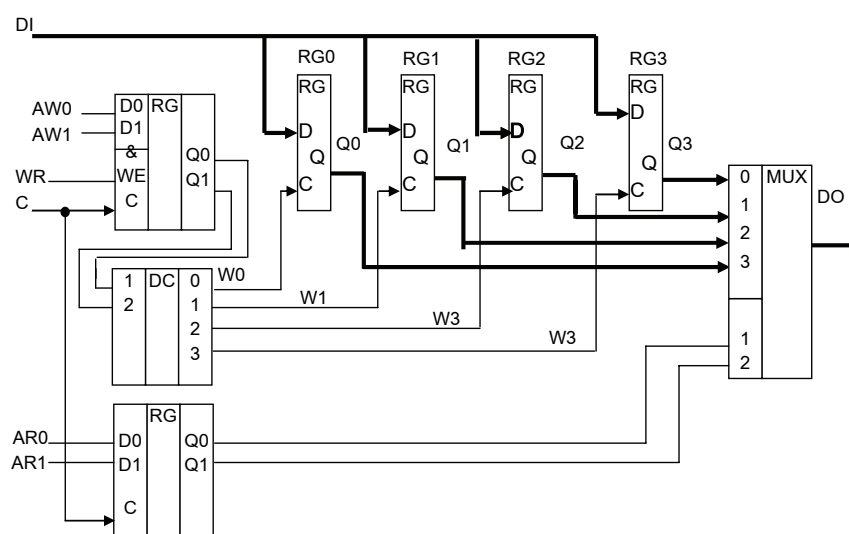
... ,  $DI_{n-1}$  } մայրուղին ամիջականորեն միացված է բոլոր ռեգիստրների D մուտքերին: Տվյալների արտածումը ռեգիստրներից իրականացվում է առանձին DO ելքային մայրուղով: Ռեգիստրային ֆայլերը ԿԸՀ-ների նկատմամբ ունեն բարձր արագագործություն, որին հասնում են ընթերցման և գրանցման գործողությունների համատեղմամբ: Այդ նպատակով ռեգիստրների ֆայլում գրանցման և ընթերցման համար ռեգիստրներն ընտրվում են համապատասխանաբար գրանցվող ռեգիստրի AW հասցեով և ընթերցվող ռեգիստրի AR հասցեով՝ 00 դեպքում ընտրվում է RG0 ռեգիստրը, 01 դեպքում՝ RG1-ը, 10 դեպքում RG2-ը, 11 դեպքում՝ RG3-ը: Ընդ որում, ժամանակի միևնույն պահին գրանցման և ընթերցման հասցեները պետք է տարբեր լինեն, քանի որ նույն ռեգիստրում գրանցման ժամանակ նրա տրիգերները կատարում են անցումներ, և անցողիկ պրոցեսի ժամանակ տվյալները կարող են ճիշտ չընթերցվել:

Համակարգի հետ սինքրոն աշխատանքի համար AW և AR հասցեները սկեռվում են ռեգիստրներում համակարգային C տակտային ազդանշանով: Ռեգիստրներում գրանցման  $W0, \dots, W3$  ազդանշանները ձևավորվում են ռեգիստրում սկեռված հասցեից՝ վերծանիչի միջոցով: Ընթերցման համար AR հասցեով ընտրված ռեգիստրի ելքային տվյալները MUX մուլտիպլեքսորի միջոցով փոխանցվում են ռեգիստրային ֆայլի ընդհանուր DO ելք:

Տվյալների փոխանակման արագությունը բարձրացնելու համար հաճախ նախատեսվում է տարբեր հանգույցներից ռեգիստրների ֆայլին տվյալներին զուգահեռ դիմելու հնարավորություն: Օրինակ, K561ИР11 և K561ИР12 ռեգիստրների ֆայլերում տվյալները ռեգիստրներից կարելի է ընթերցել միաժամանակ ըստ երկու հասցեների՝ ARA և ARB, համապատասխանաբար DOA և DOB ելքերից: Նկ.5.34-ում ցույց է տրված K561ИР11 ռեգիստրների ֆայլի կառուցվածքային սխեման և գրաֆիկական նշանը:

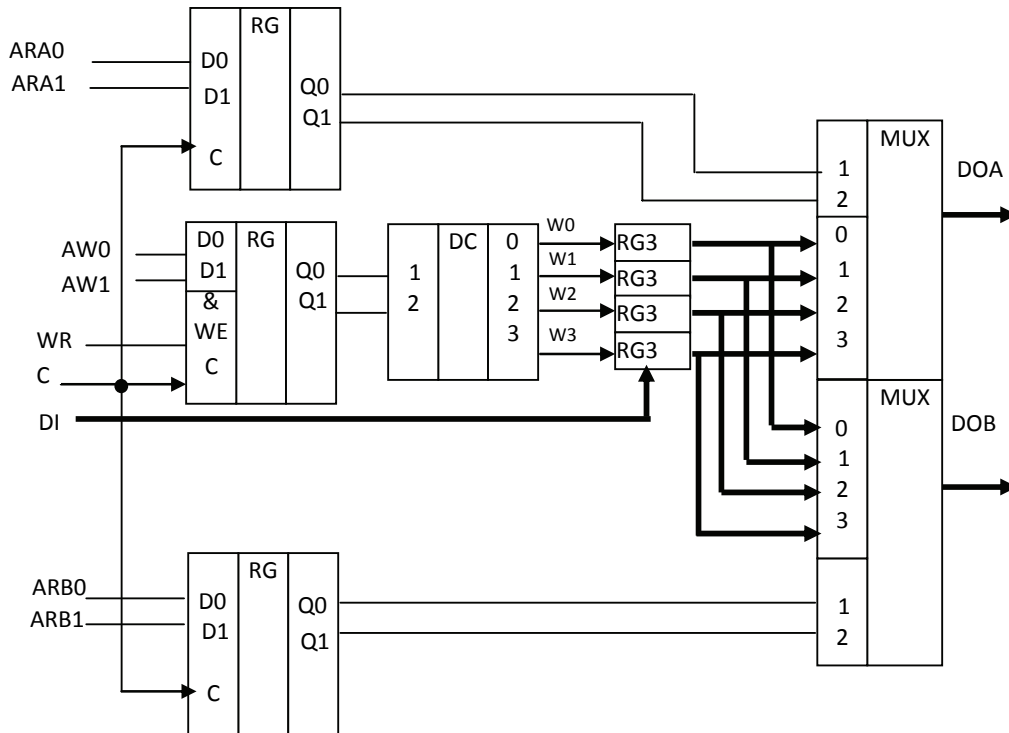
### 5.11. Տեղաշարժող ռեգիստրներ

Տեղաշարժող կամ հաջորդական ռեգիստրները, բացի տվյալների պահպանումից, կարող են նաև տվյալների հետ իրականացնել տրամաբանական և թվաբանական գործողություններ: Ռ-բիթ տեղաշարժող ռեգիստրը կառուցվում է թվով



Նկ. 5.33. Չորս ռեգիստրից կազմված ռեգիստրների ֆայլի կառուցվածքը



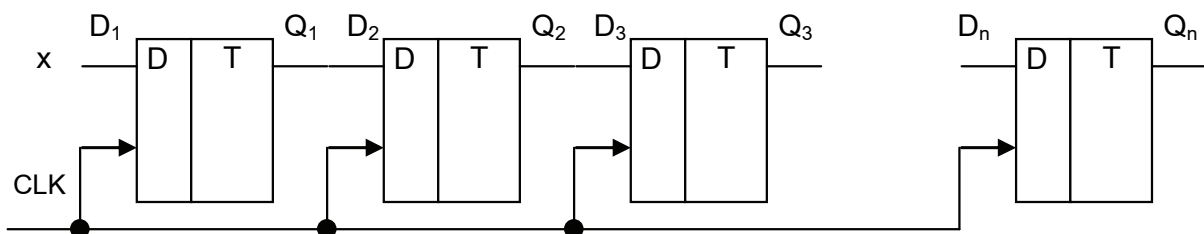


Նկ. 5.34. Երկու ելքերով ռեգիստրների ֆայլի կառուցվածքը

ո տրիգերների վրա (Նկ.5.35), որտեղ նախորդ տրիգերի ելքը միացված է հաջորդ տրիգերի տվյալների մուտքին: Տակտային իմպուլսով նախորդ տրիգերի պարունակությունը գրանցում է հաջորդ տրիգեր՝ չփոխելով 0-ների և 1-երի հաջորդականությունը: D տրիգերների վրա կառուցված տեղաշարժող ռեգիստրում տրիգերների մուտքերի ֆունկցիաներն ունեն հետևյալ տեսքը.

$$D_1^t = x^t, \quad D_r^t = Q_{r-1}^t, \quad r = 2, 3, \dots, n; \quad (5.36)$$

Այստեղից բխում է, որ որևէ t-րդ տակտում r-րդ տրիգերում գտնվող տվյալի բիթը (t+1)-րդ տակտում փոխանցվում է (r+1)-րդ տրիգեր, այսինքն՝ առկա է տվյալների տեղաշարժ: Ռեգիստրի հաջորդական մուտքին կիրառված x ազդանշանը կհայտնվի նրա ելքում n տակտերից հետո:



Նկ. 5.35. Բազմակարգ տեղաշարժող ռեգիստր

Եթե  $Q_n$ -ը ռեգիստրում գրված երկուական թվի բարձր կարգն է, տեղաշարժը կատարվում է բարձր կարգերի ուղղությամբ կամ դեպի ձախ: Պահպանվող  $X = x_n, x_{n-1}, \dots$ ,

$x_r, x_{r-1}, \dots, x_2, x_1$  թիվը մեկ կարգով տեղաշարժից հետո ձևափոխվում է  $Y=y_n, y_{n-1}, \dots, y_r, y_{r-1}, \dots, y_2, y_1$  թվի, որտեղ՝

$$y_r = \begin{cases} x_{r-1}, & n \geq r > 1 \\ x, & r = 1 \end{cases} \quad (5.37)$$

Եթե  $x=0$ , ապա դեպի ձախ մեկ բիթով տեղաշարժի արդյունքում ռեգիստրի պարունակությունը կբազմապատկվի երկուսով ըստ  $M=2^n$  մոդուլի: Արդյունքի բարձր բիթը, որի քաշային գործակիցը  $2^n$  է, տեղաշարժի հետևանքով ռեգիստրից դուրս է բերվում՝ կորչում է: Ուստի, եթե արդյունքը մեծ է կամ հավասար  $2^n$ , այսինքն՝ տեղաշարժվող թվի բարձր բիթը 1 է, ապա արդյունքում, բարձր բիթի բացակայության պատճառով, ռեգիստրում կունենանք՝  $2^*X-2^n$ : Օրինակ, եթե  $X=(01101)_2=13_{10}$  հետևաբար մեկ կարգով տեղաշարժից հետո կստացվի՝  $Y=(11010)_2=26_{10}$ : Իսկ  $X=(10011)_2=19_{10}$  դեպքում, մեկ կարգով տեղաշարժից հետո կստացվի՝  $Y=(00110)=6_{10}$ , որը  $2^*19=38$  թիվը  $2^5=32$ -ի ( $n=5$ )-ի բաժանումից ստացված մնացորդն է (կամ  $6=38-32$ ):

Եթե  $Q_n$ -ը ռեգիստրում պահվող թվի ցածր կարգն է, տեղաշարժը կատարվում է թվի ցածր կարգերի ուղղությամբ կամ դեպ աջ: Երբ  $Q_n=0$  մեկ կարգով դեպի աջ տեղաշարժը համարժեք է թվի բաժանմանը 2-ի, արդյունքի ամբողջ մասը մնում է ռեգիստրում, իսկ մնացորդը՝  $n$ -րդ տրիգերի պարունակությունը մինչև տեղաշարժը, կորչում է: Օրինակ,  $X=(10111)_2=23_{10}$ , եթե թիվը ռեգիստրում գրված է ցածր կարգով առաջ ընկած՝  $Q_5=1, Q_4=1, Q_3=1, Q_2=0, Q_1=1$ , ապա տեղաշարժից հետո կստացվի՝  $Q_5=1, Q_4=1, Q_3=0, Q_2=1, Q_1=0$ , այսինքն՝  $Y=(01011)_2=11_{10}$ :

Տարբերում են երկուսական թվի թվաբանական տեղաշարժ, երբ  $x=0$  և տրամաբանական տեղաշարժ կամ ցիկլային պտույտ, երբ  $x=Q_n$ , այսինքն բարձր կարգի տրիգերի ելքը միացված է ցածր կարգի տրիգերի մուտքին՝ կազմելով օղակ:

Եթե տեղաշարժող ռեգիստրում հաջորդ տրիգերի ելքը միացվի նախորդի մուտքին, կունենանք տեղաշարժ դեպի աջ: Տրիգերների մուտքերի հավասարումներն այս դեպքում կլինեն.

$$D_r^t = Q_{r+1}^t, \quad r = 1, \dots, n-1, \quad D_n^t = y^t, \quad (5.38)$$

որտեղ  $y$ -ը  $n$ -րդ տրիգերի մուտքին արտաքինից տրվող ազդանշանն է:

Գործնականում հաճախ անհրաժեշտ է ունենալ ռեգիստրում պահպանվող տվյալները և՛ աջ, և՛ ձախ տեղաշարժելու հնարավորություն: Այդպիսի ռեգիստրները կոչվում են դարձափոխելի: Այս ռեգիստրներն ունեն տեղաշարժի ուղղության ընտրության LR մուտք: Երբ  $LR=1$ , տեղի է ունենում տեղաշարժ դեպի ձախ, իսկ երբ  $LR=0$ , տեղի է ունենում տեղաշարժ դեպի աջ (երկու դեպքում էլ թվի բարձր կարգը գտնվում է  $n$ -րդ տրիգերում): Դարձափոխելի ռեգիստրի տրիգերների մուտքերի հավասարումները կարելի է ստանալ (5.36) և (5.38)-ից

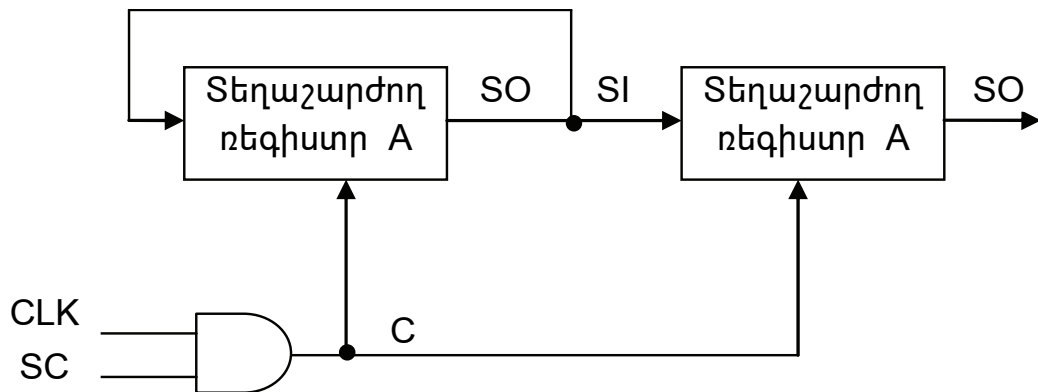
$$\begin{aligned} D_1^t &= Q_2^t \& \overline{LR} + x^t \& LR, \\ D_n^t &= y^t \& \overline{LR} + Q_{n-1}^t \& LR, \\ D_r^t &= Q_{r+1}^t \& \overline{LR} + Q_{r-1}^t \& LR, \quad r = 2, 3, \dots, n-1 \end{aligned} \quad (5.39)$$

### 5.11.1. Տվյալների հաջորդական փոխանցում թվային համակարգերում

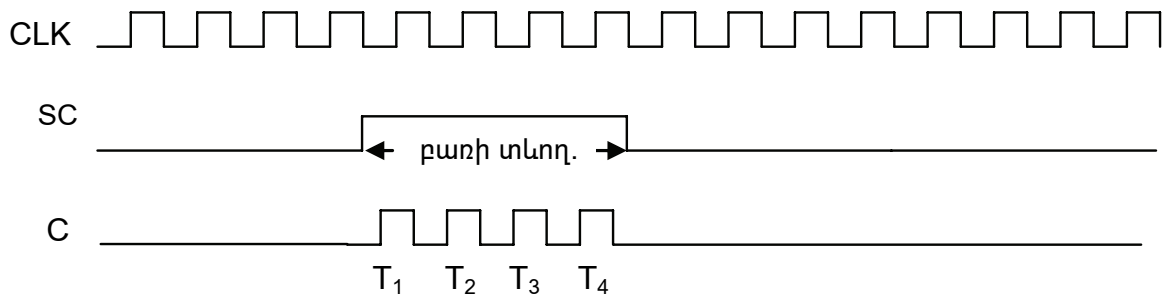
Թվային համակարգերում տվյալների փոխանցումը համակարգի բաղադրիչների միջև կարելի է իրականացնել զուգահեռ կամ հաջորդական ձևաչափով: Հաջորդական ռեժիմում աշխատող թվային համակարգում տվյալները փոխանցվում և մշակվում են բիթ առ բիթ, յուրաքանչյուր տակտում փոխանցվում կամ մշակվում է բազմակարգ բառի մեկ բիթ:

Տվյալների հաջորդական փոխանցումը կարելի է իրականացնել տեղաշարժող ռեգիստրների միջոցով: Նկ. 5.36–ում ցույց է տրված A ռեգիստրից B ռեգիստր տվյալների հաջորդական փոխանցման սխեման: A ռեգիստրի SO հաջորդական ելքից տվյալները գնում են B ռեգիստրի SI հաջորդական մուտք: Փոխանցվող տվյալների պահպանման համար A ռեգիստրի ելքը միացված է մուտքին (A ռեգիստրն աշխատում է ցիկլային տեղաշարժի ռեժիմում): B ռեգիստրի սկզբնական տվյալները դուրս են մղվում իր հաջորդական ելքից և կորչում են (հիարկե, եթե դրանք չեն պահվում մեկ այլ ռեգիստրում): Տեղաշարժի կառավարման SC մուտքը որոշում է՝ երբ և քանի տակտային ազդանշանով է իրականացվում տեղաշարժը: Դա արված է ԵՎ տարրով, որը թույլ է տալիս տակտային ազդանշաններից ազդել ռեգիստրի տրիգերների վրա, եթե  $SC=1$ :

Ենթադրենք տեղաշարժող ռեգիստրը քառակարգ է: Կառավարման համակարգը պետք է կառուցված լինի այնպես, որ ակտիվացնի  $SC=1$  ազդանշանը տակտային իմպուլսի չորս պարբերությանը համարժեք ժամանակահատվածում: Տվյալների փոխանցման ժամանակային դիագրամները ցույց են տրված նկ. 5.37-ում: Ռեգիստրների տեղաշարժը իրականացվում է տակտային CLK իմպուլսի դրական ճակատով:



Նկ. 5.36. Տեղաշարժող ռեգիստրներով տվյալների փոխանցման համակարգի կառուցվածքը



Նկ. 5.37. Տվյալների փոխանցման համակարգի ժամանակային դիագրամները

SC ազդանշանը պետք է սինքրոնացված լինի CLK իմպուլսի հետ այնպես, որ SC-ը պետք է փոխանջատվի CLK-ի բացասական ճակատից ամիջապես հետո: Հաջորդ չորս CLK իմպուլսի տևողության ընթացքում SC=1: Չորրորդ՝ T<sub>4</sub> իմպուլսից հետո SC-ն դրվում է 0 վիճակ, և հետագա տեղաշարժն արգելվում է: Ենթադրենք մինչև տեղաշարժը A ռեգիստրի պարունակությունը 1011 է, իսկ B ռեգիստրինը՝ 0010: Տվյալների հաջորդական փոխանցումը A-ից B ռեգիստր չորս տակտերում ցույց է տրված աղյուսակ 5.5-ում: Առաջին (T<sub>1</sub>) իմպուլսից հետո A-ի բարձր բիթը (ֆիզիկապես ամենաաջ բիթը) փոխանցվում է B-ի ամենացածր բիթ (ամենաձախ բիթ), միաժամանակ այդ բիթը փոխանցվում է A-ի ամենաձախ բիթ: A-ի և B-ի մյուս բիթերը տեղաշարժվում են մեկ կարգով աջ: Հաջորդ երեք իմպուլսները կատարում են համանման գործողություններ: Չորրորդ իմպուլսից հետո SC-ն դրվում է 0 վիճակ և A ու B ռեգիստրները կպարունակեն նույն տվյալները՝ 1011: Այսպիսով A-ի պարունակությունը փոխանցվեց B ռեգիստր, և A-ի պարունակությունը չփոխվեց:

Տակտային իմպուլսի պարբերությունը կոչվում է բիթի տևողություն, ռեգիստրի ամբողջ պարունակության տեղաշարժի ժամանակը կոչվում է բառի տևողություն: Տվյալների զուգահեռ փոխանցման ժամանակ տվյալի ամբողջ բառը փոխանցվում է մեկ տակտում, այսինքն՝ բառի տևողությունը հավասար է բիթի տևողությանը, իսկ հաջորդական փոխանցման ժամանակ բառի տևողությունը հավասար է բիթի տևողության և բիթերի թվի արտադրյալին: ո-բիթ տվյալների հաջորդական փոխանցումը զուգահեռի նկատմամբ ո անգամ դանդաղ է: Սակայն հաջորդական փոխանցման առավելությունն այն է, որ անկախ բառի կարգաթվից, միայն մեկ կապի գիծ է անհրաժեշտ: Տվյալների հաջորդական փոխանցումը լայնորեն կիրառվում է համակարգչային և հեռահաղորդման համակարգերում:

Աղյուսակ 5.5

Տակտեր	A ռեգիստր	B ռեգիստր	B-ի հաջորդական ելք
Սկզբ. վիճակ	0 1 1 1	0 0 1 0	0
T <sub>1</sub> -ից հետո	1 1 0 1	1 0 0 1	1
T <sub>2</sub> -ից հետո	1 1 1 0	1 1 0 0	0
T <sub>3</sub> -ից հետո	0 1 1 1	0 1 1 0	0
T <sub>4</sub> -ից հետո	1 0 1 1	1 0 1 1	1

### 5.11.2. Համապիտանի ռեգիստրներ

Տեղաշարժող ռեգիստրներն օգտագործվում են՝ զուգահեռ տվյալները հաջորդականի և հաջորդականը զուգահեռի ձևափոխման համար: Եթե ռեգիստրի տրիգերների ելքերը մատչելի են ընթերցման համար, ապա հաջորդաբար ներածված տվյալները կարելի է զուգահեռ վերցնել տրիգերների ելքերից: Եթե տեղաշարժող ռեգիստրում ավելացվի տվյալների զուգահեռ բեռնման հնարավորություն, զուգահեռ ներածված տվյալները կարելի է տեղաշարժերի միջոցով դուրս բերել հաջորդական ելքից: Այդպիսի ռեգիստրը կոչվում է զուգահեռ բեռնավորումով տեղաշարժող ռեգիստր:

Ընդհանուր դեպքում, համապիտանի ռեգիստրը կարող է օժտված լինել ստորև բերված բոլոր ֆունկցիաներով կամ դրանցից մի մասով, բայց անապառաժ պետք է ունենա տեղաշարժի ֆունկցիա.

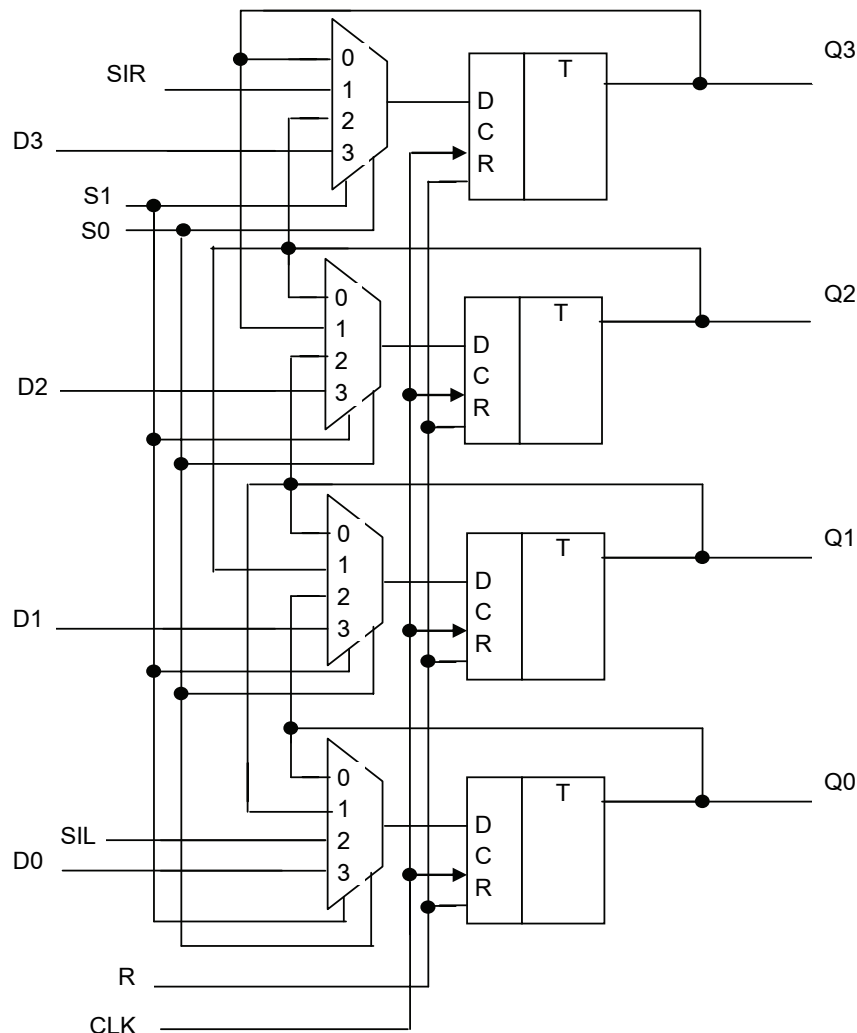
1. Ջրոյի բերման կառավարման R մուտք:
2. CLK տակտային մուտք, որը հնարավորություն է տալիս սինքրոնացնել բոլոր գործողությունները:
3. Դեպի ձախ տեղաշարժի թույլտվություն SL՝ հաջորդական մուտքը և ելքը կոնֆիգուրացվում են այդ ազդանշանով:
4. Դեպի աջ տեղաշարժի թույլտվություն SR՝ հաջորդական մուտքը և ելքը կոնֆիգուրացվում են այդ ազդանշանով: SL և SR ազդանշանների փոխարեն ռեգիստրը կարող է ունենալ տեղաշարժի թույլտվության SC և տեղաշարժի ուղղության կառավարման LR ազդանշաններ՝  $SL=SC\&LR$ ,  $SR=SC\&\overline{LR}$ :
5. Չուգահեռ բեռնավորման կառավարում PL և դրա հետ կապված տվյալների զուգահեռ n մուտքեր՝  $D_1, D_2, \dots, D_n$ :
6. Տվյալների n զուգահեռ ելքեր՝  $Q_1, Q_2, \dots, Q_n$ :
7. Պահպանման վիճակ, որում ռեգիստրի տվյալները չեն փոխվում, եթե նույնիսկ անընդհատ կիրառվում են տակտային իմպուլսներ:

Գործնականում այս բոլոր ֆունկցիաների կառավարման ազդանշանները կարող են համակցվել տարբեր եղանակներով: Համակցման այդպիսի մեկ օրինակ բերվեց վերևում՝ 4-րդ կետում:

Նկ. 5.38-ում բերված է համապիտանի ռեգիստրի կառուցվածքային սխեման, որն ունի նշված բոլոր ֆունկցիաները: Ռեգիստրը կառուցված է չորս D տրիգերով: Կառավարման ֆունկցիաների մեծ մասն իրականացվում է տրիգերների մուտքային ազդանշանների ընտրության մուլտիպլեքսորների միջոցով, որոնք կառավարվում են ընտրության  $s_1s_0$  ազդանշաններով: Երբ  $s_1s_0=00$ , ընտրվում է մուլտիպլեքսորի 0 մուտքը, ռեգիստրի ներկա վիճակի արժեքը կիրառվում է տրիգերների մուտքերին: Այս դեպքում յուրաքանչյուր տրիգերի ելքից ստեղծում է ազդանշանի փոխանցման ուղի դեպի նույն տրիգերի մուտք: Հաջորդ տակտային իմպուլսով յուրաքանչյուր տրիգերի ելքից վիճակը նորից գրանցում է այդ նույն տրիգերի մեջ, հետևաբար ռեգիստրի վիճակը չի փոխվում: Երբ  $s_1s_0=01$ , ընտրվում է մուլտիպլեքսորի 1 մուտքը, որը միացված է հաջորդ (բարձր) կարգի տրիգերի ելքին: Այս դեպքում ունենք տեղաշարժ դեպի

աջ (դեպի ցածր կարգերը), աջ տեղաշարժի հաջորդական SIR մուտքը միանում է չորրորդ տրիգերի D մուտքին: Երբ  $s_1s_0=10$ , ընտրվում է մուլտիպլեքսորի 2 մուտքը, որը միացված է նախորդ (ցածր) կարգի տրիգերի ելքին: Այս դեպքում ունենք տեղաշարժ դեպի ձախ (դեպի բարձր կարգերը), ձախ տեղաշարժի հաջորդական SIL մուտքը միանում է առաջին տրիգերի D մուտքին: Եվ վերջապես, երբ  $s_1s_0=11$ , ընտրվում է մուլտիպլեքսորների թիվ 3 մուտքը՝ զուգահեռ բեռնավորման D0, D1, D2, D3 մուտքերը: Հաջորդ տակտային իմպուլսով զուգահեռ տվյալները D0, D1, D2, D3 մուտքերից գրանցվում են ռեգիստրի տրիգերներ: Ռեգիստրի կառավարման ռեժիմները նկարա - գրված են աղյուսակ 5.6–ում:

Ձանգվածային թողարկվող ԿՄՕԿ տեղաշարժող ռեգիստրներից պարզագույնը 564IP2 ԻՍ-ն է, որը պարունակում է երկու քառակարգ ռեգիստր (նկ.5.39): Ասինքրոն R մուտքը ծառայում է ռեգիստրը 0 վիճակ դնելու համար, SI-ն սինքրոնացված հաջորդական մուտքն է, ռեգիստրի տրիգերների ելքերը մատչելի են:

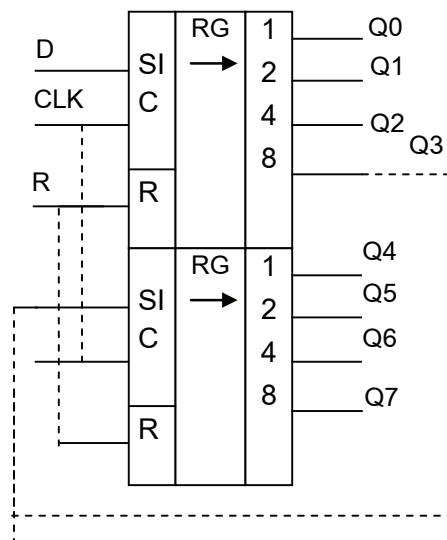


Նկ. 5.38. Համապիտանի ռեգիստրի կառուցվածքային սխեման

Աղյուսակ 5.6

Կառավարման ազդանշաններ		Կառավարման ռեժիմ
$s_1$	$s_0$	
0	0	Վիճակի պահպանում
0	1	Տեղաշարժ դեպի աջ
1	0	Տեղաշարժ դեպի ձախ
1	1	Չուգահեռ բեռնում

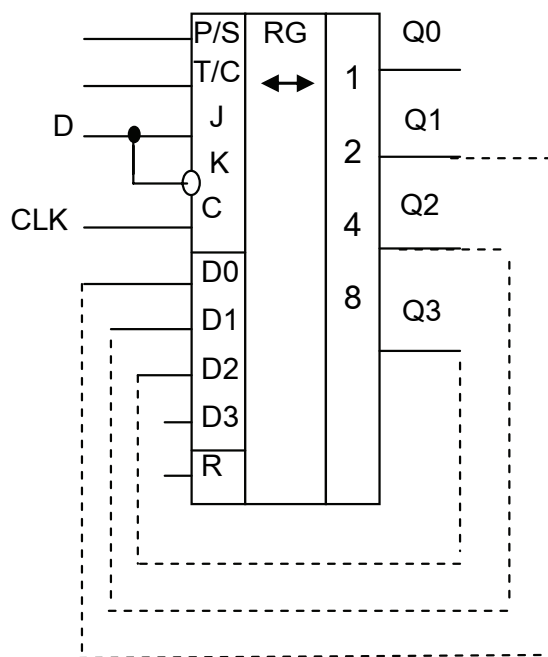
Ռեգիստրն իրականացնում է տեղաշարժ դեպի ձախ, իսկ հաջորդական ելք կարող է ծառայել 8 քաշային գործակցով բարձր կարգի ելքը: Երկու ռեգիստրները հաջորդաբար միացնելով (նկարում ցույց է տրված կետագծերով) կարելի է ստանալ 8-կարգ ռեգիստր: Այս ռեգիստրը կարող է ծառայել որպես հաջորդական կոդը զուգահեռի ձևափոխիչ. թիվը հաջորդաբար ներածվում է SI մուտքից տակտային իմպուլսներով, իսկ զուգահեռ կոդը կարդացվում է ռեգիստրի Q0, ..., Q7 ելքերից:



Նկ. 5.39. 564IP2 քառակարգ տեղաշարժող ռեգիստրի ԻՍ

Նկ. 5.40–ում ցույց է տրված 564IP9 համապիտանի ռեգիստրի գրաֆիկական սիմվոլը: ԻՍ-ի ելուստներն ունեն հետևյալ ֆունկցիաները՝ D0, D1, D2, D3 զուգահեռ մուտքերն են, J և  $\bar{K}$ -ն հաջորդական մուտքերն են (դրանց միավորումը համարժեք է D մուտքի), P/S-ը ռեժիմի կառավարման մուտքն է (P/S=1 դեպքում տեղի ունի զուգահեռ մուտք D0, D1, D2, D3 մուտքերից, P/S=1 դեպքում՝ հաջորդական մուտք D մուտքից), T/C-ելքի կոդի տեսքի կառավարում (T/C=1 դեպքում ռեգիստրի պարունակությունը Q0, Q1, Q2, Q3 ելքերում ներկայացվում է ուղիղ կոդով, T/C=0 դեպքում՝ հակադարձ կոդով): Այս ԻՍ-ով կարելի է իրականացնել մուտք-ելք հետևյալ ռեժիմները՝ (ա) զուգահեռ մուտք D0, D1, D2, D3-ից (P/S=1), զուգահեռ ելք Q0, Q1, Q2, Q3 ելքերից, (բ)

հաջորդական մուտք D-ից ( $P/S=0$ ), զուգահեռ ելք  $Q0$ ,  $Q1$ ,  $Q2$ ,  $Q3$ -ից, (զ) զուգահեռ մուտք  $D0$ ,  $D1$ ,  $D2$ ,  $D3$ -ից ( $P/S=1$ ), հաջորդական ելք  $Q3$ -ից, (դ) հաջորդական մուտք D-ից ( $P/S=0$ ), հաջորդական ելք  $Q3$ -ից: 5641P9 ԻՍ-ը կարելի է օգտագործել նաև որպես դարձափոխելի ռեգիստր: Դրա համար բավարար է կատարել նկ. 5.40–ում կետագծերով ցույց տրված միացումները: Ռեգիստրի տվյալները դեպի ձախ շեղելու համար տրվում է  $P/S=0$  կառավարող ազդանշան, մուտքային հաջորդական տվյալները տրվում են D-ին, իսկ հաջորդական տվյալները դուրս են բերվում  $Q3$ -ից: Ռեգիստրի տվյալները դեպի աջ շեղելու համար տրվում է  $P/S=1$  կառավարող ազդանշան, մուտքային հաջորդական տվյալները տրվում են  $D3$ -ին, իսկ հաջորդական տվյալները դուրս են բերվում  $Q0$ -ից:



Նկ. 5.40. 5641P9 համապիտանի ռեգիստրի գրաֆիկական սիմվոլը

Թողարկվում են նաև ավելի մեծ կարգայնությամբ և ավելի մեծ ֆունկցիոնալ հնարավորություններով տեղաշարժող ռեգիստրներ:

## 5.12. Օղակաձև հաշվիչներ

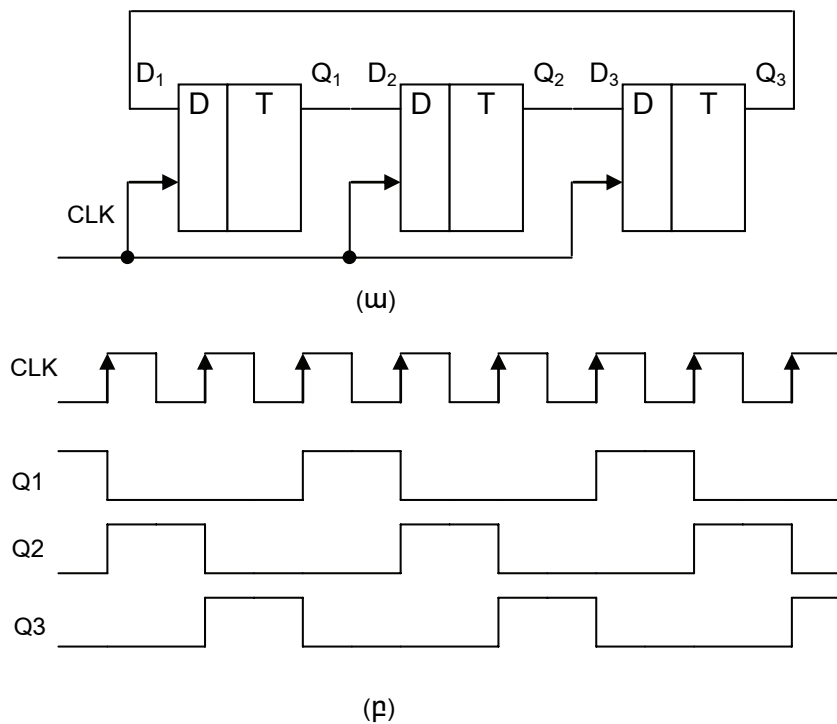
Օղակաձև հաշվիչ կարելի է ստանալ տեղաշարժող ռեգիստրից, եթե վերջին տրիգերի  $Q_n$  ելքը (նկ. 5.35) միացվի առաջին տրիգերի մուտքին՝  $D_1$ -ին: Եթե ռեգիստրի որևէ կարգում գրվի 1, իսկ մնացած կարգերում՝ 0, ապա այդ 1-ը յուրաքանչյուր տակտային իմպուլսից հետո կտեղափոխվի հաջորդ տրիգեր: Յուրաքանչյուր տրիգերի ելքում 1 ազդանշանը հայտնվում միևնույն  $f_1=f/n$  հաճախականությամբ, որտեղ  $f$ -ը տակտային իմպուլսների հաճախականությունն է: Համապատասխան սխեման և ժամանակային դիագրամները  $n=3$  դեպքում ցույց է տրված նկ. 5.41–ում:



Օղակաձև հաշվիչները մեծ կիրառություն ունեն կառավարող սարքավորումներում ազդանշանի ակտիվ մակարդակը հաջորդաբար հսկվող օբյեկտներին փոխանցելու համար: Այդ պատճառով դրանք կոչվում են նաև օղակաձև բաշխիչներ:

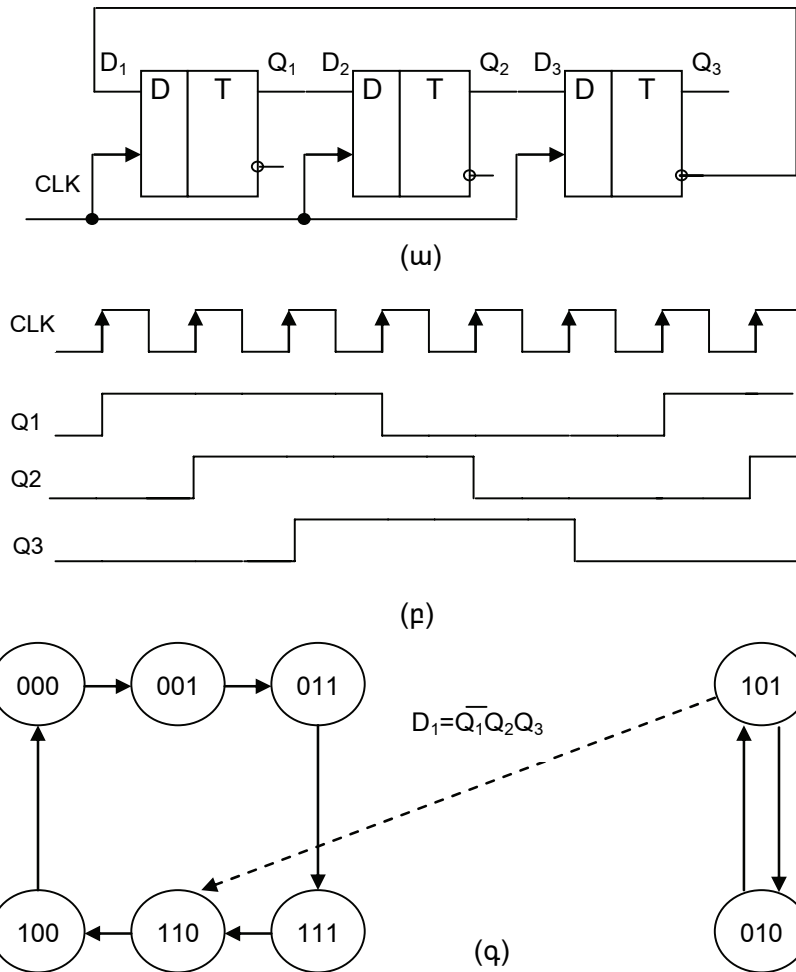
Մինչև աշխատանքի սկսելը անհրաժեշտ է հաշվիչի առաջին կարգի տրիգերում գրել 1, իսկ մնացած տրիգերներում՝ 0, քանի որ տրիգերների վիճակները սնունը միացնելուց հետո կարող են լինել կամայական:

Օղակաձև հաշվիչները չեն պարունակում արտաքին տրամաբանական տարրեր, որոնք սովորական հաշվիչներում մտցնում են հապաղումներ և սահմանափակում արագագործությունը: Այդ պատճառով էլ օղակաձև հաշվիչներն ունեն ամենամեծ արագագործությունը:



Նկ. 5.41. Եռակարգ օղակաձև հաշվիչ. (ա) սխեման, (բ) ժամանակային դիագրամները

Բարձր արագագործության համար վճարվող գինը օղակաձև հաշվիչներում անհրաժեշտ տրիգերների մեծ թիվն է՝ տրիգերների անհրաժեշտ թիվը հավասար է վերահաշվման գործակցին, այն դեպքում, երբ երկուական հաշվիչներում վերահաշվման գործակցը  $2^n$  է,  $n$ -ը տրիգերների թիվն է: Օրինակ,  $M=8$  վերահաշվման գործակցով երկուական հաշվիչում անհրաժեշտ է ունենալ 3 տրիգեր, իսկ օղակաձև հաշվիչում՝ 8 տրիգեր: Տրիգերների օգտագործման արդյունավետությունը կարելի մեծացնել երկու անգամ, այսինքն՝ ստանալ  $M=2n$ , եթե տրիգերների միջև կապերից մեկը դարձվի խաչվող, այսինքն՝ եթե տրիգերներից մեկի մուտքը միացվի նախորդ տրիգերի բացասված ելքին: Այսպիսի հաշվիչը կոչվում է խաչվող կապերով կամ Ջոնսոնի հաշվիչ: Նկ. 5.42ա-ում ցույց է տրված  $M=2 \times 3=6$  վերահաշվման գործակցով հաշվիչի սխեման, իսկ Նկ. 5.42բ-ում՝ ժամանակային դիագրամները:



Նկ. 5.42.  $M=6$  վերահաշվման գործակցով խաչվող կապերով հաշվիչ. (ա) սխեման, (բ) ժամանակային դիագրամները, (գ) անցումների գրաֆը

Օղակաձև հաշվիչներն ունեն մեծ թվով չօգտագործված վիճակներ: Եթե միացման պահին կամ աղավաղումների պատճառով հաշվիչը հայտնվի որևէ չօգտագործվող վիճակում, հնարավոր է, որ աշխատանքի ինքնուրույն շտկում տեղի չունենա: Աշխատանքի շտկման համար հաշվիչին ավելացվում է տրամաբանական շղթա, որը հետևում է հաշվիչի վիճակին:

Եթե հաշվիչը հայտնվի որևէ չօգտագործվող վիճակում, տրամաբանական շղթան տրիգերների մուտքերին կիրառում է համապատասխան ազդանշաններ, որոնք հաշվիչին ստիպում են անցնել նախատեսված վիճակներից որևէ մեկին: Սովորաբար դա արվում է առաջին տրիգերի մուտքի ֆունկցիայի համապատասխան ընտրության միջոցով: Նկ. 5.42գ-ում բերված է  $M=6$  օղակաձև հաշվիչի անցումների գրաֆը: Ավելացուկային վիճակները 010 և 101-ն են: Եթե հաշվիչը հայտնվի 101 չնախատեսված վիճակում ( $Q_3=1$ ,  $Q_2=0$ ,  $Q_1=1$ ), ապա հաջորդ տակտում այն կանցնի 010 վիճակ ( $Q_3^{t+1}=D_3^t=Q_2^t=0$ ,  $Q_2^{t+1}=D_2^t=Q_1^t=1$ ,  $Q_1^{t+1}=D_1^t=\overline{Q_3^t}=0$ ), որը նույնպես ավելցուկային է:

Հաջորդ տակտուն հաշվիչը 010-ից կանցնի 101, և այսպես շարունակ: Եթե առաջին տրիգերի մուտքի ֆունկցիան լրացվի  $Q_3 \overline{Q_2} Q_1$  բաղադրիչով՝

$$D_1' = Q_3 \overline{Q_2} Q_1 + \overline{Q_3}, \quad (5.40)$$

ապա հաշվիչը 101 վիճակից կանցնի 110 վիճակ և հետագայում կաշխատի համաձայն անցումների հիմնական գրաֆի:

### 5.13. Գծային հետադարձ կապերով տեղաշարժող ռեգիստր (ԳՀԿՏՌ)

Տվյալների փոխանցման համակարգերում օգտագործվող սխալներ հայտնաբերող և ուղղող կոդերի գեներացման և վերծանման, ինչպես նաև թվային համակարգերի թեստավորման, կեղծ պատահական հաջորդականությունների (ԿՊՀ) գեներացման և էլի շատ կիրառությունների համար օգտվում են ժամանակակից աբստրակտ հանրաշվի մեթոդներից, որոնց գործնական իրականացման համար օգտագործվում են գծային հետադարձ կապերով ռեգիստրներ:

Տեղաշարժող ռեգիստրներում  $n$  երկարությամբ երկուական հաջորդականությունները՝

$$B = \{B_0, B_1, B_2, \dots, B_{n-1}\} \quad (5.41)$$

հարմար է ներկայացնել ձևական բազմանդամների օգնությամբ՝

$$B(x) = B_0 x^0 \oplus B_1 x^1 \oplus B_2 x^2 \oplus \dots \oplus B_{n-1} x^{n-1}, \quad (5.42)$$

որտեղ  $x^k$  անդամը ցույց է տալիս  $B_k$  բիթի դիրքը հաջորդականության մեջ:

Օրինակ,  $B = 10010111$  հաջորդականությանը համապատասխանում է հետևյալ  $n=7$  աստիճանի բազմանդամը՝

$$B(x) = 1 \cdot x^0 \oplus 0 \cdot x^1 \oplus 0 \cdot x^2 \oplus 1 \cdot x^3 \oplus 0 \cdot x^4 \oplus 1 \cdot x^5 \oplus 1 \cdot x^6 \oplus 1 \cdot x^7 = 1 \oplus x^3 \oplus x^5 \oplus x^6 \oplus x^7$$

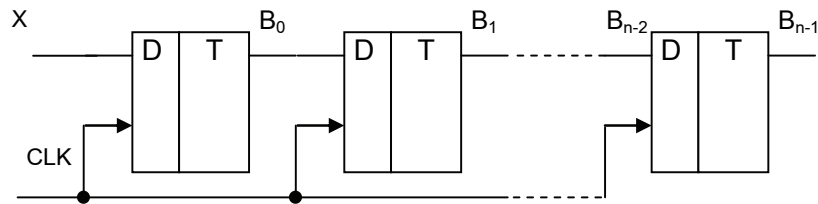
Երկուական հաջորդականության  $k$ -բիթով գծային տեղաշարժին համապատասխանում է բազմանդամի բազմապատկմանը  $x^k$ -ով՝

$$D(x) = x^k B(x) = B_0 x^k \oplus B_1 x^{k+1} \oplus B_2 x^{k+2} \oplus \dots \oplus B_{n-1} x^{k+n-1} :$$

Օրինակ, ենթադրենք  $B = 10010111$  թիվը գրանցված է 8-աբիթ տեղաշարժող ռեգիստրում (նկ. 5.43,  $n=8$ ):  $k=2$  դեպքում կստացվի՝

$$D(x) = x^2 \oplus x^5 \oplus x^7,$$

որին համապատասխանում է  $D = XX100101$ , որտեղ ցածր երկու կարգերը որոշված չեն կամ ռեգիստրի հաջորդական մուտքին կիրառված բիթերն են երկու տեղաշարժերի դեպքում, իսկ բարձր երկու կարգերը ռեգիստրից դուրս են եկել:

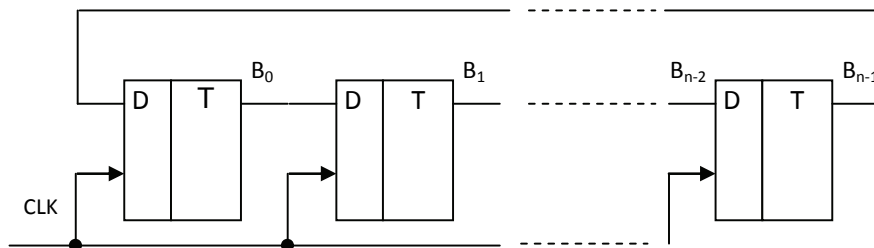


Նկ. 5.43. Երկուական թվի տեղաշարժ հաջորդական ռեգիստրում

Շրջանաձև տեղաշարժող ռեգիստրի դեպքում (նկ. 5.44,  $n=8$ ) արդյունաբար բազմանդամի ցուցիչները պետք է հաշվել ըստ մոդուլ  $n$ -ի: Քննարկվող օրինակի դեպքում՝  $B=10010111$ ,  $n=8$ ,  $k=2$ .

$$D(x) = x^k B(x) = x^2 \oplus x^5 \oplus x^7 \oplus x^{8 \bmod 8} \oplus x^{9 \bmod 8} = x^2 \oplus x^5 \oplus x^7 \oplus x^0 \oplus x^1 = 1 \oplus x \oplus x^2 \oplus x^5 \oplus x^7$$

Ռեգիստրի արդյունաբար պարունակությունը կլինի՝  $D=11100101$ :



Նկ. 5.44. Երկուական թվի տեղաշարժ շրջանաձև հաջորդական ռեգիստրում

### 5.13.1. Գծային հետադարձ կապերով տեղաշարժող ռեգիստրների (ԳՀԿՏՌ) կառուցվածքները

Յուրաքանչյուր ԳՀԿՏՌ նկարագրվում է համապատասխան բնութագրիչ բազմանդամով.

$$C(x) = C_0 x^0 \oplus C_1 x^1 \oplus C_2 x^2 \oplus \dots \oplus C_{n-1} x^{n-1} \oplus C_n x^n, \quad (5.43)$$

որտեղ  $x^k$ –ն ցույց է տալիս, որ հետադարձ կապի ազդանշանը վերցվում է տեղաշարժող ռեգիստրի  $k$ -րդ տրիգերի ելքից,  $C_0, C_1, \dots, C_n$ -ը բազմանդամի գործակիցներն են, որոնք կարող են լինել 0 կամ 1: Այդ գործակիցները որոշում են տեղաշարժող ռեգիստրի համապատասխան հետադարձ կապերի առկայությունը կամ բացակայությունը:

Ընդհանուր դեպքում  $C_0=1$ ,  $C_n=1$ ,

$$C(x) = 1 \oplus C_1 x^1 \oplus C_2 x^2 \oplus \dots \oplus C_{n-1} x^{n-1} \oplus x^n : \quad (5.44)$$

Նույն բնութագրիչ բազմանդամի նկարագրությամբ համապատասխանում են ԳՀԿՏՌ-ի երկու կառուցվածքներ՝ դուրս բերված և ներդրված մոդուլ երկուսով գումարիչներով:

Դուրս բերված գումարիչներով ռեգիստրի կառուցվածքը ցույց է տրված նկ. 5.45-ում: Երբ ռեգիստրին մուտքային հաջորդականություն չի կիրառվում ( $B(x)=0$ ), այսինքն՝ մուտքային հանգույց չունի, այն գործում է իբրև երկուական կեղծ պատահա-

կան հաջորդականությունների գեներատոր՝ ԿՊՅԳ: ԿՊՅԳ կառուցվածքում ռեգիստրի հետադարձ կապի հավասարումն ստացվում է

$$C(x)=0 \quad (5.45)$$

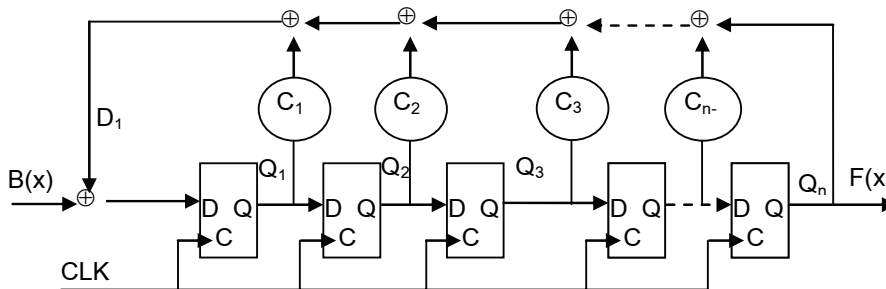
պայմանից: Այդ դեպքում (5.44) հավասարումից կստացվի՝

$$C_1x^1 \oplus C_2x^2 \oplus \dots \oplus C_{n-1}x^{n-1} \oplus x^n = 1: \quad (5.46)$$

Փոխարինելով  $x$ -ի աստիճանները համապատասխան տրիգերների ելքերով, կստանանք՝

$$D_1 = C_1Q_1 \oplus C_2Q_2 \oplus \dots \oplus C_{n-1}Q_{n-1} \oplus C_nQ_n: \quad (5.47)$$

Երբ ռեգիստրի մուտքին կիրառվում է  $B(x)$  երկուական հաջորդականություն, այն գործում է իբրև ռեկուրսիվ գտիչ՝ ելքային  $F(x)$  հաջորդականությունը կարելի է ներկայացնել իբրև մուտքային հաջորդականության և ԳՅԿՏՌ գործակիցների փաթեթ: Առաջին տրիգերի մուտքի հավասարումն ունի հետևյալ տեսքը.



Նկ. 5.45. Դուրս բերված գումարիչներով ռեգիստրի կառուցվածքը.  $B(x)=0$  դեպքը համապատասխանում է գեներատորի, իսկ  $B(x) \neq 0$  դեպքը՝ ռեկուրսիվ գտիչի կառուցվածքին

$$D_1 = C_1Q_1 \oplus C_2Q_2 \oplus \dots \oplus C_{n-1}Q_{n-1} \oplus C_nQ_n \oplus B(x): \quad (5.48)$$

Ձտիչը կոչվում է ռեկուրսիվ, եթե ժամանակի որևէ պահի ելքային ազդանշանի արժեքը կախված է ինչպես մուտքային հաջորդականության, այնպես էլ ելքային հաջորդականության արժեքներից ժամանակի նախորդ պահերին: Իսկ եթե ժամանակի որևէ պահի ելքային ազդանշանի արժեքը կախված է միայն մուտքային հաջորդականության տվյալ և նախորդ պահերի արժեքներից, գտիչը կոչվում է ոչ ռեկուրսիվ:

Ներդրված գումարիչներով կառուցվածքում (նկ. 5.46) տրիգերների մուտքերի հավասարումներն ունեն հետևյալ տեսքը՝

$$D_i = Q_n C_{i-1} \oplus Q_{i-1}, \quad i=2, \dots, n \quad (5.49)$$

և

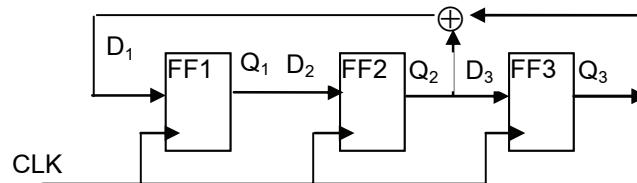
$$D_1 = Q_n \quad (5.50)$$

գեներատորի կառուցվածքում, իսկ ռեկուրսիվ գտիչի կառուցվածքում՝

$$D_1 = Q_n \oplus B(x): \quad (5.51)$$

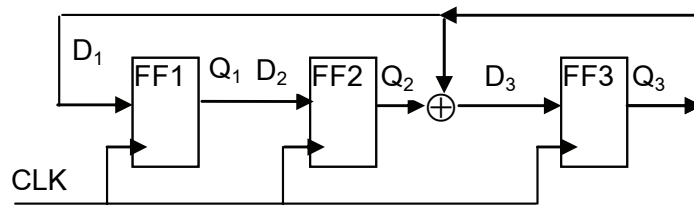
Օրինակ, հետևյալ բազմանդամին՝

( $c_0=1, c_1=0, c_2=1, c_3=1$ ), համապատասխանում են նկ. 5.47, 5.48–ում ցույց տրված հետադարձ կապերով տեղաշարժող ռեգիստրները: Հետադարձ կապի հավասարումը նկ. 5.47-ի ռեգիստրի համար ստացվում է հետևյալ կերպ՝  $C(x)=1 \oplus x^2 \oplus x^3 = 0 \Rightarrow x^0 = x^2 \oplus x^3$ :  $x^0$ -ն առաջին տրիգերի մուտքն է:



Նկ. 5.48-ի ռեգիստրի տրիգերների մուտքի հավասարումները ստացվում են կիրառելով (5.49) և (5.50) բանաձևերը (5.52) հավասարման նկատմամբ  $D_1=Q_3$ ,  $D_3=Q_2 \oplus Q_3$ :

270



Նկ. 5.48.  $C(x)=1\oplus x^2\oplus x^3$  բազմանդամով նկարագրվող ներդրված մոդուլ երկուսով գունարիչներով զՀԿՏՈ

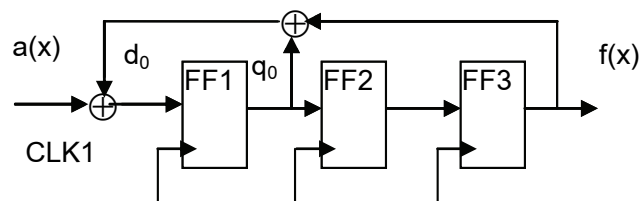
Աղյուսակ 5.7

Տակտի համարը	Նկ.5.47-ի ռեգիստր		Նկ.5.48-ի ռեգիստր	
	նախորդ վիճակը $D_1D_2D_3$	հաջորդ վիճակը $D_1D_2D_3$	նախորդ վիճակը $D_1D_2D_3$	հաջորդ վիճակը $D_1D_2D_3$
1	111	011	111	110
2	011	001	110	011
3	001	100	011	100
4	100	010	100	010
5	010	101	010	001
6	101	110	001	101
7	110	111	101	111

Տվյալների հաջորդականության անցկացումը զՀԿՏՈ-ով համարժեք է դրանց զտմանը, այսինքն՝ զՀԿՏՈ-ն զտիչ է, իսկ այն նկարագրող բազմանդամը որոշում է զտիչի ֆունկցիան:

**Օրինակ 5.3.** Նկ. 5.49-ում ցույց է տրված ռեկուրսիվ զտիչի կառուցվածքը, որը մուտքային  $a(x)$  հաջորդականությունը ձևափոխում է ելքային  $f(x)$  հաջորդականության: Ձտիչը նկարագրվում է հետևյալ բազմանդամով.

$$P_3(x)=x^3\oplus x\oplus 1: \quad (5.53)$$



զՀԿՏՈ

Նկ. 5.49.  $P_3(x)=x^3\oplus x\oplus 1$  բազմանդամով նկարագրվող ռեկուրսիվ զտիչի կառուցվածքը

Նշանակենք D-տրիգերների մուտքերը d-ով, իսկ ելքերը q-ով՝

$$q_0=d_0x, \quad (5.54)$$

x-ը ցույց է տալիս մեկ տակտով հապաղում՝  $q^+=d$ :

Ելքային  $f(x)$  ազդանշանը  $d_0$ -ի նկատմամբ ուշանում է 3 տակտով՝  $d_0$ -ն անցնում է երեք տրիգերով.

$$f(x) = d_0 x^3 : \quad (5.55)$$

Առաջին տրիգերի մուտքային ազդանշանը որոշվում է հետադարձ կապի հավասարումով.

$$d_0 = a(x) \oplus q_0 \oplus f(x) = a(x) \oplus d_0 x \oplus f(x): \quad (5.56)$$

Նկատի ունենալով, որ մոդուլ երկուսով հանումը համարժեք է գումարմանը, (5.56)-ից կարելի է որոշել

$$d_0 \oplus d_0 x = a(x) \oplus f(x)$$

և

$$d_0 = (a(x) \oplus f(x)) / (1 \oplus x) \quad (5.57)$$

տեղադրելով (5.57)-ը (5.55)-ում կստացվի

$$f(x) = (a(x) \cdot x^3 \oplus f(x) \cdot x^3) / (1 \oplus x):$$

Կատարելով հետևյալ միջանկյալ ձևափոխությունները՝

$$f(x) \oplus f(x) \cdot x = a(x) \cdot x^3 \oplus f(x) \cdot x^3,$$

$$f(x) \oplus f(x) \cdot x \oplus f(x) \cdot x^3 = a(x) \cdot x^3,$$

ի վերջո կստացվի՝

$$f(x) = a(x) \cdot x^3 / (1 \oplus x \oplus x^3) = a(x) \cdot x^3 / P_3(x) \quad (5.58)$$

Այսպիսով, դիտարկված գտիչի ելքային հաջորդականությունն ստացվում է մուտքայինը գտիչը նկարագրող բազմանդամին բաժանումից,  $x^3$  արտադրիչը ցույց է տալիս, որ ելքային հաջորդականությունը մուտքայինի նկատմամբ ուշանում է երեք տակտով: Հետադարձ կապերով գտիչը կոչվում է ռեկուրսիվ կամ անվերջ իմպուլսային բնութագրով (ԱԻԲ):

Դիտարկենք օրինակ.  $a = \{1101001\}$  կամ  $a(x) = x^6 \oplus x^5 \oplus x^3 \oplus 1$  և  $P_3(x) = x^3 \oplus x \oplus 1$ : Ձտիչի ելքային հաջորդականության ձևավորման ընթացքը ներկայացված է աղյուսակ 5.8-ում:

Ընդհանուր դեպքում նկ. 5.45-ում բերված ԳՀԿՏՌ-ի ելքային հաջորդականությունը նկարագրող բազմանդամը որոշվում է հետևյալ հավասարումից.

$$F(x) = B(x)x^n / C(x), \quad (5.59)$$

որտեղ  $C(x)$ -ը տրվում է (5.44)-ով:

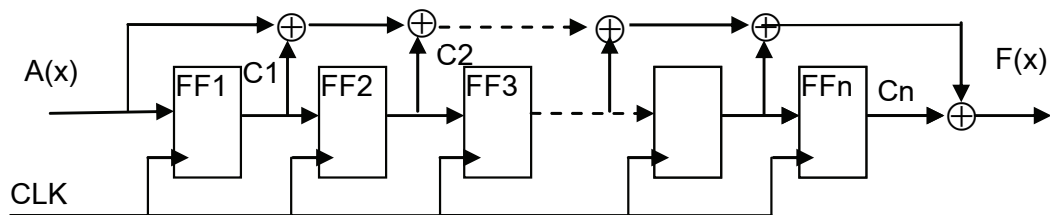


a(x)	d <sub>0</sub>	q <sub>0</sub>	q <sub>1</sub>	q <sub>2</sub> =f(x)	q <sub>0</sub>	q <sub>1</sub>	q <sub>2</sub> =f(x)
1	1	0	0	0	1	0	0
1	0	1	0	0	0	1	0
0	0	0	1	0	0	0	1
1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0
	1	1	0	0	1	1	0
	1	1	1	0	1	1	1
	0	1	1	1	0	1	1
	1	0	1	1	1	0	1
	0	1	0	1	0	1	0
	0	0	1	0	0	0	1
	1	0	0	1			

Ձտիչ կարելի է կառուցել նաև գծային ուղիղ կապերով տեղաշարժող ռեգիստրների (ԳՈԼՎՏՈ) վրա: Այդպիսի գտիչները կոչվում են ոչ ռեկուրսիվ կամ վերջավոր իմպուլսային բնութագրով (ՎԻԲ): Ոչ ռեկուրսիվ գտիչի կառուցվածքը ցույց է տրված նկ. 5.50-ում: Ոչ ռեկուրսիվ գտիչում բացակայում են հետադարձ կապերը, առկա են միայն ուղիղ կապեր՝ ըստ (5.44) բնութագրիչ բազմանդամի գործակիցների.

$$F(x) = A(x) \oplus C_1 Q_1 \oplus C_2 Q_2 \oplus \dots \oplus C_{n-1} Q_{n-1} \oplus C_n Q_n: \quad (5.60)$$

Ոչ ռեկուրսիվ գտիչի ելքային ազդանշանը որոշվում է տրիգերների ելքային ազդանշանների գծային համակցությամբ:



Նկ. 5.50. Ոչ ռեկուրսիվ գտիչի կառուցվածքը (ԳՈԼՎՏՈ)

Հաշվի առնելով, որ

$$Q^k = A(x)x^k \quad (5.61)$$

(5.60)-ից կստացվի՝

$$F(x) = A(x)(1 \oplus C_1 x \oplus C_2 x^2 \oplus \dots \oplus C_{n-1} x^{n-1} \oplus C_n x^n) = A(x)C(x): \quad (5.62)$$

Ոչ ռեկուրսիվ գտիչի ելքային հաջորդականությունը նկարագրող բազմանդամը հավասար է մուլտիպլիկացիոն հաջորդականությունը նկարագրող բազմանդամի և ԳՈԼՎՏՈ-ի բնութագրիչ բազմանդամի արտադրյալին:

### 5.13.2. Առավելագույն երկարությամբ չկրկնվող հաջորդականություններ

Գործնականում կարևոր նշանակություն ունեն այն բազմանդամները, որոնք հնարավորություն են տալիս գեներացնել առավելագույն երկարությամբ չկրկնվող հաջորդականություններ: Տեսությունից հայտնի է, որ առավելագույն չկրկնվող հաջորդականության երկարությունը՝  $N=2^n-1$  է,  $n$ -ը բազմանդամի կարգն է: Այդպիսի հաջորդականություններ գեներացնող բազմանդամները պետք է բավարարեն հետևյալ պայմաններին՝

- 1) պարզ բազմանդամ է՝ չի վերլուծվում արտադրիչների,
- 2) հանդիսանում է  $x^{2^n-1} \oplus 1$  երկանդամի բաժանարար,
- 3) չի հանդիսանում ոչ մի  $x^k \oplus 1$ ,  $k < 2^n$  տիպի երկանդամի բաժանարար:

**Օրինակ 5.4.** Ցույց տալ, որ  $P(x)=x^3 \oplus x^2 \oplus 1$  բազմանդամը բավարարում է վերը նշված երեք պայմաններին:

- 1) պարզ բազմանդամ է՝ չի բաժանվում  $x$ ,  $x \oplus 1$ ,  $x^2$ ,  $x^2 \oplus 1$ ,  $x^2 \oplus x$ ,  $x^2 \oplus x \oplus 1$  բազմանդամներից ոչ մեկին,
- 2)  $x^7 \oplus 1$  երկանդամի բաժանարար է,
- 3)  $x^6 \oplus 1$ ,  $x^5 \oplus 1$ ,  $x^4 \oplus 1$  երկանդամների բաժանարար չէ:

Առավելագույն երկարությամբ հաջորդականություններ գեներացնող պարզ բազմանդամների որոնումը պարզ խնդիր չէ: Աղյուսակ 5.9–ում բերված են այդպիսի բազմանդամների օրինակներ՝ զրոյից տարբեր գործակիցները:

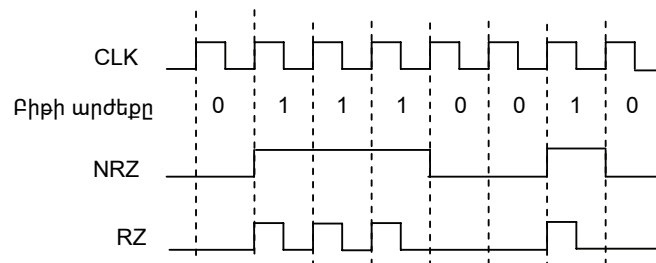
### 5.14. Գծային կապերով տեղաշարժող ռեգիստրների կիրառություններ

#### 5.14.1. Տվյալների հաջորդական փոխանցման ազդանշանային կոդեր

Հեռահաղորդակցության թվային համակարգերում թվային տվյալները փոխանցվում են երկուական հաջորդականությունների տեսքով: Այդպիսի կոդի տարածված օրինակ է առանց զրոյի վերադարձի կոդը՝ ԱԶՎ (NRZ, Non-Return-to-Zero): ԱԶՎ կոդի յուրաքանչյուր բիթ հաղորդվում է տակտային ազդանշանի մեկ պարբերության ընթացքում (նկ.5.51): Սա տվյալների կոդավորման պարզագույն, բայց մաիժամանակ ամենահուսալի եղանակն է՝ կարճ տարածությունների վրա տվյալների փոխանցման համար: Դա պահանջում է, որ տվյալներին զուգահեռ փոխանցվի նաև տակտային ազդանշանը, որով տվյալների հոսքում տրվում է բիթերի դիրքը: Հակառակ դեպքում ընդունիչում հնարավոր չի լինի որոշել անընդմեջ գնացող 1-երի կամ 0-ների քանակը, որոնք տրվում են 1-ի կամ 0-ի չփոփոխվող մակարդակով: Օրինակ, տակտային ազդանշանի բացակայության դեպքում, որը տալիս է բիթերի միջև սահմանները, նկ. 5.51–ում ցույց տրված 01110010 ԱԶՎ բիթերի հաջորդականությունը կարող է ընկալվել որպես 01010:

Աղյուսակ 5.9

N	P(x)	N	P(x)
1	0, 1	21	0, 2, 21
2	0, 1, 2	22	0, 1, 22
3	0, 1, 3	23	0, 5, 23
4	0, 1, 4	24	0, 1, 3, 4, 24
5	0, 2, 5	25	0, 3, 25
6	0, 1, 6	26	0, 1, 7, 26
7	0, 1, 7	27	0, 1, 7, 27
8	0, 1, 5, 6, 8	28	0, 3, 28
9	0, 4, 9	29	0, 2, 29
10	0, 3, 10	30	0, 1, 15, 16, 30
11	0, 2, 11	31	0, 3, 31
12	0, 3, 4, 7, 12	32	0, 1, 27, 28, 32
13	0, 1, 3, 4, 13	40	0, 2, 19, 21, 40
14	0, 1, 11, 12, 14	50	0, 1, 26, 27, 50
15	0, 1, 15	60	0, 1, 60
16	0, 2, 3, 5, 16	70	0, 1, 15, 16, 70
17	0, 3, 17	80	0, 1, 37, 38, 80
18	0, 7, 18	90	0, 1, 18, 19, 90
19	0, 1, 5, 6, 19	100	0, 37, 100
20	0, 3, 20	256	0, 1, 3, 16, 256



Նկ. 5.51. Թվային տվյալների հաջորդական փոխանցման ազդանշանային կոդեր

Այն դեպքերում, երբ տակտային ազդանշանը չի փոխանցվում, ընդունիչում այն պետք է վերականգնել տվյալների բիթերի հոսքից՝ վերլուծելով բիթերի հոսքում 0→1 և 1→0 անցումները: Երկուական տվյալների հոսքից տակտային ազդանշանի վերականգնման համար օգտագործվում են անալոգաթվային սարքեր՝ փուլի ինքնալարքով հաճախության կարգավորիչներ (ՓԻՅԿ), որոնց անսխալ աշխատանքի համար անհրաժեշտ է, որ տվյալների հոսքում լինեն բավարար քանակությամբ 0→1 և 1→0 անցումներ:

Երբ տվյալները հաղորդվում են ԱՉՎ կոդով, ՓԻՅԿ-ն կաշխատի անսխալ միայն այն դեպքում, երբ տվյալների հոսքում բացակայում են անընդհատ գնացող 0-ներ կամ 1-եր:

Ձրոյի վերադարձով կոդը՝ ՉՎ (RZ, Return-to-Zero) տարբերվում է ԱՉՎ կոդից նրանով, որ '1' բիթի դեպքում 1-ի մակարդակը պահպանվում է միայն տակտի կես պար-

բերության ընթացքում: Եթե տվյալների հոսքը պարունակում է մեծ թվով իրար հաջորդող 1-եր, ապա հաղորդվող ազդանշանում կունենանք մեծ թվով անցումներ, որոնք հեշտացնում են տակտային ազդանշանի վերկանգնումը: Բայց եթե տվյալների հոսքը պարունակում է մեծ թվով իրար հաջորդող 0-ներ, հաղորդվող ազդանշանում անցումները կբացակայեն և տակտային ազդանշանի վերկանգնումը անհնարին կլինի:

Տվյալների բարձր արագությամբ փոխանցման միջավայրերում, օրինակ, օպտիկական մանրաթելային գծերում, հաղորդվող թվային ազդանշաններից պահանջվում է, որ դրանք ըստ հաստատուն բաղադրիչի հավասարակշռված լինեն: Դա նշանակում է, որ տվյալների հոսքում 1-երի և 0-ների թվերը պետք է հավասար լինեն: Եթե 1-երի թիվը գերազանցում է 0-ների թվին համեմատաբար երկար ժամանակի ընթացքում, ապա դա հանգեցնում է ընդունիչի շեմի տեղաշարժի, որի հետևանքով վատանում է 0-ի զանազանումը 1-ից: Այս խնդրի լուծման համար առաջարկված են տվյալների կոդավորման հավասարակշռված կոդեր, որոնցում յուրաքանչյուր կոդային համակցության ավելացվում են մի քանի ավելցուկային բիթեր՝ 0-ների և 1-երի թվի հավասարեցման համար: Ակնհայտ է, որ հավասարակշռված կոդավորման դեպքում նվազում է տվյալների հաղորդման արդյունավետությունը: Օրինակ, եթե տվյալները սովորական երկուական հաջորդականությամբ կոդավորելու համար յուրաքանչյուր սիմվոլի համար անհրաժեշտ է ունենալ 8-բիթ համակցություն, բայց կոդը հավասարակշռելու համար պետք է ավելացնել ևս 2 բիթ, ապա հավասարակշռված կոդով տվյալների հաղորդման արդյունավետությունը կնվազի  $\frac{(10-8) \cdot 100\%}{8} = 25\%$ -ով:

Թվային տվյալների հաջորդականությունները հավասարակշռելու և դրանցում անցումների թիվը մեծացնելու համար գործնականում հաճախ օգտագործվում են ազդանշանների խառնիչներ (սկրամբլերներ): Խառնիչների օգտագործումը չի վատացնում տվյալների հաղորդման արդյունավետությունը:

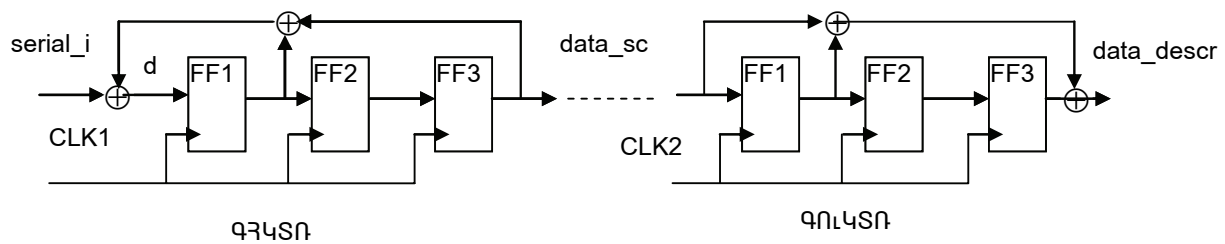
Սկրամբլերը կառուցվում է առավելագույն երկարությամբ հաջորդականությունների ԳՅԿՏՌ-ի տեսքով: Հաղորդվող տվյալների հոսքը անցկացվում է ԳՅԿՏՌ-ով: Ընդունիչում այդ տվյալները պետք է հետ ձևափոխվեն՝ սկզբնական տվյալների վերականգնման համար: Այդ վերկանգնման գործողությունը նույնպես կատարվում է ԳՅԿՏՌ-ի միջոցով, որը կոչվում է դեսկրամբլեր:

Դիտարկենք պարզագույն օրինակ, երբ սկրամբլերը նկարագրվում է հետևյալ բազմանդամով.

$$P_3(x) = x^3 \oplus x \oplus 1:$$

Դեսկրամբլերը նույնպես նկարագրվում է այդ բազմանդամով: Նկ. 5.52-ում ցույց է տրված հաջորդական տվյալների փոխանցման համակարգի կառուցվածքը, որտեղ փոխանցվող serial\_in տվյալները տրվում են ԳՅԿՏՌ1 ռեգիստրի մուտքին: Ռեգիստրը միացված է ռեկուրսվ գտիչի (անվերջ իմպուլսային բնութագրով, ԱԻԲ) կառուցվածքով: Նշանակենք D-տրիգերների մուտքերը d-ով, իսկ ելքերը՝ q-ով: Սկրամբլերի ելքային ազդանշանը d<sub>0</sub>-ի նկատմամբ ուշանում է 3 տակտով՝ d<sub>0</sub>-ն անցնում է երեք տրիգերով.

$$\text{data\_scr}(x) = d_0 x^3: \quad (5.63)$$



Նկ. 5.52. Հաջորդական տվյալների փոխանցում սկրամբլերի և դեսկրամբլերի կիրառությամբ  
Առաջին տրիգերի մուտքային ազդանշանը որոշվում է հետևյալ կերպ.

$$d_0(x) = \text{serial\_in}(x) \oplus d_0x \oplus \text{data\_scr}(x): \quad (5.64)$$

Նկատի ունենալով, որ մոդուլ երկուսով հանումը համարժեք է գումարմանը, կարելի է որոշել.

$$d_0 \oplus d_0x = \text{serial\_in}(x) \oplus \text{data\_scr}(x)$$

և

$$d_0(x) = (\text{serial\_in}(x) \oplus \text{data\_scr}(x)) / (1 \oplus x):$$

Տեղադրելով  $d_0$  –ի այս արտահայտությունը (5.63)–ում կստացվի՝

$$\text{data\_scr}(x) = (\text{serial\_in}(x) \cdot x^3 \oplus \text{data\_scr}(x) \cdot x^3) / (1 \oplus x):$$

Որոշ միջանկյալ ձևափոխություններից հետո կստացվի՝

$$\text{data\_scr}(x) = \text{serial\_in}(x) \cdot x^3 / (1 \oplus x \oplus x^3) = \text{serial\_in}(x) \cdot x^3 / P_3(x): \quad (5.65)$$

Այսպիսով, սկրամբլերի ելքային հաջորդականությունն ստացվում է մուտքային և սկրամբլերը նկարագրող բազմանդամի հարաբերությունից:

Դեսկրամբլերը (Նկ. 5.52) կառուցված է ոչ ռեկուրսիվ գտիչի (վերջավոր իմպուլսային բնութագրով, ՎԻԲ) տեսքով՝

$$\text{data\_descr}(x) = \text{data\_scr}(x) \oplus \text{data\_scr}(x) \cdot x \oplus \text{data\_scr}(x) \cdot x^3:$$

Այստեղից կարելի է որոշել՝

$$\text{data\_descr}(x) = \text{data\_scr}(x) (1 \oplus x \oplus x^3) = \text{data\_scr}(x) \cdot P_3(x) \quad (5.66)$$

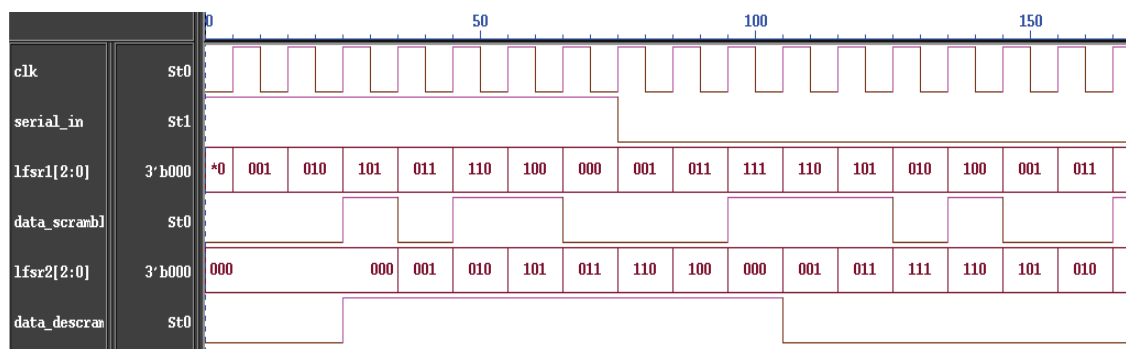
Դեսկրամբլերի ելքային հաջորդականությունն ստացվում է մուտքային և գտիչը նկարագրող բազմանդամի արտադրյալից: Տեղադրելով  $\text{data\_scr}$ –ի արժեքը (5.65)–ից (5.66)–ի մեջ, կստացվի՝

$$\text{data\_descr}(x) = \text{serial\_in}(x) \cdot x^3 \quad (5.67)$$

Այսպիսով, դեսկրամբլերի ելքային  $\text{data\_descr}$  հաջորդականությունը համընկնում է սկրամբլերի մուտքային  $\text{serial\_in}$  հաջորդականության հետ, իսկ  $x^3$  բազմապատկիչը

ցույց է տալիս, որ data\_descr հաջորդականությունը 3 տակտով ուշանում է serial\_in հաջորդականության նկատմամբ:

Նկ. 5.53-ում ցույց են տրված ազդանշանների ժամանակային դիագրամները, որոնք ստացվել են նկ. 5.52-ում ցույց տրված համակարգի նմանակումից: Այդ նմանակման ժամանակ համակարգի մուտքին տրվում է 1111\_1111\_0000\_0000 16-բիթ հաջորդականություն: Սկրամբլերի ելքում առաջին բիթը ձևավորվում է մուտքային ազդանշանի կիրառման պահից երեք տակտ հետո: Ինչպես կարելի է տեսնել այդ ժամանակային դիագրամներից, կապի գծով փոխանցվող data\_scr ազդանշանում ավելացվել են ազդանշանի փոխանջատումներ՝ մուտքային serial\_in ազդանշանի նկատմամբ: Աղյուսակ 5.10-ում բերված են ռեգիստրների վիճակների փոփոխությունները:



Նկ. 5.53. Հաջորդական տվյալների փոխանցման ազդանշանների ժամանակային դիագրամները (ստացված Verilog նմանակումից)

Աղյուսակ 5.10

տակտի թիվը, t	Serial_in	ԳՅԿՏՈՒ <sup>t+1</sup>			data_scr	ԳՈՒԿՏՈՒ <sup>t+1</sup>			data_descr
		[2]	[1]	[0]		[2]	[1]	[0]	
		0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0
3	1	1	0	1	1	0	0	0	1
4	1	0	1	1	0	0	0	1	1
5	1	1	1	0	1	0	1	0	1
6	1	1	0	0	1	1	0	1	1
7	1	0	0	0	0	0	1	1	1
8	1	0	0	1	0	1	1	0	1
9	0	0	1	1	0	1	0	0	1
10	0	1	1	1	1	0	0	0	1
11	0	1	1	0	1	0	0	1	0
12	0	1	0	1	1	0	1	1	0
13	0	0	1	0	0	1	1	1	0
14	0	1	0	0	1	1	1	0	0
15	0	0	0	1	0	1	0	1	0
16	0	0	1	1	0	0	1	0	0

Երկուական հաջորդականությունների ձևափոխումը սկրամբլերով մեծացնում է փոխանջատումների թիվը, ինչը հնարավորություն է տալիս տվյալների հաջորդականությունից վերականգնել տակատային ազդանշանը: Դա կարևոր խնդիր է այն համակարգերում, երբ սինքրոն տվյալները փոխանակվում են առանց կապի գծով ուղեկցող տակտային իմպուլսների փոխացման, օրինակ, ինտերնետ (WAN) և տեղական (LAN) ցանցերով: Այդ համակարգերում սկրամբլերը և դեսկրամբլերը նկարագրվում են 58-ակարգ պարզ բազմանդամներով:

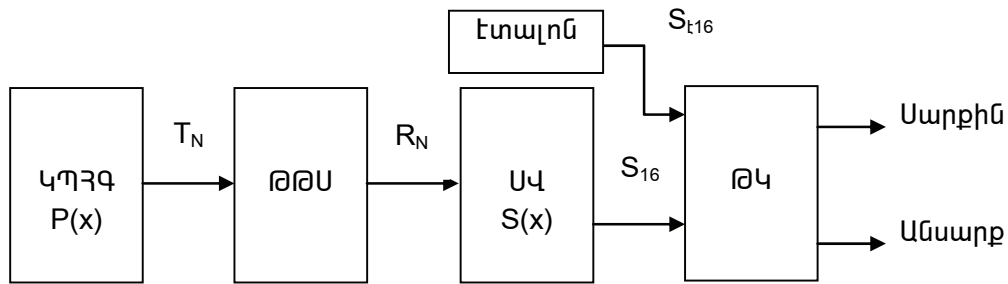
Սկրամբլերի միջոցով տվյալների հաջորդականությունների ձևափոխումն օգտագործվում է նաև տվյալների գաղտնագրման համար՝ առանց սկրամբլերի նկարագրության ֆունկցիան իմանալու դժվար է սկրամբլերով անցկացված հաջորդականությունից վերականգնել սկզբնականը:

#### 5.14.2. Թվային համակարգերի թեստավորման համակարգեր

Թվային համակարգերի թեստավորման համար լայնորեն օգտագործվում է սիգնատուր վերլուծության եղանակը, որի էությունն այն է, որ թվային հանգույցի թեստավորման համար նրա մուտքերին տրվում են թեստային հաջորդականություններ և վերլուծվում են ելքային հաջորդականությունները: Եթե ելքային հաջորդականությունները տարբերվում են նախանշվածից, ապա ստուգվող հանգույցը համարվում է անսարք: Այս միանգամայն պարզ տրմաբանական մոտեցման իրականացման դժվարությունն այն է, որ սովորաբար քիչ թե շատ մեծ թվով ներքին վիճակներ ունեցող հանգույցների ելքային հաջորդականություններն ունեն շատ մեծ երկարություններ, ուստի դրանց պահպանումը և վերլուծությունը կապահանջի մեծ ծավալի հիշողություն և երկար ժամանակ: Իրադրությունը շտկելու արդյունավետ մոտեցում է ելքային հաջորդականությունների սեղմումը որոշակի ալգորիթմներով: Ներկայումս կիրառվող ամենաարդյունավետ եղանակը երկուական հաջորդականությունների սեղմումն է սիգնատուր վերլուծիչով: Սիգնատուր վերլուծիչը առավելագույն երկարությամբ հաջորդականությունների ԳՅԿՏՌ է, որը սովորաբար 16-ակարգ է և նկարագրվում է առավելագույն երկարությամբ հաջորդականություններ գեներացնող հետևյալ պարզ բազմանդամով՝

$$S(x) = x^{15} \oplus x^{11} \oplus x^8 \oplus x^6 \oplus 1: \quad (5.68)$$

Նկ. 5.54–ում բերված է սիգնատուր վերլուծությամբ թեստավորման համակարգի պարզեցված կառուցվածքային սխեման, որտեղ մուտքային  $T_N$  թեստ հաջորդականությունը գեներացվում է  $P(x)$  բազմանդամով նկարագրվող ԳՅԿՏՌ–ի վրա կառուցված ԿՊՅԳ–ի միջոցով: Այդ հաջորդականությունը տրվում է թեստավորվող թվային սարքի (ԹԹՍ) մուտքին, որի ելքային  $R_N$  հաջորդականությունը տրվում է  $S(x)$  բազմանդամով նկարագրվող ՍՎ սիգնատուր վերլուծիչի մուտքին: Թեստավորման ավարտին սիգնատուր վերլուծիչի ռեգիստրում գրանցված 16-աբիթ երկուական թիվը՝  $S_{16}$ , թեստավորվող սարքի սիգնատուրն է:



Նկ. 5.54. Միգնատուր վերլուծությանը թեստավորման համակարգի պարզեցված կառուցվածքային սխեման

Այնուհետև, ստացված  $S_{16}$  սիգնատուրը ԹԿ թվային կոմպարատորում համեմատվում է սարքին վիճակի էտալոնային սիգնատուրի հետ՝  $S_{t16}$ , եթե դրանք համընկնում են, ապա թեստավորվող սարքը համարվում է սարքին, հակառակ դեպքում՝ անսարք:

### 5.14.3. Աղմկապաշտպանված ցիկլային կոդերի գեներացում և վերծանում

Սխալն ուղղող ցիկլիկ կոդավորումն իրականացվում է՝  $k$ -բիթ սկզբնական կոդային համակցությանը  $m$  ստուգող բիթեր ավելացնելով:

Ցիկլային կոդն օժտված է այն հատկությամբ, որ ցանկացած կոդային համակցության ցիկլային տեղաշարժմամբ ստացվում է կոդի մեկ այլ համակցություն: Ցիկլիկ կոդն օժտված է նաև հետևյալ հատկություններով.

- 1) Այն գեներացվում է  $m=(n-k)$  կարգերով  $G(x)$  գեներատորային բազմանդամով նկարագրվող ԳՈՒԿՏՈՒ-ով, որտեղ  $n$ -ը ցիկլիկ կոդի լրիվ բիթերի թիվն է՝  $n=m+k$ : Այս կոդը կոչվում է  $(n,k)$  ցիկլային կոդ:
- 2) Եթե  $G(x)$ -ը բավարարում է առավելագույն երկարությամբ հաջորդականությունների գեներացման պայմանին, ապա այդ ցիկլային կոդն ի վիճակի է հայտնաբերել կոդային համակցությունների հաղորդման բոլոր եզակի սխալները և բոլոր հարակից սխալները:

Կոդային համակցությունը գեներացվում է սկզբնական կոդային բառի բազմանդամի բազմապատկմամբ  $G(x)$  գեներատորային բազմանդամով:

Օրինակ,  $(7,4)$  կոդը կարելի է գեներացնել

$$G(x) = 1 \oplus x \oplus x^3 \quad (5.69)$$

բազմանդամով: Եթե մուտքային կոդային բառը 1011-ն է, այդ կոդին համապատասխանող բազմանդամը կլինի՝

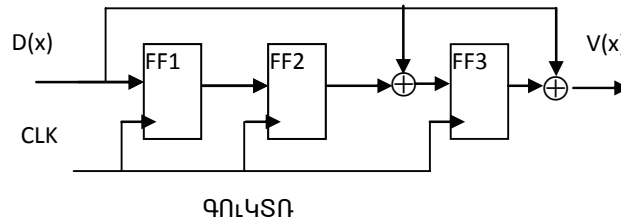
$$D(x) = 1 \oplus x \oplus x^3 :$$

Համապատասխան ցիկլային կոդի համակցությունը կլինի՝

$$\begin{aligned} V(x) &= D(x)G(x) = (1 \oplus x \oplus x^3)(1 \oplus x \oplus x^3) = 1 \oplus x \oplus x^3 \oplus x \oplus x^2 \oplus x^4 \oplus x^3 \oplus x^4 \oplus x^6 = \\ &= 1 \oplus x^2 \oplus x^6 : \end{aligned}$$



Այս բազմանդամին համապատասխանում է 1000101 կոդային հաջորդականությունը:  
Ցիկլային կոդով կոդավորող սարքի սխեման ցույց է տրված նկ. 5.55–ում:



Նկ. 5.55. (7,4) ցիկլիկ կոդի կոդավորիչ գույսՏՈ-ով

Կոդավորվող սկզբնական բառը հաջորդաբար տրվում է ռեգիստրի D(x) մուտքին, իսկ կոդավորված բառի բիթերը հաջորդաբար ստացվում են V(x) ելքում: Ռեգիստրը պետք է նախապես դրվի 0 վիճակ: Աղյուսակ 5.11–ում ցույց են տրված բոլոր կոդային համակցությունները: Սա Յենիմգի կոդ է, որում կոդային բառերի նվազագույն հեռավորությունը 3 է:

Աղյուսակ 5.11

Տվյալ(D <sub>0</sub> D <sub>1</sub> D <sub>2</sub> D <sub>3</sub> )	Կոդ(V <sub>0</sub> V <sub>1</sub> V <sub>2</sub> V <sub>3</sub> V <sub>4</sub> V <sub>5</sub> V <sub>6</sub> )	Տվյալ(D <sub>0</sub> D <sub>1</sub> D <sub>2</sub> D <sub>3</sub> )	Կոդ(V <sub>0</sub> V <sub>1</sub> V <sub>2</sub> V <sub>3</sub> V <sub>4</sub> V <sub>5</sub> V <sub>6</sub> )
0000	0000000	1000	1101000
0001	0001101	1001	1100101
0010	0011010	1010	1110010
0011	0010111	1011	1111111
0100	0110100	1100	1011100
0101	0111001	1101	1010001
0110	0101110	1110	1000110
0111	0100011	1111	1001011

Ցիկլային կոդը չբաժանվող կոդ է՝ այն հնարավոր չէ ապակոդավորել ավելորդ բիթերն ուղղակի կոդից դեն գցելով: Որպես ցիկլային կոդի դեկոդեր օգտագործվում է նույն G(x) բազմանդամով նկարագրվող ԳՅԿՏՈ: Նկ. 5.56–ում ցույց է տրված  $G(x)=1+x+x^3$  բազմանդամով նկարագրվող դեկոդերի կառուցվածքը:

Նկ. 5.56–ում ցույց տրված ցիկլիկ կոդի դեկոդերի աշխատանքը նկարագրվում է հետևյալ հավասարումներով՝

$$B(x) = x^3 Q(x) \oplus x Q(x),$$

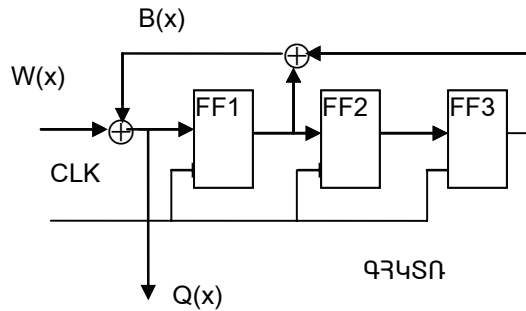
$$Q(x) = W(x) \oplus B(x) = W(x) \oplus x Q(x) \oplus x^3 Q(x):$$

Քանի որ մոդուլ 2-ով գումարումն ու հանումը համարժեք են, ուստի՝

$$Q(x) \oplus x Q(x) \oplus x^3 Q(x) = W(x)$$

ԳՅԿՏՈ-ի ելքային հաջորդականությունը կորոշվի՝

$$Q(x) = W(x) / (1 \oplus x \oplus x^3): \quad (5.70)$$



Նկ. 5.56.  $G(x)=1\oplus x\oplus x^3$  բազմանդամով նկարագրվող դեկոդերի կառուցվածքը

Այսպիսով, եթե ընդունված կոդային բառն աղավաղված չէ՝  $W(x)=V(x)$ , դեկոդավորման արդյունքում կստացվի սկզբնական  $Q(x)=V(x)/G(x)=D(x)$  բառը: Աղյուսակ 5.12–ում ցույց է տրված  $W(x)$ -ի դեկոդավորման ընթացքը  $W(x)=1010001$  օրինակով:

Հաղորդման ժամանակ  $V(x)$  բառի աղավաղման պատճառով որոշ բիթեր կարող են շրջվել: Դա կարելի է ներկայացնել կոդի սկզբնական արժեքին մոդուլ երկուսով գումարելով սխալի վեկտորը: Եթե սխալը հնարավոր է միայն մեկ բիթում, սխալի վեկտորը կարելի է ներկայացնել  $E(x)=x^j$  բազմանդամի տեսքով, որտեղ  $j$ –ն աղավաղված բիթի կարգն է: Ընդունված կոդին համապատասխանում է

$$W(x) = V(x) \oplus E(x) \quad (5.71)$$

բազմանդամը, սխալի բացակայության դեպքում է  $E(x)=0$ : Դեկոդավորման ժամանակ  $W(x)=V(x) \oplus E(x)$  բազմանդամը բաժանվում է  $G(x)$  բազմանդամին.

$$W(x)/G(x) = (V(x) \oplus E(x))/G(x) = D(x) \oplus E(x)/G(x) \quad (5.72)$$

Բաժանման արդյունքում ստացված զրոյից տարբեր մնացորդը վկայում է սխալի առկայության մասին:

Աղյուսակ 5.12

Տակտ	FF1	FF2	FF3	$W(x)$	$B(x)$	$Q(x)$
0	0	0	0	1	0	1
1	1	0	0	0	1	1
2	1	1	0	1	1	0
3	0	1	1	0	1	1
4	1	0	1	0	0	0
5	0	1	0	0	0	0
6	0	0	1	1	1	0
7	0	0	0			
	մնացորդ՝ սխալի հայտնիչ, $S_j(x)=0$ , սխալ չկա			հաղորդված $V(x)$ կոդ		սկզբնական $D(x)$ կոդը

## 5.15. Խնդիրներ

**Խնդիր 5.1.** Նախագծել հաշվիչ T տրիգերների վրա, որը հաշվում է 0, 3, 2, 1 կարգով:

**Խնդիր 5.2.** Նախագծել հաշվիչ JK տրիգերների վրա, որը հաշվում է հետևյալ կարգով.  
ա) 0,1,2, բ) 0,1,2,3,4, գ) 0,1,2,3,4,5,6,

**Խնդիր 5.3.** Նախագծել հաշվիչ RS տրիգերների վրա, որը հաշվում է 0, 1, 3, 2, 6, 4, 5, 7 կարգով:

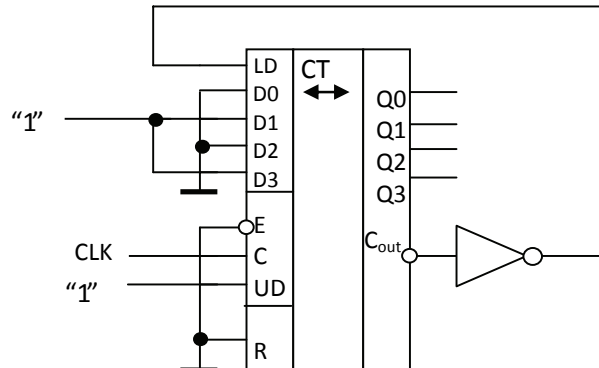
**Խնդիր 5.4.** Նախագծել հաշվիչ T տրիգերների վրա, որը հաշվում է 0, 1, 3, 7, 6, 4 կարգով:

**Խնդիր 5.5.** Նախագծել հաշվիչ JK տրիգերների վրա, որը հաշվում է 0, 4, 2, 1, 6 կարգով:

**Խնդիր 5.6.** Կառուցել  $M=5$  բաժանման գործակցով հաճախականության բաժանիչ, որի ելքում յուրաքանչյուր ցիկլում գեներացվում է մուտքային իմպուլսի տևողությամբ մեկ իմպուլս:

**Խնդիր 5.7.** Որոշել 32-բիթ կուտակող գումարիչով հաճախականության սինթեզատորի մուտքային  $M$  բիթ արժեքը, որպեսզի  $f_{in}=5$  ՄՀց-ի դեպքում ելքային հաճախականությունը լինի 5 կՀց:

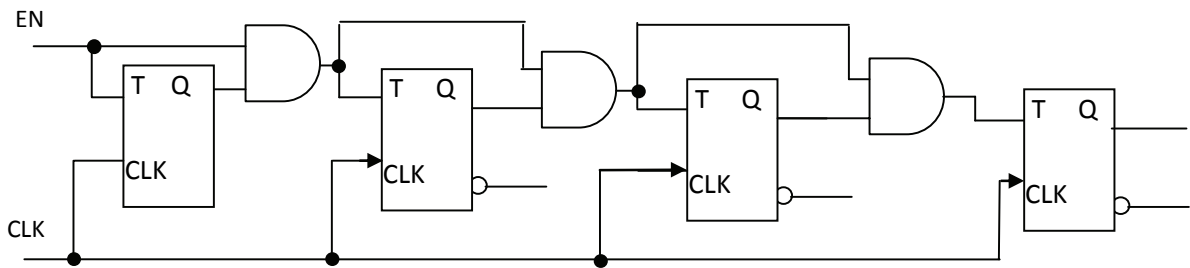
**Խնդիր 5.8.** Որոշել K561IE11 հաշվիչի վրա կառուցված նկ. 5.57-ի սխեմայի վերահաշվման գործակիցը և վիճակների փոփոխության հաջորդականությունը: Կառուցել CLK,  $Q_0$ ,  $Q_1$ ,  $Q_2$ ,  $Q_3$ , LD ազդանշանների ժամանակային դիագրամները:



Նկ. 5.57. Խնդիր 5.8-ին վերաբերող սխեման

**Խնդիր 5.9.** Կրկնել Խնդիր 5.8-ը այն դեպքի համար, երբ  $UD=0$ :

**Խնդիր 5.10.** Տրիգերների անսխալ տակտավորման պայմանից արտահայտել նկ. 5.58-ի հաշվիչի տակտավորման հաճախականությունը տրիգերի տեղակայման ժամանակով՝  $t_{su}$ , տակտային մուտքից-ելք հապաղումով՝  $t_{C-Q}$  և շեփ տարրի հապաղումով՝  $t_{and}$ :

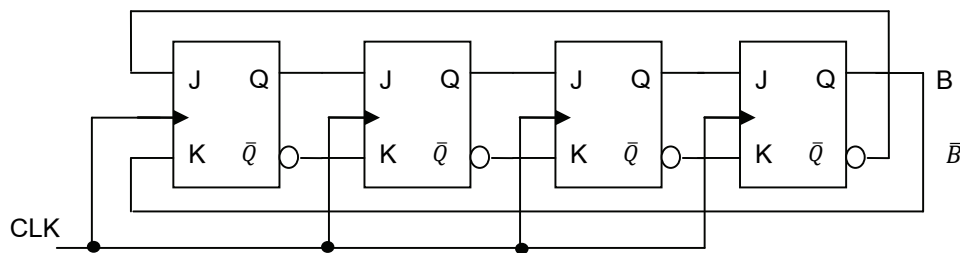


Նկ. 5.58. Խնդիր 5.9-ին վերաբերվող սխեման:

**Խնդիր 5.11.** Կրկնել խնդիր 5.10-ը՝  $n$ -կարգ հաշվիչի համար:

**Խնդիր 5.12.** Կառուցել օղակաձև հաշվիչ  $M=8$  վերահաշվման գործակցով չորս D-տրիգերների վրա: Վերլուծել չօգտագործված վիճակները: Հաշվիչին ավելացնել ինքնաշտկման շղթա՝ չօգտագործված վիճակներից ինքնուրույն դուրս գալու համար:

**Խնդիր 5.13.** Հետագոտել նկ. 5.59-ում բերված սխեման: Համրելով, որ աշխատանքն սկսվում է 0000 սկզբնական վիճակից, կառուցել տրիգերների ելքերի վիճակների ժամանակային դիագրամները: Վերլուծել չօգտագործված վիճակները: Հաշվիչին ավելացնել ինքնաշտկման շղթա՝ չօգտագործված վիճակներից ինքնուրույն դուրս գալու համար:



Նկ. 5.58. Խնդիր 5.9-ին վերաբերող սխեման

**Խնդիր 5.14.** Քառաբիթ տեղաշարժող ռեգիստրի սկզբնական վիճակը 1101 է: Ռեգիստրը տեղաշարժվել է դեպի աջ 6 տակտերով՝ մուտքում ունենալով 101101 հաջորդականությունը: Ցույց տալ ռեգիստրի պարունակությունը յուրաքանչյուր տակտից հետո:

**Խնդիր 5.15.** Կառուցել 8-ակարգ օղակաձև հաշվիչ վիճակների հետևյալ հաջորդականությամբ՝ 11111110, 11111101, ..., 01111111: Վերլուծել չօգտագործված վիճակները: Հաշվիչին ավելացնել ինքնաշտկման շղթա՝ չօգտագործված վիճակներից ինքնուրույն դուրս գալու համար:

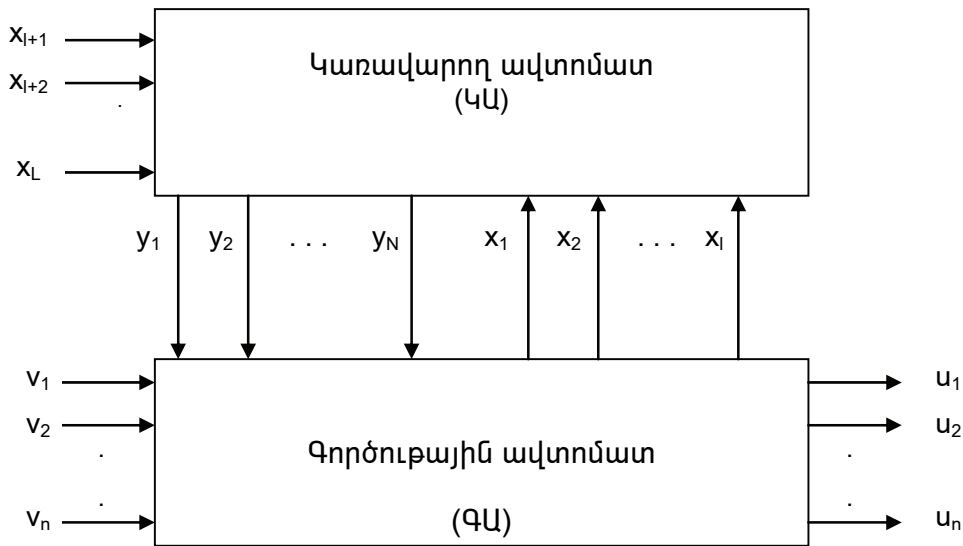
**Խնդիր 5.16.** Կառուցել  $x^0 = x^1 + x^2$  հետադարձ կապի հավասարումով ԳՀԿՏՌ: Ցույց տալ ռեգիստրի ելքում գեներացվող հաջորդականությունը:

**Խնդիր 5.17.** Կառուցել  $x^0 = x^4 + x^3$  հետադարձ կապի հավասարումով ԳՀԿՏՌ: Ցույց տալ գեներացվող առավելագույն պարբերությամբ հաջորդականությունը:

## Գլուխ 6. Միկրոծրագրային ավտոմատներ

### 6.1. Միկրոծրագրային կառավարման սկզբունքը

Երբ թվային համակարգն ունի մեծ թվով ներքին վիճակներ, իսկ աշխատանքը նկարագրվում է բարդ ալգորիթներով, ապա դժվարանում է համակարգի նկարագրությունը և վերլուծությունը որպես միասնական ավտոմատի՝ հիմնված ավտոմատային գրաֆների և անցումների աղյուսակների վրա: Այսպիսի դեպքերում հարմար է համակարգը ներկայացնել գործության (կատարողական) և կառավարող ավտոմատների կոմպոզիցիայի տեսքով (Նկ. 6.1):



Նկ. 6.1. Թվային համակարգի ներկայացումը գործության և կառավարող ավտոմատների կոմպոզիցիայի տեսքով

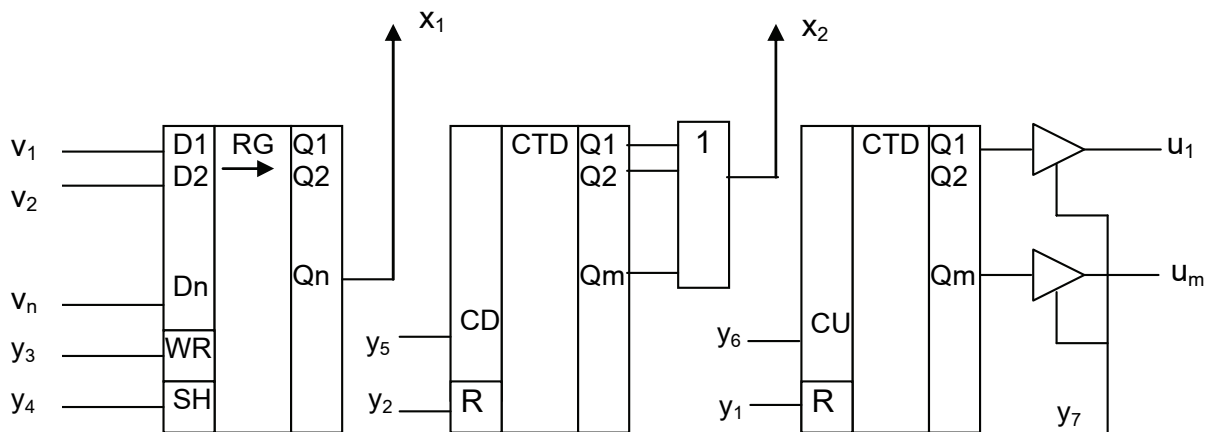
Գործության ավտոմատը (ԳԱ) համակարգի կատարողական մասն է, որտեղ կատարվում են տվյալների մշակման թվաբանական և տրամաբանական գործողություններ նրա մուտքին տրվող  $V=\{v_1, \dots, v_n\}$  երկուական փոփոխականների հետ և ելք է փոխանցում արդյունքը՝  $U=\{u_1, \dots, u_m\}$ : Յուրաքանչյուր գործողություն հրահանգավորվում է կառավարող ավտոմատից՝  $y_1$ -ից  $y_N$  կառավարող երկուական ազդանշաններով:

Կառավարող ավտոմատը (ԿԱ), ելնելով գործության ավտոմատից ստացվող  $x_1$ -ից  $x_i$  և արտաքինից տրվող  $x_{i+1}$ -ից  $x_L$  երկուական ազդանշանների արժեքներից, ժամանակի մեջ հաջորդաբար ձևավորում է  $y_1$ -ից  $y_N$  կառավարող ազդանշաններ, որոնք որոշում են գործության ավտոմատում տվյալների մշակման գործողությունների հաջորդականությունը:

Գործության ավտոմատը թվային համակարգի կատարողական մասն է: Դրա կառուցումը սկզբունքային դժվարություն չի ներկայացնում: Այն սովորաբար կառուցվում է տիպային թվային հանգույցներից՝ ռեգիստրներ, հաշվիչներ, վերձանիչներ, մուլտիպլեքսորներ, բազմաբիթ գումարիչներ և բազմապատկիչներ, կոդեր համեմատող և ձևափոխով սարքեր և այլն:

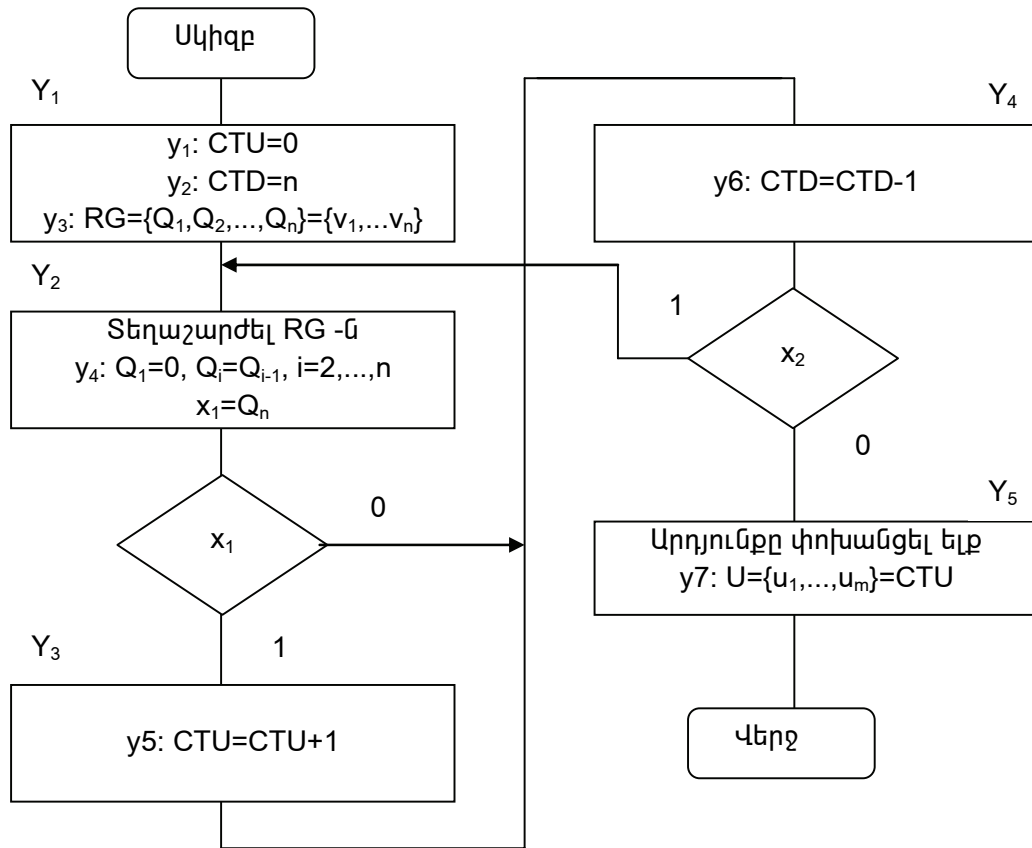
Դիտարկենք թվային համակարգի տրոհման օրինակ:

**Օրինակ 6.1.** Թվային համակարգը պետք է հաշվի մուտքային  $n$ -բիթ  $V=\{v_1, \dots, v_n\}$  երկուական հաջորդականությունում 1-երի քանակը և ելք փոխանցի այդ քանակը երկուական կոդով  $U=\{u_1, \dots, u_m\}$ : Համակարգը պետք է ստուգի մուտքային հաջորդականության յուրաքանչյուր բիթ և արդյունքին գումարի 1 յուրաքանչյուր անգամ, երբ ստուգվող բիթը 1 է: Գործությանին ավտոմատը (նկ. 6.2) բաղկացած է երկու հաշվիչներից՝ CTD հանող հաշվիչ, որը պետք է պահպանի դեռևս չստուգված բիթերի թիվը և CTU գումարող հաշվիչ, որում ձևավորվում է արդյունքը, ինչպես նաև  $n$ -բիթ տեղաշարժող ռեգիստր RG, որում գրվում է ստուգվող հաջորդականությունը և զրոյի դետեկտոր (m-մուտք ԿԱՍ տարր): Ավտոմատային ժամանակի յուրաքանչյուր տակտում ռեգիստրի պարունակությունը տեղաշարժվում է մեկ կարգով և ստուգվում է ռեգիստրի ելքային բիթի արժեքը: Հանող հաշվիչը աշխատանքի սկզբում պետք է բեռնավորվի բիթերի  $n$  թվով, հաշվիչից պետք է հանվի 1 յուրաքանչյուր բիթի ստուգմից հետո: Աշխատանքն ավարտվում է, երբ բոլոր բիթերն ստուգված են, այսինքն, երբ CTD հանող հաշվիչի պարունակությունը դառնում 0: Գումարող հաշվիչի սկզբնական վիճակը 0 է: Կառավարող ավտոմատը հրահանգավորում է գործողությունների կատարման հաջորդականությունը գործությանին ավտոմատում:  $y_1$  ազդանշանով գումարող հաշվիչը զրոյացվում է,  $y_2$ -ով հանող հաշվիչում գրվում է  $n$ ,  $y_3$ -ով մուտքային հաջորդականությունը ղուգահեռ բեռնվում է տեղաշարժող ռեգիստր,  $y_4$ -ով ռեգիստրի պարունակությունը տեղաշարժվում է մեկ կարգով,  $y_5$ -ով հանող հաշվիչից հանվում 1,  $y_6$ -ով գումարող հաշվիչին գումարվում է 1,  $y_7$ -ով արդյունքը փոխանցվում է ելք:



Նկ. 6.2. Երկուական հաջորդականությունում 1-երի թվի հաշվման գործությանին ավտոմատի կառուցվածքային սխեման

Կառավարման համար անհրաժեշտ տեղեկությունը գործությանին ավտոմատից փոխանցվում է կառավարող ավտոմատին երեք տրամաբանական ազդանշաններով՝  $x_1=1$ , եթե ստուգվող բիթը 1 է,  $x_1=0$ ՝ ստուգվող բիթը 0 է,  $x_2=1$ ՝ հանող հաշվիչի պարունակությունը 0 չէ (m-մուտք ԿԱՍ տարրի ելքը 0 չէ),  $x_2=0$ ՝ հանող հաշվիչի պարունակությունը 0 է: Կառավարող ավտոմատի աշխատանքը նկարագրող ալգորիթմը բլոկ-սխեման ցույց է տրված նկ. 6.3-ում:



Նկ. 6.3. Կառավարող ավտոմատի աշխատանքի ալգորիթմը նկարագրող բլոկ-սխեման

Շատ դեպքերում կառավարող ավտոմատի աշխատանքի ալգորիթմը բավականին բարդ է և դրա նկարագրության համար պետք է ունենալ ձևականացված եղանակներ: Դրանց ծանոթանալու համար պետք է ներածվեն որոշակի սկզբնական հասկացություններ:

Գործութային ավտոմատում տվյալների մշակման չտրոհվող ակտը, որը կատարվում է ավտոմատային ժամանակի մեկ տակտում, կոչվում է միկրոգործողություն: Միկրոգործողությունների բազմությունը կնշանակենք՝  $Y = \{y_1, \dots, y_N\}$ : Միկրոգործողությունների կատարումը հրահանգավորում է կառավարող ավտոմատը, որի ելքային ազդանշանները նույնպես նշանակվում են  $y_1, \dots, y_N$  (ինչպես ցույց է տրված նկ. 6.1–ում): Եթե տվյալ պահին պետք է կատարվի  $y_i$  միկրոգործողությունը, ապա կառավարող ավտոմատի ելքում  $y_i=1$ , եթե  $y_i$ -ին չպետք է կատարվի,  $y_i=0$ : Ավտոմատային ժամանակի միևնույն տակտում կատարվող միկրոգործողությունների համախումբը կոչվում է միկրոհրաման՝  $Y_i = \{y_{i1}, \dots, y_{iq}\} \subset Y$ : Դիտարկված օրինակում  $y_1, y_2, y_3$  միկրոգործողությունները կարող են կատարվել միաժամանակ, հետևաբար դրանք կազմում են մեկ միկրոհրաման՝  $Y_1 = \{y_1, y_2, y_3\}$ :

Միկրոհրամանների կատարման հաջորդականությունը որոշվում է  $x_1, x_2, \dots, x_L$  տրամաբանական փոփոխականների արժեքներով, որոնց մի մասը՝  $x_1, x_2, \dots, x_l$  որոշվում է նախորդ տակտում գործութային ավտոմատում կատարված միկրոգործողությու-

յունների արդյունքով, իսկ մյուս մասը՝  $x_{l+1}, x_{l+2}, \dots, x_L$ , տրվում է արտաքինից: Դիտարկված օրինակում  $x_1$  և  $x_2$  տրամաբանական փոփոխականները որոշվում են գործութային ավտոմատում կատարված տեղաշարժի և հանման միկրոգործողությունների արդյունքներով:

Ենթադրենք գործութային ավտոմատում կարող են կատարվել  $Y_1, \dots, Y_G$  միկրոհրամանները, որոնք  $Y = \{y_1, \dots, y_N\}$  բազմության ենթաբազմություններ են: Միկրոհրամանների կատարման հաջորդականությունը տրվում է  $x_1, x_2, \dots, x_L$  տրամաբանական փոփոխականներից կախված անցման ֆունկցիաներով՝  $\alpha_{ij}$ ,  $i, j = 1, \dots, G$ : Եթե  $Y_i$  միկրոհրամանից հետո պետք է կատարվի  $Y_j$ –ն, ապա  $\alpha_{ij} = 1$ : Օրինակ 1-ում տրված բլոկ սխեմայում (նկ. 6.3)  $\alpha_{12} = 1$ , այսինքն միշտ  $Y_1$ –ից հետո կատարվում է  $Y_2$ –ը անկախ որևէ պայմանից,  $\alpha_{23} = x_1$ ,  $\alpha_{24} = \bar{x}_1$ ,  $\alpha_{21} = 0$ ,  $\alpha_{25} = 0$  ֆունկցիաները ցույց են տալիս, որ  $Y_2$ –ից հետո կատարվում է  $Y_3$ –ը, եթե  $x_1 = 1$  ( $\alpha_{23} = 1$ ) և  $Y_3$ –ը, եթե  $x_1 = 0$  ( $\alpha_{24} = 1$ ),  $Y_1, Y_5$ –ը չեն կարող կատարվել  $Y_2$ –ից անջապես հետո:

Անցման ֆունկցիաները օժտված են հետևյալ հատկություններով՝

1. օրթոգոնալությամբ

$$\alpha_{ij} \ \& \ \alpha_{ig} = 0, \quad \text{երբ } j \neq g, \quad (6.1)$$

2. լրիվությամբ

$$\sum_{g=1}^G \alpha_{ig} = 1 : \quad (6.2)$$

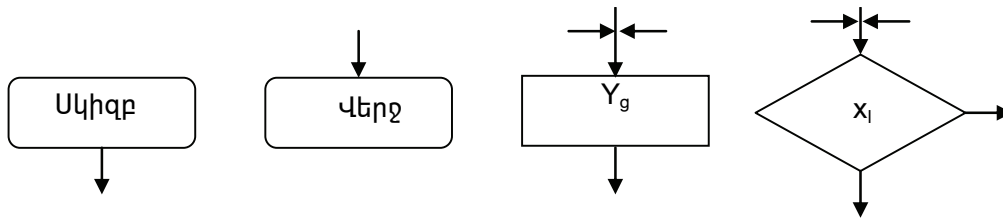
Այս հատկություններից առաջինը ցույց է տալիս, որ յուրաքանչյուր տակտում կարող է կատարվել մեկից ոչ ավել միկրոհրաման, իսկ երկրորդը, որ տվյալ տակտում պետք է անպայման կատարվի որևէ միկրոհրաման:

Միկրոհրամանների և անցման ֆունկցիաների համախումբը կոչվում է միկրոծրագիր: Միկրոծրագիրը տրվում է միկրոհրամանների և անցման ֆունկցիաների հաջորդականությամբ: Միկրոծրագիրը կառավարող ավտոմատի աշխատանքի ալգորիթմի նկարագրությունն է ծրագրի տեսքով: Այդ պատճառով կառավարող ավտոմատը կոչվում է նաև միկրոծրագրային ավտոմատ կամ ալգորիթմիկ ավտոմատ (Algorithmic State Machine, ASM):

## 6.2. Ալգորիթմի գրաֆ-սխեմա

Միկրոծրագրերի նկարագրության համար հարմար է օգտվել ալգորիթմի գրաֆ-սխեմայից (ԱԳՍ) [6]: ԱԳՍ-ն գրականության մեջ նշվում է նաև որպես ալգորիթմի բլոկ սխեմա [7,8]: ԱԳՍ-ն ուղղորդված գրաֆ է, որը պարունակում է չորս տեսակի գագաթներ (նկ.6.4)՝ սկիզբ, վերջ, օպերատորային և պայմանական: Վերջ, օպերատորային և պայմանական գագաթներն ունեն մեկական մուտք, սկիզբ գագաթը մուտք չունի: Վերջ գագաթը ելք չունի: Սկիզբ և օպերատորային գագաթներն ունեն մեկական ելք: Պայմանական գագաթն ունի երկու ելք, որոնք նշվում են 0-ով, եթե գագաթում գրված պայմանը տեղի չունի, 1-ով՝ եթե պայմանը տեղի ունի:





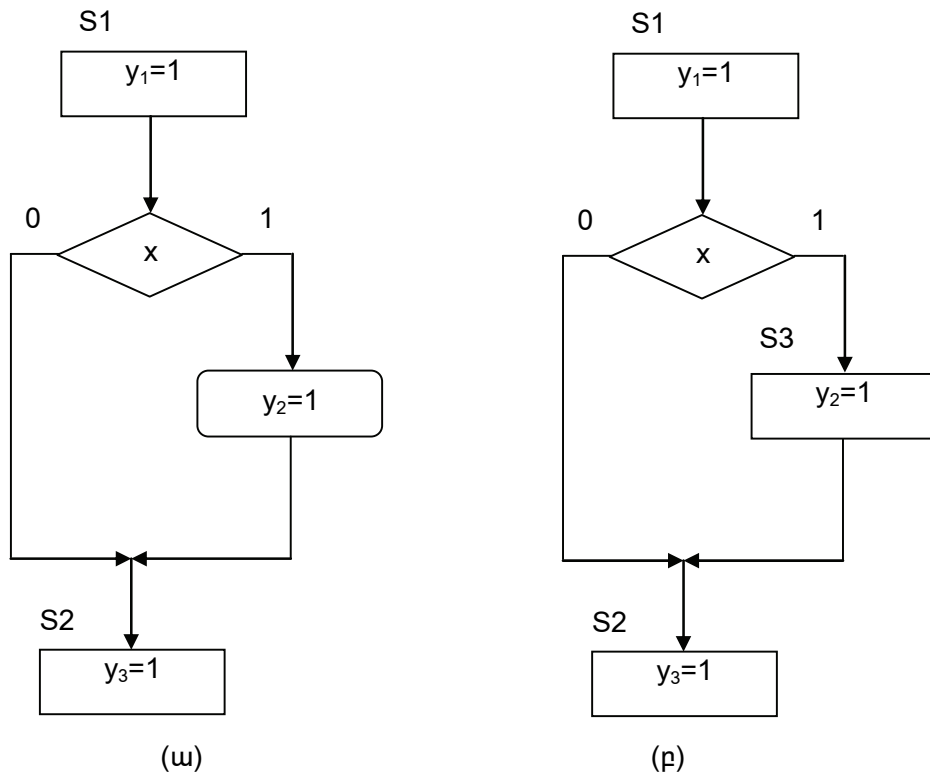
Նկ. 6.4. ԱԳՍ-ի գազաթմերի տիպերը

ԱԳՍ-ն բավարարում է հետևյալ պայմաններին.

1. գազաթմերի ելքերն ու մուտքերը միացվում են աղեղներով, որոնք միշտ ուղղված են ելքից մուտք,
2. յուրաքանչյուր ելք միացված է միայն մեկ մուտքի,
3. յուրաքանչյուր մուտք միացված է գոնե մեկ ելքի,
4. ցանկացած գազաթ գտնվում է սկիզբ գազաթից վերջ գազաթ տանող գոնե մեկ ուղու վրա,
5. յուրաքանչյուր պայմանական գազաթում գրվում է  $x_1, x_2, \dots, x_L$  տրամաբանական փոփոխականներից մեկը, թույլատրվում է տարբեր գազաթներում գրել նույն փոփոխականը,
6. պայմանական գազաթի ելքերից մեկը կարելի է միացնել իր մուտքի հետ (դա անթույլատրելի է օպերատորային գազաթի համար): Այդպիսի պայմանական գազաթը կոչվում է հետադարձ կամ սպասող,
7. յուրաքանչյուր օպերատորային գազաթում գրվում է մեկ օպերատոր (միկրոհրաման)՝  $Y_g$ , որը միկրոգործողությունների  $Y = \{y_1, \dots, y_N\}$  բազմության ենթաբազմություն է: Թույլատրվում է գրել դատարկ ենթաբազմություն՝  $Y_g = \emptyset$ , թույլատրվում է տարբեր գազաթներում գրել նույն միկրոհրամանը:

Նկ. 6.3-ում ցույց տրված բլոկ-սխեման բավարարում է ԱԳՍ-ի վերևում բերված սահմանմանը:

Հարկ է նշել, որ ԱԳՍ-ի բերված սահմանումը ավտոմատների նկարագրության համար միակը չէ, գրականությունում բերվում են նաև այլ սահմանումներ՝ կախված նկարագրվող ավտոմատի տիպից: Օրինակ, [7]-ում ավտոմատի նկարագրության համար սահմանվում է ԱԳՍ, որում ավելացվում է ևս մեկ տիպի գազաթ՝ պայմանական ելքի գործողության գազաթ (նկ. 6.5ա):



Նկ. 6.5. Պայմանական գործողության գազաթ պարունակող ԱԳՍ (ա), պայմանական գործողության գազաթի փոխարինումը օպերատորային գազաթով (բ)

S1 գազաթում կառավարող ավտոմատը գեներացնում է  $y_1$  ազդանշան, ստուգում է  $x$  պայմանը (մուտքային ազդանշանը): Եթե  $x=1$ , ապա գեներացվում է  $y_2$  ազդանշանը և կատարվում է անցումը S2 գազաթ: Իսկ եթե  $x=0$ , անցումը S2 գազաթ կատարվում է առանց  $y_2$ -ի ձևավորման: Պայմանական գործողության գազաթում կառավարող ազդանշանի գեներացումը համապատասխանում է Միլի ավտոմատում ելքային ազդանշանի ձևավորմանը: Այսպես, եթե ԱԳՍ-ի գազաթը համարենք ավտոմատի վիճակ,  $x$ -ը մուտքային ազդանշան, ապա  $y_2$ -ը ձևավորվում է Միլի ավտոմատի ելքերի ֆունկցիայով, այսինքն՝ ելքային ազդանշանի արժեքը կախված է ինչպես վիճակից, այնպես էլ մուտքային ազդանշանից՝  $y_2 = \lambda(S_1, x)$ :  $y_1$ -ի և  $y_3$ -ի ձևավորումը համապատասխանում է Մուրի մոդելին՝ ելքային ազդանշանը կախված է միայն վիճակից, բացահայտ կախված չէ մուտքային ազդանշաններից՝  $y_1 = \lambda(S_1)$ ,  $y_3 = \lambda(S_2)$ : Միշտ էլ հնարավոր է ԱԳՍ-ում խուսափել պայմանական գործողության գազաթից՝ մտցնելով լրացուցիչ օպերատորային գազաթ (նկ 6.5բ), որում նշվում է անհրաժեշտ գործողությունը՝  $y_2 = \lambda(S_3)$ : Անհրաժեշտ է նկատել, որ այս դեպքում  $y_2$ -ը ձևավորվում է նախորդ դեպքի նկատմամբ մեկ տակտ ավելի ուշ: Նկ. 5ա տարբերակում, երբ  $t$  պահին ավտոմատը գտնվում S1 վիճակում, ձևավորվում են  $y_1=1$  և  $y_2=1$  (եթե  $x=1$ ) ելքային ազդանշանները,  $t+1$  պահին ավտոմատն անցնում է S2 վիճակ՝ ձևավորելով  $y_3=1$  ազդանշան: Նկ. 5բ-ում երբ  $t$  պահին ավտոմատը գտնվում S1 վիճակում, ձևավորվում է

$y_1=1$ :  $t+1$  պահին, եթե  $x=0$ , ավտոմատն անցնում է  $S_2$  վիճակ՝ ձևավորելով  $y_3=1$ , իսկ եթե  $x=1$ , ավտոմատն անցնում է  $S_3$  վիճակ՝ ձևավորելով  $y_2=1$ :  $t+2$  պահին ավտոմատն  $S_3$ -ից անցնում է  $S_2$  վիճակ՝ ձևավորելով  $y_3=1$ :

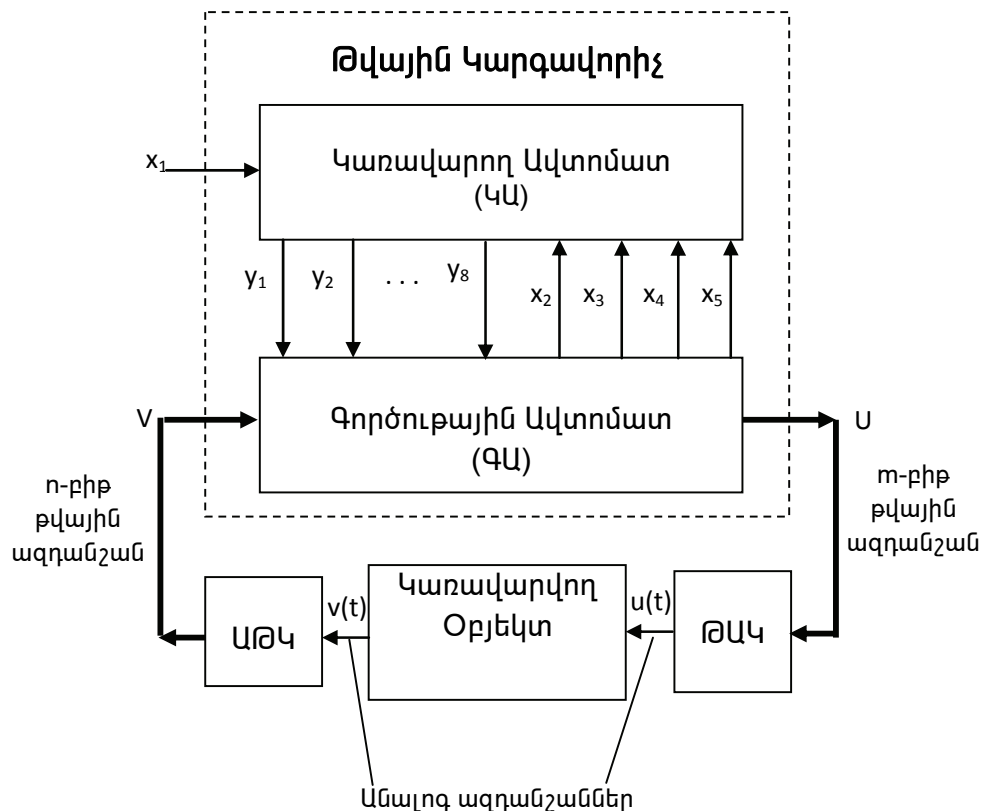
Հաջորդ բաժնում ցույց կտրվեն ԱԳՍ-ից ավտոմատային գրաֆի անցման եղանակները: Ընդ որում, ավտոմատային գրաֆը կարող է համապատասխանել ինչպես Միլի այնպես էլ Մուրի ավտոմատի մոդելին, երկու դեպքում էլ առանց պայմանական գործողությունների գազաթների: Հետագա շարադրանքում կուսումնասիրվեն միայն նկ. 6.5բ-ում ներկայացված տեսքի ԱԳՍ-ն, որը չի պարունակում պայմանական գործողության գազաթ:

Դիտարկենք թվային համակարգը գործությանին և կառավարող մասերի տրոհման և միկրոծրագրի կազմման ևս մի քանի օրինակ:

**Օրինակ 6.2.** Անհրաժեշտ է կառուցել թվային կարգավորիչ, որի մուտքին տրվում է կարգավորվող օբյեկտի ելքային պարամետրը  $n$ -ակարգ երկուական թվի տեսքով՝  $V=\{v_1, \dots, v_n\}$ , իսկ ելքում ձևավորվում է կարգավորող ազդեցությունը  $m$ -ակարգ երկուական թվի տեսքով՝  $U=\{u_1, \dots, u_m\}$ :

Նկ. 6.6-ում ցույց է տրված կառավարման համակարգի բլոկ-սխեման: Այն բաղկացած է կառավարվող օբյեկտից, թվային կարգավորիչից, անալոգ-թիվ և թիվ-անալոգ կերպափոխիչներից (համապատասխանաբար, ԱԹԿ և ԹԱԿ): Օբյեկտի  $v(t)$  ելքային պարամետրի կարգավորման համար պետք է անհրաժեշտ ձևով փոփոխել նրա մուտքային  $u(t)$  կարգավորող ազդեցությունը: Հաճախ օբյեկտի մուտքային և ելքային ազդանշանները անալոգային են: Որպեսզի  $v(t)$  անալոգային ազդանշանը ներածվի թվային կարգավորիչը, այն անհրաժեշտ է թվայնացնել՝ ենթարկել անալոգ-թիվ կերպափոխության: Նմանապես, որպեսզի թվային կարգավորիչի ձևավորած  $U$  կարգավորող ազդեցությունը կիրառվի օբյեկտին,  $U$ -ն պետք է կերպափոխվի  $u(t)$  անալոգային ազդանշանի:

Գործնականում կարգավորող ազդեցության փոփոխության տիրույթը սահմանափակ է: Օրինակ, եթե պատկերացնենք կառավարվող օբյեկտը էլեկտրական վառարան է, որի ներսում ջերմաստիճանը պետք է պահվի առաջադրված  $V_0$  արժեքին հավասար, իսկ կարգավորող ազդեցությունը վառարանի պարույրին կիրառվող լարումն է, ապա ակնհայտ է, որ լարման առավելագույն արժեքը սահմանափակված է ցանցի լարումով՝  $U_9=220$ Վ: Կարգավորիչը կարող է պարույրին կիրառվող լարումը կարգավորել 0-ից մինչև 220 Վ սահմաններում: Ընդ որում, կարգավորող ազդեցության արժեքների  $M$  թիվը, որոշվում է  $U$  կարգավորող ազդեցության  $m$  կարգաթվով՝  $M=2^m$ : Այսինքն կարգավորող ազդեցությունն ունի  $M$  ընդհատ մակարդակներ: Ընդհանուր դեպքում կարող ենք համարել, որ կարգավորող ազդեցությունը սահմանափակված է երկու կողմից.



Նկ. 6.6. Կառավարման համակարգի բլոկ-սխեման

$$U(k) := \begin{cases} A, & \text{եթե } U(k) \leq A \\ U(k), & \text{եթե } A \leq U(k) \leq B \\ B, & \text{եթե } U(k) \geq B, \end{cases} \quad (6.3)$$

որտեղ՝  $U(k)$  և  $V(k)$  վերագրման օպերատորն է,  $k$ -ն՝ ընդհանուր ավտոմատային ժամանակը:

Համարենք, որ կարգավորումն իրականացվում է “վերև-ներքև” (Up-Down) պարզագույն ալգորիթմով՝

$$U(k) := \begin{cases} U(k-1) - 1, & \text{եթե } V(k) > V_0 \\ U(k-1), & \text{եթե } V(k) = V_0 \\ U(k-1) + 1, & \text{եթե } V(k) < V_0, \end{cases} \quad (6.4)$$

$k$ -րդ տակտում  $U$ -ի ընթացիկ արժեքը փոքրացվում է 1-ով, եթե օբյեկտի կարգավորվող  $V$  պարամետրը փոքր է առաջադրված  $V_0$  արժեքից, թողնվում է անփոփոխ, եթե  $V=V_0$  և մեծացվում է 1-ով, եթե  $V < V_0$ :

Բանաձև (6.4)-ով նկարագրվող կարգավորման օրենքի իրականացման համար անհրաժեշտ է կատարել  $V$  և  $V_0$  երկուսկան թվերի համեմատում  $n$ -կարգ թվային համեմատող սարքով, 1 –ի գումարում կամ հանում դարձափոխելի հաշվիչի միջոցով,  $U$ -ի համեմատում  $A$ -ի կամ  $B$ -ի հետ  $m$ -կարգ համեմատող սարքով,  $V_0$ ,  $A$ ,  $B$  հաստատունների պահպանում ռեգիստրներում:

Կազմենք միկրոգործողությունների և տրամաբանական պայմանների ցուցակ:  $x_1$ -ով նշանակենք արտաքինից տրվող տրամաբանական փոփոխականը, որով թույլատրվում է ( $x_1=1$ ) կարգավորիչի աշխատանքը փակ կոնտուրում:  $y_1$ -ով նշանակենք  $V$ -ի և  $V_0$ -ի համեմատման միկրոգործողությունը, որի արդյունքը գնահատվում է  $x_2$  և  $x_3$  տրամաբանական փոփոխականներով:

$$x_2 = \begin{cases} 1, & \text{եթե } V = V_0 \\ 0, & \text{եթե } V \neq V_0 \end{cases}, x_3 = \begin{cases} 1, & \text{եթե } V > V_0 \\ 0, & \text{եթե } V \leq V_0 \end{cases} \quad (6.5)$$

Թվային համակարգերի վարքագծային նկարագրության համար օգտագործվում է ռեգիստրային փոխանցումների տրամաբանությունը՝ ՌՓՏ (Register Transfer Logic, RTL): Թվային համակարգը դիտվում է որպես ռեգիստրների և այդ ռեգիստրների միջև տվյալների փոխանակման գործողությունների բազմություն: Օրինակ, համակարգիչը պարունակում է առանձին ռեգիստրներ, հիշողություն (ավելի դանդաղ աշխատող ռեգիստրների բազմություն) և սարքեր (թվաբանական-տրամաբանական սարք), որոնք այդ ռեգիստրների միջև փոխանակվող տվյալների հետ կատարում են թվաբանական և տրամաբանական գործողություններ: Միկրոգործողությունների գրառման համար հարմար է օգտագործել ռեգիստրային փոխանցումների նշանակումները (Register Transfer Notation, RTN): Ներկայումս գրականության մեջ բացակայում են ռեգիստրային փոխանցումների միասնական նշանակումներ:

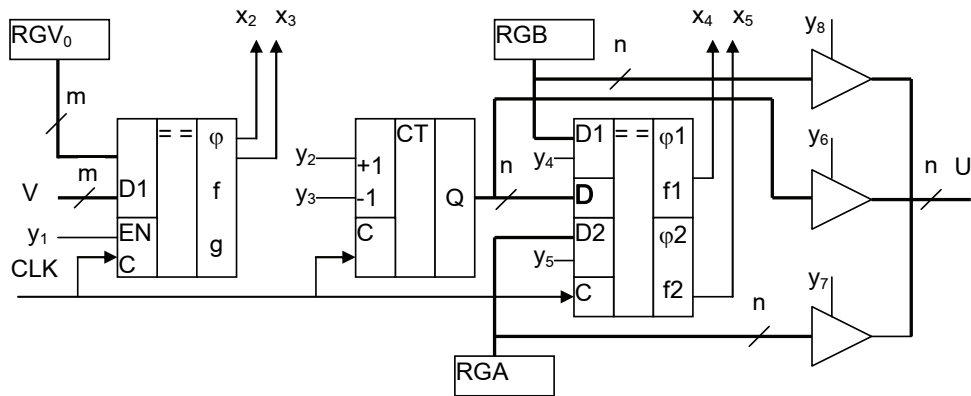
Դիտարկվող օրինակի համար կատարենք ռեգիստրային փոխանցումների հետևյալ նշանակումները՝

$$\begin{aligned} y_1 &= V \sim V_0, \\ y_2 - (CT) &:= (CT) + 1, \\ y_3 - (CT) &:= (CT) - 1, \\ y_4 - (CT) &\sim A, \\ y_5 - (CT) &\sim B, \\ y_6 - U &:= (CT), \\ y_7 - U &:= A, \\ y_8 - U &:= B, \end{aligned} \quad (6.6)$$

որտեղ (CT) - դարձափոխելի հաշվիչի պարունակությունն է, CT-ում գտնվում է U-ի ընթացիկ արժեքը, “ $\sim$ ” նշանով նշված է կողերի համեմատման գործողությունը: CT հաշվիչի պարունակության (U-ի ընթացիկ արժեքի) և A ու B հաստատումների համեմատության արդյունքները գնահատվում են  $x_4$  և  $x_5$  տրամաբանական փոփոխականներով՝

$$x_4 = \begin{cases} 1, & \text{եթե } (CT) > B \\ 0, & \text{եթե } (CT) \leq B \end{cases}, x_5 = \begin{cases} 1, & \text{եթե } (CT) < A \\ 0, & \text{եթե } (CT) \geq A \end{cases} \quad (6.7)$$

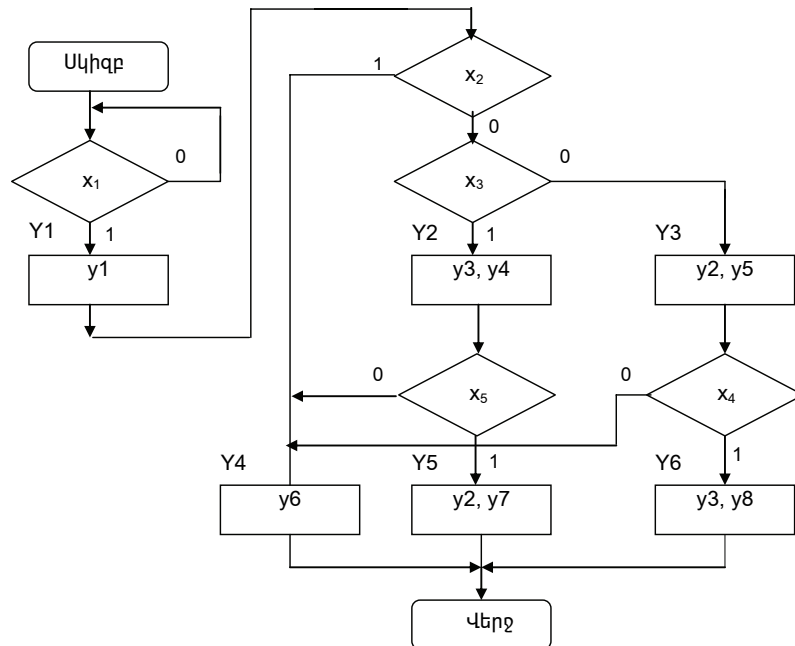
Բերված նկարագրության հիման վրա կարելի է կառուցել գործույթային ավտոմատի կառուցվածքային սխեման (նկ. 6.7):



Նկ. 6.7. Գործուբային ավտոմատի կառուցվածքային սխեման

V-ի համեմատությունը  $V_0$ -ի հետ (այն տրվում է  $RGV_0$  m-կարգ ռեգիստրից) իրագործվում է CLK տակտային ազդանշանի ակտիվ մակարդակով, եթե  $y_1=1$ : Տակտային ազդանշանով հաշվիչի պարունակությանը գումարվում է 1, եթե  $y_2=1$ , հանվում է 1, եթե  $y_3=1$ , և պարունակությունը չի փոխվում, եթե  $y_2=0$  և  $y_3=0$ : Հաշվիչի ընթացիկ պարունակությունը՝  $Q=(CT)$ , համեմատվում է RGA ռեգիստրում պահվող A հաստատունի հետ, եթե  $y_5=1$  և համեմատվում է RGB ռեգիստրում պահվող B հաստատունի հետ, եթե  $y_4=1$ : Համեմատությունները կատարվում են CLK տակտային ազդանշանի ակտիվ մակարդակի տևողության ընթացքում: Ս ելք փոխանցվող ազդանշանը ընտրվում է  $y_6, y_7, y_8$  ազդանշաններով՝ եռավիճակ ելքերով բուֆերների միջոցով:

Ելնելով կարգավորիչի աշխատանքի նկարագրությունից և միկրոգործողությունների ու տրամաբանական պայմանների ցուցակից՝ կազմվում է կարգավորիչի միկրոծրագրի ԱԳՍ-ն (նկ. 6.8):



Նկ. 6.8. Թվային կարգավորիչի միկրոծրագրի ԱԳՍ-ն

Կառավարող ավտոմատը սպասում է  $x_1$  գազաթուն այնքան ժամանակ, քանի դեռ արտաքինից չի տրվել  $x_1=1$  աշխատանքի թույլտվությունը: Ավտոմատային ժամանակի միևնույն տակտում կատարվող միկրոգործողությունները նշված են մեկ օպերատորային գազաթուն, որի վրա նշված է համապատասխան միկրոհրամանը: Օրինակ,  $y_2$  և  $y_4$  միկրոգործողությունները կազմում են  $Y_2$  միկրոհրամանը: Յուրաքանչյուր միկրոհրամանի համապատասխանում է մեկ օպերատորային գազաթ:

ԱԳՍ-ն միկրոծրագրի գրաֆիկական պատկերումն է: Միկրոծրագիրը կարելի է նաև նկարագրել տեքստային եղանակով՝ փսևդոկոդով, որի համար օգտվում են ծրագրավորման ստանդարտ կառուցվածքները՝ IF-ELSE, IF-ELSE-IF...: Նկ. 6.8-ում պատկերված ԱԳՍ-ին համապատասխանող փսևդոծրագիրը (փսևդոկոդն) ունի հետևյալ տեսքը՝

```

Սկիզբ: if( $x_1=0$ ) գնալ Սկիզբ
else  $y_1$ ;
    if( $x_2=1$ ) { $y_6$ , գնալ Վերջ}
    else if( $x_3=0$ ) { $y_2, y_5$ }
        if( $x_4=1$ ) { $y_3, y_8$ , գնալ Վերջ}
        else { $y_6$ , գնալ Վերջ};
    else if( $x_5=1$ ) { $y_2, y_7$ , գնալ Վերջ}
    else { $y_6$ , գնալ Վերջ};

```

Վերջ:

Այստեղ “Սկիզբ” և “Վերջ” համապատասխան տողերի նշակետերն են, որոնց կատարվում է անցում ծրագրի այլ կետերից:

**Օրինակ 6.3.** Բազմաբիթ երկուական թվերի հաջորդական բազմապատկիչ:

Երկու երկուական թվերի ձեռքով բազմապատկումն իրագործվում է հաջորդական գումարումների և տեղաշարժերի միջոցով: Այդ գործընթացը ցուցադրվում է ստորև բերված թվային օրինակով: Ենթադրենք բազմապատկվում են  $23_{10}=10111_2$  և  $19_{10}=10011_2$  թվերը (աղյուսակ 6.1):

Աղյուսակ 6.1

1	0	1	1	1						բազմապատկելի
1	0	0	1	1						բազմապատկիչ
1	0	1	1	1						
0	1	1	1							
0	0	0	0	0	0					
0	0	0	0	0						
1	0	1	1	1						
1	1	0	1	1	0	1	0	1		արտադրյալ



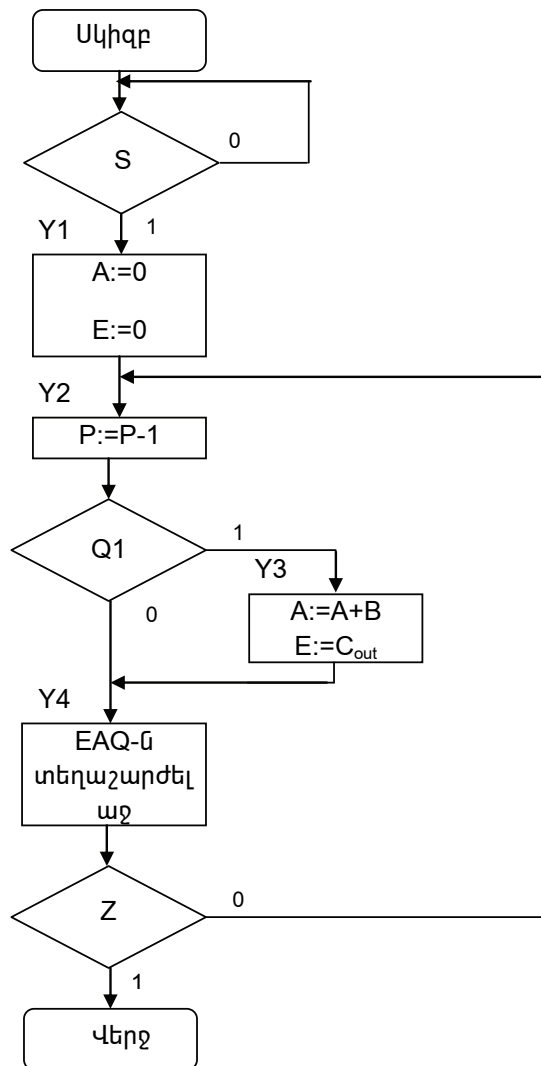


A-ում գտնվող մասնակի արտադրյալը և Q-ում գտնվող բազմապատկիչը միա-  
 ժամանակ մեկ բիթով տեղաշարժվում են աջ՝ A-ի կրտսեր բիթը տեղաշարժվում է Q-ի  
 ավագ բիթ,  $C_{out}$ -ը E-ից տեղաշարժվում է A-ի ավագ բիթ, իսկ E տրիգերը գրոյացվում  
 է: Աջ տեղաշարժի գործողությունից հետո մասնակի արտադրյալի մեկ բիթը փոխանց-  
 վում է Q, իսկ բազմապատկիչի բիթերը մեկ բիթով տեղաշարժվում են աջ: Այս կերպ Q  
 ռեգիստրի ամենաաջ բիթը, որը նշանակված է Q1-ով, միշտ պարունակում է բազմա-  
 պատկիչի այն բիթը, որը պետք է ստուգվի հաջորդ քայլում:

Բազմապատկիչի ԱԳՍ-ն ցույց է տրված նկ. 6.10-ում: Սկզբնական վիճակում  
 բազմապատկելին գտնվում է B-ում, բազմապատկիչը՝ Q-ում: Բազմապատկման գործ-  
 ընթացն սկսվում է, երբ  $S=1$ : A ռեգիստրը և E տրիգերը 0-ացվում են, իսկ P հաշվիչում  
 բեռնվում է բազմապատկիչի բիթերի թիվը՝ n:

Այնուհետև, գործընթացը մտնում է փակ օղակ, որում ձևավորվում են մասնակի  
 արտադրյալները: Ստուգվում է բազմապատկիչի Q1-ում գտնվող բիթը, եթե այն 1 է,  
 ապա B բազմապատկելին գումարվում է A-ում գտնվող մասնակի արտադրյալին, գու-  
 մարումից առաջացած անցումը փոխանցվում է E: Իսկ եթե Q1-ը 0 է, ապա A-ում  
 գտնվող մասնակի արտադրյալը թողնվում է անփոփոխ: P հաշվիչից հանվում է 1՝ ան-  
 կախ Q1-ի արժեքից: E, A, Q ռեգիստրները միացված են հաջորդաբար՝ կազմելով մեկ  
 համակցված EAQ ռեգիստր, որը տեղաշարժվում է աջ՝ ձևավորելով նոր մասնակի  
 արտադրյալը:

Յուրաքանչյուր մասնակի արտադրյալի ձևավորումից հետո ստուգվում է P հաշ-  
 վիչը: Եթե P-ի պարունակությունը 0 չէ, ապա Z կառավարող տրամաբանական մուտքը  
 0 է, և գործընթացը կրկնվում է՝ ձևավորելով նոր մասնակի արտադրյալ: Գործընթացը  
 կանգ է առնում, երբ P-ի պարունակությունը հասնում 0-ի, և Z կառավարող տրամաբա-  
 նական մուտքը հավասարվում է 1-ի: Նկատենք, որ A-ում ձևավորված մասնակի ար-  
 տադրյալը տեղաշարժվում է Q և զբաղեցնում է բազմապատկիչի տեղը: Բազմապատ-  
 կումից հետո բազմապատկիչն այլևս մատչելի չէ: Վերջնական արտադրյալը ձևավոր-  
 վում է A և Q ռեգիստրներում՝ A-ում ավագ բիթերը, Q-ում՝ կրտսեր բիթերը:



Նկ. 6.10. Բազմապատկիչի կառավարող ավտոմատի ԱԳՍ-ն

Աղյուսակ 6.2–ում բերված է բազմապատկիչի ռեգիստրների պարունակությունների փոփոխությունները բազմապատկման գործընթացի ժամանակ վեկուն բերված թվային օրինակի դեպքում՝  $Q=10011$  և  $B=10111$ :

Աղյուսակ 6.2

Տակտ	Գործընթաց	E	A	Q	P
1	Y1: Բազմապատկիչը Q-ում, $A:=0$ , $P:=5$	0	00000	10011	101
2	Y2: $P:=P-1$	0	00000	10011	100
3	$Q1=1$ : Y3: առաջին մասնակի արտադրյալը՝ $EA=A+B$	0	10111	10011	100
4	Y4: EAQ-ն տեղաշարժել աջ	0	01011	11001	100
5	$Z=0$ : Y2: $P:=P-1$	0	10111	11001	011
6	$Q1=1$ : Y3: Երկրորդ մասնակի արտադրյալը՝ $EA=A+B$	1	00010	11001	011
7	EAQ-ն տեղաշարժել աջ	0	10001	01100	011

Տակտ	Գործընթաց	E	A	Q	P
8	$Z=0$ Y2: $P:=P-1$	0	10001	01100	010
9	$Q1=0$ Y4: EAQ-ն տեղաշարժել աջ	0	01000	10110	010
10	$Z=0$ Y2: $P:=P-1$	0	01000	10110	001
11	$Q1=0$ Y4: EAQ-ն տեղաշարժել աջ	0	00100	01011	001
12	$Z=0$ Y2: $P:=P-1$	0	00100	01011	000
13	$Q1=1$ Y3: Հինգերորդ մասնակի արտադրյալը՝ $EA=A+B$	0	11011	01011	000
14	EAQ-ն տեղաշարժել աջ	0	01101	10101	000
	$Z=0$ Վերջնական արտադրյալը AQ	0	01101	10101	000

**Օրինակ 6.4.** Բազմաբիթ երկուական թվերի հաջորդական բաժանում:

Նկ. 6.11ա-ում ցույց է տրված անկյունով բաժանման օրինակ, բաժանելին 243 է, բաժանարարը՝ 23: Առաջին քայլում փորձ է արվում բաժանելիի առաջին նիշը՝ 2-ը բաժանել 23-ի, որը չի հաջողվում: Հաջորդ քայլում փորձ է արվում բաժանել 24-ը 23-ի, որոշվում է քանորդի առաջին թվանշանը՝ 1: Կատարվում է  $24-23=1$  հանումը, բաժանելիի վերջին թվանշանը՝ 3 ներքև է բերվում և կազմվում ընթացիկ բաժանելին՝ 13, որը չի բաժանվում 23-ի: Ուստի քանորդի հաջորդ թվանշանը կլինի 0: Մնացորդը կլինի 13, իսկ քանորդը՝ 10: Երկուական կոդերով բաժանումը կատարվում է նույն կարգով, միայն այն տարբերությամբ, որ քանորդի յուրաքանչյուր թվանշան կարող է լինել 1 կամ 0: Ստորև ցույց են տրված բաժանման քայլերը՝ առաջին չորս քայլերում բաժանելին հաջորդաբար տեղաշարժվում է ձախ և փորձ է արվում բաժանելիի առաջին չորս նիշերը՝ 1, 11, 111, 1111, բաժանել 10111-ի: Այդ քայլերը չեն հաջողվում, և քանորդի ավագ չորս բիթերում գրանցվում է 0000: Հաջորդ տեղաշարժից հետո տեղաշարժված բաժանելիի ավագ հինգ բիթերում ստացվում է 11110, որից հանվում է բաժանարարը՝ 10111, քանորդի համապատասխան բիթը (հիգերորդը՝ հաշված ավագ բիթից) կարգվում է 1 վիճակ: Հերթական տեղաշարժից հետո մասնակի մնացորդում ստացվում է 1110, որը փոքր է 10111-ից: Ուստի, քանորդի հաջորդ բիթում գրվում է 0 և կատարվում է մասնակի մնացորդի տեղաշարժ: Մասնակի մնացորդը կլինի 11101, որից հանվում է բաժանարարը, իսկ քանորդի հերթական բիթը՝ կարգվում է 1: Նոր մնացորդը վերջին տեղաշարժից հետո կլինի 1101, որը փոքր է բաժանարարից: Ուստի, քանորդի վերջին բիթը կլինի 0 իսկ վեջնական մնացորդը՝ 1101:

243	23	(A)	1	1	1	1	0	0	1	1	10111 (B)
23	010		1	0	1	1	1				00001010 → քանորդ(Q)
13					1	1	1	0	1		
					1	0	1	1	1		
							1	1	0	1	→ մնացորդ(R)
(ա)										(բ)	

Նկ. 6.11. Բաժանման օրինակ. (ա) տասական թվանշաններով, (բ) երկուական թվանշաններով:

Պետք է նախագծել թվային սարք, որն իրագործում է տրված  $A$  և  $B$   $n$ -բիթ երկուական թվերի բաժանումը: Բաժանման արդյունքը տրվում է երկու  $n$ -բիթ ելքերով՝  $Q$  քանորդով և  $R$  մնացորդով: Նկ. 6.11-ում ցույց տրված ալգորիթմը կարելի է իրականացնել  $A$  բաժանելին հաջորդաբար տեղաշարժելով դեպի ձախ՝ ավագ բիթը ներածելով  $R$  ռեգիստր: Յուրաքանչյուր տեղաշարժի գործողությունից հետո  $R$ -ը համեմատվում է  $B$ -ի հետ: Եթե  $R \geq B$ , ապա քանորդի համապատասխան բիթը կարգվում է 1 վիճակ և  $R$ -ից հանվում է  $B$ -ն: Եթե  $R < B$ , ապա  $Q$ -ի համապատասխան բիթը դրվում 0 վիճակ: Այս ալգորիթմի փսևդոկոդը ցույց է տրված նկ. 6.12-ում:  $R||A$  նշանակումը համապատասխանում է 2 $n$ -բիթ տեղաշարժող ռեգիստրի, որի կրտսեր  $n$  բիթերը գտնվում են  $A$ -ում, իսկ ավագ  $n$ -բիթերը՝  $R$ -ում: Բաժանման յուրաքանչյուր տակտում  $Q$  ռեգիստրի համապատասխան բիթը դրվում է 1 կամ 0 վիճակ: Դրա իրագործման ամենապարզ եղանակը յուրաքանչյուր տակտում  $Q$ -ի կրտսեր բիթ 1 կամ 0 ներմուծելն է՝ դեպի ձախ տեղաշարժի միջոցով:

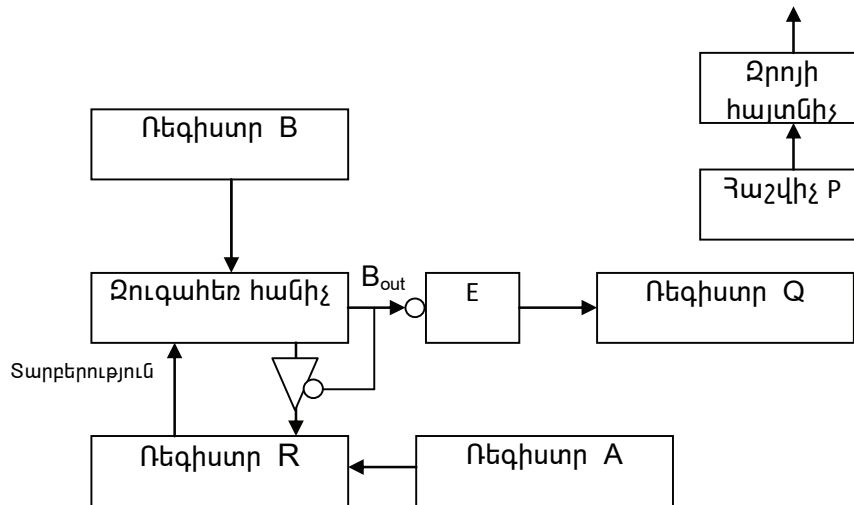
```

R=0;
for i=0 to n-1 do
    left-shift R||A;
    if R≥B then Qi=1; R=R-B;
else Qi=0;
end if;
end for;

```

Նկ. 6.12. Բաժանման ալգորիթմի փսևդոկոդը

Բաժանիչի կատարողական մասը ցույց է տրված նկ. 6.13-ում: Հանելին պահվում է  $B$  ռեգիստրում, քանորդը՝  $Q$  ռեգիստրում, մասնակի մնացորդը ձևավորվում է  $A$  և  $R$  ռեգիստրներում: Բաժանման ավարտին մնացորդը գտնվում է  $R$  ռեգիստրում: Ձուգահեռ հանիչը  $R$  ռեգիստրից հանում է  $B$  ռեգիստրի պարունակությունը: Հանման ժամանակ առաջացող փոխառության  $B_{out}$  բիթի ինվերս արժեքը պահվում է  $E$  տրիգերում: Սկզբում քանորդի բիթերի թիվը բեռնվում է  $P$  հաշվիչ: Այնուհետև, յուրաքանչյուր մասնակի արդյունքի ձևավորումից հետո հաշվիչից հանվում է 1: Եթե հանման գործողությունից ստացվում է փոխառություն՝  $B_{out}=1$ , ապա քանորդի ռեգիստրի հերթական բիթին վերագրվում է 0, իսկ հանման արդյունքը չի գրանցվում  $R$  ռեգիստր,  $R||A$  ռեգիստրների զույգը տեղաշարժվում է ձախ: Իսկ եթե հանման գործողությունից փոխառություն չի ստացվում՝  $B_{out}=0$ , ապա քանորդի ռեգիստրի հերթական բիթին վերագրվում է 1, հանման արդյունքը գրանցվում է  $R$  ռեգիստր,  $R||A$  ռեգիստրների զույգը տեղաշարժվում է ձախ: Երբ  $P$  հաշվիչի պարունակությունը հասնում է 0-ի, վերջնական արդյունքը ձևավորված է  $R$ ,  $Q$  ռեգիստրներում և բաժանման գործընթացն ավարտված է:

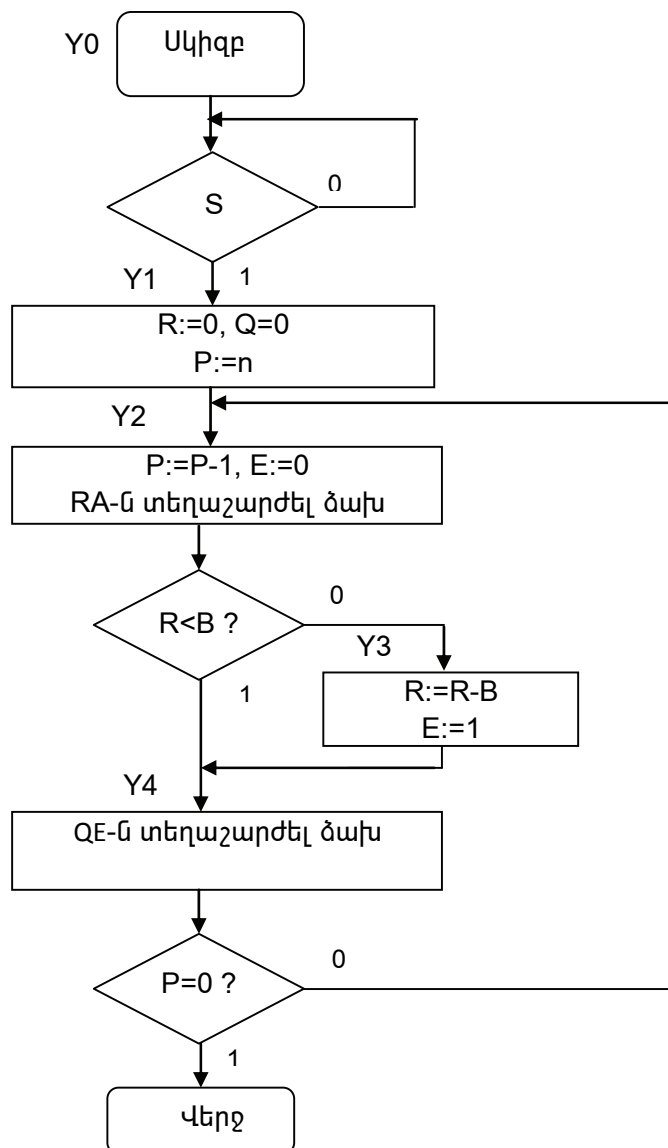


Նկ. 6.13. Երկուական թվերի բաժանիչի կատարողական ավտոմատը

Բազմաթիվ երկուական թվերի բաժանման ալգորիթմի գրաֆ-սխեման ցույց է տրված նկ. 6.14-ում: Կառավարող ավտոմատը սպասում է սկզբնական վիճակում մինչև S սկիզբ ազդանշանի ստանալը: Որից հետո սկսվում է բաժանման գործընթացը: Նախ կարգվում են ռեգիստրների սկզբնական վիճակները՝  $R=0$ ,  $Q=0$ ,  $P=n$ : Այնուհետև, փակ ցիկլով P հաշվիչից հանվում է 1, RA-ն տեղաշարժվում է ձախ, E-ն դրվում է 0 վիճակ: R-ից հանվում է B-ն: Այդ տարբերությունը կազմում է մասնակի մնացորդը: Եթե հանումից առաջանում է փոխառություն՝  $R < B$ , ապա E-ն մնում է 0 վիճակում: Հակառակ դեպքում E-ն դրվում է 1 վիճակ: QE-ն տեղաշարժվում է ձախ՝ E-ի վիճակը ներածվում է Q ռեգիստրի կրտսեր բիթ: Այս գործընթացը կրկնվում է մինչև  $P=0$ -ի ստացումը: Երբ  $P=0$ , բաժանումն ավարտված է:

Աղյուսակ 6.3-ում ցույց են տրված ռեգիստրների պարունակությունները ալգորիթմի աշխատանքի յուրաքանչյուր տակտում, երբ կատարվում է  $243_{10}/23_{10}=11110011_2/10111_2$  բաժանումը: Համարվում է, որ բոլոր ռեգիստրները, բացի E-ից, 8-բիթ են:

Միկրոժրագրային կառավարման սկզբունքը հնարավորություն է տալիս իրականացնել համակարգված նախագծում, որի արդյունքում կարելի է ստանալ բարձր հուսալիության և արագագործ թվային համակարգ: Միաժամանակ համակարգված նախագծումը հեշտացնում է անհրաժեշտության դեպքում նախագծում պահանջվող փոփոխություններ մտցնելը:



Նկ. 6.14. Բազմաբիթ երկուական թվերի բաժանման ալգորիթմի գրաֆ սխեման

Աղյուսակ 6.3

Տակտ	Վիճակ	R	A	B	Q	E	P
1	Y1	00000000	11110011	00010111	00000000	X	00001000
2	Y2	00000001	11100110	00010111	00000000	0	00000111
3	Y4	00000001	11100110	00010111	00000000	0	00000111
4	Y2	00000011	11001100	00010111	00000000	0	00000110
5	Y4	00000011	11001100	00010111	00000000	0	00000110
6	Y2	00000111	10011000	00010111	00000000	0	00000101
7	Y4	00000111	10011000	00010111	00000000	0	00000101
8	Y2	00001111	00110000	00010111	00000000	0	00000100
9	Y4	00001111	00110000	00010111	00000000	0	00000100

Աղյուսակ 6.3-ի շարունակություն

Տակտ	վիճակ	R	A	B	Q	E	P
10	Y2	00011110	01100000	00010111	00000000	0	00000100
11	Y3	00000111	01100000	00010111	00000000	1	00000011
12	Y4	00000111	01100000	00010111	00000001	1	00000011
13	Y2	00001110	11000000	00010111	00000001	0	00000010
14	Y4	00001110	11000000	00010111	00000010	0	00000010
15	Y2	00011101	10000000	00010111	00000010	0	00000001
16	Y3	00000110	10000000	00010111	00000010	1	00000001
17	Y4	00000110	10000000	00010111	00000101	1	00000001
18	Y2	00001101	00000000	00010111	00000101	0	00000000
19	Y4	00001101	00000000	00010111	00001010		00000000

### 6.3. Միկրոծրագրային ավտոմատի սինթեզ

Միկրոծրագրային ավտոմատի սինթեզը ըստ ԱԳՍ-ի կատարվում է երկու փուլով՝ ԱԳՍ-ի նշում և ավտոմատի անցումների գրաֆի կառուցում:

Միկրոծրագրային Միլի ավտոմատի սինթեզի համար ԱԳՍ-ի նշումը  $a_1, a_2, \dots, a_m$  սինվոլներով կատարվում է հետևյալ ալգորիթմով [6] (ալգորիթմ Ա1)՝

- 1)  $a_1$  սինվոլով նշվում է սկիզբ գազաթին հաջորդող գազաթի մուտքը և վերջնական գազաթի մուտքը,
- 2) օպերատորային գազաթին հաջորդող գազաթների մուտքերը նշվում են  $a_2, \dots, a_m$  սինվոլներով,
- 3) որևէ գազաթի մուտքը կարող է նշվել միայն մեկ սինվոլով,
- 4) տարբեր գազաթների մուտքեր նշվում են տարբեր սինվոլներով (վերջ գազաթը բացառություն է կազմում):

Նկ. 6.8-ում ցույց տրված ԱԳՍ-ն Ա1 ալգորիթմով նշելուց հետո կունենա նկ.6.15-ում ցույց է տրված տեսքը:

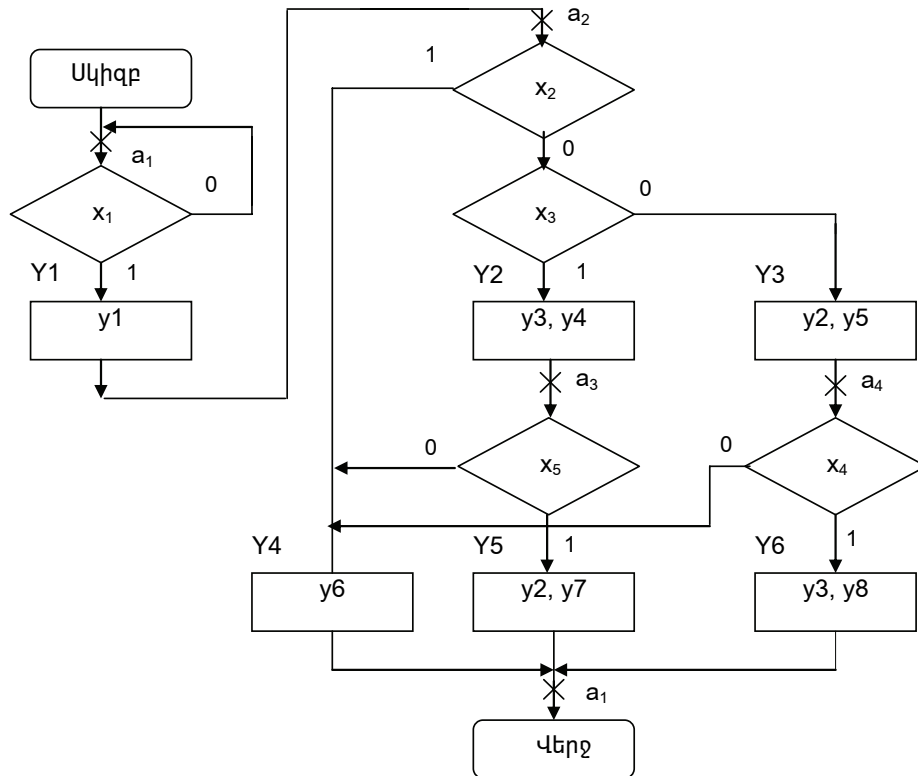
ԱԳՍ-ից ավտոմատի անցումների գրաֆին կան աղյուսակին անցնելիս ԱԳՍ-ի  $a_1, a_2, \dots, a_m$  նշանները համապատասխանում են ավտոմատի գրաֆի գազաթներին (ավտոմատի վիճակներին): Հաջորդ խնդիրը, որ պետք է լուծել ԱԳՍ-ից ավտոմատի գրաֆին անցնելու համար,  $a_1, a_2, \dots, a_m$  նշանների միջև ուղիների նկարագրությունն է, որը համապատասխանում է ավտոմատի անցումների նկարագրությանը:

Որևէ  $a_m$  նշանից  $a_s$  նշան տանող ուղին կարելի է նկարագրել հետևյալ ձևով՝

$$a_m X(a_m, a_s) Y(a_m, a_s) a_s, \quad (6.8)$$

որտեղ՝  $X(a_m, a_s)$  – տվյալ ուղու վրա գտնվող տրամաբանական գազաթներին համապատասխանող փոփոխականների կոնյունկցիան է, ընդ որում, եթե ուղին անցնում է  $x_i$  տրամաբանական գազաթից դուրս եկող 1 սլաքով, այդ կոնյունկցիայում  $x_i$ -ն վերցվում է ուղիղ (չբացասված) ձևով, իսկ եթե 0 սլաքով, ապա՝ բացասված ձևով:  $a_m$ -ից դեպի  $a_s$  ուղու վրա  $X(a_m, a_s)$  կոնյունկցիան ընդունում է 1 արժեք: Օրինակ, նկ. 6.15-ից

կարելի է գրել՝  $X(a_2, a_3) = \overline{x_2 x_3}$ ։  $Y(a_m, a_s)$ -ը  $a_m$  և  $a_s$  նշանները միացնող ուղու վրա գտնվող օպերատորային գազաթի պարունակությունն է (միկրոհրամանը), օրինակ՝  $Y(a_2, a_3) = \{y_3, y_4\}$ ։ Հետևելով (6.8)-ի գրառման ձևին՝ կունենանք՝  $a_2 X(a_2, a_3) Y(a_2, a_3) a_3 = a_2 \overline{x_2 x_3} \{y_3, y_4\} a_3$ ։



Նկ. 6.15. Նկ. 6.8-ում ցույց տրված ԱԳՍ-ն Ա1 ալգորիթմով նշելուց հետո

Եթե տվյալ ուղու վրա տրամաբանական գազաթ չկա, (6.8)-ում ընդունվում է  $X(a_m, a_s) = 1$ ։ Օրինակ, նկ. 6.16-ում պատկերված ԱԳՍ-ում  $X(a_3, a_4) = 1$ ։

Եթե որևէ ուղու վրա օպերատորային գազաթ չկա, ընդունվում է, որ կատարվում է դատարկ օպերատոր՝  $Y(a_m, a_s) = \emptyset$ , և ուղին նկարագրվում է հետևյալ կերպ՝

$$a_m X(a_m, a_s) a_s \quad (6.9)$$

Օրինակ, նկ. 6.16-ի ԱԳՍ-ում  $a_1$ -ից  $a_3$  տանող ուղին կնկարագրվի՝  $a_1 X(a_1, a_3) a_3 = a_1 x_1 a_3$ ։

Եթե ԱԳՍ-ում կան կոնտուրներ, որոնք ընդգրկում են միայն պայմանական գազաթներ (նկ. 6.17), այդ կոնտուր մտնող  $a_m$  նշանի համար կարելի է գրել՝

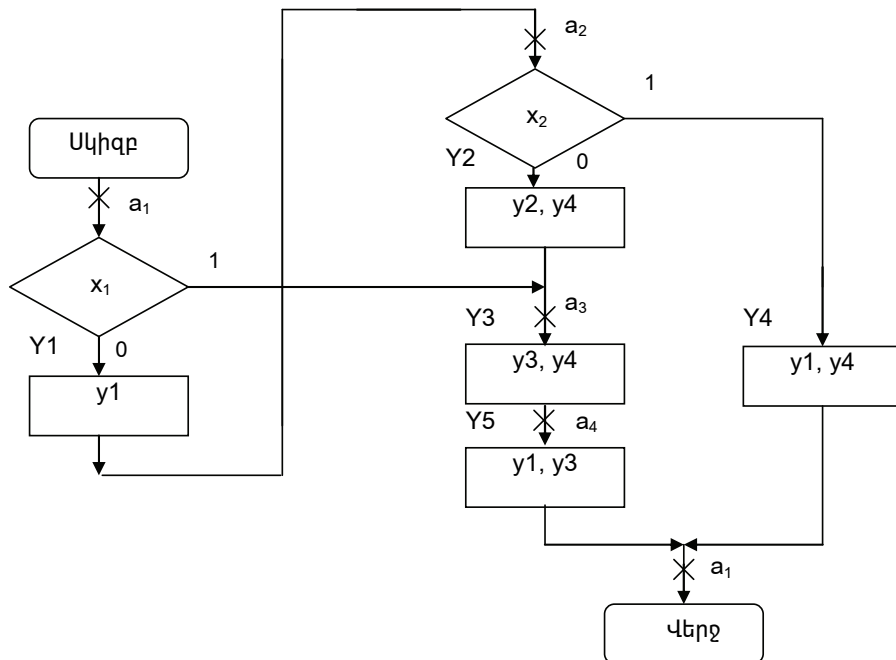
$$a_m x_{m_1}^{\sigma_1} \& x_{m_2}^{\sigma_2} \& \dots \& x_{m_k}^{\sigma_k} \& \dots, \quad (6.10)$$

որը կունենա անվերջ երկարություն։  $x_{m_k}$  -ն կոչվում է սպասող գազաթ։ Քանի դեռ  $x_{m_k} = \sigma_k$ , ոչ մի գործողություն չի կատարվում։ Այս դեպքում համարվում է, որ գոյություն ունի

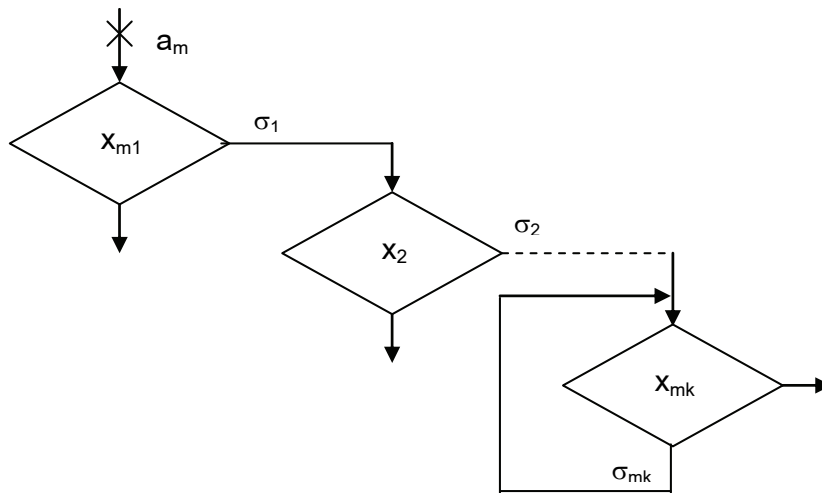


$$a_m x_{m_1}^{\sigma_1} \& x_{m_2}^{\sigma_2} \& \dots \& x_{m_k}^{\sigma_k} a_m \quad (6.11)$$

տեսքի ուղի: Օրինակ, նկ. 6.15-ի ԱԳՍ-ից կունենանք՝  $a_1 \bar{x}_1 a_1$  :



Նկ. 6.16. ԱԳՍ-ի օրինակ՝  $a_3$ -ից  $a_4$  ուղին տրամաբանական զագաթ չի պարունակում՝  
 $a_1 X(a_1, a_3) Y(a_1, a_3) a_3 = a_1 x_1 a_3$



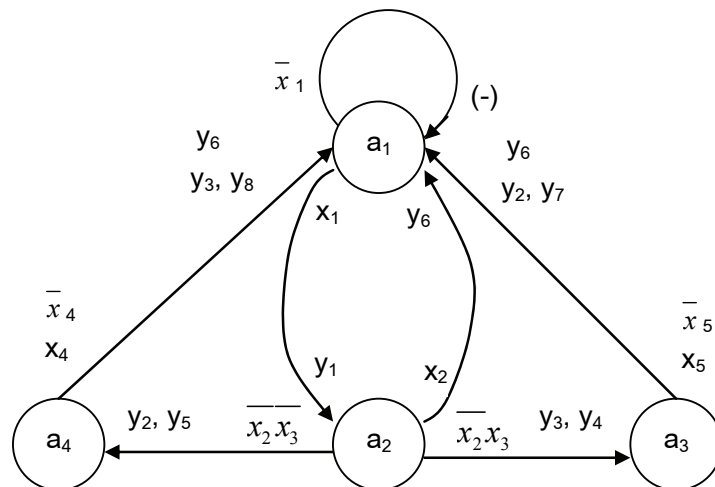
Նկ. 6.17. Միայն պայմանական զագաթներ ընդգրկող կոնտուր պարունակող ԱԳՍ-ի օրինակ

Նշված ԱԳՍ-ից համապատասխան ավտոմատի անցումների գրաֆը կառուցվում է հետևյալ եղանակով: ԱԳՍ-ի  $a_i$  նշանին համապատասխանության մեջ է դրվում գրաֆի մեկ զագաթ, որը նույնպես նշվում է  $a_i$ -ով: Եթե ԱԳՍ-ում գոյություն ունի  $a_m$  նշանից  $a_s$  նշանը տանող ուղի, ապա գրաֆում  $a_m$  և  $a_s$  զագաթները միացվում են  $a_m$ -ից  $a_s$  ուղղված աղեղով: Աղեղի սկզբի մոտ գրվում է այդ ուղղուն համապատասխանող  $X(a_m, a_s)$  կոն-

յունկցիան, իսկ ծայրի մոտ՝  $Y(a_m, a_s)$  միկրոհրամանը: Եթե ԱԳՍ-ում գոյություն ունեն  $a_m$ -ից  $a_s$  տանող մեկից ավելի ուղիներ (օրինակ, նկ. 6.15-ում  $a_3$ -ից  $a_1$  գոյություն ունեն երկու ուղիներ՝  $a_3 \times_5 Y_5 a_1$  և  $a_3 \overline{x_5} Y_4 a_1$ ), ապա միևնույնն է գրաֆի  $a_m$  և  $a_s$  գագաթները միացվում են միայն մեկ աղեղով, որի սկզբի մոտ՝ իրար տակ գրվում են ուղիներին համապատասխանող կոնյունկցիաները, իսկ ծայրի մոտ՝ նորից իրար տակ գրվում են համապատասխան միկրոհրամանները:

Նկ. 6.18-ում բերված է նկ. 6.15-ում ցույց տրված ԱԳՍ-ին համապատասխանող ավտոմատի անցումների գրաֆը, որտեղ (-)-ով նշված է դատարկ (չկատարվող) օպերատորը:

Մեծ թվով գագաթների և անցումների դեպքում միկրոծրագրային ավտոմատի նկարագրությունը գրաֆով կորցնում է ակնառությունը: Այդ պատճառով ավելի նպատակահարմար է ավտոմատը նկարագրել անցումների աղյուսակի միջոցով: ԱԳՍ-ից ստացվող անցումների աղյուսակը Միլի ավտոմատի համար տրվում է չորս սյուններով՝  $a_m$  և  $a_s$  - սկզբնական վիճակը և անցման վիճակը,  $X(a_m, a_s)$  -  $a_m$ -ից  $a_s$  ուղու տրամաբանական փոփոխականների կոնյունկցիան, որը տվյալ անցման դեպքում ընդունում է 1 արժեք,  $Y(a_m, a_s)$  ելքային փոփոխականները, որոնք ընդունում են 1 արժեք տվյալ անցման վրա: Աղյուսակի յուրաքանչյուր տող համապատասխանում է ԱԳՍ-ի մեկ ուղու: Աղյուսակ 6.4-ում բերված է նկ. 6.15-ում տրված ԱԳՍ-ին համապատասխանող Միլի ավտոմատի անցումների աղյուսակը:



Նկ. 6.18. Նկ. 6.15-ում ցույց տրված ԱԳՍ-ին համապատասխանող ավտոմատի անցումների գրաֆը

Միկրոծրագրային Մուրի ավտոմատի սինթեզի համար ԱԳՍ-ի նշումը  $a_1, a_2, \dots, a_m$  սիմվոլներով կատարվում է հետևյալ ալգորիթմով [6] (ալգորիթմ Ա2)՝

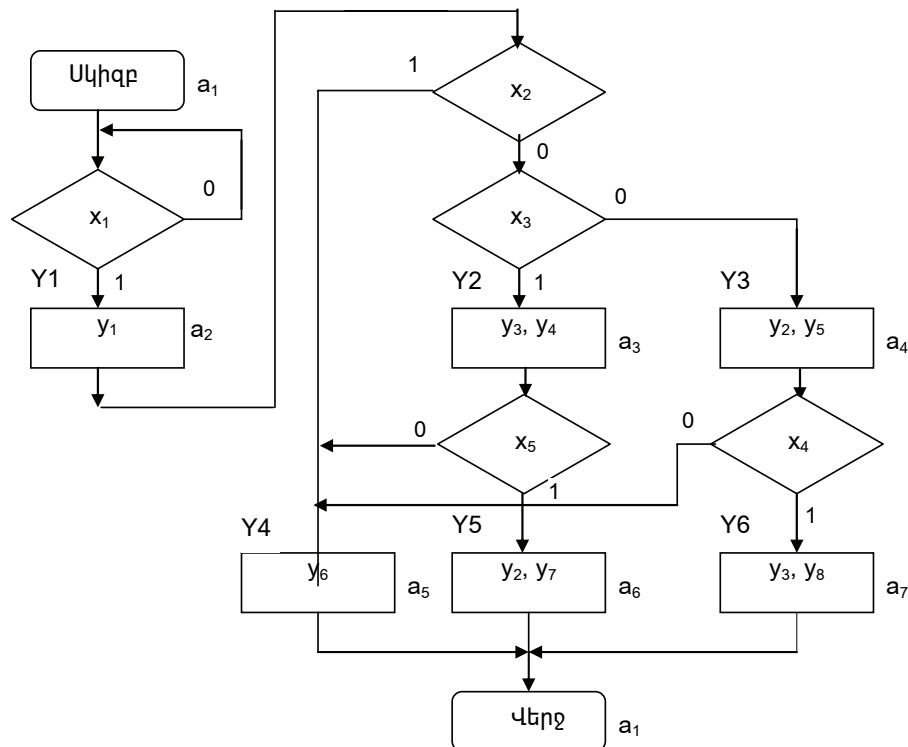
- 1)  $a_1$  սիմվոլով նշվում է սկիզբ և վերջ գագաթները,
- 2) տարբեր օպերատորային գագաթներ նշվում են տարբեր սիմվոլներով,
- 3) բոլոր օպերատորային գագաթները պետք է նշված լինեն:

Նկ. 6.19-ում ցույց է տրված նկ. 6.8-ի ԱԳՍ-ն նշված Ա2 ալգորիթմով:

Նշված ԱԳՍ-ից Մուրի ավտոմատի անցումների գրաֆն ստանալու համար ԱԳՍ-ի ուղիները որոշվում են (6.9) արտահայտության տեսքով՝  $a_m X(a_m, a_s) a_s$ : ԱԳՍ-ի յուրաքանչյուր նշանի համապատասխանության մեջ է դրվում ավտոմատային գրաֆի մեկ գագաթ: Եթե ԱԳՍ-ում գոյություն ունի  $a_m$ -ից  $a_s$  տանող ուղի, ապա գրաֆում այդ գագաթները միացվում են աղեղով: Աղեղի վրա գրվում է  $X(a_m, a_s)$  կոնյունկցիան: Գրաֆի որևէ  $a_i$  գագաթի մոտ գրվում է ԱԳՍ-ում  $a_i$ -ով նշված օպերատորային գագաթի պարունակություն՝  $Y_i = \{y_{i1}, \dots, y_{iq}\}$  միկրոհրամանը:

Աղյուսակ 6.4

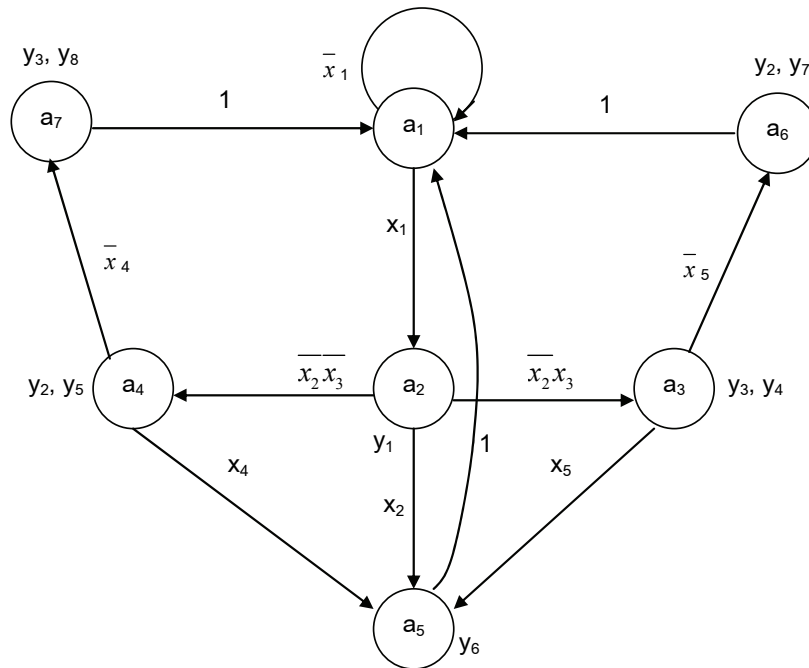
$a_m$	$a_s$	$X(a_m, a_s)$	$Y(a_m, a_s)$
$a_1$	$a_1$	$\bar{x}_1$	-
	$a_2$	$x_1$	$y_1$
$a_2$	$a_1$	$\bar{x}_2$	$y_6$
	$a_3$	$\overline{x_2 x_3}$	$y_3, y_4$
	$a_4$	$\overline{x_2 x_3}$	$y_2, y_5$
$a_3$	$a_1$	$\bar{x}_5$	$y_6$
		$x_5$	$y_2, y_7$
$a_4$	$a_1$	$\bar{x}_4$	$y_6$
		$x_4$	$y_3, y_8$



Նկ. 6.19. Ա2 ալգորիթմով (Մուրի ավտոմատ) նշված նկ. 6.8-ի ԱԳՍ-ը

Թույլատրվում է գրել դատարկ միկրոհրաման՝  $Y_i = \emptyset$ : Եթե ԱԳՍ-ում գոյություն ունեն  $a_m$ -ից

$a_s$  տանող մի քանի ուղիներ, միևնույնն է՝ այդ գագաթները միացվում են մեկ աղեղով, իսկ համապատասխան կոնյունկցիաները աղեղի վրա գրվում են իրար տակ: Նկ. 6.20-ում ցույց է տրված նկ. 6.19-ի ԱԳՍ-ին համապատասխանող ավտոմատային գրաֆը:



Նկ. 6.20. Նկ. 6.19-ում ցույց տրված Մուրի ավտոմատի ԱԳՍ-ին համապատասխանող ավտոմատային գրաֆ

Մուրի ավտոմատի անցումների աղյուսակն ունի երեք սյունակ, ընդ որում՝  $Y(a_m)$  բազմության ելքային ազդանշանները գրվում են  $a_m$  վիճակի հետ նույն սյունակում: Աղյուսակ 6.5-ում բերված է նկ. 6.19-ում տրված ԱԳՍ-ին համապատասխանող Մուրի ավտոմատի անցումների աղյուսակը:

#### 6.4. Միկրոծրագրային ավտոմատի իրականացում

Միկրոծրագրային ավտոմատի կառուցման համար անհրաժեշտ է նախ կոդավորել ավտոմատի վիճակները, ապա ստանալ տրիգերների մուտքի ու ավտոմատի ելքի ֆունկցիաները: Դրա համար հարմար է նախ կառուցել միկրոծրագրային ավտոմատի կառուցվածքային աղյուսակը, որն ստացվում է՝ անցումների աղյուսակին ավելացնելով վիճակների կոդերի սյունակներն ու տրիգերների մուտքերի սյունակը: Տրիգերների մուտքերի սյունակում նշվում է այն մուտքի անվանումը, որը տվյալ տողին համապատասխանող անցման վրա պետք է ընդունի 1 արժեք: Տրիգերների մուտքերի անվանումները պետք է գրել նույն կարգով, ինչ որ վիճակի փոփոխականները:

Օրինակ, աղյուսակ 6.4-ով նկարագրվող Միլի ավտոմատն ունի 4 վիճակ, հետևաբար նրա վիճակները կարելի է կոդավորել երկու բիթերով՝  $Q_2Q_1$ : Ավտոմատի վիճակի փոփոխականների պահպանման համար անհրաժեշտ է ունենալ երկու տրիգեր:

Աղյուսակ 6.5

$a_m Y(a_m)$	$a_s$	$X(a_m, a_s)$
$a_1$	$a_1$	$\overline{x_1}$
	$a_2$	$x_1$
$a_2, y_1$	$a_3$	$\overline{x_2 x_3}$
	$a_4$	$\overline{\overline{x_2 x_3}}$
	$a_5$	$x_5$
$a_3, y_3 y_4$	$a_5$	$x_5$
	$a_6$	$\overline{x_5}$
$a_4, y_2 y_5$	$a_5$	$x_4$
	$a_7$	$\overline{x_4}$
$a_5, y_6$	$a_1$	1
$a_6, y_2 y_7$	$a_1$	1
$a_7, y_3 y_8$	$a_1$	1

Աղյուսակ 6.6-ում բերված է աղյուսակ 6.4-ով ներկայացված Միլի ավտոմատի կառուցվածքային աղյուսակը, որտեղ ենթադրվում է, որ օգտագործվում են D տրիգերներ: Տրիգերների մուտքի ֆունկցիաները՝  $D_2, D_1$ , որոշվում են բնութագրիչ աղյուսակից՝  $D = Q^+$ :

Կառուցվածքային աղյուսակից կարելի է դուրս գրել ավտոմատի ելքերի և տրիգերների մուտքերի բուլյան ֆունկցիաների բանձները դիզյունկտիվ նորմալ տեսքով, որի համար նույն ֆունկցիայի բոլոր տողերին համապատասխանող կոնյունկցիաները միացվում են ԿԱՄ գործողությամբ: Աղյուսակ 6.6-ից կստացվի՝

Աղյուսակ 6.6

Տողի #	$a_m$	$K(a_m) = Q_2 Q_1$	$a_s$	$K(a_s) = Q_2^+ Q_1^+$	$X(a_m, a_s)$	$Y(a_m, a_s)$	$I(a_m, a_s)$
1	$a_1$	00	$a_1$	00	$\overline{x_1}$	-	-
2			$a_2$	01	$x_1$	$y_1$	$D_1$
3	$a_2$	01	$a_1$	00	$x_2$	$y_6$	-
4			$a_3$	10	$\overline{x_2 x_3}$	$y_3, y_4$	$D_2$
5			$a_4$	11	$\overline{\overline{x_2 x_3}}$	$y_2, y_5$	$D_2 D_1$
6	$a_3$	10	$a_1$	00	$\overline{x_5}$	$y_6$	-
7					$x_5$	$y_2, y_7$	-
8	$a_4$	11	$a_1$	00	$\overline{x_4}$	$y_6$	-
9					$x_4$	$y_3, y_8$	-

$$\begin{aligned}
y_1 &= \overline{Q_2} \overline{Q_1} x_1, \\
y_2 &= \overline{Q_2} Q_1 \overline{x_2} \overline{x_3} + Q_2 \overline{Q_1} x_5, \\
y_3 &= \overline{Q_2} Q_1 \overline{x_2} x_3 + Q_2 Q_1 x_4, \\
y_4 &= \overline{Q_2} Q_1 \overline{x_2} x_3,
\end{aligned}
\tag{6.12}$$

$$\begin{aligned}
y_5 &= \overline{Q_2} Q_1 \overline{x_2} \overline{x_3}, \\
y_6 &= \overline{Q_2} Q_1 x_2 + Q_2 \overline{Q_1} \overline{x_5} + Q_2 Q_1 \overline{x_4},
\end{aligned}
\tag{6.13}$$

$$\begin{aligned}
y_7 &= Q_2 \overline{Q_1} x_5, \\
y_8 &= Q_2 Q_1 x_4,
\end{aligned}
\tag{6.14}$$

$$\begin{aligned}
D_1 &= \overline{Q_2} \overline{Q_1} x_1 + \overline{Q_2} Q_1 \overline{x_2} \overline{x_3}, \\
D_2 &= \overline{Q_2} Q_1 \overline{x_2} x_3 + \overline{Q_2} Q_1 \overline{x_2} \overline{x_3} :
\end{aligned}
\tag{6.15}$$

Գործնականում հանդիպող ավտոմատներում այդ բանաձևերը չափազանց բարդ տեսքի են, դրանք տասնյակ կամ հարյուրավոր ֆունկցիաներ են՝ կախված տասնյակ կամ հարյուրավոր փոփոխականներից: Նման դեպքերում բուլյան ֆունկցիաների այդպիսի համակարգերի պարզեցման համար դասական նվազարկման եղանակների կիրառությունը կարող է արդյունավետ չլինել: Փորձը ցույց է տալիս, որ ֆունկցիաների համակարգի զգալի պարզեցման կարելի հասնել համակարգում կրկնվող ֆունկցիաների կամ ենթաֆունկցիաների դասերի առանձնացմամբ և դրանց համատեղ նվազարկմամբ: Միկրոծրագրային ավտոմատի նվազեցված տրամաբանական շղթաների կառուցման պարզագույն եղանակը աղյուսակի մույն տողում գրված ելքերի և տրիգերների մուտքերի ֆունկցիաների համատեղ իրականացումն է: Այսպես, աղյուսակ 6.6-ի 5-րդ տողի  $\overline{Q_2} Q_1 \overline{x_2} \overline{x_3}$  կոնյունկցիան կարող է օգտագործվել  $y_2$ ,  $y_5$ ,  $D_2$ ,  $D_1$  ֆունկցիաների կառուցման ժամանակ: Հեշտ է նակտել, որ  $D_1 = y_1 + y_5$ ,  $D_2 = y_4 + y_5$ :

Աղյուսակ 6.5-ով նկարագրվող Մուրի ավտոմատն ունի 7 վիճակ, հետևաբար նրա վիճակները կարելի է կոդավորել երեք բիթով՝  $Q_3 Q_2 Q_1$ : Աղյուսակ 6.7-ում բերված է աղյուսակ 6.5-ով ներկայացված Մուրի ավտոմատի կառուցվածքային աղյուսակը:

Նույն եղանակով աղյուսակ 6.7-ից կարելի դուրս գրել ավտոմատի ելքի և տրիգերների մուտքի ֆունկցիաները: Նկատենք, որ Մուրի ավտոմատի  $Y(a_m)$  ելքի ֆունկցիաները որոշվում են միայն  $K(a_m)$  վիճակի փոփոխականներով:

Աղյուսակ 6.7

Տողի #	$a_m Y(a_m)$	$K(a_m) = Q_3 Q_2 Q_1$	$a_s$	$K(a_s) = Q_3^+ Q_2^+ Q_1^+$	$X(a_m, a_s)$	$I(a_m, a_s)$
1	$a_1$	000	$a_1$	000	$\bar{x}_1$	-
2			$a_2$	001	$x_1$	$D_1$
3	$a_2, y_1$	001	$a_3$	010	$\bar{x}_2 x_3$	$D_2$
4			$a_4$	011	$\bar{x}_2 \bar{x}_3$	$D_2 D_1$
5			$a_5$	100		$D_3$
6	$a_3, y_3 y_4$	010	$a_5$	100	$x_5$	$D_3$
7			$a_6$	101	$\bar{x}_5$	$D_3 D_1$
8	$a_4, y_2 y_5$	011	$a_5$	100	$x_4$	$D_3$
9			$a_7$	110	$\bar{x}_4$	$D_3 D_2$
10	$a_5, y_6$	100	$a_1$	000	1	-
11	$a_6, y_2 y_7$	101	$a_1$	000	1	-
12	$a_7, y_3 y_8$	110	$a_1$	000	1	-

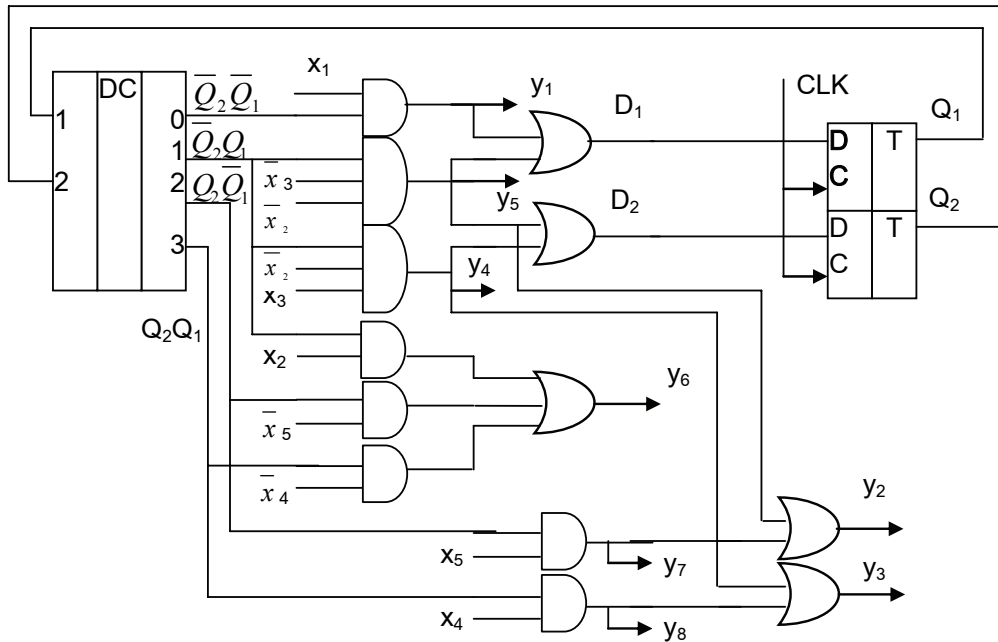
$$\begin{aligned} y_1 &= \bar{Q}_3 \bar{Q}_2 Q_1, \\ y_2 &= \bar{Q}_3 Q_2 Q_1 + Q_3 \bar{Q}_2 Q_1, \end{aligned} \quad (6.17)$$

$$\begin{aligned} y_3 &= \bar{Q}_3 Q_2 \bar{Q}_1 + Q_3 Q_2 \bar{Q}_1, \\ y_4 &= \bar{Q}_3 Q_2 \bar{Q}_1, \end{aligned} \quad (6.18)$$

$$\begin{aligned} y_5 &= \bar{Q}_3 Q_2 Q_1, \\ y_6 &= Q_3 \bar{Q}_2 \bar{Q}_1, \end{aligned} \quad (6.19)$$

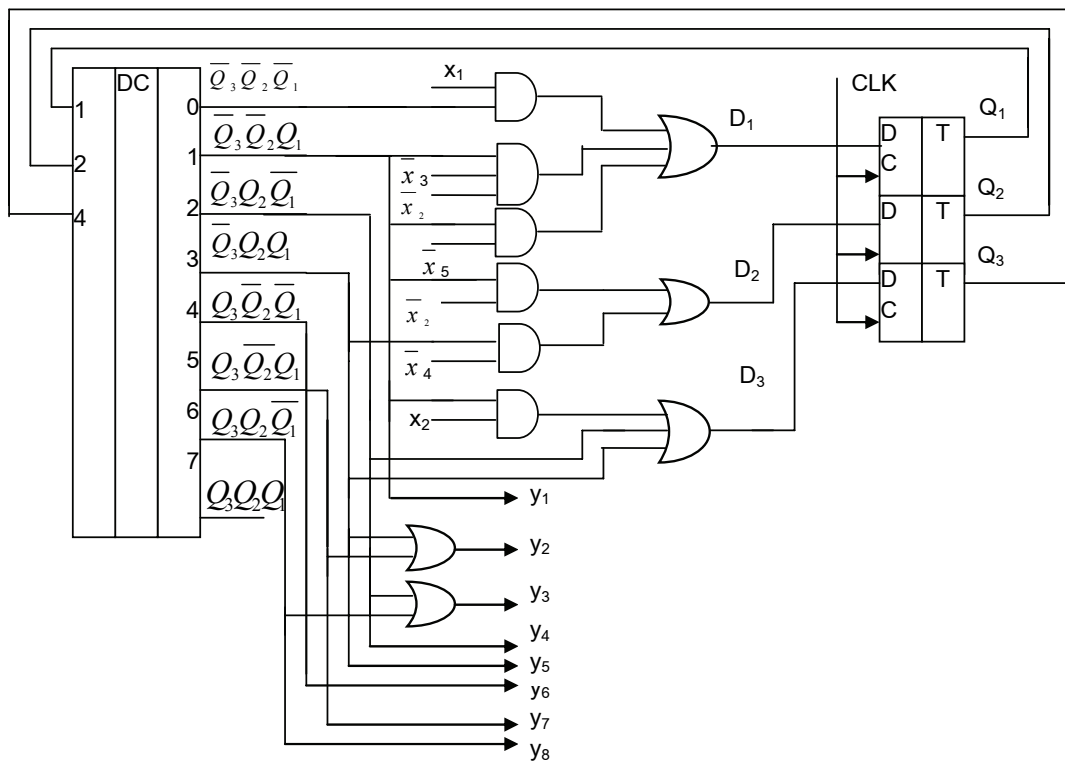
$$\begin{aligned} y_7 &= Q_3 \bar{Q}_2 Q_1, \\ y_8 &= Q_3 Q_2 \bar{Q}_1, \end{aligned} \quad (6.20)$$

$$\begin{aligned} D_1 &= \bar{Q}_3 \bar{Q}_2 \bar{Q}_1 x_1 + \bar{Q}_3 \bar{Q}_2 Q_1 \bar{x}_2 \bar{x}_3 + \bar{Q}_3 Q_2 \bar{Q}_1 \bar{x}_5, \\ D_2 &= \bar{Q}_3 \bar{Q}_2 Q_1 \bar{x}_2 x_3 + \bar{Q}_3 \bar{Q}_2 Q_1 \bar{x}_2 \bar{x}_3 + \bar{Q}_3 Q_2 Q_1 \bar{x}_4 = \bar{Q}_3 \bar{Q}_2 Q_1 \bar{x}_2 + \bar{Q}_3 Q_2 Q_1 \bar{x}_4, \\ D_3 &= \bar{Q}_3 \bar{Q}_2 Q_1 x_2 + \bar{Q}_3 Q_2 \bar{Q}_1 x_5 + \bar{Q}_3 Q_2 \bar{Q}_1 \bar{x}_5 + \bar{Q}_3 Q_2 Q_1 x_4 + \bar{Q}_3 Q_2 Q_1 \bar{x}_4 = \bar{Q}_3 \bar{Q}_2 Q_1 x_2 + \bar{Q}_3 Q_2 \bar{Q}_1 + \bar{Q}_3 Q_2 Q_1 : \end{aligned} \quad (6.21)$$



Նկ. 6.21. Միլի ալտոմատ՝ կառուցված (6.12)-(6.16) հավասարումներով

Համապատասխան սխեմաները բերված են նկ. 6.21-ում և 6.22-ում: Քանի որ վիճակի փոփոխականների համակցությունները հաճախ են հանդիպում (6.12)- (6.21) ֆունկցիաներում, ապա հարմար է դրանք ստանալ վերծանիչների միջոցով:



Նկ. 6.22. Մուրի ալտոմատ՝ կառուցված (6.17)-(6.21) հավասարումներով



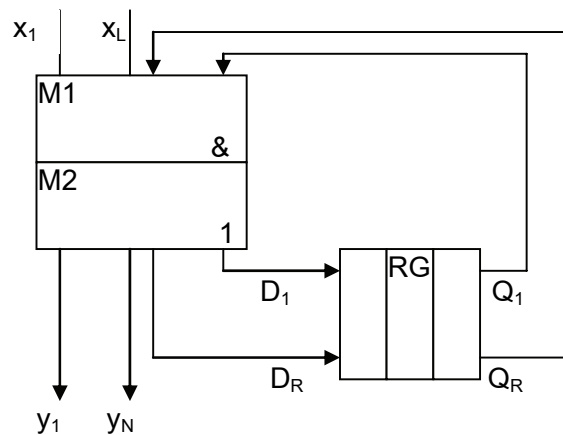
### 6.5. Միկրոծրագրային ավտոմատների կառուցումը ծրագրավորվող միկրոսխեմաներով

Միկրոծրագրային ավտոմատը կարող է ուղղակի եղանակով կառուցվել ծրագրավորվող տրամաբանական մատրիցների՝  $\overline{OSU}$ , և արտաքինից միացվող զուգահեռ ռեգիստրի միջոցով (նկ. 6.23):

Ավտոմատի համակցական մասը, որը տրվում է ելքերի և տրիգերների մուտքի ֆունկցիաների միջոցով իրականացվում է մեկ տրամաբանական մատրիցով, եթե

$$L + R \leq s, N + R \leq t, B \leq q, \quad (6.22)$$

որտեղ՝  $L$ ,  $N$ ,  $R$  - կառուցվող ավտոմատի մուտքերի, ելքերի և վիճակի փոփոխականների (տրիգերների) թվերն են,  $B$  – կոնյունկցիաների թիվն է՝ կառուցվածքային աղյուսակի տողերի թիվը, իսկ  $s$ ,  $t$ ,  $q$ –ն տրամաբանական մատրիցի մուտքի գծերի, ելքի գծերի և կոնյունկցիայի գծերի թվերն են՝  $(s, t, q)$ - մատրից:



Նկ. 6.23. Միկրոծրագրային ավտոմատի սխեման ծրագրավորվող տրամաբանական մատրիցի և զուգահեռ ռեգիստրի միջոցով

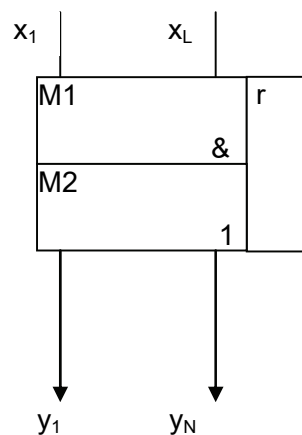
Ծրագրավորվող ԻՍ-ի  $M1$  մատրիցում ձևավորվում են կառուցվածքային աղյուսակի տողերին համապատասխանող կոնյունկցիաները, իսկ  $M2$ -ում՝ ավտոմատի ելքային և տրիգերների մուտքերի ֆունկցիաները՝  $M1$ -ի ելքային գծերի (կոնյունկցիաների) դիզյունկցիաների միջոցով: Աղյուսակ 6.4-ով տրված Միլի ավտոմատն ունի հետևյալ պարամետրերը՝  $L=5$ ,  $R=2$ ,  $N=8$ ,  $B=9$ : Ընդ որում, առաջին տողին համապատասխանող կոնյունկցիան չի մասնակցում ավտոմատի ելքային և տրիգերների մուտքի ֆունկցիաներում, ուստի այն կարելի է չձևավորել: Այսպիսով, աղյուսակ 6.4-ով տրված Միլի ավտոմատն իրականացնելու համար անհրաժեշտ է ունենալ երկու տրիգեր (երկկարգ ռեգիստր) և ծրագրավորվող մատրից հետևյալ պարամետրերով՝  $s \geq 5+2=7$ ,  $t \geq 8+2=10$ ,  $q \geq 8$ : Աղյուսակ 6.8-ում բերված է ծրագրավորվող մատրիցի ծրագրավորման աղյուսակը:

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$Q_1$	$Q_2$	$y_1$	$y_2$	$y_3$	$y_4$	$y_5$	$y_6$	$y_7$	$y_8$	$D_1$	$D_2$
1	-	-	-	-	0	0	1	-	-	-	-	-	-	-	1	-
-	1	-	-	-	0	1	-	-	-	-	-	1	-	-	-	-
-	0	1	-	-	0	1	-	-	1	1	-	-	-	-	-	1
-	0	0	-	-	0	1	-	1	-	-	1	-	-	-	1	1
-	-	-	-	0	1	0	-	-	-	-	-	1	-	-	-	-
-	-	-	-	1	1	0	-	1	-	-	-	-	1	-	-	-
-	-	-	0	-	1	1	-	-	-	-	-	1	-	-	-	-
-	-	-	1	-	1	1	-	-	1	-	-	-	-	1	-	-

Հիշողությամբ ԾՏՍ-ն կամ ծրագրավորվող տրամաբանական սարքը՝ ԾՏՍ, ԻՍ-ի վրա պարունակում է նաև հիշողության ռեգիստր, որը հնարավորություն է տալիս ավտոմատն ամբողջությամբ իրագործել մեկ ԻՍ-ի միջոցով: ԾՏՍ-ն ունի նկ. 6.24-ում ցույց տրված կառուցվածքը՝ բաղկացած է M1, M2 մատրիցներից և RG ռեգիստրից: ԾՏՍ-ն բնութագրվում է  $(s, t, q, r)$  քառյակով: Ավտոմատը կարելի է ուղղակի իրագործել մեկ ԾՏՍ-ի ԻՍ-ի միջոցով, եթե

$$L \leq s, N \leq t, B \leq q, R \leq r, \quad (6.23)$$

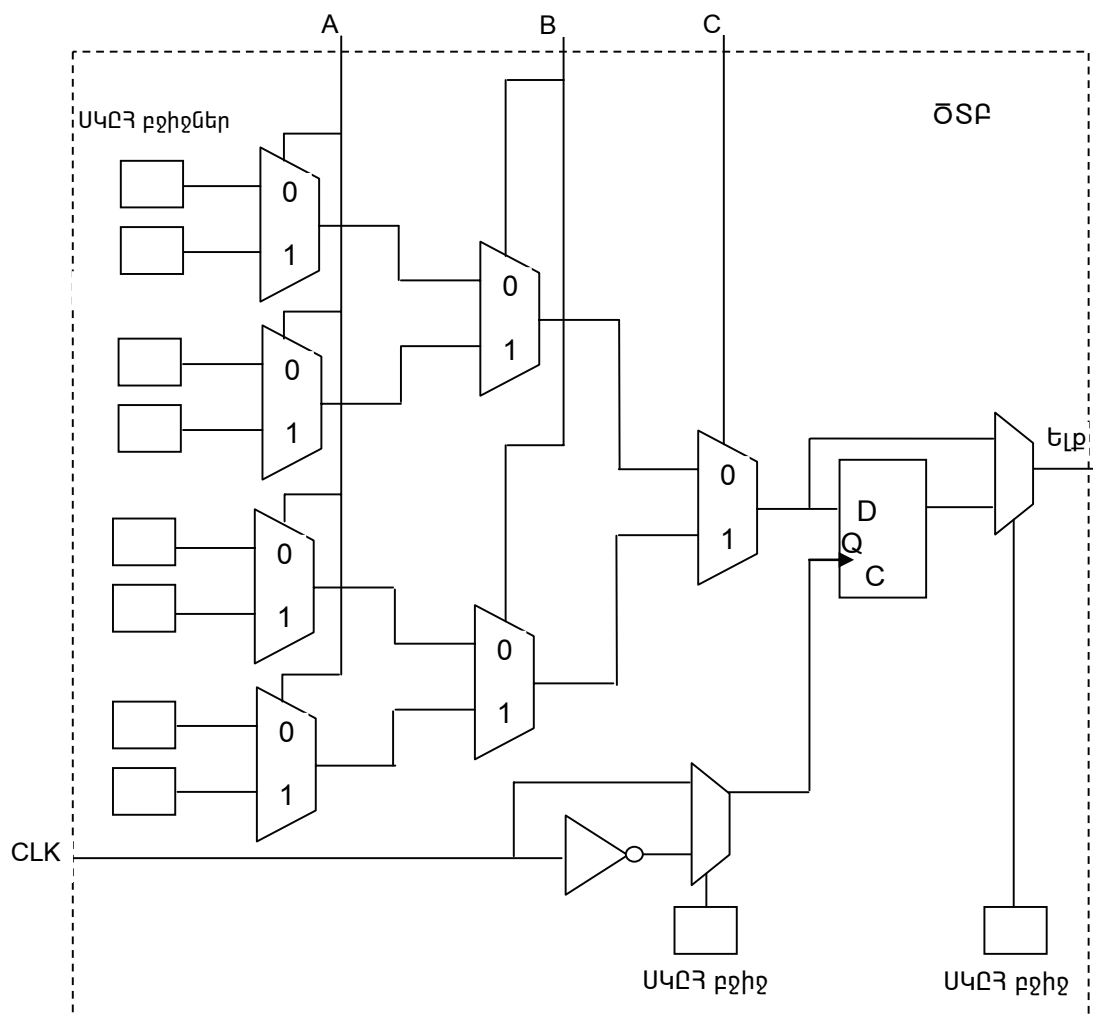
որտեղ  $r$ -ը ԾՏՍ-ի ԻՍ-ի ռեգիստրի կարգաթիվն է: ԾՏՍ-ի ծրագրավորման աղյուսակն ունի նույն տեսքը, ինչ ԾՏՍ-ինը:



Նկ. 6.24. Հիշողությամբ ծրագրավորվող տրամաբանական մատրից

Թվային համակարգերի կառուցման համար մեծ հնարավորություններ են ընձեռնում ԿԾՓԶ ծրագրավորվող ԻՍ-երը, որոնք պարունակում են մեծ թվով ծրագրավորվող տրամաբանական բլոկներ (ԾՏԲ): Տրամաբանական բլոկներից յուրաքանչյուրը բաղկացած է ընթերցվող տրամաբանական աղյուսակներից (LUT) և տրիգերներից, ինչը թույլ է տալիս իրականացնել վերջավոր ավտոմատներ: Այդպիսի մեկ բլոկի սխեմատիկ պատկերը ցույց է տրված նկ. 6.25-ում: Նկարում ցույց տրված տարբերա-

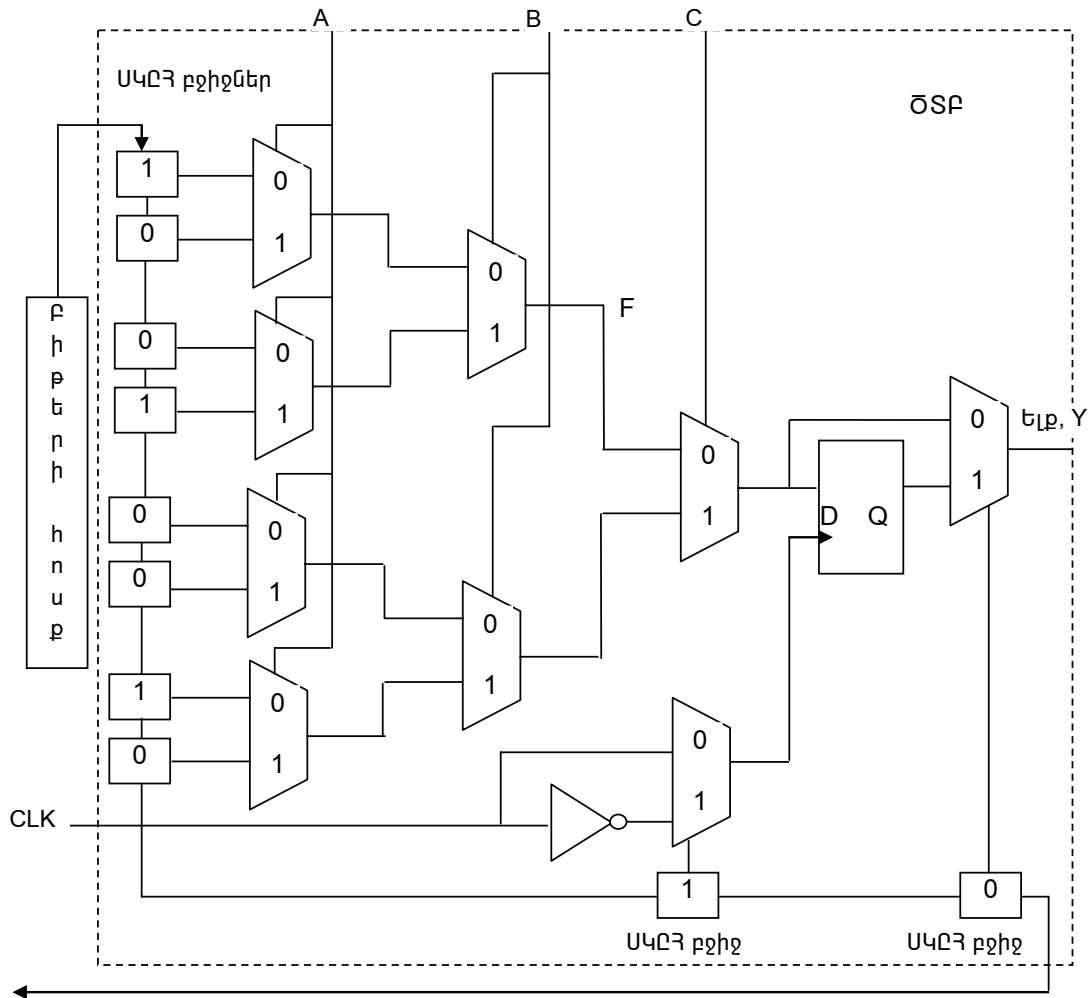
կում իրականացվում է երեք փոփոխականի (A,B,C) ֆունկցիա, որի իսկության աղյուսակն ունի  $2^3$  տող: Այդ կառուցվածքը կոչվում է 3-մուտք աղյուսակ (3-LUT): Սովորաբար, ԿԾՓԶ-երում օգտագործվում են ավելի շատ մուտքերով աղյուսակներ՝ տրամաբանական ֆունկցիաների իրականացման համար: ԾՏԲ-ի ստատիկ կամայական ընտրությամբ հիշողության (ՍԿԸՀ) բջիջները ծրագրավորում են բլոկի տրամաբանական ֆունկցիան և կառուցվածքը: Ելքի մուլտիպլեքսորի ընտրության մուտքը ծրագրավորող բիթի միջոցով ԾՏԲ-ի ելքը կարելի վերցնել մուլտիպլեքսորից, դրանով իսկ իրականացնելով համակցական տրամաբանական բլոկ, կամ տրիգերի ելքից՝ իրականացնելով հաջորդական տրամաբանական բլոկ: Տակտային ազդանշանի ուղու վրա գտնվող մուլտիպլեքսորի ընտրության մուտքը ծրագրավորող բիթի միջոցով կարելի տվյալների սինքրոնացումն իրականացնել տակտային ազդանշանի աճող կամ ընկնող ճակատով:



Նկ. 6.25. Ծրագրավորվող տրամաբանական բլոկ

Օգտագործողի կողմից ԿԾՓԶ-ի ծրագրավորումն իրականացվում է ՍԿԸՀ բջիջներում գրանցվող բիթերի հաջորդական ներմուծումով, ինչպես ցույց է տրված Նկ. 6.26-ում: Այդ նպատակով ծրագրավորման ժամանակ ՍԿԸՀ բջիջները պետք է

դասավորվեն հաջորդական շղթայով՝ տեղաշարժող ռեգիստրի կառուցվածքով: Ծրագրավորման ժամանակի կրճատման նպատակով կարելի է կառուցել մի քանի հաջորդական շղթաներ, որոնք կարելի է բեռնավորել միաժամանակ:



Նկ. 6.26. ՎԾՓԶ-ի տրամաբանական բլոկի ծրագրավորում

Երբ բիթերի արժեքները արդեն բեռնված են աղյուսակում, դրանց կարելի է դիմել A,B,C մուլտիբրից: Յուրաքանչյուր A,B,C հավաքածու ակտիվացնում է 8:1 մուլտիպլեքսորների ծառի մեկ ուղի: Օրինակ,  $A=B=C=0$  հավաքածուի դեպքում բլոկի ելք կփոխանցվի աղյուսակի ամենավերևի բջջի պարունակությունը՝ 1,  $A=1, B=C=0$  հավաքածուի դեպքում՝ հաջորդ բջջի պարունակությունը՝ 0: Դիտարկվող ծրագրավորման օրինակում ԾԼԲ-ի ելք է տրվում ամիջապես 8:1 մուլտիպլեքսորի ելքը, քանի որ ելքի ընտրության բիթն ունի 0 արժեք: Այս օրինակում իրագործվում է  $Y = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C}$  ֆունկցիան:

ՎԾՓԶ ԻՍ-ով թվային համակարգերի նախագծումը կատարվում է ավտոմատացված նախագծման ծրագրային գործիքներով: Նախագծի սկզբնական նկարագրությունը տրվում է սխեմաների նկարագրության լեզուներով (ՍՆԼ, HDL): ՍՆԼ կոդի

մշակումից հետո ամբողջ նախագծման ընթացքը ավտոմատացված է և կարող է իրագործվել մի քանի ժամվա ընթացքում: ԿԾՓՁ ԻՍ-երը սովորաբար ունեն բարձր արժեք և հազվադեպ են կիրառվում սերիալային թողարկվող սարքերում: Դրանք հիմնականում օգտագործվում են թվային համակարգերի նախատիպերի արագ կառուցման և փորձարկման համար:

Պետք է նկատել նաև, որ ծրագրավորվող ԻՍ-երի վրա կառուցված թվային համակարգը սովորաբար ունի սխեմատեխնիկական մեծ ավելցուկություն և որպես հետևանք՝ ավելի ցածր արագագործություն, քան տրամաբանական բջիջների վրա կառուցման դեպքում:

## 6.6. Խնդիրներ

**Խնդիր. 6.1.** Նկ. 6.9-ի պարզեցված սխեման լրացնել կառավարող ավտոմատից տրվող  $y_i$  կառավարող ազդանշաններով և CLK տակտային ազդանշանով: Նկ. 10-ի ԱԳՍ-ին ավելացնել  $y_i$  միկրոգործողությունները:

**Խնդիր. 6.2.** Նշել նկ. 6.3-ում բերված ԱԳՍ-ն Ա1 ալգորիթմով (Միլի ավտոմատ) և կառուցել համապատասխան անցումների գրաֆը:

**Խնդիր. 6.3.** Նշել նկ. 6.3-ում բերված ԱԳՍ-ն Ա2 ալգորիթմով (Մուրի ավտոմատ) և կառուցել համապատասխան անցումների գրաֆը:

**Խնդիր. 6.4.** Նշել նկ. 6.3-ում բերված ԱԳՍ-ն Ա1 ալգորիթմով (Միլի ավտոմատ) և ստանալ ավտոմատի կառուցվածքային աղյուսակը: Կազմել ավտոմատի ելքերի և տրիգերների մուտքերի ֆունկցիաները, եթե օգտագործվում են D տրիգերներ:

**Խնդիր. 6.5.** Նշել նկ. 6.3-ում բերված ԱԳՍ-ն Ա2 ալգորիթմով (Մուրի ավտոմատ) և ստանալ ավտոմատի կառուցվածքային աղյուսակը: Կազմել ավտոմատի ելքերի և տրիգերների մուտքերի ֆունկցիաները, եթե օգտագործվում են D տրիգերներ:

**Խնդիր. 6.6.** Սինթեզել Միլի ավտոմատ ըստ Խնդիր 6.3-ում ստացված ԱԳՍ-ի: Ստանալ ավտոմատի իրականացման համար հիշողությամբ ԾՏՄ-ի ծրագրավորման աղյուսակ:

**Խնդիր. 6.7.** Սինթեզել Մուրի ավտոմատ ըստ Խնդիր 6.3-ում ստացված ԱԳՍ-ի: Կառուցել կառավարող ավտոմատի սխեման՝ օգտագործելով D տրիգերներ, տրամաբանական տարրեր և վերծանիչ:

**Խնդիր. 6.8.** Կառուցել երկուական թվերի բազմապատկիչի լրիվ սխեման, երբ  $n=4$ : Կառավարող ավտոմատը վերցնել Խնդիր 6.7-ից, իսկ կատարողական ավտոմատը կառուցել ըստ նկ. 6.9-ի՝ ընտրելով համապատասխան հանգույցների համար անհրաժեշտ թվային միկրոսխեմաներ՝ հաշվիչ, ռեգիստրներ, գումարիչ:

**Խնդիր 6.9.** Նշել նկ. 6.14-ում բերված ԱԳՍ-ն Ա2 ալգորիթմով (Մուրի ավտոմատ) և ստանալ ավտոմատի կառուցվածքային աղյուսակը, երբ  $n=8$ : Կազմել ավտոմատի ելքերի և տրիգերների մուտքերի ֆունկցիաները, եթե օգտագործվում են D տրիգերներ:

**Խնդիր 6.10.** Կառուցել տեղաշարժերի և հանումների ալգորիթմով աշխատող 8-բիթ երկուական թվերի բաժանիչի կառավարման սխեման, որը համապատասխանում է նկ. 6.14-ի ԱԳՍ-ին:

**Խնդիր 6.11.** Ձևափոխել նկ. 6.10-ի ԱԳՍ-ն այնպես, որ բազմապատկման համար անհրաժեշտ տակտերի թիվը կախված չլինի բազմապատկիչի արժեքից:

## ԲԱԺԻՆ ԵՐՐՈՐԴ

### Թվային համակարգերի ավտոմատացված նախագծում

#### Գլուխ 7. Թվային համակարգերի նկարագրություն Վերիլոգ լեզվով

##### 7.1. Ներածություն

Թվային սխեմաների նկարագրության լեզուները (ՍՆԼ, Hardware Description Languages, HDL) նախատեսված են նախագծվող թվային համակարգերի աշխատանքի ալգորիթմի կամ տրամաբանության նկարագրության և աշխատանքի մոդելավորման համար: Դրանք իրենց բնույթով ծրագրավորման լեզուներ են: Սակայն ՍՆԼ-ները ծրագրային միջոցների մշակման լեզուներից (օրինակ, C++) տարբերվում են մի քանի հատկություններով, որոնցից հիմնականն այն է, որ ՍՆԼ-ները հնարավորություն են տալիս մոդելավորել սխեմաներում միաժամանակ ընթացող պրոցեսները: Ավտոմատացված էլեկտրոնային նախագծման ոլորտում առավել տարածված են սխեմաների նկարագրության VHDL (Very High Speed Hardware Description Language) և Վերիլոգ (Verilog) լեզուները: Թվային համակարգի աշխատանքի տրամաբանության և մոդելավորման տեսանկյունից VHDL և Վերիլոգ լեզուները համարժեք են: Ավտոմատացված նախագծման գործընթացներում Վերիլոգն ավելի տարածված է՝ հիմնականում սխեմաների ավելի հարմար և կոմպակտ նկարագրության շնորհիվ: Վերիլոգ լեզուն նման է ծրագրավորման C լեզվին, սակայն այն ավելի ընկալելի է սխեմատեխնիկական նախագծողների կողմից:

Վերիլոգ լեզուն մշակվել է Ֆիլ Մուրբիի (Phil Moorby) կողմից անցյալ դարի 80-ական թվականների վերջում: Հետագայում (1995-ին) այն դարձավ IEEE (Institute of Electrical and Electronic Engineering) ստանդարտ [11]: Ժամանակի ընթացքում Verilog լեզուն զարգացել է, և ստանդարտը վերաթողարկվել է 2001 և 2005 թթ. [12,13]:

Նախնական փուլում ՍՆԼ-ներն օգտագործվում էին մոդելավորման միջոցով նախագծվող համակարգի տրամաբանության ստուգման համար: Նախագծողներն այնուհետև պետք է համակարգը ներկայացնեին տրամաբանական տարրերի և դրանց միացումների միջոցով: Նախագծվող համակարգերի ֆունկցիոնալ հնարավորությունների ընդարձակման ու բարդացման հետևանքով դժվարանում և երկարաձգվում է նախագծումը: Տրամաբանական սինթեզի ծրագրային գործիքների ստեղծումը արմատապես փոխեց թվային նախագծման մեթոդները: Թվային համակարգերը կարող են ՍՆԼ-ների միջոցով նկարագրվել ռեգիստրային փոխանցումների մակարդակով ՌՓՄ (RTL), այնուհետև, այդ նկարագրությունից սինթեզի գործիքն ստանում է կառուցվող ֆիզիկական սարքի սխեման: Այսպիսով, նախագծողը պետք է նկարագրի ռեգիստրների միջև տվյալների հոսքը՝ տվյալների մշակման գործընթացները: Սխեմաների կազմման ամբողջ աշխատանքը իրականացվում է սինթեզի միջոցով:

Վերիլոգը բավականին բարդ ծրագրավորման լեզու է՝ օժտված ծրագրավորման համար անհրաժեշտ բոլոր հնարավորություններով: Այստեղ կծանոթանանք լեզվին

այն նվազագույն մակարդակով, որը թույլ կտա նկարագրել բացառապես թվային սխեմաներ:

## **7.2. Ավտոմատացված թվային նախագծման ընթացքը**

Ինտեգրալ իրականացմամբ թվային համակարգի ավտոմատացված նախագծման ընթացքը ցույց է տրված նկ. 7.1-ում:

Բլոկները համապատասխանում են նախագծման քայլերին, իսկ բլոկների միջև նշված են նախագծի ներկայացման փաստաթղթերը և ֆայլերը:

Սպեցիֆիկացիան նկարագրում է նախագծվող համակարգի վերացական ֆունկցիոնալությունը, ինտերֆեյսը, ամբողջական ճարտարապետությունը: Այս փուլում համակարգը նախագծողները դեռևս կարող են չիմանալ, թե ինչպես է համակարգն իրականացվելու թվային շղթաների տեսքով: Սպեցիֆիկացիայից ստեղծվում է վարքագծային նկարագրությունը, որի միջոցով պետք է վերլուծվեն նախագծվող համակարգի ֆունկցիաները, պարամետրերը, համապատասխանությունը ստանդարտներին և այլ բարձր մակարդակի հատկություններ և բնութագրեր:

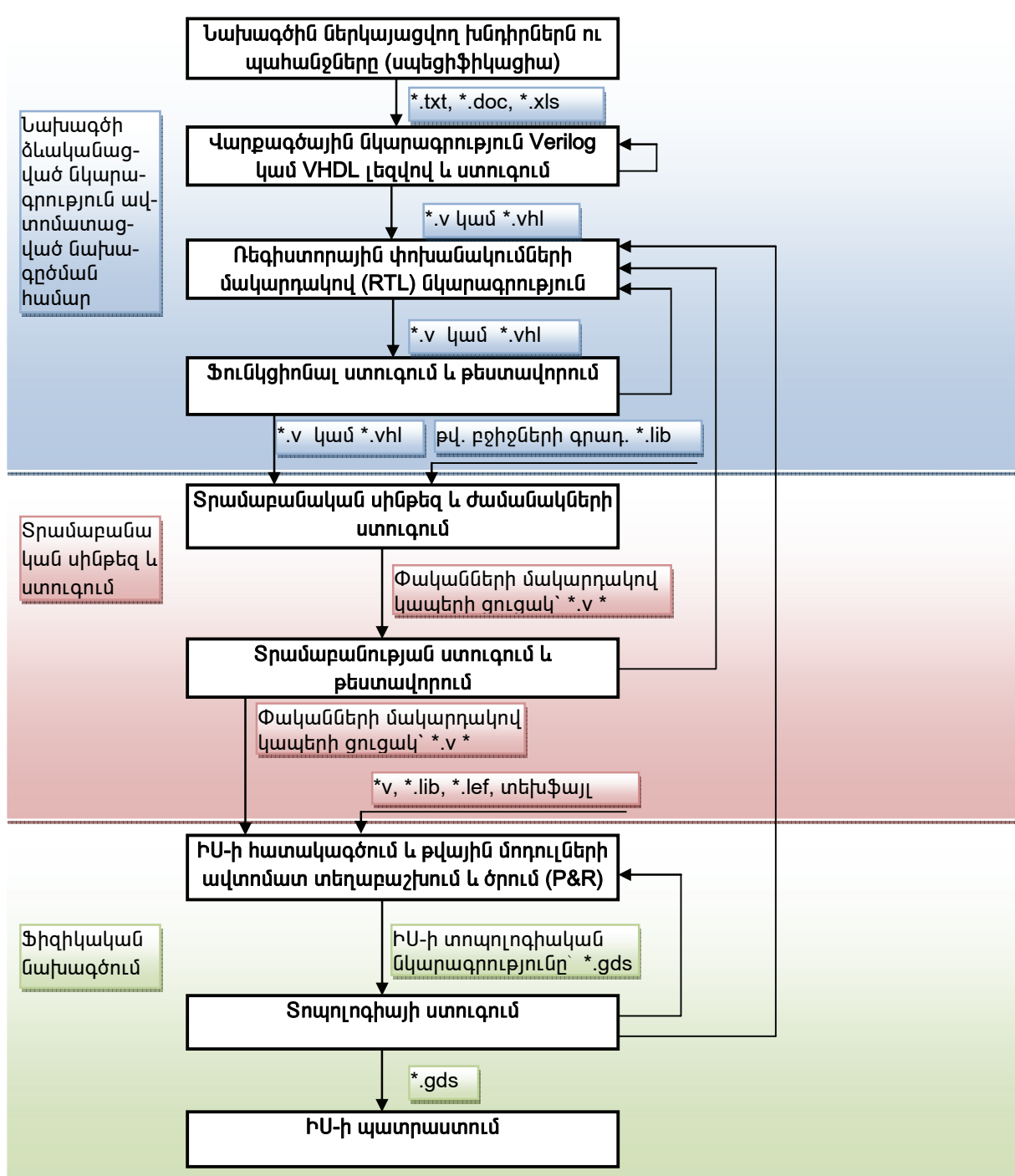
Վարքագծային նկարագրությունը հաճախ գրվում է ՍՆԼ-ով, օրինակ, Վերիլոգով: Վերիլոգով գրված նկարագրությունը՝ կողը, կարելի է նմանակել Վերիլոգ նմանակիչի (սիմուլյատորի) միջոցով և արդյունքները վերլուծել սպեցիֆիկացիայի նկատմամբ:

Վարքագծային նկարագրությունը ձևափոխվում է (ձեռքով կամ ավտոմատացված) ՌՓՄ (RTL) նկարագրության: ՌՓՄ կոդ մշակողը պետք է նկարագրի տվյալների մշակման ֆունկցիաներն այնպես, որ դրանք կարողանան իրականացվել թվային շղթաներով: Ստացված ՌՓՄ նկարագրությունը նույնպես ստուգվում է Վերիլոգ նմանակման միջոցով: Այս փուլից հետո նախագծումը կատարվում է ավտոմատացված նախագծման գործիքներով:

Տրամաբանական սինթեզի գործիքները ՌՓՄ նկարագրությունը ձևափոխում են թվային շղթայի նկարագրության, որը բաղկացած է տրամաբանական տարրերից և դրանց միջև կապերից: Այսպիսի նկարագրությունը կոչվում է տրամաբանական տարրերի մակարդակի կապերի ցուցակ (netlist): Տրամաբանական սինթեզի գործիքը միաժամանակ կատարում է նախագծի օպտիմալացում՝ արագագործության, մակերեսի կամ տարրերի թվի և սպառվող հզորության չափանիշներով: Վերիլոգ լեզվով ներկայացված կապերի ցուցակը նույնպես ստուգվում է Վերիլոգ նմանակման միջոցով:

Տրամաբանական տարրերի մակարդակի կապերի ցուցակով՝ ավտոմատ տեղաբաշխման և ծրման գործիքով, ստեղծվում է կառուցվող համակարգի տոպոլոգիական պատկերը (layout), որից պատրաստվում է ինտեգրալ սխեման:

Կարևոր է հիշել, որ նախագծման որակյալ արդյունք ստանալու համար անհրաժեշտ է խորապես տիրապետել ավտոմատացված նախագծման ծրագրային գործիքներին և ունենալ թվային նախագծման անհրաժեշտ գիտելիքներ:



Նկ. 7.1. Ինտեգրալ կատարմամբ թվային համակարգի ավտոմատացված նախագծման ընթացքը

ՄՆԼ-ով թվային համակարգերի նախագծումը նման է համակարգչային ծրագրերի մշակմանը: Շղթաների մշակումը հանգում է դրանց տեքստային նկարագրությանը և կարգաբերմանը: Վերլիդգը դարձել է սխեմաների նկարագրության ստանդարտ լեզու: Այն ունի բազմաթիվ օգտակար հատկություններ.

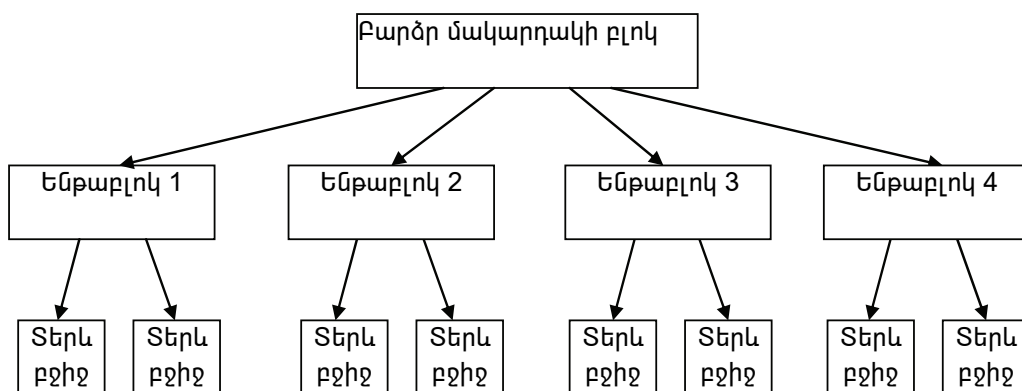


- Վերիլոգը սխեմաների նկարագրության ընդհանուր գործածության լեզու է, որը հեշտ է սովորել և օգտագործել: Այն սինտաքսով շատ նման է C ծրագրավորման լեզվին: C ծրագրավորման փորձ ունեցող նախագծողներին հեշտ է սովորել Վերիլոգ:
- Վերիլոգը թույլ է տալիս նույն կողում ունենալ վերացարկման տարբեր մակարդակների մոդելներ՝ վարքագծային, ՌՓՄ կամ տրամաբանական տարրերի մակարդակի:
- Նախագծողը պետք տիրապետի մեկ լեզվի՝ օրինակ, Վերիլոգի, նախագծվող համակարգի նկարագրության, նմանակման և սինթեզի համար: Տարբեր ընկերությունների կողմից մշակված սինթեզի գործիքները համատեղելի են Վերիլոգի հետ:
- Էլեկտրոնային բաղադրիչների՝ գրադարանների մատակարարները տրամադրում են բաղադրիչների Վերիլոգ մոդելները՝ սինթեզից ստացված կապերի ցուցակի նմանակման համար:

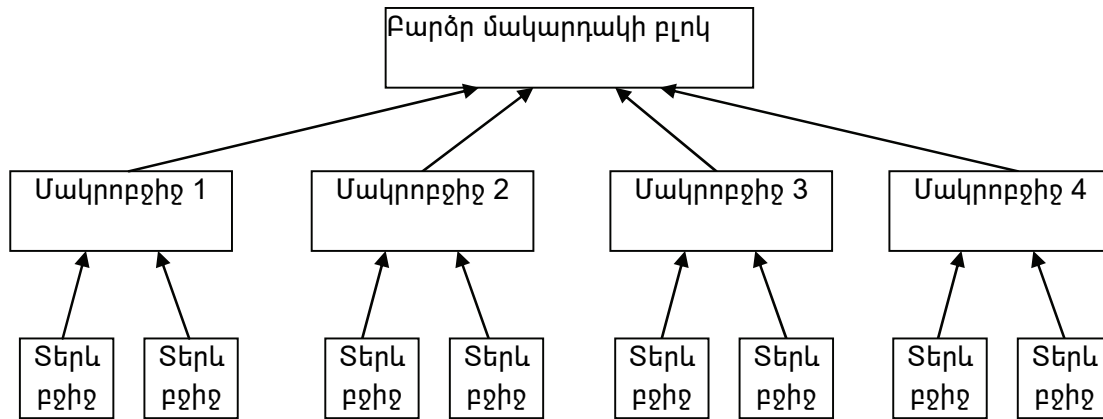
### 7.2.1. Հիերարխիկ նախագծման և մոդելավորման սկզբունքները

Օգտագործվում է հիերարխիկ նախագծման երկու մոտեցում՝ վերևից ներքև և ներքևից վերև: Վերևից ներքև նախագծման դեպքում սահմանվում է ամենաբարձր մակարդակի բլոկը, և որոշվում են այն ենթաբլոկները, որոնցով պետք է կառուցվի բարձր մակարդակի բլոկը: Այնուհետև ենթաբլոկները, իրենց հերթին, նույնպես տրոհվում են ավելի ցածր մակարդակի ենթաբլոկների, մինչև որ ստացվեն ենթաբլոկներ՝ բջիջներ, որոնք այլևս չեն տրոհվում: Չտրոհվող բջիջները հիերարխիկ նախագծման ընթացքը նկարագրող ծառի տերևներն են, իսկ արմատը ամենաբարձր մակարդակի բլոկն է՝ նախագծվող համակարգը: Վերևից ներքև նախագծման ծառը ցույց է տրված նկ. 7.2-ում:

Ներքևից վերև նախագծման դեպքում սկզբում որոշվում են մատչելի բլոկները և բջիջները՝ տերև բջիջները, որոնց միջոցով կարելի է կառուցել համակարգը: Այնուհետև այդ բջիջներից կառուցվում են ավելի բարձր մակարդակի բլոկներ, մինչև նախագծի ամենաբարձր մակարդակի բլոկը: Ներքևից վերև նախագծման ընթացքը ցույց է տրված նկ. 7.3-ում:



Նկ. 7.2. Վերևից ներքև հիերարխիկ նախագծման ծառը



Նկ. 7.3. Ներքևից վերև հիերարխիկ նախագծման ընթացքը

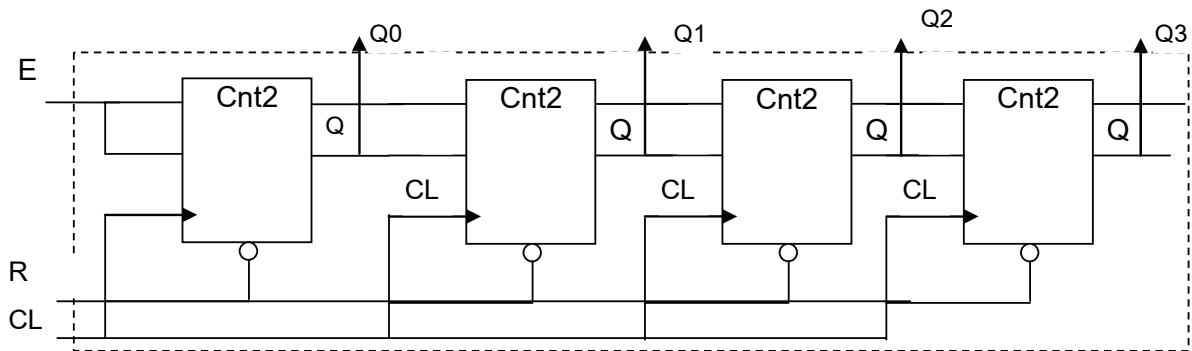
Գործնականում թվային նախագծման ընթացքում կիրառվում է վեկից ներքև և ներքևից վերև մոտեցումների համակցությունը: Համակարգային նախագծողը սահմանում է բարձր մակարդակի բլոկը: Տրամաբանական նախագծողները որոշում են, թե ինչպես պետք է նախագծի ֆունկցիաները տրոհվեն բլոկների և ենթաբլոկների: Միաժամանակ, շղթաները նախագծողները մշակում են տերմ մակարդակի բջիջների սխեմաներն այնպես, որ դրանք բավարարեն բլոկների և ենթաբլոկների կառուցման պահանջներին: Նախագծման գործընթացները հանդիպում են այն կետում, որտեղ տրամաբանական նախագծողները տրոհել են նախագիծը այնպիսի ստորին մակարդակների, որ դրանք կարող են իրականացվել տերմ մակարդակի բջիջներով:

Հիերարխիկ նախագծման մոտեցումները ցուցադրելու նպատակով հետևենք տակտային ազդանշանի աճող ճակատով տակտավորվող սինքրոն քառակարգ հաշվիչի նախագծման ընթացքին:

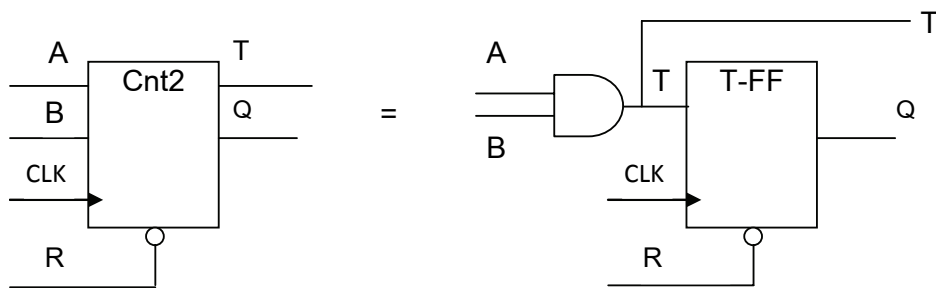
Կարգավորված կառուցվածքով քառակարգ սինքրոն հաշվիչի սխեման ցույց է տրված նկ. 7.4-ում (համեմատել նկ. 5.3-ի սխեմայի հետ): Հաշվիչն ունի տակտային մուտք՝ CLK, ասինքրոն զրոյացման մուտք՝ R, հաշվի թույլտվության մուտք՝ E և հաշվի քառաբիթ կոդի ելքեր՝ Q0, Q1, Q2, Q3: Այն բաղկացած է 1-բիթ հաշվիչներից՝ cnt2, որոնցից յուրաքանչյուրը բաղկացած է սինքրոն T-տրիգերից և 2ԵՎ տարրից, ինչպես ցույց է տրված նկ. 7.5-ում: T-տրիգերն ունի տակտային մուտք՝ CLK, T-մուտք, ասինքրոն զրոյացման մուտք՝ R և մեկ ելք՝ Q: 1-բիթ հաշվիչն ունի տակտային մուտք՝ CLK, ասինքրոն զրոյացման մուտք՝ R, կառավարման երկու մուտք՝ A և B, երկու ելք՝ T և Q: T-տրիգերներով հաշվիչը կառուցվում է ըստ հետևյալ հավասարումների.

$$T_i = a_i \& b_i = T_{i-1} \& q_{i-1}, i \geq 2$$

$$T_1 = E$$

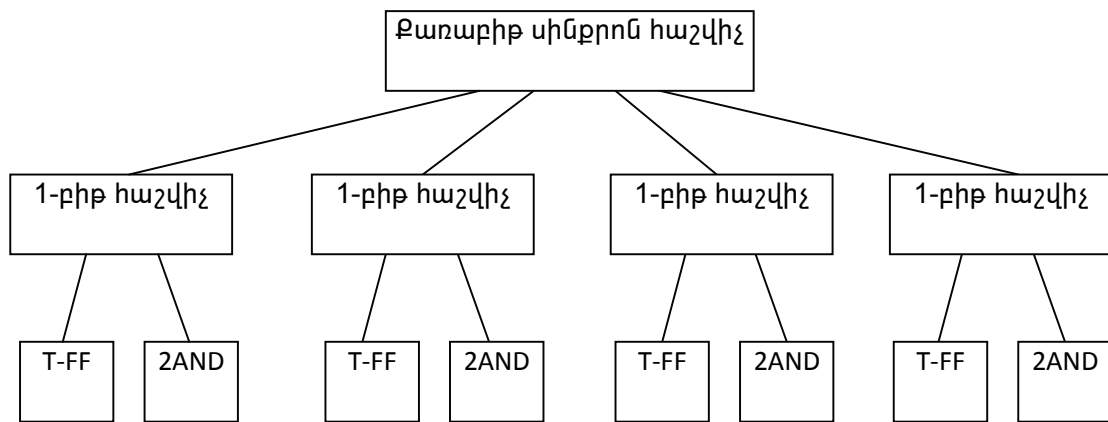


Նկ. 7.4. Քառակարգ սինքրոն հաշվիչ



Նկ. 7.5. 1-բիթ հաշվիչ կազմված T-տրիգերից և 2ԵՎ տարրից

Այսպիսով, քառակարգ հաշվիչը կառուցված է հիերարխիկ սկզբունքով: Նախագծման հիերարխիան ցույց է տրված նկ. 7.6-ում:



Նկ. 7.6. Քառակարգ հաշվիչի նախագծման հիերարխիան

Վերևից ներքև նախագծման ժամանակ սկզբուն սահմանվում է քառակարգ սինքրոն հաշվիչի ֆունկցիան, որը բարձր մակարդակի բլոկն է: Այնուհետև քառաբիթ հաշվիչը կառուցվում է 1-բիթ հաշվիչներից: 1-բիթ հաշվիչը կառուցվում է T-տրիգերից և 2ԵՎ տարրից: Այսպիսով, մեծ բլոկները տրոհվում են ավելի փոքրերի, մինչև որ այլևս հնարավոր չէ դրանք տրոհել: Ներքևից վերև նախագծման դեպքում գործընթացը գնում է հակառակ ուղղությամբ՝ փոքր ենթաբլոկները համակցվում են ավելի մեծ բլոկների մեջ: Օրինակ, T-տրիգերը կարող է կառուցվել տրամաբանական տարրերից, որոնք իրենց հերթին կառուցվում են տրանզիստորներից: Վերևից ներքև և ներքևից վերև նախագծման գործընթացները հանդիպում են 1-բիթ հաշվիչի մակարդակում:

### 7.2.3. Վերիլոգ մոդուլներ

Վերիլոգում հիերարխիկ մոդելավորման սկզբունքն իրականացվում է մոդուլների միջոցով: Մոդուլը Վերիլոգ լեզվի հիմնական բաղադրիչն է, որից կառուցվում է նախագիծը: Մոդուլը կարող է լինել մեկ բաղադրիչ կամ ավելի ցածր մակարդակի բաղադրիչների հավաքածու: Սովորաբար բաղադրիչները խմբավորվում են մոդուլի մեջ ընդհանուր ֆունկցիոնալություն ունենալու նպատակով, որը կարող է օգտագործվել նախագծի բազմաթիվ տեղերում: Մոդուլն ավելի բարձր մակարդակի բլոկում ապահովում է պահանջվող ֆունկցիոնալությունը իր մատույցի ինտերֆեյսի՝ մուտքերի և ելքերի միջոցով, բայց թաքցնում է իր ներքին կառուցվածքը բարձր մակարդակի բլոկից: Դա թույլ է տալիս, անհրաժեշտության դեպքում, փոփոխել մոդուլի ներքին կառուցվածքը առանց նախագծի մնացած մասերի վրա ազդելու:

Նկ. 7.6-ում քառաբիթ հաշվիչը, 1-բիթ հաշվիչը և T-տրիգերը մոդուլների օրինակներ են: Վերիլոգում մոդուլը հայտարարվում է **module** բանալի բառով: Մոդուլի նկարագրությունը պետք է ավարտվի համապատասխան **endmodule** բանալի բառով: Յուրաքանչյուր մոդուլ պետք է ունենա անվանում և տերմինալների ցուցակ, որը նկարագրում է մոդուլի մուտքերն ու ելքերը:

**module** <մոդուլի\_անվանում> (<մոդուլի մուտք/ելք գծերի ցուցակ>);

```
...
<մոդուլի ներքին նկարագրություն>
...
...
endmodule
```

Մասնավորապես, 1-բիթ հաշվիչը կարող է որոշվել հետևյալ մոդուլով.

```
module cnt2 (CLK, R, A, B, T, Q);
.
.
.
<1-բիթ հաշվիչի ֆունկցիայի նկարագրություն>
.
.
endmodule
```

Վերիլոգը վարքագծային և կառուցվածքային նկարագրության լեզու է: Յուրաքանչյուր մոդուլի ներքին ֆունկցիոնալությունը կարելի է նկարագրել վերացարկման չորս մակարդակով՝ կախված նախագծի պահանջներից: Մոդուլի վարքագիծը արտաքին միջավայրում կախված չէ նրա ներքին նկարագրության վերացարկման մակարդակից: Մոդուլի ներքին նկարագրությունը անտեսանելի է արտաքին միջավայրից: Հետևաբար, մոդուլի ներքին նկարագրությունը կարելի է փոխել առանց նրան շրջապատող միջավայրի փոփոխությունների: Վերացարկման այդ չորս մակարդակները կուսումնասիրվեն հետագա բաժիններում: Այստեղ բերենք դրանց սահմանումները:

- Վարքագծային (behavioral) կամ ալգորիթմական մակարդակ. սա արքայական ամենաբարձր մակարդակն է: Մոդուլը կարող է իրականացվել ալգորիթմի մակարդակով՝ առանց հաշվի առնելու դրա սխեմատիկական իրականացման մանրամասները: Այս մակարդակով նախագծումը շատ նման է C լեզվով ծրագրավորմանը:
- Տվյալների հոսքի (dataflow) մակարդակ. Այս մակարդակում մոդուլը նախագծվում է տվյալների հոսքի նկարագրությամբ: Նախագծողը տեղյակ է, թե ինչպես են տվյալները փոխանակվում ռեգիստրների միջև և ինչպես են դրանք մշակվում:
- Տրամաբանական տարրերի (gate) մակարդակ. մոդուլն իրականացվում է տրամաբանական տարրերի և դրանց միացումների միջոցով: Այս մակարդակով նախագծումը նման է տրամաբանական սխեմաների կառուցմանը և նկարագրությանը:
- Փոխանցատիչների (switch) մակարդակ. սա վերացարկման ամենացածր մակարդակն է Վերիլոգում: Մոդուլը կարող է իրականացվել առանձին փոխանցատիչների՝ ՄՕԿ տրանզիստորների և դրանց կապերի նկարագրության միջոցով: Այս մակարդակով նախագծումը պահանջում է թվային շղթաների տրանզիստորների մակարդակով իրականացման մանրամասների իմացություն:

Վերիլոգը թույլ է տալիս նույն նախագծում ունենալ վերացարկման տարբեր մակարդակների նկարագրություններ: Ավտոմատացված թվային նախագծման մեջ ռեգիստրային փոխանցումների մակարդակ (ՌՓՄ, RTL) տերմինը հաճախ օգտագործվում է Վերիլոգ նկարագրության համար, որտեղ օգտագործվում են վարքագծային և տվյալների հոսքի նկարագրության կառուցվածքներ՝ պայմանով, որ այդ նկարագրությունները ընդունելի են տրամաբանական սինթեզի ծրագրային գործիքների կողմից:

Եթե նախագիծը բաղկացած է բազմաթիվ մոդուլներից, յուրաքանչյուր մոդուլ Վերիլոգով կարելի է նկարագրել թվարկված չորս մակարդակներից յուրաքանչյուրով: Սակայն նախագծի հասունացման հետ միաժամանակ մոդուլները աստիճանաբար ներկայացվում են տրամաբանական տարրերի մակարդակով նկարագրությամբ:

Որքան բարձր է վերացարկման մակարդակը, այնքան նախագիծն ավելի ճկուն է և տեխնոլոգիայից անկախ: Որքան մոտենում ենք փականների մակարդակին՝ այսինքն ֆիզիկական իրականացման մակարդակին, այնքան նախագիծը դառնում է տեխնոլոգիայից կախված: Տրամաբանությունում նույնիսկ փոքր փոփոխությունները կարող են նախագծում լուրջ փոփոխությունների պատճառ դառնալ:

Մոդուլը նմուշ է (template), որից կարելի է ստեղծել կոնկրետ օբյեկտներ: Երբ որևէ մոդուլ կանչվում է, Վերիլոգն այդ մոդուլից ստեղծում է մեկ կոնկրետ օբյեկտ: Յուրաքանչյուր օբյեկտ պետք է ունենա անհատական անուն, փոփոխականներ, պարամետրեր և մուտքի/ելքի ինտերֆեյս: Մոդուլի նմուշից օբյեկտների ստեղծումը կոչվում է տեղադրում (instantiation), իսկ այդ օբյեկտները կոչվում են օրինակներ (instances): Ստորև բերված օրինակում ամենաբարձր մակարդակի բլոկը (synchronous\_counter մոդուլը) cnt2 1-րիժ հաշվիչի մոդուլից (նմուշից) ստեղծում է չորս օրինակ՝ inst0, inst1,

inst2, inst3 անհատական անուններով: cnt2-ի յուրաքանչյուր օրինակ պարունակում է մեկ T տրիգերի և մեկ 2ԵՎ տարրի օրինակներ: Վերիլոգ ծրագրի տեքստում մեկ տողով մեկնաբանությունները նշվում են երկու թեք գծերով` '//':

**Օրինակ 7.1.** Քառակարգ հաշվիչի Վերիլոգ կոդը.

// Ամենաբարձր մակարդակի մոդուլն անվանվել է synchronous\_counter  
// այն բաղկացած է չորս 1-բիթ հաշվիչներից: Միացումները ցույց են տրված նկ. 7.4-ում

**module** synchronous\_counter(CLK, R, E, Q);

**output** [3:0] Q; // Ելքային ազդանշանների հայտարարում

// Q –ն վեկտոր է, կբացատրվի հետագայում

**input** CLK, R; // մուտքային ազդանշանների հայտարարում

**wire** [3:0] T; //օբյեկտների միջև կապեր՝ լարեր, կբացատրվի հետագայում

cnt2 inst0(CLK, R, E, E, T[0], Q[0]); //1-բիթ հաշվիչի չորս օրինակներ

cnt2 inst1(CLK, R, Q[0], T[0], T[1], Q[1]);

cnt2 inst2(CLK, R, Q[1], T[1], T[2], Q[2]);

cnt2 inst3(CLK, R, Q[2], T[2], T[3], Q[3]);

**endmodule**

Ստորև ցույց է տրված cnt2 մոդուլի նկարագրությունը՝ այն բաղկացած է մեկ T տրիգերից և մեկ AND2 տարրից: Համարվում է, որ T տրիգերի՝ T\_FF և AND2 տարրի մոդուլները նկարագրված են նախագծում, բայց այստեղ բերված չեն: Միացումները ցույց են տրված նկ. 7.5-ում:

**module** cnt2(CLK, R, A, B, T, Q);

**output** Q, T; // մուտք/ելք ազդանշանների հայտարարում

**input** CLK, R, A, B;

T\_FF n\_1(CLK, R, T, Q); // T\_FF-ի օրինակ: Օրինակի անունն է՝ n\_1

AND2 n\_2(A, B, T); // AND2-ի օրինակ: Օրինակի անունն է՝ n\_2

**endmodule**

Վերիլոգ լեզվում արգելված են միմյանց մեջ ներդրված մոդուլները: Որևէ մոդուլի նկարագրության մեջ (**module** և **endmodule** բանալիային բառերի միջև) չի կարող պարունակվել մեկ այլ մոդուլի նկարագրություն: Դրա փոխորեն մեկ մոդուլում կարելի է օգտագործել մեկ այլ մոդուլից ստեղծված օրինակներ: Կարևոր է իրարից տարբերել մոդուլը և դրանից ստեղծված օրինակները: Մոդուլի սահմանումը նկարագրում է նրա ինտերֆեյսի աշխատանքը: Օգտագործելու համար պետք է մոդուլից ստեղծել օրինակներ:

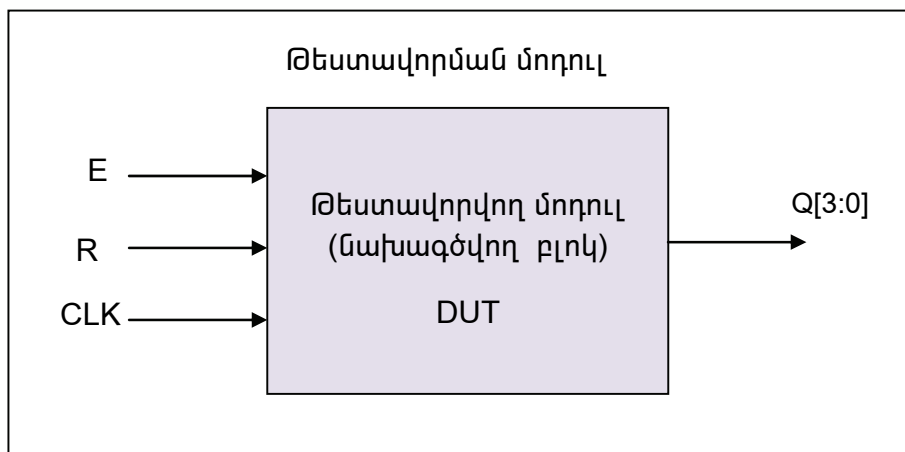
## 7.2.4 Նախագծի նմանակում և ստուգում

Երբ նախագծի որևէ բլոկ նախագծված է, այն պետք է ստուգել: Բլոկի ֆունկցիոնալությունը թեստավորվում է՝ նրա մուտքերին կիրառելով ազդակներ և ստուգելով ելքերի արձագանքները:

Թեստավորման համակարգը պետք է ներառի ազդակների գեներացման բլոկ և ստուգվող բլոկի օրինակ: Այդպիսի թեստավորման բլոկը նույնպես կարելի է նկարագրել Վերիլոգ լեզվով՝ այն Վերիլոգ մոդուլ է: Թեստավորման մոդուլը ավելի հաճախ անվանում են փորձասեղան (test bench):

Թեստավորման մոդուլը կարելի է կառուցել երկու մոտեցմամբ.

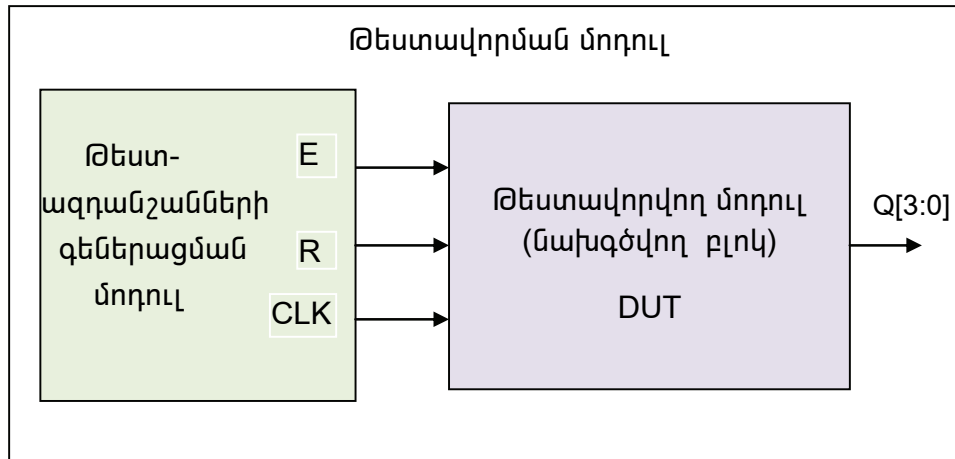
1. Փորձարկման համար մուտքային ազդանշանները գեներացվում են թեստավորման մոդուլի միջից և կիրառվում են թեստավորվող մոդուլի օրինակի (device under test, DUT) մուտքերին (նկ. 7.7): Ստուգվող մոդուլի օրինակի ելքերը մատչելի են թեստավորման մոդուլում: Նկ. 7.7-ում ստուգվող բլոկը քառաբիթ սինքրոն հաշվիչն է, թեստային ազդանշաններն են՝ E, CLK, R: Ելքային Q[3:0] ազդանշանների վերլուծությամբ կարելի է եզրակացություն անել նախագծված բլոկի ֆունկցիոնալ համապատասխանության մասին:



Նկ. 7.7. Թեստավորման մոդուլ՝ թեստային ազդանշանները (վեկտորները) գեներացվում են թեստավորման մոդուլում

2. Փորձարկման համար մուտքային թեստային ազդանշանները գեներացվում են առանձին թեստային ազդանշանների ձևավորման մոդուլում: Թեստավորման մոդուլում ստեղծվում են թեստավորվող մոդուլի և թեստային ազդանշանների մոդուլի օրինակներ (նկ. 7.8):

Երկու մոտեցումներն էլ հավասարապես արդյունավետ են:



Նկ. 7.8. Թեստավորման մոդուլ՝ թեստային ազդանշանները (վեկտորները) գեներացվում են առանձին մոդուլում

### 7.2.5. Վերիլոգ նախագծի օրինակ

Քննարկված հիերարխիկ նախագծման և մոդելավորման սկզբունքները ակնառու դարձնելու նպատակով ուսումնասիրենք քառաբիթ սինքրոն հաշվիչի ամբողջական Վերիլոգ նկարագրության և նմանական օրինակը: Թեստավորման համակարգը կկառուցենք նկ. 7.7-ում ցույց տրված կառուցվածքով՝ թեստավորման մոդուլ, որում տեղակայված է թեստավորվող բլոկի օրինակը: Թեստավորման համար թեստավորման մոդուլում գեներացվում են մուտքային ազդանշանները և հսկվում են թեստավորվող բլոկի ելքերը: Նշենք, որ Վերիլոգ մոդուլների նկարագրություններում պետք է չէ այս փուլում փորձել հասկանալ բոլոր կառուցվածքները և սինտաքսը: Այս փուլում միայն պետք է ձգտել հասկանալ նախագծման գործընթացը: Մենք կուսումնասիրենք Վերիլոգ լեզվի կառուցվածքները և սինտաքսը հետագա բաժիններում:

**Նախագծվող բլոկ.** կկիրառվի վերևից ներքև հիերարխիկ նախագծման մոտեցում: Սկզբում կգրենք ամենաբարձր մակարդակի Վերիլոգ մոդուլը՝ `synchronous_counter` (տես օրինակ 7.1 ).

```
module synchronous_counter(CLK, R, E, Q);
```

```
output [3:0] Q;
```

```
input CLK, R;
```

```
wire [3:0] T;
```

```
//1-բիթ հաշվիչի չորս օրինակներ
```

```
cnt2 inst0(CLK, R, E, E, T[0], Q[0]);
```

```
cnt2 inst1(CLK, R, Q[0], T[0], T[1], Q[1]);
```

```
cnt2 inst2(CLK, R, Q[1], T[1], T[2], Q[2]);
```

```
cnt2 inst3(CLK, R, Q[2], T[2], T[3], Q[3]);
```

```
endmodule
```



Այս բլոկում ունենք cnt2 մոդուլի չորս օրինակ: Հետևաբար պետք է ունենալ cnt2 մոդուլի նկարագրությունը (տե՛ս նկ. 7.5 ).

```
module cnt2(CLK, R, A, B, T, Q);
```

```
    output Q, T;
    input CLK, R, A, B;
```

```
    T_FF    n_1(CLK, R, T, Q);
    AND2    n_2(A, B, T);
```

```
endmodule
```

Քանի որ այս մոդուլում օգտագործվել են T\_FF և AND2 մոդուլների օրինակներ, պետք է նկարագրել նաև այդ մոդուլները.

```
module T_FF (CLK, R, T, Q);
```

```
    output Q;
    input CLK, R, T;
```

```
// Այստեղ կան շատ մոր կառուցվածքներ, ուշադրություն մի դարձրեք ֆունկցիոնալությանը,
// կենտրոնացեք վերևից ներքև նախագծման գործընթացի վրա
```

```
    reg Q;
    always @(posedge R or negedge CLK)
    if (R)
        Q <= 1'b0;
    else if (T==0)
        Q <= Q;
    else
        Q <= ~Q;
endmodule
```

```
module AND2 (A, B, T);
```

```
    output T;
    input A,B;
```

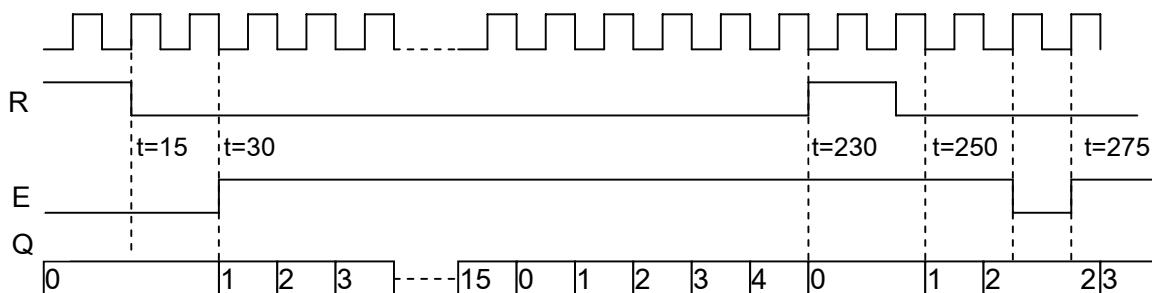
```
    assign T=A&B;
```

```
endmodule
```

Բոլոր մոդուլները որոշված են մինչև ամենացածր մակարդակի տերև- բջիջները՝ մոդուլները: Նախագծվող բլոկը պատրաստ է:

**Թեստավորման մոդուլ.** սա ամենաբարձր մակարդակի մոդուլն է, որում տեղադրվում է նախագծված բլոկի մեկ օրինակ՝ թեստավորման համար: Այն պետք է գրել այնպես, որ հնարավոր լինի ստուգել քառաբիթ հաշվիչի սպասվող ֆունկցիոնալությունը: Այս մոդուլում պետք է առաջին հերթին նկարագրվեն մուտքային թեստային ազդանշանները՝ CLK, R, E: Դրա համար կարելի է օգտվել նախապես կազմված ժամանակային դիագրամներից: Այդպիսի դիագրամների մեկ օրինակ ցույց է տրված նկ. 7.9-ում: CLK տակտային իմպուլսների պարբերությունը ընդունենք, որ 10 միավոր է: Սկզբնական զրոյացման ազդանշանը պահվում է ակտիվ վիճակում ժամանակի 15 միավորի ընթացքում, այն նորից ակտիվ վիճակ է ընդունում 230-ից 245 ժամանակահատվածում: Հաշվի թույլտվության E ազդանշանն ակտիվանում է  $t=30$  պահին, նորից անցնում է պասիվ վիճակի  $t=265$  պահին և նորից ակտիվանում է  $t=275$  պահին:

Այժմ կարելի է գրել թեստային ազդանշանների Վերիլոգ նկարագրությունը թեստավորման մոդուլում: Թեստավորման մոդուլի Վերիլոգ կոդը բերված է ստորև: Մի խորացեք կոդի սինտաքսի մեջ, ուղղակի փորձեք հասկանալ, թե ինչպես է թեստավորվում սինթրոն հաշվիչը այդ միջավայրում:



Նկ. 7.9. Թեստավորման մուտքային ազդանշանների և սպասվող ելքային ազդանշանների ժամանակային դիագրամները

```
module testbench;
```

```
reg CLK;
```

```
reg R;
```

```
reg E;
```

```
wire[3:0] Q;
```

```
// թեստավորվող բլոկի օրինակի տեղակայում, օրինակի անունն է dut  
synchronous_counter dut(CLK, R, E, Q);
```

```
// CLK ազդանշանի գեներացում՝ պարբերությունը 10 միավոր է, կիսապարբերությունը՝ 5.
```

```
initial
```

```
CLK = 1'b0; // CLK-ը դնել 0 վիճակ
```

```
always
```

```
#5 clk = ~clk; // CLK-ը ժխտել ժամանակի յուրաքանչյուր 5 միավորից հետո
```

```
// R ազդանշանի ձևավորում՝ 0-ից 15 միջակայքում R = 1, 15-ից 230 միջակայքում R = 0
```

```
// 230-ից 245 միջակայքում R = 1, այնուհետև՝ R = 0
```

```
initial
```

```
begin
```

```

R = 1'b1;
#15 R = 1'b0; //կատարվում է նախորդ հրամանի նկատմամբ ժամանակի 15 միավոր
//ուշացումով
#215 R = 1'b1; //կատարվում է նախորդ հրամանի նկատմամբ 215 միավոր ուշացումով:
//Այսինքն, R = 1'b1 հրամանը կկատարվի ժամանակի 230 պահին:
#15 R = 1'b0;
end

```

```

// E ազդանշանի ձևավորում՝ 0-ից 30 միջակայքում E =0, 30-ից 265 միջակայքում E =1
// 265-ից 275 միջակայքում E =0, այնուհետև՝ E =1

```

```

initial
begin
E = 1'b0;
#30 E = 1'b1;
#235 E = 1'b0;
#10 E = 1'b1;
#20 $finish; //ավարտել նմանակումը
end

```

```

// ելքային ազդանշանների փոփոխությունների գրանցում

```

```

initial
$monitor($time, " Output Q = %d", Q);

```

```

endmodule

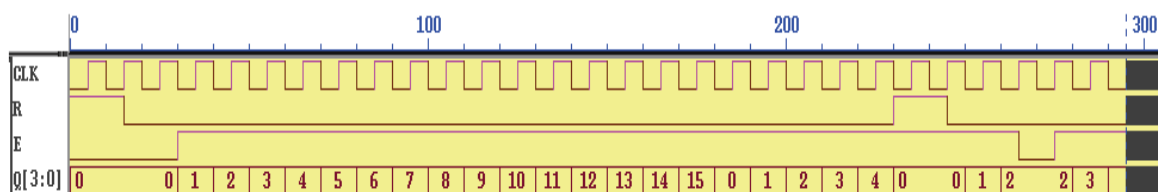
```

Քանի որ թեստավորման բլոկը պատրաստ է, կարող ենք կատարել նմանակումը և ստուգել նախագծված սինթրոն հաշվիչի աշխատանքի ճշտությունը: Նմանակումից ստացված ելքային ազդանշանների վիճակները բերված են ստորև, ձախից նշված է ընթացիկ ժամանակը:

```

0 Output Q = 0 110 Output Q = 9 200 Output Q = 2
30 Output Q = 1 120 Output Q = 10 210 Output Q = 3
40 Output Q = 2 130 Output Q = 11 220 Output Q = 4
50 Output Q = 3 140 Output Q = 12 230 Output Q = 0
60 Output Q = 4 150 Output Q = 13 250 Output Q = 1
70 Output Q = 5 160 Output Q = 14 260 Output Q = 2
80 Output Q = 6 170 Output Q = 15 280 Output Q = 3
90 Output Q = 7 180 Output Q = 0 290 Output Q = 4
100 Output Q = 8 190 Output Q = 1

```



Նկ. 7. 10. Վերիլոգ նմանակումից ստացված ժամանակային դիագրամները (Synopsys VirSim), վերը նշված է նմանակման ժամանակը, ձախից՝ փոփոխականների անվանումները

### 7.2.6. Խնդիրներ

**Խնդիր 7.1.** Քառաբիթ գումարիչը բաղկացած է չորս միակարգ լրիվ գումարիչից.

ա. սահմանել 1-բիթ լրիվ գումարիչի մոդուլը՝ FA: Հարկ չկա որոշել մոդուլի ներքին բովանդակությունը կամ մուտքի/ելքի գծերի ցուցակը:

բ. սահմանել քառաբիթ գումարիչի մոդուլը՝ add4: Հարկ չկա որոշել մոդուլի ներքին բովանդակությունը կամ մուտք/ելք գծերի ցուցակը: Ստեղծել FA մոդուլի չորս օրինակ add4 մոդուլում և անվանել դրանք i0, i1, i2, i3:

**Խնդիր 7.2.** Սահմանել քառաբիթ թվերի համեմատման մոդուլ ըստ (3.39) (3.40) բանաձևերի (հարկ չկա նկարագրել ֆունկցիոնալությունը): Սահմանել քառաբիթ թվերի համեմատման մոդուլ, որում պետք է տեղադրել միակարգ թվերի համեմատման մոդուլի չորս օրինակ (տե՛ս նկ. 3.32):

## 7.3 Վերիոգ լեզվի սիմվոլները և տվյալների տիպերը

### 7.3.1. Վերիոգ լեզվի բաղադրիչները

Վերիլոգ լեզվում օգտագործվող պայմանավորվածությունները շատ նման են C ծրագրավորման լեզվում օգտագործվողներին: Վերիլոգ լեզվով ծրագրի տեքստը բաղկացած է լեզվի միավորներից: Լեզվի յուրաքանչյուր միավոր բաղկացած է մեկ կամ մի քանի սիմվոլներից: Լեզվի միավորների տեղաբաշխումը տեքստում կարող է լինել ազատ ոճով՝ բացակները և նոր տողի անցումները կարևոր չեն ուղղագրության տեսանկյունից, բայց դրանք շատ կարևոր են՝ տեքստին ընթեռնելի տեսք տալու համար:

Լեզվի միավորները կարելի է ներկայացնել հետևյալ խմբերով՝

- Օպերատոր
- Բացակ
- Մեկնաբանություն
- Թիվ
- Սիմվոլների շղթա
- Իդենտիֆիկատոր (նույնարար)
- Բանալի բառ

Օպերատորները կարող են լինել եզակի, կրկնակի կամ եռակի գործողությունների նշաններ, որոնք օգտագործվում են արտահայտություններում: Եզակի օպերատորը դրվում է օպերանդից ձախ, կրկնակին՝ երկու օպերանդների արանքում: Օրինակ,  $x \sim y$  արտահայտությունում “~”-ը եզակի օպերատոր է՝ նշանակում է ժխտում,  $z = x + y$  արտահայտությունում “+”-ը կրկնակի օպերատոր է: Վերիլոգ լեզվում կա միայն մեկ եռակի օպերատոր՝ պայմանական օպերատոր: Օրինակ,  $q = s ? x : y$ , նշանակում է՝ q-ին վերագրել x, եթե  $s = 1$ , և վերագրել y, եթե  $s = 0$ :

Բացակը կարող է լինել տեղանցում (\b), թաք (\t), նոր տող (\n): Բացակները Վերիլոգ ծրագրում անտեսվում են, բացի այն դեպքերից, երբ դրանք բաժանում են ծրագրի միավորները: Բացակը տեքստային շղթաներում չի անտեսվում:

Մեկնաբանություններն օգտագործվում են ծրագրին ընթեռնելի դարձնելու և բացատրություններ տալու համար: Վերիլոգ ծրագրի տեքստում մեկնաբանությունը կարելի ավելացնել երկու ձևով՝ մեկ տողով մեկնաբանությունը սկսվում է “//” նշանով և ավարտվում է նոր տողի բացակով, բազմաթիվ տողերով մեկնաբանությունը սկսվում է “/\*”-ով և ավարտվում է “\*/”-ով: Մեկ տողով մեկնաբանության օրինակ՝

q=s ? x: y; // q-ին վերագրել x, եթե s=1, և վերագրել y, եթե s=0:

Թվերը Վերիլոգում լինում են չափսով՝ չափսի բացահայտ նշումով և առանց չափսի: Չափսով թվերը ներկայացվում են հետևյալ չափաձևով՝ <չափսը>’< հիմքը>’< թիվը>: <չափսը> գրվում է միայն տասական հիմքով և ցույց է տալիս թվում բիթերի թիվը: < հիմքը> կարող է լինել երկուական ('b կամ 'B), ութական ('o կամ 'O), տասնվեցական ('h կամ 'H), տասական ('d կամ 'D): Տասնվեցական թվանշանները կարելի գրել և՛ մեծատառով, և՛ փոքրատառով: Չափսերով թվերի օրինակներ.

4'b1001 // 4-բիթ երկուական թիվ,  
5 'D 3 // 5-բիթ տասական թիվ,  
3'b01x // 3-բիթ երկուական թիվ, x-ը նշանակում է անհայտ արժեք,  
12'hx // 12-բիթ անհայտ տասնվեցական թիվ՝  
16'hz // 16-բիթ բարձր դիմադրության արժեքով թիվ:

Առանց չափսի թվերը գրվում են առանց <չափսը> բացահայտ դաշտի և համարվում է, որ դրանց չափսը որոշվում է համակարգչի և Վերիլոգ նմանակիչի կարգայնությամբ՝ առնվազն 32 բիթ: Տասական թվերը կարելի գրել առանց նշելու < հիմքը>: Օրինակներ՝

659 //տասական թիվ,  
'h 837FF // տասնվեցական թիվ,  
'o7460 // ութական թիվ,  
4af // անթույլատրելի է՝ տասնվեցական թիվը պետք սկսվի “h”-ով:

Չափսով բացասական թվի <չափս>-ից առաջ դրվում է “-“ նշան՝

-8'd6 // 8-բիթ բացասական թիվ, որը պահվում է իբրև 6 թվի երկուսի լրացումը՝ 8'b11111010,  
-6'sd3 // s-ը այստեղ ցույց է տալիս, որ գործողությունները պետք է կատարվեն նշանով  
//թվերի գործողությունների կանոններով,  
4'd-2 // անթույլատրելի է:

Թվերի ընթեռնելիությունը լավացնելու համար թվանշանները կարելի է բաժանել տողատակի գծիկով, օրինակ՝

16'b0011\_0101\_0001\_1111

Տողատակի գծիկն անթույլատրելի է առաջին թվանշանից առաջ:

Սիմվոլների շղթան կամ ուղղակի շղթան սիմվոլների հաջորդականություն է ներառված կրկնակի չակերտների միջև: Հետևյալ օրինակում՝

*"Hello world!"*

յուրաքանչյուր սիմվոլ ներկայացվում 1-բայթ ASCII արժեքով: "Hello world!" շղթան պարունակում է 12 ASCII սիմվոլներ՝  $8 \times 12 = 96$  բիթ:

**Բանալի բառերը** հատուկ իդենտիֆիկատորներ են, որոնք վերապահված են լեզվի կառուցվածքների սահմանման համար: Բանալի բառերը պետք է լինեն փոքրատառ: \$ նշանին հետևող բանալի բառերը մեկնաբանվում են իբրև հակարգային առաջադրանքներ (ենթածրագրեր) կամ ֆունկցիաներ, օրինակ՝

```
$display ("display a message"); //ցուցադրել display a message տեքստը,  
$finish; // ծրագրի կատարման ավարտ
```

Երբ բանալի բառին նախորդում է “~” նշանը, այդ բանալի բառը Վերիլոգ ծրագրի կոմպիլատորի կողմից ընկալվում է իբրև կոմպիլացման դիրեկտիվ: Օրինակ,

```
`define wordsize 8
```

դիրեկտիվով կոմպիլատորին ասվում է, որ ծրագրում հանդիպող `wordsize իդենտիֆիկատորը պետք է փոխարինել 8-ով:

**Իդենտիֆիկատորները** օբյեկտներին տրվող անուններ են, որոնք թույլ են տալիս դրանց դիմել ծրագրում: Իդենտիֆիկատորները բաղկացած են տառերից, թվերից, տողատակի գծիկից և դոլարի նշանից: Իդենտիֆիկատորը չի կարող սկսվել թվանշանով կամ դոլարի նշանով: Օրինակներ՝

```
reg value; // reg-ը բանալի բառ է, value-ն իդենտիֆիկատոր է,  
input clk; // input-ը բանալի բառ է, clk-ը՝ իդենտիֆիկատոր:
```

### 7.3.2 Տվյալների տիպերը

Վերիլոգում տվյալների տիպերը սահմանվում են թվային սարքերում հանդիպող տվյալների պահպանման և փոխանցման տարրերի և շղթաների ներկայացման համար:

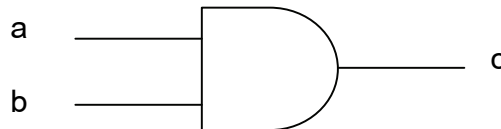
Վերիլոգ լեզվում ազդանշանների արժեքների բազմությունը կազմված է չորս հիմնական արժեքից՝

- 0 – տրամաբանական 0 արժեք կամ ոչ ճշմարիտ (false) պայման,
- 1 - տրամաբանական 1 արժեք կամ ճշմարիտ (true) պայման,
- x - անհայտ տրամաբանական արժեք,
- z - բարձր դիմադրության վիճակ (Hi-Z):

Տվյալների տիպերը բաժանվում են երկու հիմնական խմբի՝ ռեգիստր տիպի և ցանց (միացման լար, net) տիպի: Այս երկու խմբերը տարբերվում են դրանց արժեքներ վերագրելու և դրանցում տվյալների պահպանման ձևով: Դրանք, նաև, ներկայացնում են սխեմայի տարբեր տիպի սարքեր:

### 7.3.2.1 Ցանցեր

Ցանցերը ներկայացնում են սխեմայի տարրերի միացումները: Ինչպես իրական սխեմայում, դրանց վրա ազդանշանի արժեքները հավասար են այն տարրի ելքի ազդանշանին, որի ելքին միացված է տվյալ ցանցը: Նկ. 7.11-ում *c* ցանցը միացված է 2ԵՎ տարրի ելքին: *c*-ն միշտ կունենա այդ տարրի ելքային ազդանշանի արժեքը, որը հաշվվում է ինչպես *a*&*b*:



Նկ. 7.11. *c* ցանցը տարվում 2ԵՎ տարրի ելքից

Ցանցերը ծրագրում հայտարարվում են **wire** (հաղորդալար) բանալի բառով: Եթե ցանցը հայտարարված չէ որպես վեկտոր, ապա այն համարվում է սկալյար՝1-բիթ: Որևէ ցանցի՝ ըստ պայմանավորվածության (default) արժեքը *z* է (բացի **triereg** տիպի ցանցից, որի ըստ պայմանավորվածության արժեքը *x* է): Ցանցն ունի այն տանող տրամաբանական տարրի ելքի արժեքը: Եթե ցանցը չունի տանող տարր, այն ստանում է *z* արժեք: Ցանցի հայտարարման օրինակներ՝

```
wire c; // նկ. 2.1-ում c ցանցը
wire a,b; // նկ. 2.1-ում a, b ցանցերը
wire d = 1'b0; // 0 ֆիքսված արժեքով ցանց
wire w1, w2; // հայտարարվում է երկու ցանցեր
```

Երբ **z** արժեքը տրված է որևէ տարրի մուտքի կամ մասնակցում է որևէ արտահայտությունում, ապա արդյունքը նույնն է, ինչ **x** արժեքի դեպքում:

Նկատեք, որ ցանցը բանալի բառ չէ, այն ներկայացնում է տվյալների տիպերի խումբ: Տվյալների այդ տիպերն են՝

**wire** - հիմնականում օգտագործվող տիպն է,

**wand, wor , triand, trior** - օգտագործվում են մոնտաժային տրամաբանության մոդելավորման համար, լուծում են նույն ցանցին միացված ազդանշանների կոնֆլիկտները,

**tri** - օգտագործվում է, երբ ցանցը տարվում է մեկից ավելի տարրերի ելքերից, լուծում է նույն ցանցին միացված ազդանշանների կոնֆլիկտները,

**tri0, tri1** – օգտագործվում է սնման աղբյուրի *vdd*, *vss* գծերին մեծ դիմադրությամբ միացված ցանցերի նկարագրության համար,

**triereg** - օգտագործվում է էլեկտրական լիցքը պահպանող ցանցերի նկարագրության համար:

Ցանցերի նշված տիպերը հնարավորություն են տալիս ճշգրիտ մոդելավորել ազդանշանային գծերի միացումները՝ երկուղղված փոխանցման փականներ, ռեգիստիվ MOS տրանզիստորներով շղթաներ, դիմամիկ MOS շղթաներ, էլեկտրական լիցքի բաժանում և այլն: Նույն ցանցին միացված բազմաթիվ տարրերի ելքերի ազդանշան-

ների միջև կոնֆլիկտները լուծվում են՝ դրանց համապատասխան ուժեր վերագրելով:  
Ուժը տրվում է երկու բաղադրիչներով՝

ա) Ցանցի 0 արժեքի ազդանշանի ուժ՝ **strength0**, որը կարող է ունենալ հետևյալ արժեքները (տրված են ըստ ուժի նվազման կարգի)՝

**supply0** – սնման աղբյուրի՝ հողի 0,

**strong0**– ուժեղ 0, սովորաբար տրամաբանական տարրի ելք,

**pull0** – մեծ դիմադրությամբ հողին միացված ցանցի ուժ,

**weak0**– թուլ 0,

**highz0**– բարձր դիմադրության ուժի 0:

բ) Ցանցի 1 արժեքի ազդանշանի ուժ՝ **strength1**, որը կարող է ունենալ հետևյալ արժեքները (տրված են ըստ ուժի նվազման կարգի)՝

**supply1** – սնման աղբյուրի 1,

**strong1**– ուժեղ 1, սովորաբար տրամաբանական տարրի ելք,

**pull1** – մեծ դիմադրությամբ սնման աղբյուրին միացված ցանցի ուժ,

**weak1**– թուլ 1,

**highz1**– բարձր դիմադրության ուժի 1:

Ուժի հայտարարությամբ ցանցի օրինակ,

**wire (strong1, pull0) mynet;**//mynet ցանցի 1 ազդանշանի ուժը **strong1** է, 0-ինը՝ **pull0**:

Եթե միևնույն ցանցին միացված են տարբեր ուժերի ազդանշաններ, ցանցում հաստատվում է ավելի ուժեղ ազդանշանի արժեքը: Եթե ցանցին միացված են նույն ուժի տարբեր ազդանշաններ, արդյունքն անհայտ է՝ **x**: (**highz0**, **highz1**) և (**highz1**, **highz0**) համակցություններն արգելված են:

### 7.3.2.2 Ռեգիստրներ

Ռեգիստրները ներկայացնում են տվյալների պահպանման տարրեր: Ռեգիստրը պահպանում է արժեքը, մինչև նրանում նոր արժեք գրանցվելը: Չպետք է շփոթել ռեգիստր տերմինը Վերիլոգում և թվային սխեմաներում՝ տրիգերների վրա կառուցված թվային շղթային: Վերիլոգում ռեգիստր է կոչվում ցանկացած փոփոխական, որը պահպանում է արժեքը: Ի տարբերություն ցանցի, ռեգիստրը որևէ տարրի ելքից տարվելու կարիք չունի: Ի տարբերություն ռեգիստր թվային շղթայի, Վերիլոգում ռեգիստրը տակտային ազդանշանի կարիք չունի, ռեգիստրի արժեքը կարելի փոխել նմանակման ցանկացած պահի՝ նրան վերագրելով նոր արժեք:

Ռեգիստրը հայտարարվում է **reg** բանալի բառով: Ռեգիստր տիպի տվյալի՝ ըստ պայմանավորվածության արժեքը **x** է: Ռեգիստրի հայտարարման և օգտագործման օրինակ՝

**reg reset;** // հայտարարել **reset** իդենտիֆիկատորով ռեգիստր

**initial** // այս կառուցվածքը կբացատրվի հետագայում



**begin**

reset = 1'b1; // reset ռեգիստրին վերագրել 1'b1 արժեք

#100 reset = 1'b0; // 100 միավոր ժամանակից հետո վերագրել 1'b0 արժեք

**end**

Ռեգիստրը կարելի նաև հայտարարել իբրև նշանով փոփոխական: Ցանցերն ու ռեգիստրները կարելի է հայտարարել որպես վեկտորներ՝ բազմաբիթ բիթերով: Եթե բիթերի թիվը հայտարարված չէ, ցանցը կամ ռեգիստրը համարվում է սկալյար՝ 1-բիթ:

**wire** a; // սկալյար ցանց՝ ըստ պայմանավորվածության

**wire** [7:0] bus; // 8-բիթ դոդ

**wire** [31:0] busA, busB, busC; // 32-բիթ լայնությամբ 3 դոդ

**reg** clock; // սկալյար ռեգիստր՝ ըստ պայմանավորվածության

**reg** [3:0] v; // 4-բիթ ռեգիստր՝ բաղկացած v[3], v[2], v[1] և v[0] բաղադրիչներից  
// (սկսած ավագ բիթից՝ դեպի կրտսեր բիթը)

**reg** [0:40] virtual\_addr; // virtual\_addr անվանումով 41-բիթ ռեգիստր

**reg signed** [63:0] m; // 64-բիթ նշանով արժեքներ պահպանող ռեգիստր

Վեկտորը կարելի է հայտարարել [բարձր# : ցածր#] կամ [ցածր# : բարձր#], բայց ցանկացած դեպքում վեկտորի ավագ բիթը քառակուսի փակագծի ձախ բիթն է: Վերևում բերված օրինակում virtual\_addr ռեգիստրի ավագ բիթը virtual\_addr[0]-ն է:

Կարելի է հասցեավորել վեկտորի առանձին բիթերը կամ բիթերի մի մասը՝

busA[7] // busA վեկտորի 7-րդ բիթը

bus[2:0] // bus վեկտորի երեք կրտսեր բիթը, bus[0:2] ձևը ճիշտ չէ, քանի որ ավագ բիթը  
// պետք է լինի ձախից

virtual\_addr[0:1] // virtual\_addr վեկտորի երկու ավագ բիթը

**integer** տիպը նշանով կամ առանց նշանի վեկտոր ռեգիստր է, որի լայնությունը որոշվում է համակարգչի կարգայնությամբ, բայց ամենաքիչը 32 բիթ է: **integer**-ը բանալի բառ է՝

**integer** i; // 32-բիթ ռեգիստր

**integer**-ը օգտագործվում է հաշիվ ներկայացնող փոփոխականների հայտարարման համար: Թեկուզ այդ նպատակով կարելի է օգտագործել **reg** տիպի փոփոխական, բայց ավելի հարմար է օգտագործել **integer**-ը: **reg** տիպը պահպանում է միայն առանց նշանի տվյալներ, այն դեպքում, երբ **integer**-ը պահպանում է նշանով տվյալներ: Օրինակ,

**integer** counter; // counter ընհանուր գործածության ռեգիստր.

**Initial**

counter = -1; // -1 թիվը պահվում է counter ռեգիստրում

Իրական տիպի հաստատունները և ռեգիստրները հայտարարվում են **real** բանալի բառով: Թվային համակարգերի նկարագրության համար **real** տիպը չի օգտագործվում: Այն կարող է օգտակար լինել խառը ազդանշանների համակարգերի թվային նմանակման համար: **real** տիպին կարելի է վերագրել արժեքներ տասնորդական կամ

էքսպոնենցիալ չափաձևով: Երբ **real** արժեք է վերագրվում **integer** տիպին, այդ արժեքը կլորացվում է դեպի մոտակա ամբողջ արժեքը: Օրինակ,

```
real delta; // delta-ն հայտարարել real տիպի
initial
begin
    delta = 4e10; // delta-ին վերագրել 4e10 էքսպոնենցիալ արժեք
    delta = 2.13; // delta-ին վերագրել 2.13 տասնորդական արժեք
end
integer i; // հայտարարել i-ն integer տիպի
initial
i = delta; // i-ն ստանում է 2 արժեք (2.13-ի կլորացումից)
```

Վերիլոգ նմանակումը կատարվում է ժամանակի նկատմամբ: Նմանակման ժամանակի արժեքի պահպանման համար օգտագործվում է հատուկ **time** ռեգիստր տվյալի տիպ: Ժամանակ տիպի ռեգիստրները հայտարարվում են **time** բանալի բառով: **time** տիպի ռեգիստրի լայնությունը (բիթերի թիվը) կախված է համակարգչի տիպից, բայց ամենաքիչը 64-բիթ է: Նմանակման ժամանակի ընթացիկ արժեքը ստանալու համար պետք է դիմել **\$time** համակարգային առաջադրանքին (task): Օրինակ.

```
time save_sim_time; // սահմանել save_sim_time անունով time տիպի փոփոխական
initial
    save_sim_time = $time; // պահպանել ընթացիկ ժամանակի արժեքը save_sim_time
    //ռեգիստրում
```

Նմանակման ժամանակը չափվում է վայրկյաններով: Սակայն նմանակման ժամանակի և ֆիզիկական շրթայում իրական ժամանակի միջև հարաբերակցությունը կարող է սահմանվել օգտագործողի կողմից: Նմանակման ժամանակի միավորը կարելի է տալ կոմպիլատորի **timescale** դիրեկտիվով: Օրինակ,

```
`timescale 1ns/1ps
```

դիրեկտիվով սահմանվում է նմանակման ժամանակի միավորը հավասար 1 նվ, իսկ ժամանակի հաշվման ճշտությունը 1 պվ է:

### 7.3.2.3 Ձանգվածներ

Ռեգիստր և ցանց տիպի տվյալները կարող են հայտարարվել նաև զանգվածների տեսքով: Կարելի է հայտարարել ցանկացած չափողականությամբ զանգված: Ձանգվածը հայտարարվում է հետևյալ չափաձևով`

<անվանումը><ինդեքս1><ինդեքս2>..**<ինդեքսN>**:

Օրինակներ`

```
integer count[0:7]; // 8 հատ count անունով integer տիպի փոփոխականների զանգված
reg bool[31:0]; // 32 հատ bool անունով 1-բիթ ռեգիստրներ
time chk_point[1:100]; // 100 հատ checkpoint անունով time տիպի փոփոխականներ
reg [4:0] port_id[0:7]; // 8 հատ port_id անունով 5-բիթ ռեգիստրների զանգված
integer matrix[4:0][0:255]; // integer տիպի փոփոխականների երկչափ զանգված
reg [63:0] array_4d [15:0][7:0][7:0][255:0]; //64-բիթ ռեգիստրների քառաչափ զանգված
```

```
wire [7:0] w_array2 [5:0]; // ցանց տիպի 8-բիթ 6 վեկտորների զանգված
wire w_array1[7:0][5:0]; // ցանց տիպի 1-բիթ փոփոխականների երկչափ զանգված
```

Չպետք է շփոթել զանգվածը ռեգիստր կամ ցանց տիպի վեկտորների հետ: Վեկտորը n-աբիթ լայնությամբ մեկ տարր է: Ջանգվածը բաղկացած է բազմաթիվ տարրերից՝ յուրաքանչյուրը 1-բիթ կամ n-աբիթ լայնությամբ:

Ստորև բերվում են վերևը սահմանված զանգվածների տարրերին արժեքների վերագրման օրինակներ՝

```
count[5] = 0; // count զանգվածի 5-րդ տարրին վերագրել 0 արժեք
chk_point[100] = 0; // check_point զանգվածի 100-րդ տարրին վերագրել 0 արժեք
port_id[3] = 0; // port_id զանգվածի 3-րդ տարրին (5-բիթ) վերագրել 0
matrix[1][0] = 33559; // matrix զանգվածի [1][0] տարրին վերագրել 33559
array_4d[0][0][0][0][15:0] = 0; // [0][0][0][0] հասցեով ռեգիստրի 15:0 բիթերին վերագրել 0
port_id = 0; // արգելված է՝ փորձ է արվում գրել ամբողջ զանգվածում: Գրել
// կարելի է զանգվածի միայն մեկ տարրում
```

Թվային համակարգերի նմանական համար հաճախ անհրաժեշտ է մոդելավորել ռեգիստրների ֆայլեր, ՍԿԸՀ և ՍԸՀ հիշողություններ: Վերիլոգում հիշողությունը մոդելավորվում է ռեգիստրների միաչափ զանգվածով: Ջանգվածի յուրաքանչյուր տարր կամ բառ հասցեավորվում է զանգվածի միակ ինդեքսով: Յուրաքանչյուր բառ կարող է լինել 1-բիթ կամ n-բիթ: Հիշողության հայտարարման օրինակներ՝

```
reg mem1bit[0:1023]; // mem1bit անունով 1-բիթ բառերի հիշողություն՝ 1024 բառ
// ծավալով
reg [7:0] membyte[0:1023]; // membyte անունով 8-բիթ բառերի հիշողություն՝ 1024
//բառով
membyte[511] // ընտրվում է մեկ 8-բիթ բառ 511 հասցեից:
```

### 7.3.2.4 Պարամետրեր

Վերիլոգը թույլ է տալիս մոդուլում սահմանել հաստատուններ **parameter** բանալի բառի միջոցով: Պարամետրը չի կարելի օգտագործել որպես փոփոխական: Պարամետրի տիպը և չափսը նույնպես կարելի է սահմանել.

```
parameter period = 5; // սահմանվում է period հաստատուն
parameter size = 64; // սահմանվում է size հաստատուն
parameter signed [15:0] sample; // ֆիքսված նշանով և միջակայքով պարամետր sample
```

Ծրագրում պետք է հնարավորինս խուսափել ֆիքսված արժեքներից: Պարամետրերի օգտագործմամբ կարելի է փոխել մոդուլի վարքագիծը՝ փոխելով պարամետրի արժեքը: Պարամետրի արժեքը կարելի է որոշել մոդուլի ստեղծման ժամանակ.

```
module abcd();
parameter period=5;//period պարամետրի սահմանում
endmodule
```

Մոդուլում պարամետրի տրված արժեքը հնարավոր է փոխել մոդուլի օրինակի տեղադրման ժամանակ: Դա հնարավորություն է տալիս նույն մոդուլն օգտագործել պարամետրի տարբեր արժեքներով: Օրինակ.

```
module top();
abcd #12 i1(); //abcd մոդուլի i1 օրինակի տեղադրում period պարամետրին 12 արժեք
           // վերագրելով
abcd #25 i2(); //abcd մոդուլի i2 օրինակի տեղադրում period պարամետրին 25 արժեք
           // վերագրելով
endmodule
```

Եթե մոդուլում սահմանված են մեկից ավելի պարամետրեր, դրանց արժեքների վերադրոշման ժամանակ պետք է պահպանել մոդուլում դրանց հայտնվելու հաջորդականությունը.

```
module abcd();
    parameter period=5; //period պարամետրի սահմանում 5 արժեքով
    parameter size=10; // size պարամետրի սահմանում 10 արժեքով
endmodule
```

```
module top();
abcd #(12, 8) i1(); //abcd մոդուլի i1 օրինակի տեղադրում period պարամետրին
           //վերագրելով 12 արժեք, size պարամետրին` 8 արժեք
abcd #25 i2(); //abcd մոդուլի i2 օրինակի տեղադրում period պարամետրին
           //վերագրելով 25 արժեք
endmodule
```

Եթե մոդուլում սահմանված պարամետրերից մեկի կամ մի քանիսի արժեքները պետք է փոխել մոդուլի օրինակի տեղադրման ժամանակ, այդ վերագրումը կարելի է կատարել բացահայտ նշելով պարամետրի անունը.

```
module top();
abcd #(.size(8)) i1(); // abcd մոդուլի i1 օրինակի տեղադրում size
           //պարամետրին` 8 արժեք վերագրելով
endmodule
```

Այն դեպքերում, երբ պետք ունենալ պարամետր, որին մոդուլի ներսում տրված արժեքը հնարավոր չլինի փոխել, պետք է այդպիսի պարամետրը լոկալ հայտարարվի` **localparam** բանալի բառով: Հետևյալ օրինակում սահմանված period լոկալ պարամետրի արժեքը հնարավոր չէ փոխել մոդուլի օրինակի տեղադրման ժամանակ:

```
module abcd();
    parameter half_period=10; // half_period պարամետրի սահմանում 10 արժեքով
    localparam period=20; //period լոկալ պարամետրի սահմանում 20 արժեքով
endmodule
```

Հետևյալ օրինակում period լոկալ պարամետրի արժեքը հնարավոր է անբացահայտ փոխել մոդուլի օրինակի տեղադրման ժամանակ:

```
module abcd();
    parameter half_period=10; // պարամետրի սահմանում 10 արժեքով
    localparam period=2* half_period; // լոկալ պարամետրի սահմանում` 20 արժեքով
endmodule
```

```
module top();
```

```
abcd #10 i1(); // abcd մոդուլի i1 օրինակի տեղադրում half_period պարամետրին
//10 արժեք վերագրելով: Միաժամանակ period պարամետրն ընդունում 20
արժեք
endmodule
```

### 7.3.2.5 Սիմվոլների շղթաներ

Սիմվոլների շղթաները կարող են պահվել ռեգիստրներում: **reg** փոփոխականի լայնությունը (բիթերի թիվը) պետք է բավարար լինի շղթան տեղավորելու համար: Շղթայի յուրաքանչյուր սիմվոլ (ASCII սիմվոլ) զբաղեցնում 8 բիթ: Եթե ռեգիստրի լայնությունը շղթայի բիթերի թվից մեծ է, ապա Վերիլոգը շղթայից ձախ բիթերը լրացնում է 0-ներով: Եթե ռեգիստրի լայնությունը շղթայի բիթերի թվից փոքր է, ապա Վերիլոգը կտրում է չտեղավորվող ձախ բիթերը: Ապահով է շղթան հայտարարել մի քիչ ավելի երկար, քան անհրաժեշտ է: Օրինակ.

```
reg [8*19:1] string_value; // հայտարարել 19 բայթ լայնությամբ reg տիպի
փոփոխական
initial
string_value = "Hello Verilog World"; // շղթան կարող է տեղավորվել string_value
//ռեգիստրում
```

Հատուկ սիմվոլները թույլ են տալիս չափաձևել ցուցադրվող տեքստերը: Հատուկ սիմվոլները կարող են ցուցադրվել միայն, եթե դրանք նախորդվում են \ կամ % հատուկ սիմվոլներով, ինչպես ցույց է տրված աղյուսակ 7.1–ում:

Աղյուսակ 7.1

Հատուկ սիմվոլներ	Չուցադրվող սիմվոլը
\n	նոր տող
\t	թափ
%%	%
\\	\
\"	"
\ooo	1-3 ութական թվանշանով գրված սիմվոլ

### 7.3.3 Համակարգային առաջադրանքներ և կոմպիլատորի դիրեկտիվներ

Վերիլոգը տրամադրում է ստանդարտ ներդրված համակարգային առաջադրանքներ (ենթածրագրեր), որոնք հեշտացնում են ծրագրավորումը: Այդպիսիք են էկրանի վրա տվյալների կամ տեքստի ցուցադրումը, փոփոխականների արժեքների հսկումը, նմանակման կանգը կամ ավարտը և էլի շատ ստանդարտ առաջադրանքներ, որոնք կարելի գտնել Վերիլոգ ծրագրի նկարագրության ստանդարտներում [11-14]: Համակարգային առաջադրանքները տրվում են **\$<բանալի բառ>** ձևով: Ամենակարևոր առաջադրանքներից է ինֆորմացիայի ցուցադրումը.

```
$display(p1, p2, p3,....., pn);
```

որտեղ p1, p2, p3,..., pn- կարող են լինել չակերտների մեջ ներառած սիմվոլների շղթաներ կամ փոփոխականներ:

Շղթաները կարելի է չափաձևել՝ օգտագործելով աղյուսակ 7.2–ում ցույց տված կառավարող սիմվոլները:

Աղյուսակ 7.2

Չափաձև	Ցուցադրում
%d կամ %D	Ցուցադրել փոփոխականը տասական թվանշաններով
%b կամ %B	Ցուցադրել փոփոխականը երկուական թվանշաններով
%s կամ %S	Ցուցադրել սիմվոլների շղթա
%h կամ %H	Ցուցադրել փոփոխականը տասնվեցական թվանշաններով
%c կամ %C	Ցուցադրել ASCII սիմվոլ
%m կամ %M	Ցուցադրել հիերախիկ անվանում
%v կամ %V	Ցուցադրել փոփոխականի ուժը
%o կամ %O	Ցուցադրել փոփոխականը ութական թվանշաններով
%t կամ %T	Ցուցադրել ընթացիկ ժամանակի չափաձևով
%e կամ %E	Ցուցադրել իրական թիվը էքսպոնենցիալ չափաձևով
%f կամ %F	Ցուցադրել իրական թիվը տասնորդական չափաձևով
%g կամ %G	Ցուցադրել իրական թիվը տասնորդական կամ էքսպոնենցիալ չափաձևով՝ որն ավելի կարճ է

Ստորև բերված են **\$display** առաջադրանքի կիրառության օրինակներ:

**Օրինակ 7.2.** Ցուցադրել սիմվոլների շղթաներ:

```
module displaying_variables;
reg [8*12:1] stringvar;

initial //Այս վարքագծային նկարագրության կառուցվածքի մասին ավելի ուշ
begin //բլոկի սկիզբը
    stringvar = "Hello World"; //փոփոխականին վերագրել շղթա տիպի տվյալ
    $display("%s", stringvar); //ցուցադրել stringvar-ի արժեքը շղթա չափաձևով
    $display("Hello World"); // ցուցադրել Hello World շղթան
    $display("%s is stored as %h", stringvar, stringvar); // ցուցադրել stringvar-ի
//արժեքը երկու անգամ. նախ շղթա, ապա տասնվեցական չափաձևով
end //բլոկի վերջ
endmodule
```

Նմանակումից հետո համակարգչի էկրանին կտեսնենք՝

```
# Hello World
# Hello World
# Hello World is stored as 0048656c6c6f20576f726c64
```

**Օրինակ 7.3.** Ցուցադրել նմանակման ընթացիկ ժամանակը:

**\$display(\$time);** // ցուցադրել նմանակման ընթացիկ ժամանակի արժեքը

Ենթադրենք ցուցադրվում է # 230: Դա նշանակում է, որ **\$display(\$time)** առաջադրանքի կատարման ժամանակ նմանակման ընթացիկ ժամանակի արժեքը եղել է ժամանակի 230 միավոր:

#### Օրինակ 7.4. Ցուցադրել փոփոխականի արժեքը:

```
//Ենթադրենք պետք է ցուցադրել addr փոփոխականի արժեքը և, որ $display  
առաջադարանքի կատարման պահին նմանական ժամանակը 200 է:  
reg [31:0] data_word;  
$display("At time %d data_word is %h", $time, data_word); //ցուցադրել տասնվեցական  
$display("At time %d data_word is % b", $time, data_word); //ցուցադրել երկուական
```

Կցուցադրվի՝

```
#At time 200 data_word is 1f5e
```

```
#At time 200 data_word is 0001111101011110
```

Ժամանակի 200 պահին data\_word փոփոխականի տասնվեցական արժեքը եղել է 1f5e, երկուական արժեքը՝ 0001111101011110:

#### Օրինակ 7.5. Հատուկ սիմվոլների ցուցադրում:

```
//Հատուկ սիմվոլների ցուցադրում՝ նոր տող(\n) և \  
$display("This is a \n multiline string with a \\ sign");
```

Կցուցադրվի՝

```
# This is a
```

```
# multiline string with a \ sign
```

Փոփոխականների արժեքների փոփոխությունների հսկումը (monitoring) կարելի է իրականացնել **\$monitor** համակարգային առաջադրանքով.

**\$monitor**(p1,p2,p3,.....,pn);

Այս առաջադրանքը ցուցադրում է p1,p2,p3,.....,pn փոփոխականների արժեքները: Չափաձևերը տրվում են ինչպես **\$display** առաջադրանքում: Ողջ ցուցակի պարամետրերի արժեքները ցուցադրվում են յուրաքանչյուր անգամ, երբ դրանցից որևէ մեկը փոխվում է: Ի տարբերություն **\$display**-ի, **\$monitor**-ը պետք է նշվի միայն մեկ անգամ: Ժամանակի յուրաքանչյուր պահի միայն մեկ **\$monitor** առաջադրանքի ցուցակ կարող է ակտիվ լինել: Եթե ծրագրում կան մեկից ավելի **\$monitor** առաջադրանքներ, ապա դրանցից ակտիվ կլինի միայն վերջինը: Փոփոխականների արժեքների փոփոխությունների հսկումը կարելի է միացնել կամ անջատել հետևյալ առաջադրանքներով՝

**\$monitoron**;

**\$monitoroff**;

Հսկումը միանում է ավտոմատ, երբ նմանակումն սկսվում է և կարող է կառավարվել **\$monitoron** և **\$monitoroff** առաջադրանքներով: Ստորև բերված է **\$monitor** առաջադրանքի կիրառության օրինակ:

#### Օրինակ 7.6. Նմանական ընթացքում փոփոխականների հսկում:

```
//հետևել նմանական ժամանակին և ազդանշանների փոփոխություններին  
module variable_monitoring;  
reg a, b;  
initial  
begin  
    $monitor($time, " Value of signals a = %b b = %b", a,b); // $time համակարգային
```

```
// առաջադրանքը վերադարձնում է նմանական ընթացիկ ժամանակը
a=1'b0; b=1'b0;
#10 a=1'b0; b=1'b1; // #-ը նախորդում է ժամանակի աճի տրվող արժեքին, որն
//ավելացվում է նմանական ժամանակին
#10 a=1'b1; b=1'b0;
#10 a=1'b1; b=1'b1;
```

**end**

Նմանակումից հետո կցուցադրվի՝

# 0 Value of signals a = 0 b = 0

# 10 Value of signals a = 0 b = 1

# 20 Value of signals a = 1 b = 0

# 30 Value of signals a = 1 b = 1

Թվարկենք ևս երկու համակարգային առաջադրանք՝

**\$stop** առաջադրանքով նմանակումը կանգ է առնում: Այն կարելի վերսկսել, օգտագործվում է նմանական արդյունքների ուսումնասիրման համար ինտերակտիվ ռեժիմում:

**\$finish** առաջադրանքը ավարտում է նմանակումը:

**Օրինակ 7.7.** Նմանական կառավարում:

```
// Ընդհատել նմանակումը ժամանակի 100 պահին միջանկյալ արդյունքների
//ուսումնասիրման համար, այնուհետև կարելի շարունակել նմանակումը
// Նմանակումն ավարտել ժամանակի 1000 պահին
initial // ժամանակը = 0
begin
```

```
    a = 0;
```

```
    b = 1;
```

```
    #100 $stop; // նմանակումը կընդհատվի ժամանակը = 100 պահին
```

```
    #900 $finish; // նմանակումը կավարտվի ժամանակը = 1000 պահին
```

**end**

Կոմպիլատորի դիրեկտիվները սահմանվում են “<բանալի բառ>” կառուցվածքով:

**`define** դիրեկտիվն օգտագործվում է տեքստային մակրոսներ սահմանելու համար.

**`define** macro\_name value// մակրոսի անվանումը և արժեքը

Վերիլոգ կոմպիլատորը մակրոսի տեքստը փոխարինում է նրա արժեքով, որտեղ որ հանդիպում է **<macro\_name>**-ի:

**Օրինակ 7.8.** **`define** դիրեկտիվի օգտագործում:

//սահմանել մակրոսի տեքստ, որով սահմանվում է բառի լայնությունը

**'define** WORD\_SIZE 32 //ծրագրի տեքստում բառի լայնության համար պետք է

//օգտագործել 'WORD\_SIZE տեքստային մակրոսը



**`include`** դիրեկտիվը թույլ է տալիս Վերիլոգ ծրագրի տեքստում ներառել մոդուլներ այլ ֆայլերից:

**`include`** file\_name // file\_name – ներառվող ֆայլի անվանումը:

**`include`** դիրեկտիվը հաճախ է օգտագործվում թեստավորման մոդուլի ֆայլում նախագծի մոդուլի ֆայլի ներառման համար: Օրինակ. ներառել design.v ֆայլում գտնվող նախագծի մոդուլները թեստավորման մոդուլում, որը գտնվում է testbench.v ֆայլում.

```
'include design.v // այս դիրեկտիվը գտնվում է թեստավորման մոդուլի ֆայլում
module testbench;
...
controller dut( ); //controller անվանումով մոդուլը գտնվում է design.v ֆայլում
...
endmodule // testbench
```

**`timescale`** դիրեկտիվի կիրառությունը քննարկվել է 7.3.2.2 բաժնում:

## 7.4. Մոդուլներ և մատույցներ

### 7.4.1 Վերիլոգ մոդուլներ

Արդեն քննարկվել են հիերարխիկ նախագծման սկզբունքները: Վերիլոգ հիերարխիկ նախագիծը բաղկացած է մոդուլներից: Մինչ այժմ մենք մոդուլը դիտարկում էինք առանց մուտք/ելք կապերի (ինտերֆեյսի) և ներքին պարունակության նկարագրության: Այժմ մենք կգբաղվենք մոդուլի այդ մանրամասների քննարկմամբ: Մոդուլը կազմված է առանձին մասերից՝

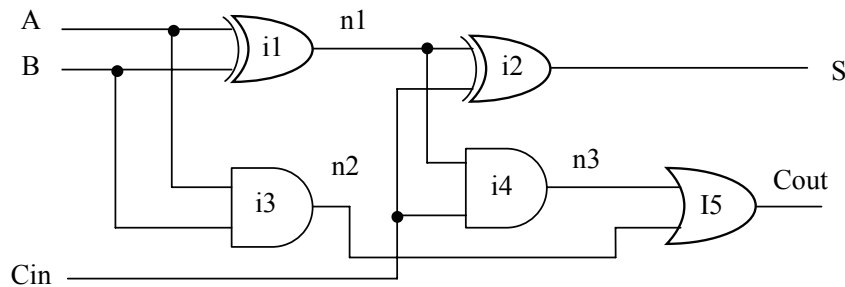
- մոդուլի անվանում, մուտք/ելք մատույցների ցուցակ;
- մատույցների հայտարարում (**input**, **output**);
- պարամետրեր (**parameter**);
- ռեգիստրների, ցանցերի և այլ փոփոխականների հայտարարում (**reg**, **wire**);
- տվյալների վերագրումներ (**assign**);
- հիերարխիկ ցածր մակարդակի մոդուլների օրինակների տեղադրում (**instantiation**);
- վարքագծային նկարագրության կառուցվածքներ (**always**, **initial**);
- ֆունկցիաներ և խնդիրներ (**function**, **task**);
- մոդուլի ավարտ (**endmodule**)

Մոդուլը միշտ սկսվում է **module** բանալի բառով: Մոդուլի անվանումը, մուտքի/ելքի մատույցների ցուցակը և հայտարարումը, պարամետրերի սահմանումը պետք է տեղադրվեն մոդուլի սկզբում: Մնացած բաղադրիչները կարող են տեղադրվել ցանկացած հաջորդականությամբ: Մոդուլում պարտադիր են՝ **module** բանալի բառը,

մոդուլի անվանումը և **endmodule** բանալի բառը, մնացած բաղադրիչները պարտադիր չեն: Օրինակ, եթե մոդուլը չունի տվյալների փոխանակման մուտքի/ելքի մատույցներ, ապա մուտքի/ելքի մատույցների ցուցակ և մատույցների հայտարարում (**input**, **output**) բաղադրիչները կբացակայեն:

Նույն Վերիլոգ ֆայլում կարելի է որոշել բազմաթիվ մոդուլներ:

Նկ. 7.12–ում ցույց է տրված 1-բիթ լրիվ գումարիչի կառուցվածքային սխեման:

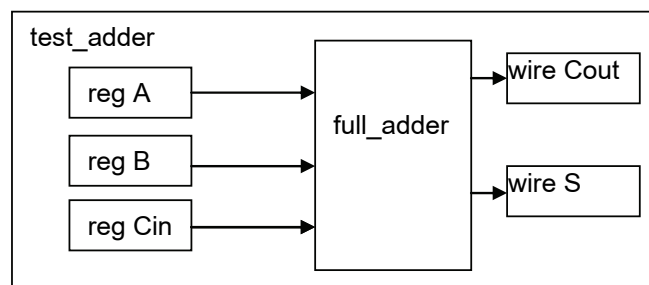


Նկ. 7.12. 1-բիթ լրիվ գումարիչի սխեման

**Օրինակ 7.9.** 1-բիթ լրիվ գումարիչը մոդելավորող `full_adder` անվանումով Վերիլոգ մոդուլ:

```
module full_adder(Cin, A, B, S, Cout); //մոդուլի անվանումը և մուտքի/ելքի
                                     //մատույցների ցուցակը
input Cin, A, B;                    //մուտքի մատույցների հայտարարում
output S, Cout;                     //ելքի մատույցների հայտարարում
wire n1, n2, n3;                   //ցանց տիպի փոփոխականների հայտարարում
xor i1(n1, A, B); //օգտագործվող տրամաբանական տարրերի
                               //օրինակների տեղադրում,
xor i2(S, Cin, n1); // ուշադրություն դարձրեք տրամաբանական
                               //տարրերի կապերին
and i3(n2, A, B);
and i4(n3, Cin, n1);
or i5(Cout, n2, n3);
endmodule
```

Այժմ նկարագրենք բարձր մակարդակի մոդուլը՝ `test_adder`, որտեղ տեղադրվում է `full_adder` մոդուլը: Այս մոդուլով կարելի է ստուգել `full_adder` մոդուլը:



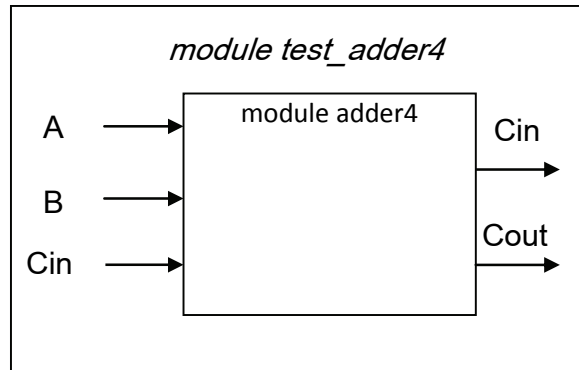
Նկ. 7.13. `full_adder` մոդուլի թեստավորումը `test_adder` անվանումով թեստավորման միջավայրում

### Օրինակ 7.10. *full\_adder* մոդուլի թեստավորում

```
`timescale 1ns/10ps //մանական քայլը 1նվ է, ճշտությունը՝ 10պվ
module test_adder; //այս մոդուլը մուտքի/ելքի կապեր չունի
reg cin, a, b; //նկարագրվում են reg տիպի, նրանց վերագրվելու են արժեքներ
wire s, cout; // wire տիպ՝ դրանց արժեքները ձևավորվում են
//տրամաբանական տարրերի ելքերում
full_adder dut(cin, a, b, s, cout); //գումարիչի մոդուլի օրինակի տեղադրում
// initial վարքագծային բլոկ, կուսումնասիրենք ավելի ուշ
Initial
    begin //մեկից ավելի հրամաններ պարունակող բլոկը սկսվում է
        //begin և ավարտվում end բանալի բառերով
        $monitor($time, "a = %b, b= %b, cin= %b, cout= %b, s= %b\n", a,b,cin,cout,s);//
        // $monitor համակարգային ֆունկցիա՝
        //մանական ժամանակ փոփոխականների արժեքները
        //տեսնելու համար
a=1'b0; b=1'b0; cin=1'b0; //մուտքերին տրվում են թեստային
        // հավաքածուներ՝ վեկտորներ
        #10 a=1'b0; b=1'b0; cin=1'b1; // #10 – նշանակում է 10 միավոր
// հապաղում
        #10 a=1'b0; b=1'b1; cin=1'b0;
        #10 a=1'b0; b=1'b1; cin=1'b1;
        #10 a=1'b1; b=1'b0; cin=1'b0;
        #10 a=1'b1; b=1'b0; cin=1'b1;
        #10 a=1'b1; b=1'b1; cin=1'b0;
        #10 a=1'b1; b=1'b1; cin=1'b1;
        #10 $finish; //ավարտել նմանակումը
    end // բլոկի ավարտ
endmodule
```

### 7.4.2 Մոդուլների մատույցներ

*Մատույցները* ծառայում են ինտերֆեյս մոդուլի և նրա շրջապատի միջև: Համակարգի մյուս բաղադրիչները կարող են տվյալներ փոխանակել մոդուլի հետ միայն մատույցների միջոցով: Մոդուլի սահմանումը պարունակում է մատույցների ցուցակը: Դիտարկենք նկ. 7.14–ի քառաբիթ գումարիչը: Այս նկարում test\_adder4–ը բարձր մակարդակի մոդուլն է, որում տեղադրվում է քառաբիթ գումարիչը (adder4): test\_adder4 մոդուլը մատույցներ չունի: Քառաբիթ գումարիչին մուտքային տվյալները տրվում են test\_adder մոդուլից մուտքային A, B, Cin մատույցների միջոցով, իսկ գումարիչից S, Cout արդյունքը test\_adder մոդուլ է փոխանցվում ելքի մատույցներով:



Նկ. 7.14. *adder4 մոդուլի մուտքի/ելքի մատույցները, որոնցով տվյալները փոխանակվում են թեստավորման միջավայրի հետ*

Մատույցների ցուցակի բոլոր մատույցները պետք է հայտարարվեն մոդուլում:  
Մատույցը հայտարարվում է բանալի բառով`

**input** - մուտքի մատույց

**output** - ելքի մատույց

**inout** - երկուղղված մատույց

Օրինակ, քառաբիթ գումարիչի մուտքի/ելքի մատույցների ցուցակը և մատույցների հայտարարությունն ունեն հետևյալ տեսքը`

```

module adder4(A, B, Cin, S, Cout);
//մատույցների հայտարարման սկիզբ
output [3:0] S;
output Cout;

input [3:0] A, B;
input Cin;
// մատույցների հայտարարման ավարտ
...
<մոդուլի ներքին նկարագրություն>
...
endmodule

```

Նկատենք, որ Վերիլոգում բոլոր մատույցները անբացահայտ հայտարարվում են **wire** տիպի: Սակայն, եթե ելքի մատույցը պահպանում է վիճակը, այն պետք է հայտարարվի **reg** տիպի: Ստորև բերված օրինակում D տրիգերի ելքը ծառայում է միաժամանակ մոդուլի ելքի մատույց և հայտարարվում է **reg** տիպի փոփոխական:

```

module DFF(q, d, clk, reset);
input d, clk, reset;
output q;
reg q; // ելքի q մատույցը պահպանում է արժեքը, հետևաբար այն հայտարարվում է reg տիպի
...
...
endmodule

```

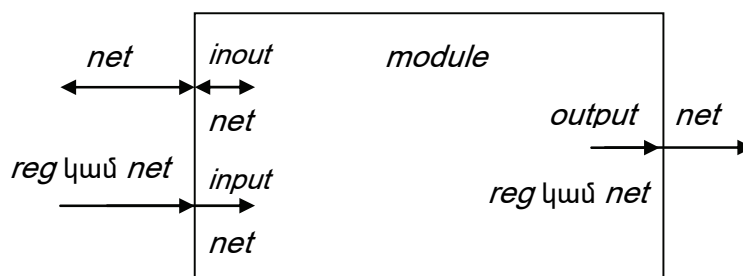
Մատույցների ֆունկցիաները և միացման կանոնները ցույց են տրված նկ. 7.15–ում: Մատույցը կազմված է երկու մասից՝ ներքին և արտաքին:

Մուտքի մատույցը մոդուլի ներսում միշտ **net** տիպի է, իսկ արտաքինից այն կարող է միացված լինել **net** կամ **reg** տիպի փոփոխականի:

Մոդուլի ներսից ելքի մատույցը կարող է լինել **net** կամ **reg** տիպի, արտաքինից այն կարող է միացված լինել միայն **net** տիպի փոփոխականի:

Մուտքի/ելքի երկուղղված **inout** մատույցը և մոդուլի ներսից և դրսից կարող է միացված լինել միայն **net** տիպի փոփոխականի:

Թույլատրվում է տարբեր չափսի մատույցներ միացնել միմյանց, երբ կատարվում են միջմոդուլային միացումներ: Սակայն սովորաբար Վերիլոգ կոմպիլյատորը գեներացնում է մատույցների չափսերի անհամաձայնության զգուշացում (warning):



Նկ. 7.15. Մատույցների ֆունկցիաները և միացման կանոնները

Վերիլոգում կարելի է մատույցները թողնել չմիացած: Օրինակ, շատ դեպքերում օգտագործվում են որոշ ելքի մատույցներ միայն ծրագրի կարգաբերման նպատակով և դրանք արտաքին ազդանշանների միացնելը կարող է իմաստ չունենալ: Կարելի է մատույցը թողնել չմիացած մոդուլի օրինակի տեղադրման ժամանակ, ինչպես ցույց է տրված ստորև:

```
module full_adder4 (Cin, A, B, S, Cout);
    full_adder i1(cin, a, b, s, ); // Cout ելքի մատույցը միացված չէ
```

Հետևյալ օրինակում մատույցների միացման կանոնները պահպանված չեն:

```
module top; // full_adder4 մոդուլի թեստավորման բարձր մակարդակի մոդուլ
    //հայտարարել միացման փոփոխականները
    reg [3:0]a,b;
    reg cin;
    reg [3:0] s;
    wire cout;

    //տեղադրել full_add4-ի օրինակը, անվանել այն i1
    full_adder4 i1(a, b, cin, s, cout); //չթուլատրված միացում՝ full_adder4 մոդուլի ելքի s
        //մատույցը միացված է ռեգիստրի տիպի s փոփոխականի top մոդուլում
    .
    < full_adder4-ի մուտքային ազդանշաններ՝ սիմվոլներ>
    .
endmodule
```

Այստեղ խախտումն այն է, որ *s* փոփոխականը *top* մոդուլում հայտարարվել է **reg** տիպի: Խնդիրը լուծվում է, եթե *top* մոդուլում *s* փոփոխականը հայտարարվի **wire** տիպի:

Մոդուլի տեղադրված օրինակում օգտագործվող ազդանշանների և մոդուլի ներսում սահմանված մատույցների միջև կա միացումների իրականացման երկու եղանակ՝ ըստ մատույցների կարգավորված ցուցակի և ըստ մատույցների անվանումների: Չի թույլատրվում այս երկու եղանակները խառնել՝ օգտագործել միաժամանակ:

Կարգավորված ցուցակով միացումների դեպքում մոդուլի օրինակում ազդանշանները պետք է թվարկվեն նույն հաջորդականությամբ, ինչ դրանց համապատասխանող մատույցները մոդուլի սահմանման մեջ:

Այսպես **օրինակ 7.9**-ում սահմանվել է `full_adder` մոդուլը հետևյալ մատույցներով, **module** `full_adder(Cin, A, B, S, Cout);` // մոդուլի մատույցների ցուցակը

Իսկ **օրինակ 7.10**-ում սահմանված `test_adder` մոդուլում տեղադրվել է այդ մոդուլի *dut* օրինակը.

```
full_adder dut(c_in, x, y, sum, c_out); //գումարիչի մոդուլի օրինակի տեղադրում, այստեղ
// ազդանշանները թվարկված են ճիշտ նույն հաջորդականությամբ, ինչ
//համապատասխան մատույցները մոդուլի մատույցների ցուցակում՝ c-in
//ազդանշանը միանում է Cin մատույցին, x-ը՝ A մատույցին և այլն
```

Մեծ թվով մատույցների դեպքում դժվար է հիշել մատույցների հաջորդականությունը: Վերիլոգն ընձեռնում է ազդանշանների միացումները մատույցներին ըստ անվանումների կատարելու հնարավորություն: Այդ դեպքում ազդանշանների հերթականությունը մոդուլի տեղադրված օրինակում կարող է լինել կամայական: Օրինակ 7.10-ում տեղադրել `full_adder` մոդուլի օրինակը՝ ազդանշանները միացնել ըստ անվանումների.

```
full_adder dut(.A(x), .B(y), .Cin(c_in), .Cout(c_out), .S(sum));
```

### 7.4.3 Հիերարխիկ անվանումներ

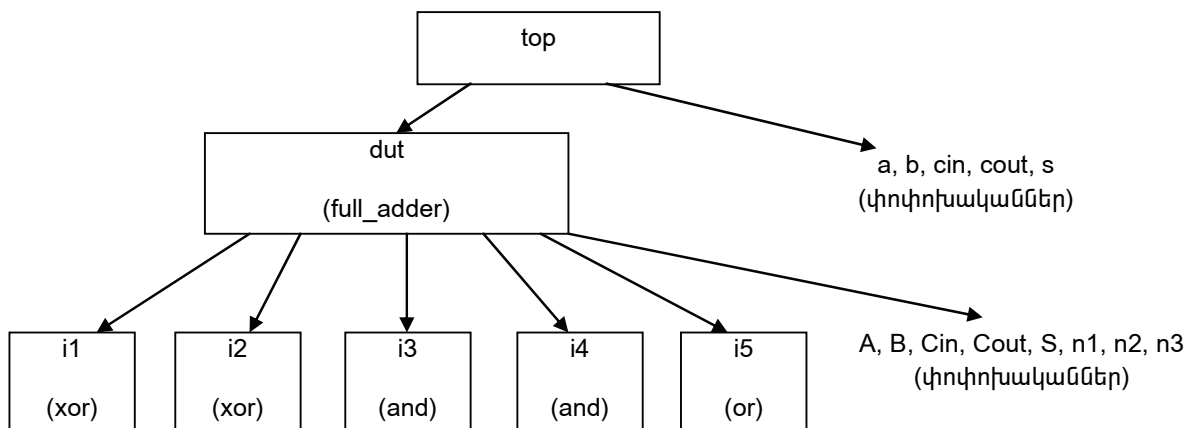
Ինչպես նշվել է, Վերիլոգը հնարավորություն է տալիս իրագործել հիերարխիկ նախագծում: Յուրաքանչյուր մոդուլի օրինակ, փոփոխական կամ ազդանշան սահմանվում է իր անվանումով: Յուրաքանչյուր իդենտիֆիկատոր նախագծում ունի իր ուրույն տեղը նախագծի հիերարխիայում: Հիերարխիկ անվանման հղումով յուրաքանչյուր իդենտիֆիկատոր նախագծում ստանում է անհատական անվանում: Հիերարխիկ անվանումը կետով (".") բաժանված իդենտիֆիկատորների ցուցակ է: Դա թույլ է տալիս դիմել որևէ իդենտիֆիկատորի նախագծի ցանկացած տեղից՝ նշելով այդ իդենտիֆիկատորի լրիվ հիերարխիկ անվանումը:

Նկ. 7.16-ում ցույց է տրված օրինակ 7.9-ի լրիվ գումարիչի և նկ. 7.14-ի թեստավորման նախագծի հիերարխիան:

Նկ 7.16-ի հիերարխիկ անվանումները հետևյալն են.

top	top.dut.i1	top.dut.A
top.a	top.dut.i2	top.dut.B
top.b	top.dut.i3	top.dut.Cin
top.cin	top.dut.i4	top.dut.Cout
top.cout	top.dut.i5	top.dut.S
top.s	top.dut.n1	top.dut.n3
top.dut	top.dut.n2	

Օրինակ, `$display("n1=%b\n", top.dut.n1)` կառուցվածքով կցուցադրվի full\_adder մոդուլի n1 փոփոխականի արժեքը



Նկ. 7.16. Լրիվ գումարիչի թեստավորման նախագծի հիերարխիան

#### 7.4.4. Խնդիրներ

**Խնդիր 7.3.** Վերիլոգով գրել հետևյալ թվերը.

- ա. Տասական 121 թիվը ներկայացնել իբրև 8-բիթ չափսով երկուական թիվ:
- բ. 8-բիթ տասնվեցական թիվը, որի բոլոր բիթերը **z** են:
- գ. Տասական -3 թիվը 4-բիթ երկուական չափաձևով: Գրել այս թիվը՝ 2-ի լրացումով չափաձևով:

**Խնդիր 7.4.** Հետևյալ շղթաներից որո՞նք են ճիշտ: Սխալներն ուղղել:

- ա. "This string displays the % sign":
- բ. "sum = ad1 + ad2":
- գ. "Octal number \007":
- դ. "This is a backslash \ character\n":

**Խնդիր 7.5.** Հետևյալ իդենտիֆիկատորներից որո՞նք են ճիշտ:

- ա. system1:
- բ. 1reg:
- գ. \$latch:

**Խնդիր 7.6.** Հայտարարել հետևյալ փոփոխականները:

- ա. bus անվանումով 10-բիթ ցանց տիպի վեկտոր:

բ. 64-բիթ վեկտոր ռեգիստր, ավագ բիթի կարգաթիվը 63 է:

գ. count անվանումով ամբողջ փոփոխական:

դ. delay անվանումով ժամանակ տիպի փոփոխական:

ե. array\_20 անվանումով զանգված, որը պարունակում է ամբողջ տիպի 20 տարր:

զ. stack անվանումով հիշողություն, որը բաղկացած է 64-բիթ 256 բառից:

**Խնդիր 7.7.** Ի՞նչ կցուցադրվի հետևյալ առաջադրանքներով՝

ա. latch = 4'd12;

\$display("The current value of latch = %b\n", latch);

բ. in\_reg = 3'd2;

\$monitor(\$time, " In register value = %b\n", in\_reg[2:0]);

գ. `define MEM\_SIZE 1024

\$display("The maximum memory size is %h", `MEM\_SIZE);

**Խնդիր 7.8.** Որո՞նք են մոդուլի հիմնական բաղադրիչները: Այդ բաղադրիչներից որո՞նք են պարտադիր:

**Խնդիր 7.9.** Մոդուլը, որը չի փոխադրում արտաքին միջավայրի հետ, արդյո՞ք ունի մուտքի/ելքի մատույցներ: Պե՞տք է այդպիսի մոդուլի սահմանման մեջ լինի մատույցների ցուցակ:

**Խնդիր 7.10.** Չորս զուգահեռ մուտքով shift\_reg4 տեղաշարժող ռեգիստրն ունի din[3:0] և clk մուտքի ու qout[3:0] ելքի գծեր:

(ա) Գրե՞ք այդ մոդուլի սահմանումը՝ նշելով միայն մատույցների ցուցակը և մատույցների հայտարարությունը, մոդուլի ներքին պարունակությունը նկարագրելու հարկ չկա:

(բ) Հայտարարել թեստավորման բարձր մակարդակի մոդուլը, սահմանել REG\_IN (4 bit) և CLK (1 bit) որպես ռեգիստր տիպի փոփոխականներ, REG\_OUT (4 bit)-ը՝ որպես ցանց տիպի փոփոխական: Տեղադրել shift\_reg-ի sr1 անվանումով օրինակը և մատույցները միացնել հաջորդաբար:

(գ) Միացնել sr1 օրինակի ազդանշանները ըստ մատույցների անվանումների:

(դ) Գրել sr1 օրինակի հիերարխիկ անվանումը: Գրել sr1 օրինակի clk and din մատույցների հիերարխիկ անվանումները:



## 7.5 Տրամաբանական տարրերի մակարդակով մոդելավորում

### 7.5.1 Վերիլոգ ներդրված պրիմիտիվներ

Թվային համակարգերի մոդելավորումը կարելի է իրականացնել վերացարկման տարրեր մակարդակներով՝ տրամաբանական տարրերի մակարդակով, տվյալների հոսքերի նկարագրման մակարդակով, վարքագծային նկարագրության մակարդակով: Տրամաբանական տարրերի մակարդակով մոդելավորումը ցածր վերացարկման մակարդակի մոդելավորում է, որը միարժեք համապատասխանում է տրամաբանական տարրերի վրա կառուցված թվային շղթային: Տարրերի մակարդակով մոդելավորման ուսումնասիրությունը հնարվորություն կտա սահուն անցնել ավելի բարձր մակարդակի մոդելավորման եղանակներին:

Հիմնական տրամաբանական տարրերին համապատասխանող պրիմիտիվները (մոդուլները) ներկառուցված են Վերիլոգում, և կարիք չկա դրանք սահմանել: Այդ տրամաբանական պրիմիտիվները թվարկված են աղյուսակ 7.3-ում, համապատասխան տարրերի սինվոլները մուտք/ելք ազդանշանային գծերի նշանակումներով ցույց են տրված նկ. 7.17-ում:

Աղյուսակ 7.3

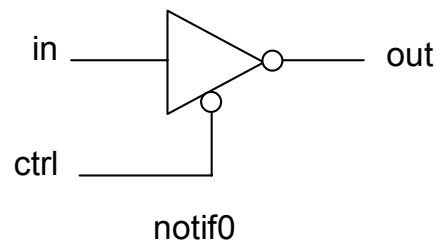
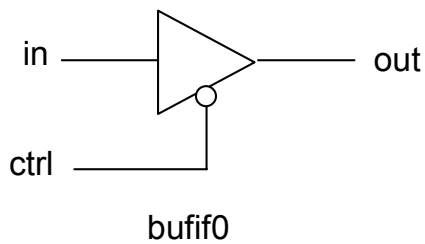
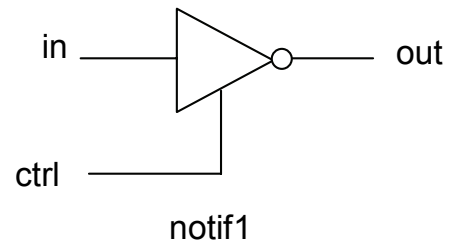
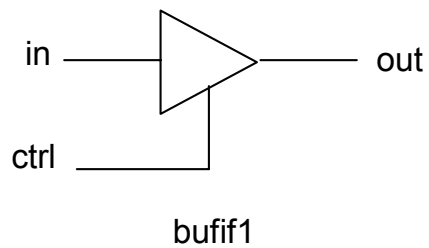
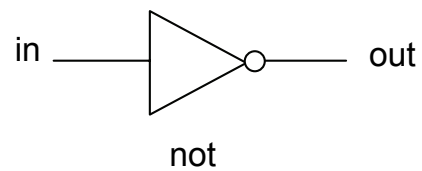
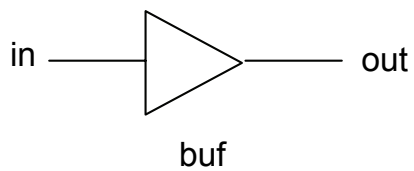
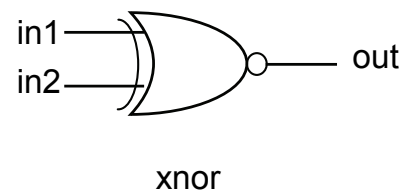
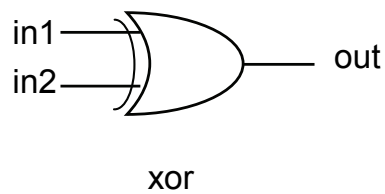
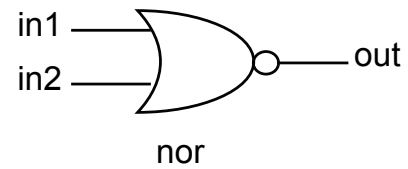
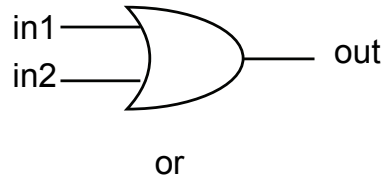
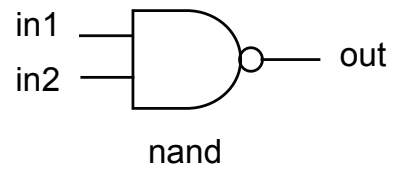
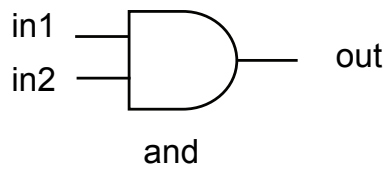
Պրիմիտիվի Վերիլոգ անվանումը	Համապատասխան տրամաբանական տարրը
and	ԵՎ
nand	ԵՎ-ՈՉ
or	ԿԱՄ
nor	ԿԱՄ-ՈՉ
xor	Բացառող-ԿԱՄ
xnor	Բացառող-ԿԱՄ-ՈՉ
buf	Բուֆեր
not	Շրջող բուֆեր՝ շրջիչ
bufif1	Եռավիճակ ելքով բուֆեր, ուղիղ կառավարող մուտքով
notif1	Եռավիճակ ելքով շրջող բուֆեր, ուղիղ կառավարող մուտքով
bufif0	Եռավիճակ ելքով բուֆեր, ժխտված կառավարող մուտքով
notif0	Եռավիճակ ելքով շրջող բուֆեր, ժխտված կառավարող մուտքով

Այս պրիմիտիվների օրինակները Վերիլոգ ծրագրում տեղադրվում են ինչպես Վերիլոգ մոդուլները: Ստորև բերված են պրիմիտիվների օրինակների տեղադրման օրինակներ:

```

and a1(out, in1, in2); //սկզբում պետք է նշել ելքի մատույցը
buf b1(OUT1, IN);
notif0 n0 (out, in, ctrl); //սկզբում ելքի մատույցը, հետո մուտքը, վերջում
// կառավարումը
nand na1_3inp(OUT, IN1, IN2, IN3); //երեք մուտքով տարրի օրինակ
and (OUT, IN1, IN2); //օրինակի տեղադրում առանց անվանման, թույլատրվում է

```



Նկ. 7.17. Վերիլոգում ըստ նախնական պայմանավորվածության օգտագործվող տրամաբանական տարրեր

Աղյուսակ 7.4–ում ցույց են տրված Վերիլոգում ներդրված տրամաբանական տարրերի իսկության աղյուսակները՝ հաշվի առնելով նաև x և z մակարդակները:

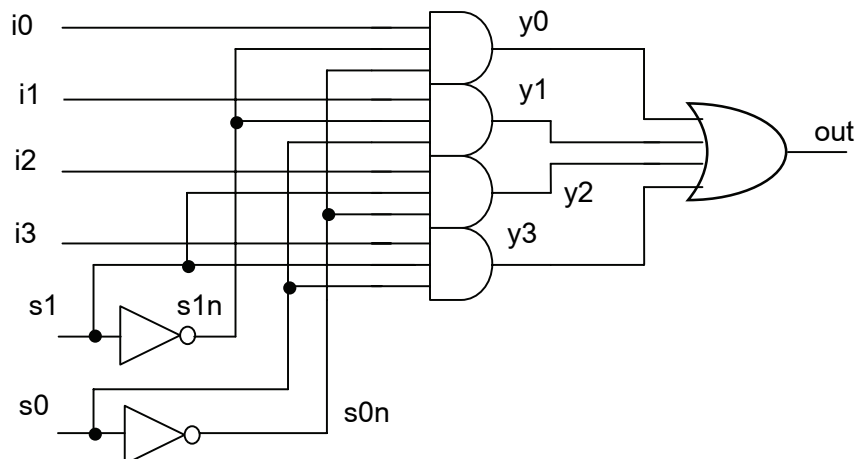
and					nand				
	in2					in2			
in1	0	1	x	z	in1	0	1	x	z
0	0	0	0	0	0	1	1	1	1
1	0	1	x	x	1	1	0	x	x
x	0	x	x	x	x	1	x	x	x
z	0	x	x	x	z	1	x	x	x
or					nor				
	in2					in2			
in1	0	1	x	z	in1	0	1	x	z
0	0	1	x	x	0	1	0	x	x
1	1	1	1	1	1	0	0	0	0
x	x	1	x	x	x	x	0	x	x
z	x	1	x	x	z	x	0	x	x
xor					xnor				
	in2					in2			
in1	0	1	x	z	in1	0	1	x	z
0	0	1	x	x	0	1	0	x	x
1	1	0	x	x	1	0	1	x	x
x	x	x	x	x	x	x	x	x	x
z	x	x	x	x	z	x	x	x	x
buf		not							
in	out	in	out						
0	0	0	1						
1	1	1	0						
x	x	x	x						
z	x	z	x						
bufif1					notif1				
	ctrl					ctrl			
i1	0	1	x	z	i1	0	1	x	z
0	z	0	L	L	0	z	1	H	H
1	z	1	H	H	1	z	0	L	L
x	z	x	x	x	x	z	x	x	x
z	z	x	x	x	z	z	x	x	x
bufif0					notif0				
	ctrl					ctrl			
in	0	1	x	z	in	0	1	x	z
0	0	z	L	L	0	1	z	H	H
1	1	z	H	H	1	0	z	L	L
x	x	z	x	x	x	x	z	x	x
z	x	z	x	x	z	x	z	x	x

Կառավարվող ելքի վիճակով բուֆերներն օգտագործվում են այն դեպքերում, երբ ելքում ակտիվ տրամաբանական մակարդակ պետք է հաստատվի միայն թույլտվության առկայությամբ: Այսպիսի իրավիճակ հանդիպում է, երբ մեկից ավելի ելքեր միացված են միևնույն ազդանշանային գծին: Ժամանակի ցանկացած պահի այդ ելքերը կառավարող ազդանշաններից մեկը միայն կարող է ակտիվ լինել, այլապես այդ գծի ազդանշանի մակարդակը կարող է լինել  $x$ : Կառավարվող բուֆերների մուտքային և կառավարող ազդանշանների որոշ համակցությունների դեպքում ելքի անհայտ մակարդակը նշված է  $L$ , որի հնարավոր արժեքներն են  $0$  կամ  $z$ : Մյուս անհայտ մակարդակը՝  $H$ , ներկայացնում է  $1$  կամ  $z$  հնարավոր արժեքները:

### 7.5.2 Տրամաբանական տարրերի մակարդակով մոդելավորման օրինակներ

Օրինակ 7.9-ում քննարկվեց լրիվ գումարիչի մոդելավորումը տրամաբանական տարրերով: Դիտարկենք ևս մեկ օրինակ:

**Օրինակ 7.11.** Տրամաբանական տարրերի մակարդակով 4:1 մուլտիպլեքսոր: Նախ դիտարկենք նկ. 7.18-ում ցույց տրված սխեման:



Նկ. 7.18. 4:1 մուլտիպլեքսորի սխեման եՎ, ԿԱՄ, ՈՉ տարրերով

Վերիլոգ մոդուլի նկարագրությունը բերված է ստորև.

```
module mux41_g(out, i0, i1, i2, i3, s1, s0);
```

```
// հայտարարել մուտքի/ելքի մատույցները
```

```
output out;
```

```
input i0, i1, i2, i3;
```

```
input s1, s0;
```

```
// ներքին ցանցերը
```

```
wire s1n, s0n;
```

```
wire y0, y1, y2, y3;
```

```
// տրամաբանական տարրերի տեղադրում ըստ սխեմայի
```

```
not (s1n, s1);
```

```

not (s0n, s0);

and (y0, i0, s1n, s0n);
and (y1, i1, s1n, s0);
and (y2, i2, s1, s0n);
and (y3, i3, s1, s0);

or (out, y0, y1, y2, y3);

endmodule

```

Մուլտիպլեքսորի թետավորման մոդուլի նկարագրությունը բերված է ստորև։ Թեստավորման ժամանակ ստուգվում է մուլտիպլեքսորի աշխատանքը մուտքային մի քանի հավաքածուների համար։ Նմանակման ընթացքում ազդանշանների փոփոխություններին կարելի է հետևել **\$monitor** համակարգային առաջադրանքի կիրառությամբ։

```

module test_mux41; // այս մոդուլը մատույց չունի
// Հայտարարել նմանակման ժամանակ մուլտիպլեքսորի մուտքերին տրվող
// ազդանշանները
reg IN0, IN1, IN2, IN3;
reg S1, S0;

// հայտարարել ելքայի փոփոխականը
wire OUT;

// տեղադրել մուլտիպլեքսորի մոդուլի օրինակը
mux41_g mux (OUT, IN0, IN1, IN2, IN3, S1, S0); //ազդանշանների հաջորդականությունը
// պետք է համապատասխանի mux41 մոդուլի մատույցների հաջորդականությանը
// կազմել մուտքային հավաքածուները
initial
    begin
        IN0 = 1; IN1 = 0; IN2 = 1; IN3 = 0; // տվյալների մուտքերի մակարդակները
        #10 S1 = 0; S0 = 0; // ընտրել IN0-ը
        #10 S1 = 0; S0 = 1; // ընտրել IN1-ը
        #10 S1 = 1; S0 = 0; //ընտրել IN2-ը
        #10 S1 = 1; S0 = 1; //ընտրել IN3-ը
        #10 IN0 = 0; IN1 = 1; IN2 = 0; IN3 = 1; // տվյալների մուտքերի մակարդակները
        #10 S1 = 0; S0 = 0; // ընտրել IN0-ը
        #10 S1 = 0; S0 = 1; // ընտրել IN1-ը
        #10 S1 = 1; S0 = 0; //ընտրել IN2-ը
        #10 S1 = 1; S0 = 1; //ընտրել IN3-ը
        $monitor($time," IN0= %b, IN1= %b, IN2= %b, IN3= %b, S1 = %b, S0 = %b, OUT = %b\n", IN0, IN1, IN2, IN3, S1, S0, OUT);
    end
endmodule

```

Նմանակման արդյունքները ցույց են տրված ստորև.

```

0 IN0= 1, IN1= 0, IN2= 1, IN3= 0, S1=0, S0=0, OUT=1
10 IN0= 1, IN1= 0, IN2= 1, IN3= 0, S1=0, S0=0, OUT=0
20 IN0= 1, IN1= 0, IN2= 1, IN3= 0, S1=0, S0=0, OUT=1
30 IN0= 1, IN1= 0, IN2= 1, IN3= 0, S1=0, S0=0, OUT=0

```

```

40 IN0= 0, IN1= 1, IN2= 0, IN3= 1, S1=0, S0=0, OUT=0
50 IN0= 0, IN1= 1, IN2= 0, IN3= 1, S1=0, S0=0, OUT=1
60 IN0= 0, IN1= 1, IN2= 0, IN3= 1, S1=0, S0=0, OUT=0
70 IN0= 0, IN1= 1, IN2= 0, IN3= 1, S1=0, S0=0, OUT=1

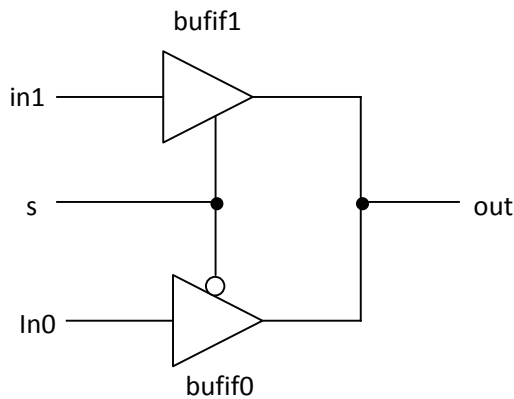
```

**Օրինակ 7.12.** Կառավարվող բուֆերներով կառուցված 2:1 մուլտիպլեքսորի սխեման և պարզեցված իսկության աղյուսակը ցույց են տրված նկ. 7.19–ում:

```

module mux21(s, in0, in1, out);
input s;
input in0, in1;
output out;
bufif0 b1(out, in0,s);
bufif1 b2(out, in1,s);
endmodule

```



s	out
0	In0
1	In1

Նկ. 7.19. Կառավարվող բուֆերներով կառուցված 2:1 մուլտիպլեքսոր

2:1 մուլտիպլեքսորի թեստավորման մոդուլը.

```

module test_mux21;
reg SEL;
reg D0, D1;
mux21 mux_b(SEL, D0, D1, OUT);
initial
begin
$monitor ("SEL = %b, D1= %b, D0= %b, OUT = %b \n", SEL, D1, D0, OUT);
SEL=1'b0; D1=1'b0; D0=1'b0;
#10 D1=1'b0; D0=1'b1;
#10 D1=1'b1; D0=1'b0;
#10 D1=1'b1; D0=1'b1;
#10 SEL=1'b1; D1=1'b0; D0=1'b0;
#10 D1=1'b0; D0=1'b1;
#10 D1=1'b1; D0=1'b0;
#10 D1=1'b1; D0=1'b1;
end
endmodule

```

**Օրինակ 7.13.** 4:1 մուլտիպլեքսորի օրինակ կառուցված mux21 մուլտիպլեքսորների միջոցով:

```

module mux41_b(out, i0, i1, i2, i3, s1, s0);
output out;
input i0, i1, i2, i3;
input s1, s0;

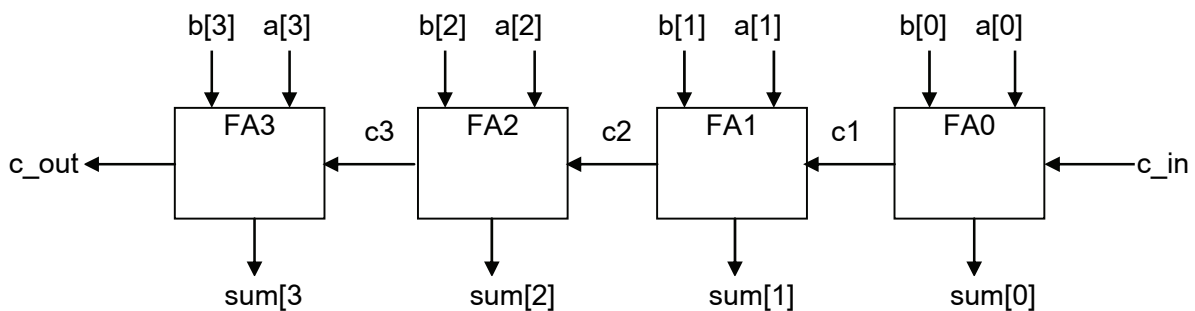
wire n1, n2;

mux21 mux_b1(s0, iN0, iN1, n1);
mux21 mux_b2(s0, iN2, iN3, n2);
mux21 mux_b3(s1, n1, n2, OUT);
endmodule

```

Այս մոդուլը կարելի է թեստավորել օրինակ 7.11-ի test\_mux41 թեստավորման մոդուլով: mux41-ի օրինակի տեղադրման ժամանակ պետք է փոխել մոդուլի անվանումը mux41\_g-ից mux41\_b :

**Օրինակ 7.14.** քառաբիթ երկուական գումարիչ՝ կառուցված օրինակ 7.9-ի 1-բիթ լրիվ գումարիչով:



Նկ. 7.20. Քառաբիթ գումարիչի կառուցվածքային սխեման

```

// Քառաբիթ գումարիչի Վերիլոգ մոդուլը
module adder4(sum, c_out, a, b, c_in);

// մուլտիպլեքս մատույցների հայտարարում
output [3:0] sum;
output c_out;
input [3:0] a, b;
input c_in;

// ներքին ցանցեր
wire c1, c2, c3;

// տեղադրել 1-բիթ գումարիչի օրինակներ
full_adder FA0(c_in, a[0], b[0], sum[0], c1);
full_adder FA1(c1, a[1], b[1], sum[1], c2);
full_adder FA2(c2, a[2], b[2], sum[2], c3);
full_adder FA3(c3, a[3], b[3], sum[3], c_out);

endmodule

```

Այս մոդուլի աշխատունակությունը կարելի է ստուգել հետևյալ թեստավորման մոդուլով:

```

module test_adder4;
// հայտարարել փոփոխականները
reg [3:0] A, B;
reg C_IN;
wire [3:0] SUM;
wire C_OUT;

// տեղադրել քառաբիթ գումարիչի օրինակ, այն անվանել fa4
adder4 fa4(SUM, C_OUT, A, B, C_IN);

initial
begin
    $monitor($time," A= %b, B=%b, C_IN= %b, \ C_OUT= %b, SUM= %b\n",
    A, B, C_IN, C_OUT, SUM);
end

// մուտքային թեստ-վեկտորներ
initial
begin
    A = 4'd0; B = 4'd0; C_IN = 1'b0;
    #5 A = 4'd3; B = 4'd4;
    #5 A = 4'd2; B = 4'd5;
    #5 A = 4'd9; B = 4'd9;
    #5 A = 4'd10; B = 4'd15;
    #5 A = 4'd10; B = 4'd5; C_IN = 1'b1;
end
endmodule

```

Նմանկումից ստացված տվյալները բերված են ստորև.

```

0 A= 0000, B=0000, C_IN= 0, \ C_OUT= 0, SUM= 0000
5 A= 0011, B=0100, C_IN= 0, \ C_OUT= 0, SUM= 0111
10 A= 0010, B=0101, C_IN= 0, \ C_OUT= 0, SUM= 0111
15 A= 1001, B=1001, C_IN= 0, \ C_OUT= 1, SUM= 0010
20 A= 1010, B=1111, C_IN= 0, \ C_OUT= 1, SUM= 1001
25 A= 1010, B=0101, C_IN= 1, \ C_OUT= 1, SUM= 0000

```

### 7.5.3 Տարրերի հապաղումները

Մինչ այժմ մենք Վերիլոգ մոդելներում հաշվի չենք առել տրամաբանական տարրերի հապաղումները: Վերիլոգը թույլ է տալիս նախագծողին օգտագործել տարրերի հապաղումները՝ իրականությանն ավելի մոտիկ նմանական արդյունքներ ստանալու համար: Տրամաբանական տարրերի համար Վերիլոգում որոշված են տարրի ելքի ազդանշանի փոխանջատման երեք տիպի հապաղում՝ (ա) աճի հապաղում, երբ տարրի ելքն անցնում է 1 վիճակ որևէ այլ վիճակից (0, x, z), (բ) անկման հապաղում, երբ տարրի ելքն անցնում է 0 վիճակի որևէ այլ վիճակից (1, x, z), (գ) անջատման հապաղում, երբ տարրի ելքն անցնում է z վիճակի որևէ այլ վիճակից (0, 1, x): Երբ տարրի ելքն անցնում է x վիճակի, հապաղմանը վերագրվում է թվարկված երեք հապա-



դումներից նվազագույնը:

Եթե Վերիլոգ նկարագրությունում միայն մեկ հապաղում է նշված, բոլոր երեք հապաղումներին վերագրվում է միևնույն արժեքը.

```
and #(delay_time) a1(out, i1, i2); // delay_time հապաղումը վերաբերում է բոլոր  
// անցումներին
```

Եթե երկու արժեքներ են նշված, դրանք վերագրվում են աճին ու անկմանը, իսկ անջատման հապաղումը համարվում է հավասար դրանցից նվազագույնին:

```
and #(rise_val, fall_val) a2(out, i1, i2); // աճի և անկման հապաղումներ
```

Նշված են բոլոր երեք հապաղումները.

```
bufif0 #(rise_val, fall_val, turnoff_val) b1 (out, in, control); // աճի, անկման և անջատման  
// հապաղումներ
```

Ստորև բերվում են տրամաբանական տարրերին հապաղումների վերագրման օրինակներ:

```
and #(5) a1(out, i1, i2); //բոլոր անցումներին վերագրվում է 5 միավոր հապաղում  
and #(4,6) a2(out, i1, i2); // աճի հապաղումը = 4, անկման հապաղումը= 6  
bufif0 #(3,4,5) b1 (out, in, control); // աճ = 3, անկում = 4, անջատում = 5
```

Վերիլոգը տրամադրում է հապաղումների կառավարման լրացուցիչ հնարավորություն: Թվարկված երեք տիպի հապաղումներից յուրաքանչյուրի համար կարելի է նշել երեք արժեք՝ նվազագույն (min), տիպային (typ), առավելագույն (max): Նմանակումն սկսելիս կարելի է ընտրել այդ արժեքներից ցանկացածը (բայց միայն մեկը)՝ նմանակիչի կանչի հրամանի օպցիայի միջոցով: Min/Typ/Max արժեքներն օգտագործվում են այն սարքերի նմանակման ժամանակ, որոնց հապաղումները փոխվում են ԻՍ-ի պատրաստման տեխնոլոգիական պրոցեսի կամ աշխատանքի ժամանակ ջերմաստիճանի փոփոխություններից:

Հապաղման արժեքի ընտրությունը Min/Typ/Max տարբերակներից կարելի է Վերիլոգ նմանակիչին տալ նմանակիչի կանչի օպցիայի միջոցով: Ստորև ցույց են տրված հապաղումների վերագրման օրինակներ՝ Սինոփսիսի VCS նմանակիչի կիրառության համար: Ենթադրվում է, որ հապաղումների վերագրումով մոդուլը գտնվում է test\_delays.v Վերիլոգ ֆայլում

```
//նմանկումը կատարել առավելագույն հապաղման արժեքով՝ +maxdelays-ը
```

```
//նմանկիչի կանչի օպցիան է
```

```
> vcs test_delays.v +maxdelays
```

```
//նմանկումը կատարել նվազագույն հապաղման արժեքով՝ +mindelays-ը նմանկիչի
```

```
// կանչի օպցիան է
```

```
> vcs test_delays.v +mindelays
```

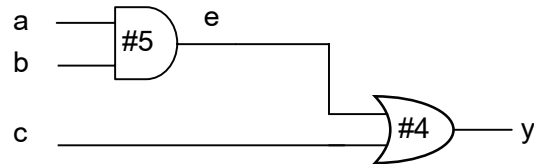
```
//նմանկումը կատարել տիպային հապաղման արժեքով՝ +typdelays-ը նմանկիչի
```

```
// կանչի օպցիան է
```

```
> vcs test_delays.v +typdelays
```

### 7.5.4. Հապաղումներով նմանակման օրինակ

Նկ 7.21–ում ցույց տրված սխեմայում օգտագործված ԵՎ և ԿԱՄ տարրերի հապաղումները 5 և 4 միավոր են համապատասխանաբար: Այդ սխեմայի Վերիլոգ մոդուլը բերված է ստորև.



Նկ. 7.21. Հապաղումներով մոդելավորվող սխեման

```

// Սահմանել Clogic անվանմամբ մոդուլ
module Clogic (out, a, b, c);
// մուտք/ելք մատույցների հայտարարում
output out;
input a,b,c;

// ներքին ցանցեր
wire e;

// տեղադրել պրիմիտիվների օրինակներ
and #(5) and1(e, a, b); //5 միավոր հապաղում and1 տարրի համար
or #(4) or1(out, e, c); //4 միավոր հապաղում or1 տարրի համար

endmodule

Այս մոդուլը թեստավորվում է հետևյալ միջավայրում.
// թեստավորման մոդուլ (բարձր մակարդակի մոդուլ)
module test_delay;

reg A, B, C; // հայտարարել փոփոխականները
wire Y;

Clogic dut( Y, A, B, C); // տեղադրել թեստավորվող մոդուլի օրինակը

// զենեքացնել մուտքային ազդանշանները
initial
begin
  A= 1'b0; B= 1'b0; C= 1'b0;
  #10 A= 1'b1; B= 1'b1; C= 1'b1;
  #10 A= 1'b1; B= 1'b0; C= 1'b0;
  #20 $finish; //նմանակումն ավարտել ժամանակի 40 միավոր հետո
end

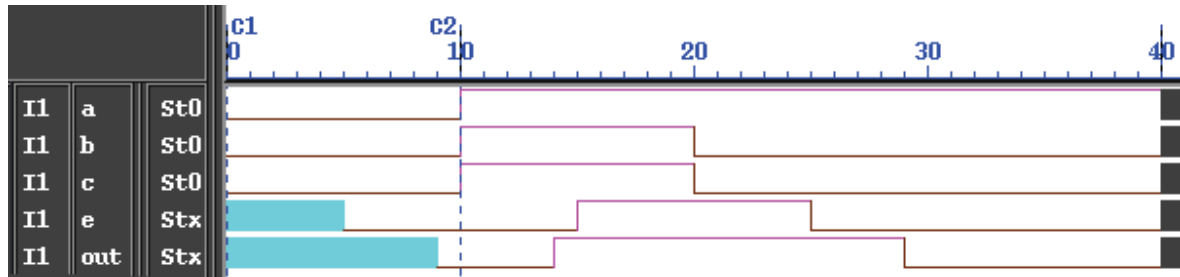
endmodule

```

Նմանակումից ստացված ժամանակային դիագրամները ցույց են տրված նկ. 7.22–ում՝ e և out ազդանշանների սկզբնական արժեքները անհայտ են՝ x, նմանակման ժամանակի t=10 պահին a, b, և c մուտքերն անցում են 1 արժեքի: Այդ պահից 4 միավոր հետո out ելքը անցնում է 1-ի, իսկ 5 միավոր հետո e ելքն է փոխանջատվում 1: Ժամա-

նակի  $t=20$  պահին  $b$ , և  $c$  մուտքերն անցում են 0 արժեքի: Որից 5 միավոր հետո  $e$  ելքն է փոխանջատվում 0, իսկ ևս 4 միավոր հետո  $out$  ելքն անցնում է 0-ի:

Օգտակար կլինի մանրամասն ուսումնասիրել նմանակումից ստացված ժամանակային դիագրամները և համադրել նկ. 7.21–ի սխեմայի հետ:



Նկ. 7.22. Clogic մոդուլի նմանակման ժամանակային դիագրամները

## 7.5.6. Խնդիրներ

**Խնդիր 7.11.** Մշակել 2 մուտքով ԵՎ, ԿԱՄ, ԿԱՄ-ՈՉ, ՈՉ տրամաբանական տարրերի Վերիլոգ մոդուլները՝ օգտագործելով ներդրված 2 մուտքով `rand` պրիմիտիվը: Անվանել այդ մոդուլները՝ `my_and2`, `my_or2`, `my_nor2`, `my_not`. Ստուգել այդ մոդուլների ճշմարտացիությունը նմանակման միջոցով՝ նախապես գրելով համապատասխան թեստավորման մոդուլ:

**Խնդիր 7.12.** Մշակել ԿԱՄ-ՈՉ պրիմիտիվների վրա կառուցված RS տրիգերի Վերիլոգ մոդուլ:

**Խնդիր 7.13.** Նմանակել օրինակ 7.12-ի մուլտիպլեքսորը ստորև բերված աղյուսակում ցույց տրված հապաղումներով: Ուսումնասիրել նմանակումից ստացված ժամանակային դիագրամները:

Աղյուսակ 7.5

	Min	Typ	Max
Աճ	1	2	3
Անկում	3	4	5
Անջատում	5	6	7

## 7.6. Տվյալների հոսքի մոդելավորում

### 7.6.1. Շարունակական վերագրում

Փոքր թվային շղթաների դեպքում տրամաբանական տարրերի մակարդակով մոդելավորումը շատ հարմար է, քանի որ նախագծողը կարող է տեղադրել յուրաքանչյուր տարր և կատարել անհրաժեշտ միացումները: Սակայն մեծ համակարգերի դեպքում տարրերի թիվը չափազանց մեծ է, և այդ մոտեցումը արդյունավետ չէ: Պետք է օգտվել ավելի բարձր վերացարկման մակարդակի մոդելավորումից: Տվյալների հոսքի մոդելավորումն իրականացվում է ցանց տիպի փոփոխականներին ազդանշանների վերագրման (assignment) միջոցով: Վերագրումների միջոցով նկարագրությունից ավտո-

մատ նախագծման գործիքներով կարելի է կառուցել համարժեք մոդելը՝ տրամաբանական տարրերի մակարդակով: Ավտոմատացված սինթեզի գործիքներով տվյալների հոսքի մակարդակով նկարագրությունից կարելի է նաև ստանալ նախագծվող համակարգի օպտիմալացված սխեման:

Շարունակական վերագրումը (continues assignment) տվյալների հոսքի մոդելավորման հիմնական օպերատորն է: Այն փոխարինում է տրամաբանական տարրերին ավելի բարձր մակարդակի նկարագրությամբ, մասնավորապես, տրամաբանական ֆունկցիաների մակարդակով: Շարունակական վերագրումը սկսվում է **assign** բանալի բառով: Ստորև բերված են մի քանի օրինակներ.

```
wire out;
assign out = i1 & i2;

wire [15:0] addr;
assign addr[15:0] = addr1_bits[15:0] ^ addr2_bits[15:0];

wire c_out;
wire [3:0] sum;
assign {c_out, sum[3:0]} = a[3:0] + b[3:0] + c_in;
```

Շարունակական վերագրման ձախ մասի փոփոխականը պետք է լինի անպայման ցանց տիպի՝ սկալյար կամ վեկտոր՝ կարող է լինել նաև փոփոխականների միակցում (concatenation)՝ {c\_out, sum[3:0]}: Աջ մասի փոփոխականները կարող են լինել ցանց կամ ռեգիստր տիպի, կարող են լինել ֆունկցիայի կանչ: Նշենք, որ գոյություն ունի նաև ոչ շարունակական վերագրման եղանակ՝ վարքագծային վերագրում, որին կծանոթանանք ավելի ուշ:

Ցանց տիպի փոփոխականին կարելի վերագրում կատարել նաև նրա հայտարարման ժամանակ՝ անբացահայտ վերագրում: Օրինակ, հետևյալ երկու վերագրումները համարժեք են.

```
wire out;
assign out = in1 & in2;

և

wire out = in1 & in2;
```

Շարունակական վերագրումը միշտ ակտիվ է՝ աջ մասի արտահայտության արժեքը հաշվվում է ցանկացած պահի, երբ նրա մեջ մտնող որևէ փոփոխական փոխում է արժեքը և վերագրվում ձախ մասին:

Շարունակական վերագրման ժամանակ կարելի է օգտագործել հապաղում, որը որոշում է աջ մասի հաշվված արժեքը ձախ մասի փոփոխականին վերագրելու ժամանակը: Հապաղումներն օգտագործվում են իրական շղթաներում ընթացող պրոցեսների մոդելավորման համար, ինչպես դա արվում է տրամաբանական տարրերի հապաղումների մոդելավորման դեպքում: Օրինակ.

```
assign #10 y = a & b; // շարունակական վերագրման հապաղում
```

Հապաղումը նշվում է **assign** բանալի բառից հետո: *a* կամ *b* փոփոխականների ցանկացած փոփոխության դեպքում հաշվվում է *a* & *b* արտահայտության արժեքը, բայց ստացված արժեքը վերագրվում է ձախ մասին 10 միավոր ուշացմամբ՝ *a* կամ *b* մուտքի փոփոխությունը ելք է հասնում 10 միավոր հապաղումով: Կարևոր է նկատել, որ եթե մուտքի իմպուլսի տևողությունը փոքր է վերագրման հապաղումից, ապա վերագրումը ելք չի հասնում: Ստորև բերված Վերիլոգ մոդուլը և նկ. 7.23–ում ցույց տրված դրա նմանակումից ստացված ժամանակային դիագրամները բացատրում են այս օրինակում հապաղումով վերագրման մանրամասները:

```

module delay10;
reg a, b;
wire y;
assign #10 y = a & b;

initial
begin
    a=1'b0; b=1'b0;
    #20 a=1'b1; b=1'b1;
    #20 a=1'b0;
    #20 a=1'b1;
    #5 a=1'b0;
    #10 $stop;
end
endmodule

```



Նկ. 7.23. *delay10* մոդուլի նմանակման ժամանակային դիագրամները

1. Երբ *a* և *b* ազդանշանները  $t=20$  պահին անցնում են 1 արժեքի, *y* ելքը 1 արժեքի է անցնում ժամանակի 10 միավորով ավելի ուշ՝  $t = 30$  պահին:
2. Երբ  $t=40$  պահին *a*-ն անցնում է 0-ի, ելքը փոխվում է 0-ի  $t=50$  պահին:
3.  $t=60$  պահին *a*-ն անցնում է 1-ի և ժամանակի 10 միավոր չանցած՝  $t=65$  պահին նորից անցնում 0-ի: Ելքի նոր արժեքի հաշվման պահին՝  $t=60$ -ից 10 միավոր հետո,  $a=0$ , հետևաբար ելքը ստանում է 0 արժեք: Վերագրման հապաղումից կարճ իմպուլսները ելք չեն անցնում:

Հապաղումը կարելի տալ նաև անբացահայտ վերագրման ժամանակ՝

```
wire #10 y = a & b;
```

### 7.6.2 Արտահայտություններ, օպերատորներ և օպերանդներ

Տվյալների հոսքերի մոդելավորման ժամանակ թվային համակարգը նկարագրվում է ոչ թե տրամաբանական տարրերի, այլ արտահայտությունների միջոցով: Արտահայտությունը օպերանդների և օպերատորների համակցություն է: Օպերանդ կարող է լինել ցանկացած տիպի տվյալ՝ փոփոխական կամ հաստատուն, ինչպես նաև համակարգային կամ օգտագործողի կողմից սահմանված ֆունկցիա, որը վերադարձնում է ցանկացած տիպի տվյալ: Օպերատորները օպերանդների միջև սահմանում են որոշակի գործողություններ՝ ցանկալի արդյունք ստանալու համար:

Վերիլոգում օգտագործվում են օպերատորների հետևյալ խմբերը՝ թվաբանական, տրամաբանական, համեմատության, բիթ առ բիթ գործողությունների, սեղմման (reduction), տեղաշարժի, միակցման և պայմանական: Աղյուսակ 7.6 –ում ցույց են տրված բոլոր օպերատորները:

Աղյուսակ 7.6

Օպերատորի տիպը	Օպերատորի սիմվոլը	Կատարվող գործողությունը	Մասնակցող օպերանդների թիվը
Թվաբանական	*	բազմապատկում	երկու
	/	բաժանում	երկու
	+	գումարում	երկու
	-	հանում	երկու
	%	մոդուլ	երկու
	**	աստիճան	երկու
Տրամաբանական	!	տրամաբանական ժխտում	մեկ
	&&	տրամաբանական ԵՎ	երկու
		տրամաբանական ԿԱՄ	երկու
Համեմատության	>	մեծ	երկու
	<	փոքր	երկու
	>=	մեծ կամ հավասար	երկու
	<=	փոքր կամ հավասար	երկու
Հավասարություն	==	հավասարություն	երկու
	!=	անհավասարություն	երկու
	===	դեպքով հավասարություն	երկու
	!==	դեպքով անհավասարություն	երկու
Բիթ առ բիթ	~	բիթ առ բիթ ժխտում	երկու
	&	բիթ առ բիթ ԵՎ	երկու
		բիթ առ բիթ ԿԱՄ	երկու
	^	բիթ առ բիթ Բացառող-ԿԱՄ	երկու
	~^ կամ ^~	բիթ առ բիթ Բացառող-ԿԱՄ-ՈՉ	երկու
Սեղմման	~	սեղմման ժխտում	մեկ
	&	սեղմման ԵՎ	մեկ
	~&	սեղմման ԵՎ-ՈՉ	մեկ
		սեղմման ԿԱՄ	մեկ
	~	սեղմման ԿԱՄ-ՈՉ	մեկ
	^	սեղմման Բացառող-ԿԱՄ	մեկ
	~^ կամ ^~	սեղմման Բացառող-ԿԱՄ- ՈՉ	մեկ

Աղյուսակ 7.6-ի շարունակություն

Օպերատորի տիպը	Օպերատորի սիմվոլը	Կատարվող գործողությունը	Մասնակցող օպերանդների թիվը
Տեղաշարժի	<<	տեղաշարժ ձախ	երկու
	>>	տեղաշարժ աջ	երկու
	<<<	թվաբանական տեղաշարժ ձախ	երկու
	>>>	թվաբանական տեղաշարժ աջ	երկու
Միակցման	{}	միակցում	ցանկացած
	{}	կրկնություն-միակցում	ցանկացած
Պայմանական	?:	պայմանական ընտրություն	երեք

+ և - թվաբանական օպերատորներն օգտագործվում են ինչպես մեկ փոփոխականի դեպքում՝ փոփոխականի նշանը տալու համար, այնպես էլ երկու փոփոխականի դեպքում՝ թվաբանական գումարման կամ հանման համար:

Դեպքով հավասարության և անհավասարության օպերանդների առանձնահատկություններին կծանոթանանք ավելի ուշ:

Ատահայտություններում գործողությունների կատարման կարգը կարելի է տալ սովորական փակագծերի օգնությամբ կամ հետևելով գործողությունների Վերիլոգում սահմանված առաջնահերթություններին, որոնք նման են C ծրագրավորման լեզվում ընդունված կարգին: Աղյուսակ 7.7-ում օպերանդները դասակարգված են ըստ առաջնահերթությունների նվազման կարգի: Նույն տողում նշված օպերատորներն ունեն հավասար առաջնահերթություն:

Արտահայտություններում գործողությունները կատարվում են ձախից աջ, բացի պայմանական ընտրության արտահայտություններից, որտեղ գործողությունները կատարվում են աջից ձախ: Հետևյալ օրինակում՝  $A+B-C$ ,  $A$ -ին գումարվում է  $B$ , որից հետո  $(A+B)$  արդյունքից հանվում է  $C$ :

Երբ արտահայտության մեջ օպերատորները տարբեր առաջնահերթություններով են, գործողությունների կատարման կարգը սահմանվում է ըստ առաջնահերթությունների: Օրինակ,  $A+B/C$  արտահայտությունում սկզբում  $B$ -ն բաժանվում է  $C$ -ի, ապա  $B/C$  արդյունքը գումարվում է  $A$ -ին:

Աղյուսակ 7.7

Օպերանդներ	Առաջնահերթություն
+, -, !, ~ (մեկ օպերանդով)	Ամենաբարձր առաջնահերթություն
*, /, %	
+, - (երկու օպերանդով)	
<<, >>, <<<, >>>>	
<, >, <=, >=	
==, !=, ==, !=	
&	
^, ~^	
&&	
?:	Ամենացածր առաջնահերթություն

### 7.6.3 Թվաբանական, տրամաբանական, համեմատության և հավասարության օպերատորներ

Դիտարկենք թվաբանական գործողությունների օրինակներ.

```
reg [4:0] Q;  
integer L;  
reg [4:0] A=5'b01100;  
reg [4:0] B=5'b00101;  
integer J=12;  
integer K=5;  
Q=A+B;      //արդյունքում Q=5'b10001  
Q=A-B;      // արդյունքում Q=5'b00111  
Q=A*B;      // արդյունքում Q=5'b11100  
Q=A/B;      // արդյունքում Q=5'b00010  
Q=A%B;      // արդյունքում Q=5'b00010  
L=J*K;      // արդյունքում L =60  
L=J/K;      // արդյունքում L =2
```

Եթե որևէ օպերանդ պարունակում է անհայտ բիթ, ուրեմն արդյունքն անհայտ է՝

```
reg [4:0] A=5'b0110x;  
reg [4:0] B=5'b00101;  
Q=A+B;      //արդյունքում Q=5'bx
```

**Տրամաբանական օպերատորի** գործողության արդյունքում վերադարձվում է 1 բիթ՝ 0 (կեղծ, false), 1 (ճշմարիտ, true), x (երկիմաստ, ambiguous) արժեքով: Եթե որևէ օպերանդ հավասար չէ զրոյի, այն համարվում է տրամաբանական 1-ի (ճշմարիտ): Եթե որևէ օպերանդ հավասար է զրոյի, այն համարվում է տրամաբանական 0-ի (կեղծ): Եթե որևէ օպերանդ պարունակում է x կամ z բիթ, այն համարվում է x-ի, և նմանակիչի համար այն համարվում է կեղծ պայմանի: Տրամաբանական օպերատորի օպերանդներ կարող են լինել արտահայտություններ: Որպեսզի կարիք չլինի հիշել օպերանդների առաջնահերթությունները, խորհուրդ է տրվում տրամաբանական գործողություններում օգտագործել փակագծեր: Միաժամանակ, փակագծերը տրամաբանական արտահայտությունները դարձնում են ավելի ընթերցելի: Բերենք տրամաբանական օպերանդների օգտագործման մի քանի օրինակ՝

A = 5; B = 0;

Q=!A; //համարվում է ՈՉ(տրամաբանական-1), արդյունքում Q=0

Q=! B; //համարվում է ՈՉ(տրամաբանական -0), արդյունքում Q=1

Q=A&&B; //համարվում է (տրամաբանական-1 ԵՎ տրամաբանական -0), արդյունքում Q=0

Q=A||B; //համարվում է (տրամաբանական-1 ԿՎՄ տրամաբանական -0), արդյունքում Q=1

Q=(A==5)&&(B==0); //արդյունքում Q=1

Q=(A==5)&&(B!=0); //արդյունքում Q=0

A = 4'b0x01; B = 4'b0110;

Q=A&&B; //համարվում է (x ԵՎ տրամաբանական -1), արդյունքում Q=x

Q=A||B; //համարվում է (տրամաբանական-x ԿՎՄ տրամաբանական -1), արդյունքում Q=1



**Համեմատության օպերատորներն** օգտագործվում են այն արտահայտություններում, որոնց հաշվման արդյունքում վերադարձվում է տրամաբանական 1, եթե արտահայտությունը ճշմարիտ է, և վերադարձվում է տրամաբանական 0, եթե արտահայտությունը կեղծ է: Եթե արտահայտությունում առկա է x կամ z, ապա արտահայտության արժեքը x է: Օրինակներ.

A=5'b01100;

B=5'b00101;

C=5'bxx101;

Q=(A> B) ; //արդյունքում Q=1

Q=(A<= C) ; //արդյունքում Q=x

Q=( A<= C)&&(B> C);//արդյունքում Q=x

**Հավասարության օպերատորներն** օգտագործվում են այն արտահայտություններում, որոնց հաշվման արդյունքում վերադարձվում է տրամաբանական 1, եթե համեմատվող օպերանդները հավասար են՝ արտահայտությունը ճշմարիտ է, և վերադարձվում է տրամաբանական 0, եթե համեմատվող օպերանդները հավասար չեն՝ արտահայտությունը կեղծ է: Համեմատությունը կատարվում է բիթ առ բիթ: Եթե օպերանդներն ունեն անհավասար թվով բիթեր, ապա պակասող բիթերը լրացվում են զրոներով:

Կարևոր է նշել տրամաբանական հավասարության և դեպքով հավասարության տարբերությունը: Եթե տրամաբանական հավասարության արտահայտությունում առկա է x կամ z, այդ արտահայտության արժեքը x է: Դեպքով հավասարության դեպքում համեմատվում են նաև x և z արժեքները: Համեմատության արդյունքը կարող է լինել միայն 0 կամ 1, այն երբեք չի ընդունում x արժեք:

A=5'b01100;

B=5'b00101;

C=5'bxx101;

Q=(A== B) ;//արդյունքում Q=0

Q=(A!= B) ;//արդյունքում Q=1

Q=(A== C) ;//արդյունքում Q=x

Q=(A== =C) ;//արդյունքում Q=0

Q=(A!= =C) ;//արդյունքում Q=1

Q=( A<= C) && (B> C);//արդյունքում Q=x

#### **7.6.4. Բիթ առ բիթ գործողությունների, սեղմման, տեղաշարժի, միակցման և պայմանական օպերատորներ**

**Բիթ առ բիթ օպերատորով** գործողությունը կատարվում է օպերանդների նույնանուն բիթերի միջև: Եթե օպերանդներն ունեն անհավասար թվով բիթեր, պակասող բիթերը լրացվում են զրոներով: z արժեքը դիտվում է որպես x: Աղյուսակ 7.8–ում ցույց են տրված բիթ առ բիթ օպերատորների իսկության աղյուսակները (համեմատե՛ք համապատասխան տրամաբանական տարրերի իսկության աղյուսակների հետ, աղյուսակ 7.4):

Բիթ առ բիթ ԵՎ (&)				Բիթ առ բիթ ԿԱՍ ( )				Բիթ առ բիթ ՈՉ (~)	
Օպ2				Օպ2				Օպ	
Օպ1	0	1	x	Օպ1	0	1	x	Օպ	Ելք
0	0	0	0	0	0	1	x	0	
1	0	1	x	1	1	1	1	1	
x	0	x	x	x	x	1	x	x	
Բիթ առ բիթ Բացառող-ԿԱՍ (^)				Բիթ առ բիթ Բացառող-ԿԱՍ-ՈՉ (^~)					
Օպ2				Օպ2					
Օպ1	0	1	x	Օպ1	0	1	x		
0	0	1	x	0	1	0	x		
1	1	0	x	1	0	1	x		
x	x	x	x	x	x	x	x		

Ստորև ցույց են տրված բիթ առ բիթ օպերատորների օրինակներ.

A=5'b01100;  
 B=5'b00101;  
 C=5'bxx101;  
 Q=~A; //արդյունքում Q=5'b10011  
 Q=A& B ; //արդյունքում Q=5'b00100  
 Q=A|B ; //արդյունքում Q=5'b01101  
 Q=A^B ; //արդյունքում Q=5'b01001  
 Q=A^~B ; //արդյունքում Q=5'b10110  
 Q=A&C ; //արդյունքում Q=5'b0x100  
 Q=A|C ; //արդյունքում Q=5'bx1101

Կարևոր է տարբերել տրամաբանական և բիթ առ բիթ օպերատորները: Տրամաբանական օպերատորները վերադարձնում են 1-բիթ արդյունք՝ 0, 1, x: Օրինակ, Q=A&B-ն վերադարձնում է Q=1, իսկ Q=A&B-ն՝ Q=5'b00100:

**Սեղմման օպերատորն** ունի միայն մեկ օպերանդ, այն իրագործում է բիթ առ բիթ գործողություն օպերանդի բոլոր բիթերի հետ՝ սկսած ձախից, և վերադարձնում 1 բիթ արդյունք: Դիտարկենք հետևյալ օրինակները.

A=5'b01100;  
 Q= &A; //համարժեք է Q= 0&1&1&0&0 =0  
 Q= |A; // համարժեք է Q=0|1|1|0|0=1  
 Q= ^A ; // համարժեք է Q=0^1^1^0^0=0  
 Q=A^~B ; // համարժեք է Q=0^~1^~1^0^0=1

Սեղմման օպերատորներն օգտագործվում են երկուական վեկտորի բիթերի ստուգման համար: &A-ն վերադարձնում է 1 միայն այն դեպքում, երբ բոլոր բիթերը 1 են: |A-ն վերադարձնում է 0 միայն այն դեպքում, երբ բոլոր բիթերը 0 են: ^A-ն ստուգում է A վեկտորում է 1 բիթերի թվի զույգությունը: ^~A-ն ստուգում է A վեկտորում է 1 բիթերի թվի կենտությունը:

**Տեղաշարժի օպերատորները** տեղաշարժում են երկուական վեկտորի բիթերը դեպի ձախ կամ աջ նշված թվով բիթերով: Տրամաբանական տեղաշարժի դեպքում ազատված բիթերի դիրքերը լցվում են 0-ներով, դուրս մնացած բիթերը կորչում են:

A=5'b01100;

Q=A<<2; // A-ն տեղաշարժել դեպի ձախ 2 բիթով, արդյունքում Q=5'b10000

Q=A>>1; // A-ն տեղաշարժել դեպի աջ 1 բիթով, արդյունքում Q=5'b00110

Թվաբանական տեղաշարժի դեպքում ձախ տեղաշարժելիս ազատված բիթերը լցվում են 0-ներով: Աջ տեղաշարժելիս օպերատորը ելնում է այն արտահայտության համատեքստից, թե ինչպես լցնի ազատված բիթերը: Առանց նշանի վեկտորի ազատված բիթերը լցվում են 0-ներով: Նշանով վեկտորի ազատված բիթերը լցվում են նշանային կարգի բիթի (ավագ բիթի) արժեքով:

**reg signed** [5:0] A, Q; //A-ն և Q-ն նշանով վեկտորներ են՝ reg signed տիպի

A=5'sb011000; //A-դրական է

Q=A>>2; // արդյունքում Q=5'b000110

B=5'b111000; // B-ն բացասական է

Q=B>>2; // արդյունքում Q=5'b111110

**Միակցման օպերատորը** ( { , } ) թույլ է տալիս միացնել բազմաթիվ օպերանդներ մեկ ընդհանուր վեկտորում: Օպերանդները պետք է պարտադիր լինեն չափավոր, որպեսզի հնարավոր լինի որոշել արդյունարար վեկտորի չափսը: Օպերանդները միակցման փակագծերի մեջ միմյանցից բաժանվում են ստորակետով:

A=5'b01100;

B=5'b00101;

C={A,B}; //արդյունքում C=10'b01100\_00101, հիշեք, որ ( \_ ) նշանը թվի արժեքի վրա չի

//ազդում՝ օգտագործվում է թվերը դիտարկելի դարձնելու համար

d={a, b[3:0], w, 3'b101}; //այս օրինակը համարժեք է հետևյալին

d={a, b[3], b[2], b[1], b[0], w, 1'b1, 1'b0, 1'b1};

**Կրկնությամբ միակցման** դեպքում({ { , } }) կարելի է նշել, թե քանի անգամ կրկնել որևէ բաղադրիչ արդյունարար վեկտորում:

A=5'b01100;

B=5'b00101;

C={3{A}}; // A-ն կրկնել 3 անգամ, արդյունքում C=10'b01100\_01100\_01100

C={2{A}, {B}, 2{1'b0}}; // արդյունքում C=10'b01100\_01100\_00101\_00

**Պայմանական օպերատորն** (?:) ունի երեք օպերանդ, որոնք իրարից բաժանվում են երկու օպերատորներով և տրվում է հետևյալ չափաձևով՝

condition\_expr ? true\_expr : false\_expr ;

Սկզբում հաշվվում է condition\_expr տրամաբանական արտահայտության արժեքը: Եթե այդ արժեքը ճշմարիտ է (տրամաբանական 1), ապա վերադարձվում է true\_expr արտահայտության արժեքը, իսկ եթե կեղծ է (տրամաբանական 0), ապա վերադարձվում է false\_expr արտահայտության արժեքը: Եթե condition\_expr տրամաբանական արտահայտության արժեքը ստացվում է x, ապա համեմատվում են true\_expr և false\_expr արտահայտությունների արժեքները բիթ առ բիթ՝ չհամընկնող բիթերը փոխարինվում են x-երով: Դիտարկենք կիրառության օրինակներ:

Հետևյալ օրինակում մոդելավորվում է եռավիճակ ելքով բուժեր.

```
wire [15:0] busa = drive_busa ? data : 16'bz;
```

busa անվանմամբ 16-բիթ դողին վերագրվում է data անվանմամբ տվյալների վեկտորը, եթե drive\_busa արտահայտության արժեքը տրամաբանական 1 է, հակառակ դեպքում այդ դողը գտնվում է բարձր դիմադրության վիճակում (z):

Պայմանական օպերատորով կարելի է մոդելավորել 2:1 մուլտիպլեքսոր՝

```
wire y=sel ? a:b; //եթ s=1, ապա y=a, իսկ եթե s=0, ապա y=b
```

Պայմանական օպերատորները կարող են միմյանց մեջ ներդրվել: Հետևյալ օրինակը մոդելավորում է s1, s0 ընտրության մուլտիպլեքսոր 4:1 մուլտիպլեքսոր՝

```
wire y=s1 ? ( s0 ? in3:in2 ) : (s0 ? in1:in0); //եթ s1=1, ապա ընտրվում է in3 կամ in2, իսկ եթե  
//s1=0, ապա ընտրվում է in1 կամ in0
```

### 7.6.5. Տվյալների հոսքերի մակարդակով մոդելավորման օրինակներ

Որևէ թվային համակարգ կարող է մոդելավորվել տրամաբանական տարրերի մակարդակով, տվյալների հոսքերի մակարդակով և վարքագծային նկարագրության մակարդակով: Դիտարկենք տվյալների հոսքերի մոդելավորման օրինակներ.

**Օրինակ 7.15.** Գրել Վերիլոգ կոդ

$$y1 = \bar{b}c + a\bar{c}d$$

$$y2 = a\bar{b}c + b\bar{d}$$

տրամաբանական ֆունկցիաներով նկարագրվող համակցական սարքի նմանակման համար:

```
module comb_circuit1(a,b,c,d,y1,y2);
```

```
// մուլտիպլեքսորների հայտարարում
```

```
input a,b,c,d;
```

```
output y1,y2;
```

```
//ելքերը նկարագրող տրամաբանական արտահայտություններ
```

```
assign y1=~b&c | a&~c&d;
```

```
assign y2=a&~b&c | b&~d;
```

```
endmodule
```

**Օրինակ 7.16.** Մոդելավորել 4:1 մուլտիպլեքսոր՝ օգտագործելով տրամաբանական արտահայտություններ:

```
module mux4:1 (s1, s0, in0, in1, in2, in3, y);
```

```
input i0, i1, i2, i3;
```

```
input s1, s0;
```

```
output y;
```

```
assign y=(~s1 & ~s0 & i0)|
```

```
(~s1 & s0 & i1) |
```

```
(s1 & ~s0 & i2) |
```

```
(s1 & s0 & i3) ;
```

```
endmodule
```

**Օրինակ 7.17.** Մոդելավորել 4-բիթ անցման հաջորդական փոխանցմամբ երկուական գումարիչ՝ օգտագործելով տրամաբանական և թվաբանական արտահայտություններ: Սկզբում ներկայացնենք 1-բիթ գումարիչի մոդուլը:

```
module full_adder(cin, a,b,s,cout):
    output s;
    output cout;
    input a, b;
    input cin;

    // երկուական գումարիչի տրամաբանական արտահայտությունը
    assign {cout, s} = a + b + c_in; //արդյունքը ներկայացվում է երկու բիթով՝ ավագ
    // բիթը՝ cout, կրտսեր բիթը՝ s
endmodule
```

Քառաբիթ գումարիչի կառուցման համար այս մոդուլի օրինակները կարելի է տեղադրել օրինակ 7.14-ի adder4 մոդուլում: Գումարիչի աշխատունակությունը կարելի է ստուգել test\_adder4 թեստավորման մոդուլով:

**Օրինակ 7.18.** Փոխանցման կանխագուշակմամբ քառաբիթ գումարիչ: Վերիլոգ մոդուլում պետք է տրամաբանական արտահայտություններով նկարագրել (3.12)-(3.16) տրամաբանական ֆունկցիաները, որոնք կրկնվում են ստորև հարմարության համար.

$$P_i = A_i \oplus B_i$$

$$G_i = A_i \& B_i$$

$$S_i = P_i \oplus C_i$$

$$C_{i+1} = G_i + P_i C_i$$

$$C_2 = G_1 + P_1 C_1$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 (G_1 + P_1 C_1) = G_2 + P_2 G_1 + P_2 P_1 C_1$$

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$$

$$C_5 = G_4 + P_4 C_4 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 C_1$$

```
module fulladd4(sum, c_out, a, b, c_in);
    // ելքեր և մուտքեր
    output [3:0] sum;
    output c_out;
    input [3:0] a,b;
    input c_in;

    // ներքին ցանցեր
    wire p0,g0, p1,g1, p2,g2, p3,g3;
    wire c4, c3, c2, c1;

    // հաշվել անցման տարածման p փոփոխականները
    assign p0 = a[0] ^ b[0],
    p1 = a[1] ^ b[1],
```

```

p2 = a[2] ^ b[2],
p3 = a[3] ^ b[3];

// հաշվել անցման գեներացման g փոփոխականները
assign g0 = a[0] & b[0],
g1 = a[1] & b[1],
g2 = a[2] & b[2],
g3 = a[3] & b[3];

// հաշվել գումարիչի յուրաքանչյուր կարգից անցման բիթը
assign c1 = g0 | (p0 & c_in),
c2 = g1 | (p1 & g0) | (p1 & p0 & c_in),
c3 = g2 | (p2 & g1) | (p2 & p1 & g0) | (p2 & p1 & p0 & c_in),
c4 = g3 | (p3 & g2) | (p3 & p2 & g1) | (p3 & p2 & p1 & g0) |
(p3 & p2 & p1 & p0 & c_in);
// հաշվել գումարի բիթերը
assign sum[0] = p0 ^ c_in,
sum[1] = p1 ^ c1,
sum[2] = p2 ^ c2,
sum[3] = p3 ^ c3;

// c4-ը վերագրել c_out-ին
assign c_out = c4;

endmodule

```

Գումարիչի աշխատունակությունը կարելի է ստուգել օրինակ 7.14-ի test\_adder4 թեստավորման մոդուլով:

**Օրինակ 7.19.** Բազմաբիթ երկուական թվերի կոմպարատոր: Կոմպարատորը նկարագրվում է (3.27)-(3.29) ֆունկցիաներով: Հետևյալ վերիլոգ կոդում օգտագործվում է `define կոմպիլատորի դիրեկտիվը: Դա թույլ է տալիս digital\_comparator մոդուլն օգտագործել բիթերի ցանկացած թվերի դեպքում: Պետք է փոխել միայն `define դիրեկտիվում նշել բիթերի անհրաժեշտ թիվը:

`define width 8 // կոմպիլացիայի ժամանակ (`width)-ը փոխարինվելու է 8-ով:

```

module digital_comparator (a,b, f, fi, g);

// ելքեր և մուտքեր
output f, fi, g;
input [`width-1:0] a,b; //վեկտորի չափսը տրված է (`width)-ով

assign f=(a>b),
fi=(a==b),
g=(a<b);

endmodule

```

### 7.6.6. Խնդիրներ

**Խնդիր 7.14.** Նկարագրել 1-բիթ հանիչի Վերիլոգ մոդուլը ըստ 3.5 իսկության աղյուսակի: Սկզբում աղյուսակից դուրս գրել նվազարկված Bout և D ֆունկցիաները դիզյունկտիվ նորմալ տեսքով: Տեղադրել հանիչի մոդուլի օրինակը թեստավորման մոդուլում և ստուգել բոլոր 8 մուտքային հավաքածուների դեպքում:

**Խնդիր 7.15.** Նկարագրել (3.49) հավասարումներով տրված Յեմինգի (7,4) կոդի կոդավորիչ:

**Խնդիր 7.16.** Նկարագրել միակարգ թվերի համեմատման մոդուլ ըստ (3.39) և (3.40) հավասարումների: Նկարագրել քառաբիթ թվերի համեմատման մոդուլ՝ տեղադրելով միակարգ թվերի համեմատման մոդուլների օրինակներ բարձր մակարդակի մոդուլում:

### 7.7 Վարքագծային մոդելավորում

Թվային համակարգերի աճող բարդության հետ միասին շատ կարևոր է դառնում անսխալ որոշումներ կայացնելը նախագծման ամենավաղ փուլում: Դրա համար նախագծողը պետք է հնարավորություն ունենա գնահատել տարբեր ճարտարապետություններ և ալգորիթմներ՝ նախքան նախագծվող համակարգի օպտիմալ ճարտարապետության և ալգորիթմի ընտրությունների կատարումը: Գնահատության ամենավաղ փուլում է ալգորիթմական մակարդակով, երբ պարատադիր չէ դեռևս մտնել դրա տարրերի մակարդակով մանրամասների մեջ, կարևորը ալգորիթմի վարքագծի ստուգումն է: Գնահատությունը և ալգորիթմի ընտրելուց և ստուգելուց հետո միայն կարելի է ձեռնամուխ լինել դրանց սխեմաների մակարդակով իրականացմանը:

Վերիլոգը հնարավորություն է ընձեռում նկարագրել թվային համակարգի ֆունկցիոնալությունը ալգորիթմական մակարդակով: Այլ կերպ ասած, նախագծողը հնարավորություն ունի նկարագրել կառուցվող համակարգի վարքագիծը: Վերիլոգը հարուստ է վարքագծային նկարագրության բազմազան կառուցվածքներով, որոնք նախագծողին ընձեռում են դրանց օգտագործման մեծ ճկունություն: Վարքագծային մոդելավորումը նման է C լեզվով ծրագրավորմանը:

#### 7.7.1. Կառուցվածքային ընթացակարգեր (պրոցեդուրներ)

Վերիլոգում գոյություն ունի կառուցվածքային ընթացակարգերի երկու տիպ, որոնք հայտարարվում են՝ **always** և **initial** բանալի բառերով: Դրանք վարքագծային մոդելավորման հիմնական կառուցվածքներն են: Վարքագծային մոդելավորման բոլոր հրամանները պետք է գետեղված լինեն են այս կառուցվածքներում: Ի տարբերություն C կամ Պասկալ ծրագրավորման, որոնցում ակտիվությունները (գործողությունները) կատարվում են հաջորդաբար, Վերիլոգում ակտիվություններն ընթանում են զուգահեռաբար, ինչպես էլեկտրական շղթաներում: Յուրաքանչյուր **always** և **initial** կառուցվածքներ կայացնում է առանձին ակտիվության ընթացք: Յուրաքանչյուր ակտիվության ընթացք սկսվում է ժամանակի 0 պահին: **always** և **initial** կառուցվածքները չեն կարող իրար մեջ ներդրվել:

**initial** կառուցվածքում գետեղված բոլոր հրամանները և վերագրումները կազմում են մեկ **initial** բլոկ: **initial** բլոկում գործողություններն սկսվում են 0 պահին և կատարվում են միայն մեկ անգամ: Յուրաքանչյուր **initial** բլոկի կատարում կարող է ավարտվել՝ անկախ մնացած բլոկներից: Բազմաթիվ վարքագծային հրամաններ կարող են խմբավորվել **initial** բլոկում՝ օգտագործելով **begin** և **end** բանալի բառերը: Ստորև ցույց են տրված **initial** կառուցվածքի օրինակներ:

```
module stimulus;
```

```
reg x,y, a,b, m;
```

```
initial
```

```
  m = 1'b0; //մեկ հրամանի դեպքում (begin ... end)-ը կարող է բացակայել
```

```
initial
```

```
begin
```

```
  #5 a = 1'b1; //բազմաթիվ հրամանները պետք է խմբավորվեն (begin ... end)-ի միջև
```

```
  #25 b = 1'b0;
```

```
end
```

```
initial
```

```
begin
```

```
  #10 x = 1'b0;
```

```
  #25 y = 1'b1;
```

```
end
```

```
initial
```

```
  #50 $finish;
```

```
endmodule
```

Այս օրինակի **initial** բլոկները սկսում են զուգահեռաբար կատարվել 0 պահից: Եթե հրամանից առաջ դրված է հապաղում, այդ հրամանը կկատարվի համապատասխան հապաղումից հետո: Այս բլոկների հրամանների կատարման հաջորդականությունը հետևյալն է.

Ժամանակ կատարված հրամանը

```
0      m = 1'b0;
```

```
5      a = 1'b1;
```

```
10     x = 1'b0;
```

```
30     b = 1'b0;
```

```
35     y = 1'b1;
```

```
50     $finish;
```

**initial** բլոկներն օգտագործվում են սկզբնական վիճակների տրման, նմանական ընթացքի հսկման և այլ գործողությունների համար, որոնք պետք է կատարվեն միայն մեկ անգամ նմանական ողջ ընթացքում: **initial** բլոկները չեն կարող օգտագործվել թվային համակարգի սինթեզի համար գրվող ռեգիստրների մակարդակով փոխանակումների նկարագրության Վերիլոգ կոդերում:

**initial** բլոկները կարելի է համատեղել փոփոխականների հայտարարության հետ: Դա թույլ է տալիս կրճատել ծրագրի տեքստը: Օրինակ,

```
reg clock = 0; //հայտարարել clock անունով ռեգիստր և վերագրել 0 արժեք
```



Սա համարժեք է հետևյալին.

```
reg clock; //հայտարարել clock անունով ռեգիստր
initial clock = 0; // վերագրել 0 արժեք
```

**always** կառուցվածքում զետեղված բոլոր հրամանները և վերագրումները կազմում են մեկ **always** բլոկ: **always** բլոկում գործողությունների կատարումն սկսվում է 0 պահին և օղակաձև անվերջ շարունակվում է: **always** կառուցվածքն օգտագործվում է թվային շղթաների ակտիվության ընթացքի մոդելավորման համար, որն անընդհատ կրկնվում է: Օրինակ, ստորև ցույց տրված տակտային ազդանշանի գեներատորի մոդուլի ելքային ազդանշանը վիճակը փոխում է յուրաքանչյուր կես պարբերությունը մեկ.

```
module clock_gen (clock);
output clock;
reg clock;

initial
  clock = 1'b0; // clock փոփոխականի սկզբնական վիճակը դնել 0
always
  #10 clock = ~clock; // կես պարբերությունը մեկ փոխել վիճակը (պարբերությունը = 20)
initial
  #1000 $finish;
endmodule
```

Նկատել, որ clock փոփոխականի սկզբնական վիճակը տրվում է առանձին **initial** հրամանով: Եթե clock-ի սկզբնական վիճակը տրվեր **always** բլոկի ներսում, յուրաքանչյուր անգամ **always** բլոկ մտնելիս այն նորից կկատարվեր: Նկատենք նաև, որ նմանակման ավարտը պետք է տրվի **initial** կառուցվածքում: Եթե չկա նմանակման դադարեցման **\$stop** կամ **\$finish** հրաման, նմանակումը կշարունակվի անվերջ: Եթե նմանեցնենք իրական թվային համակարգին, ապա **\$stop**-ը մոդելավորում է ընդհատումը, **\$finish**-ը՝ սնուցման անջատումը:

Բազմաթիվ վարքագծային հրամաններ կարող են խմբավորվել մեկ **always** բլոկում՝ օգտագործելով **begin** և **end** բանալի բառերը:

### 7.7.2. Ընթացակարգային վերագրումներ (Procedural Assignments)

Ընթացակարգային վերագրումներով ընթացիկ արժեքներ են տրվում **reg**, **integer**, **real** կամ **time** տիպի փոփոխականներին: Որևէ փոփոխականի տրված արժեքը մնում է անփոփոխ, քանի դեռ որևէ ընթացակարգային վերագրում այն չի փոխել: Դրանով ընթացակարգային վերագրումը տարբերվում է անընդհատ վերագրումից, որն օգտագործվում է տվյալներ հոսքերի մոդելավորման համար: Անընդհատ վերագրման աջ մասի արտահայտության արժեքն անընդհատ տրվում է ձախ մասի ցանցին:

Գոյություն ունի ընթացակարգային վերագրումների երկու տիպ՝ արգելափակող (blocking) և չարգելափակող (nonblocking):

**Արգելափակող վերագրումները** կատարվում են այն հերթականությամբ, ինչպես դրանք գետեղված են հաջորդական բլոկում: Դիտարկենք հետևյալ օրինակը.

```
reg x, y, z;
reg [15:0] a, b;
integer count;

initial //բլոկը վարքագծային հրամանները պետք է լինեն initial կամ always բլոկներում
begin
  x = 0; y = 1; z = 1; //սկալյարի վերագրում
  count = 0; // integer տիպի փոփոխականի վերագրում
  a = 16'b0; b = a; //վեկտորի վերագրում
  #15 a[2] = 1'b1; //ընտրված բիթի վերագրում հապաղումով
  #10 b[15:13] = {x, y, z} //միակցման արդյունքի վերագրում վեկտորի ընտրված մասին
  count = count + 1; // integer տիպի փոփոխականի վերագրում
end
```

**begin ... end** բլոկում հրամանները կատարվում են հաջորդաբար՝  $y = 1$  հրամանը կատարվում է միայն  $x = 0$  հրամանից հետո,  $count = count + 1$  հրամանը կատարվում է վերջինը:

- $t=0$  պահին կատարվում են  $x = 0$ -ից մինչև  $b = a$  հրամանները,
- $time = 15$  պահին կատարվում է  $a[2] = 1$  հրամանը,
- $time = 25$  պահին կատարվում են  $b[15:13] = \{x, y, z\}$  և  $count = count + 1$  հրամանները:

Քանի որ բլոկում առկա են 15 և 10 միավոր հապաղումներ, ապա  $count = count + 1$  հրամանը կատարվում է  $time = 25$  պահին:

Նկատենք, որ ընթացակարգային վերագրման ժամանակ, եթե աջ մասի բիթերի թիվը մեծ է ձախ մասի բիթերի թվից, աջ մասի բիթերի ավել մասը կտրվում է, որպեսզի հավասարվի ձախ մասի բիթերի թվին: Պահվում են կրտսեր բիթերը՝ կտրվում են ավագ բիթերը: Եթե աջ մասի բիթերի թիվը փոքր է ձախ մասից, ապա ձախ մասի ավագ բիթերը լցվում են զրոներով:

**Չարգելափակող վերագրումները** թույլ են տալիս վերագրումները ժամանակի մեջ իրականացնել առանց հրամանների մեկ մասի կատարման արգելափակման հաջորդական բլոկում՝ ավելի վաղ հադիպող հրամանների կողմից: Չարգելափակող վերագրման օպերատորն է “<=”: Նկատենք, որ այդ նույն օպերատորը օգտագործվում է նաև փոքր կամ հավասար հարաբերության օպերատորի համար: “<=” օպերատորի ֆունկցիան Վերիլոգ կոմպիլատորը մեկնաբանում է համատեքստից. վերագրման ժամանակ իբրև չարգելափակող վերագրում, որևէ արտահայտության մեջ՝ իբրև հարաբերության օպերատոր: Հետևյալ օրինակում ցույց են տրված չարգելափակող վերագրումներ՝ նախորդ օրինակի արգելափակող վերագրումների մեկ մասը փոխարինվել են չարգելափակողներով: Այս օրինակը նաև ցույց է տալիս արգելափակող և չարգելափակող վերագրումների տարբերությունը:

```
reg x, y, z;
reg [15:0] a, b;
integer count;
```

//բոլոր վարքագծային հրամանները պետք է լինեն initial կամ always բլոկներում

**initial**

**begin**

x = 0; y = 1; z = 1; //սկալյարի վերագրում

count = 0; // integer տիպի փոփոխականի վերագրում

a = 16'b0; b = a; //վեկտորի վերագրում

a[2] <= #15 1'b1; //ընտրված բիթի վերագրում հապաղումով

b[15:13] <= #10 {x, y, z} //միակցման արդյունքի վերագրում վեկտորի ընտրված մասին

count <= count + 1; // integer տիպի փոփոխականի վերագրում

**end**

Այս օրինակում x = 0-ից մինչև b = a կատարվում են t=0 պահին: Որից հետո երեք չարգելափակող վերագրումները կատարվում են նույն պահին:

1. a[2] = 1 կատարվում է 15 միավոր հապաղումով (t = 15 պահին)

2. b[15:13] = {x, y, z} կատարվում է 10 միավոր հապաղումով (t = 10 պահին)

3. count = count + 1 կատարվում է առանց հապաղման (t = 0 պահին)

Դիտարկենք արգելափակող և չարգելափակող վերագրումների տարբերությունը բացահայտող ևս մեկ օրինակ.

**reg a, b;**

**initial**

**begin**

a=1'b0;

b=1'b1;

a=b;

b=a;

**end**

Այս **initial** բլոկի կատարումից հետո a և b ռեգիստրներում կստացվի՝

a=1

b=1

Եթե վերջին երկու վերագրումների հաջորդականությունը փոխենք **initial** բլոկում, ապա արդյունքը կլինի՝

a=0

b=0

Վերագրումների հաջորդականությունից արդյունքի փոփոխությունը համարժեք է մրցումներին թվային շղթաներում:

Այժմ **initial** բլոկում վերջին երկու վերագրումները դարձնենք չարգելափակող.

**reg a, b;**

**initial**

**begin**

a=1'b0;

b=1'b1;

a<=b;

b<=a;

**end**

Այս դեպքում **initial** բլոկի կատարումից հետո  $a$  և  $b$  ռեգիստրներում կստացվի՝

$a=1$

$b=0$

Չարգելափակող վերագրումները կատարվում են միաժամանակ (զուգահեռ)՝  $a \leq b$  վերագրումը չի արգելափակում այդ նույն պահին  $b \leq a$  վերագրումը:  $b$ -ին վերագրվում է  $a$ -ի այն արժեքը, որն առկա էր նախքան  $a \leq b$  վերագրման կատարումը: Չարգելափակող վերագրումների հաջորդականությունը բլոկում չի ազդում արդյունքի վրա:

Այս օրինակներում միևնույն բլոկում տեղադրվել են արգելափակող և չարգելափակող վերագրումներ՝ միայն դրանց արդյունքը ցույց տալու համար: Ընդհանրապես, խորհուրդ չի տրվում դրանք խառնել մեկ բլոկում: Սինթեզի գործիքները դա արգելում են:

Կարևոր է հասկանալ չարգելափակող վերագրումների նշանակությունը թվային նախագծերում: Դրանք օգտագործվում են թվային շղթայում տվյալների համատեղ փոխանակումների մոդելավորման համար, երբ այդ փոխանակումները տեղի են ունենում նույն՝ այսինքն՝ որևէ պատահարի տեղի ունենալու պահին: Դիտարկենք հետևյալ օրինակը, որտեղ երեք միաժամանակյա տվյալների փոխանակումները տեղի են ունենում տակտային իմպուլսի դրական ճակատի հայտնվելու պահին:

**always @(posedge clock)**

**begin**

**reg1** <= #1 in1;

**reg2** <= @(negedge clock) in2 ^ in3;

**reg3** <= #1 reg1; // reg1-ի արժեքը նախքան այս բլոկ մտնելը

**end**

Տակտային ազդանշանի դրական ճակատի հայտնվելու յուրաքանչյուր պահի (**@(posedge clock)**) տեղի է ունենում չարգելափակող վերագրումների հետևյալ հաջորդականությունը.

1. clock իմպուլսի դրական ճակատի պահին ընթերցվում են աջ մասի փոփոխականները: Հաշվվում են աջ մասի արտահայտությունների արժեքները և պահվում են նմանակիչի ներսում:
2. Ձախ մասի փոփոխականների վերագրումը կատարվում է ըստ վերագրումների ներսում տրված հապաղումների՝ **reg1**-ում գրվում է clock-ի դրական ճակատից 1 միավոր հետո, **reg2**-ում clock-ի հաջորդ բացասական ճակատի պահին (**@(negedge clock)**), **reg3**-ում clock-ի դրական ճակատից 1 միավոր հետո:

**always** բլոկում վերագրումների հաջորդականությունը կարևոր չէ, քանի որ ձախ մասի փոփոխականներին վերագրվում են նմանակիչի ներսում պահվող արժեքները: Օրինակ, **reg3**-ին վերագրվում է **reg1**-ի հին արժեքը, որն ընթերցվել էր clock իմպուլսի դրական ճակատի պահին, նախքան **reg1**-ին նոր արժեք վերագրելը: Հետևաբար, **reg1**, **reg2**, և **reg3** ռեգիստրների վերջնական արժեքները կախված չեն վերագրումների կատարման հաջորդականությունից:

Բոլոր այն դեպքերում, երբ թվային համակարգում տեղի է ունենում տվյալների միաժամանակյա փոխանակում որևէ պատահարի պահի, խորհուրդ է տրվում օգտագործել չարգելափակող վերագրումների՝ արգելափակողների փոխարեն: Նման դեպքերում արգելափակող վերագրումների օգտագործումը կարող է առաջ բերել մրցումներ, քանի որ վերջնական արդյունքը կախված է վերագրումների կատարման հաջորդականությունից:

### 7.7.3. Նմանակման ժամանակի կառավարում

Վերիլոգում կան նմանակման ժամանակի վարքագծային կառավարման տարբեր եղանակներ: Եթե Վերիլոգ ծրագրում բացակայում են ժամանակի կառավարման հրամանները, ապա նմանակման ժամանակը առաջ չի գնա: Ժամանակի կառավարումը հնարավորություն է տալիս որոշել նմանակման ժամանակի պահերը, երբ վարքագծային հրամանները պետք է կատարվեն: Ժամանակի կառավարումը կարելի է իրագործել երեք եղանակով՝ հապաղումների վրա հիմնված կառավարում, պատահարների վրա հիմնված կառավարում և մակարդակի նկատմամբ զգայուն կառավարում:

#### 7.7.3.1. Հապաղումների վրա հիմնված ժամանակի կառավարում

Հապաղման վրա հիմնված ժամանակի կառավարման դեպքում տրված հապաղման չափը որոշում է նմանակման ընթացքում տվյալ հրամանի հանդիպման և դրա կատարման միջև ընկած ժամանակահատվածը: Հապաղումը տրվում է # սինվոլով: Հապաղման չափը կարելի է տալ թվով, իդենտիֆիկատորով, արտահայտությամբ, min:typ:max կառուցվածքով: Կա հապաղմամբ կառավարման երեք տիպ՝ սովորական հապաղմամբ, վերագրման ներքին հապաղմամբ, զրո հապաղմամբ:

**Սովորական հապաղմամբ կառավարման** դեպքում զրոյից տարբեր հապաղումը նշվում է ընթացակարգային վերագրումից առաջ: Օրինակներ.

```
`define dly 100    // սահմանել dly հապաղման մեծության արժեքը
module regular_delay;

parameter coarse_delay = 50; // սահմանել հապաղման մեծության պարամետրեր
parameter fine_delay = 5;
reg a, c, d; //հայտարարել փոփոխականները
reg [2:0] b, d;

initial
begin
a = 0; // առանց հապաղման կառավարման վերագրում
#4 b = 3'd3; // հապաղումը տրված է թվով
# coarse_delay c = 0; // հապաղումը տրված է իդենտիֆիկատորով՝ պարամետրով
#(coarse_delay + fine_delay) d = a+b; // հապաղումը տրված է արտահայտությամբ
#b a = a+ 1; // հապաղումը տրված է իդենտիֆիկատորով՝ b փոփոխականով
#(2:3:4) d = 0; // հապաղումը տրված է min:typ:max կառուցվածքով
#`dly c=1; // հապաղումը տրված է կոմպիլյատորի դիրեկտիվով
#20 $finish;
end
endmodule
```

Ընթերցողին խորհուրդ է տրվում հաշվել այս օրինակում յուրաքանչյուր վերագրման կատարման պահը և այն ստուգել նմանակումով:

Վերագրման **ներքին հապաղմամբ կառավարման** դեպքում հապաղման արժեքը նշվում է վերագրման օպերատորից աջ:

```
reg a, b, c;  
initial  
begin  
  a = 0; b = 1;  
  c = #5 a + b; //վերցնել a և b-ի արժեքները t=0 պահին, հաշվել a+b-ն, սպասել  
  // ժամանակի 5 միավոր, այնուհետև վերագրել c-ին  
end
```

Սովորական հապաղմամբ վերագրման և ներքին հապաղմամբ կառավարման տարբերությունն այն է, որ սովորական հապաղումն ուշացնում է ամբողջ վերագրման հրամանի կատարումը, իսկ վերագրման ներքին հապաղման դեպքում աջ մասի արտահայտությունը հաշվվում է ընթացիկ պահին, բայց վերագրվում է ձախ մասին ուշացումով:

**always-initial** բլոկների կատարման հաջորդականությունը որոշակի չէ. դրանք կարող են կատարվել ցանկացած հաջորդականությամբ: **Ջրո հապաղմամբ ժամանակի կառավարումը** թույլ է տալիս կառավարել **always-initial** բլոկների կատարման հաջորդականությունը: Դիտարկենք հետևյալ երկու **initial** բլոկները.

```
initial  
#0 x = 1; // զրո հապաղմամբ ժամանակի կառավարումը
```

```
initial  
x = 0;
```

Այստեղ  $x = 1$ ,  $x = 0$  վերագրումները պետք կատարվեն միաժամանակ: Սակայն, **#0** հապաղման շնորհիվ  $x = 1$  վերագրումը կկատարվի վերջինը՝ արդյունքը կլինի  $x = 1$ : Եթե **#0** հապաղումը չլինի, արդյունքը կլինի 0 կամ 1, որը մինչև ծրագրի կատարումը հնարավոր չէ կանխագուշակել, այսինքն կունենանք 0 և 1 արժեքների մրցում: Սակայն, **#0** հապաղման կիրառումը խորհուրդ չի տրվում, քանի որ այն կարող է մտցնել նմանական և իրական սարքի վարքագծի անհամաձայնություն:

### 7.7.3.2. Պատահարների վրա հիմնված ժամանակի կառավարում (event based timing control)

Պատահարը ռեգիստրի կամ ցանցի արժեքի փոփոխություն է: Պատահարը կարելի է օգտագործել՝ որևէ հրամանի կամ հրամանների բլոկի կատարումը թույլատրելու համար: Գոյություն ունի պատահարներով ժամանակի կառավարման չորս տիպ՝ սովորական պատահարով (regular event control), անվանումով պատահարով (named event control), պատահարների ցուցակով (sensitivity list), փոփոխականի մակարդակի նկատմամբ զգայուն (level-sensitive timing control)։

**Սովորական պատահարով կառավարման** համար օգտագործվում է **@** սիմվոլը, որին հաջորդում է այն փոփոխականի անվանումը փակագծերում, որի փոփոխությունը հանդիսանում է պատահար: Այդ պատահարը կարող է լինել փոփոխականի արժեքի փոփոխություն կամ դրական, կամ բացասական անցում (ճակատ): Դրական են համարվում 0-1, 0-x, 0-z, x-1, z-1 անցումները: Բացասական են համարվում 1-0, 1-x, 1-z, x-0, z-0 անցումները: Դրական անցման համար օգտագործվում է **posedge** բանալի բառը, իսկ բացասականի համար՝ **negedge**-ը: Սովորական պատահարով կառավարման օրինակներ.

```
@(date) a = b; //a = b կատարվում է date ազդանշանի փոփոխության պահին
@(posedge clock) q = ~q & j | q & ~k; // q-ին վերագրվում է աջ մասի արտահայտությունը
// clock ազդանշանի դրական ճակատի պահին
@(negedge reset) q = 0; // q = 0-ն կատարվում է reset ազդանշանի բացասական
//ճակատի պահին
```

Վերիլոգը հնարավորություն է տալիս հայտարարել **անվանումով պատահար** և այդ պատահարը տեղի ունենալու պահին ակտիվացնել (trigger) այն: Այդ պատահարը այնուհետև կարելի է օգտագործել ուրիշ հրամանների ժամանակի կառավարման արտահայտություններում, որպեսզի գործողությունների կատարումը թույլատրվի այս պատահարի ակտիվացմամբ: Անվանումով պատահարները միաժամանակ ընթացող պրոցեսների սինքրոնացման մոդելավորման հզոր գործիք են:

Անվանումով պատահարը հայտարարվում է **event** բանալի բառով, այն տվյալ չի պահում: Պատահարի ակտիվացումը տրվում է “->” սիմվոլով: Պատահարի ակտիվացումը ստուգվում է **@** սիմվոլով: Անվանումով պատահարի հայտարարման, ակտիվացման և ստուգման օրինակ.

```
//Այս օրինակում մոդելավորվում է միկրոկոնտրոլլերի ընդհանուր գործածության 8
//ռեգիստրի պարունակության փոխանցումը դեզի հիշողություն, երբ ստացվում է
//ընդհատման պահանջ int0 կամ int1 գծով
```

```
event interrupt; //հայտարարել interrupt անվանումով պատահարը
always @(posedge clock) // սպասել clock ազդանշանի դրական ճակատին
begin
if(int0 || int1) // եթե ստացվում է ընդհատման պահանջ int0=1 կամ int1=1 գծով
->interrupt; //ապա ակտիվացնել շրջել interrupt պատահարը
end

always @( interrupt) //ստուգել՝ սպասել interrupt պատահարի ակտիվացմանը (շրջվելուն),
//երբ interrupt-ը շրջվում է, ընդհանուր գործածության 8 ռեգիստրների
// պարունակությունը փոխանցել load_stack անունով ռեգիստր:
load_stack = {R0, R1, R2, R3, R4, R5, R6, R7}; // օգտագործվել է միակցման { }
//օպերատորը
```

Հաճախ որևէ հրամանի կամ հրամանների բլոկի կատարումը ակտիվացվում է ոչ թե մեկ, այլ մի քանի փոփոխականների կամ պատահարների փոփոխությամբ: Նման դեպքերում այդ փոփոխականները կամ պատահարները կարող են տրվել մեկ ընդհանուր **ցուցակով**: Ցուցակում դրանք միմյանցից բաժանվում են ստորակետով կամ **or** բա-

նալի բառով: Նման ցուցակը կոչվում է նաև զգայունության ցուցակ (sensitivity list):

```
// clk տակտային իմպուլսի դրական ճակատով կառավարվող D տրիգեր: Ցուցակում  
// պատահարները միմյանցից բաժանված են or բանալի բառով
```

```
always @( posedge clk or negedge reset) //սպասել posedge clk կամ  
//negedge reset պատահարներին
```

```
begin
```

```
if(!reset) //եթե reset=0, տրիգերը կարգել 0 վիճակ:
```

```
  q <=0;
```

```
else
```

```
  q <=d; //հակառակ դեպքում տրիգերում գրանցել d
```

```
end
```

```
// 2:1 մուլտիպլեքսոր: Ցուցակում փոփոխականները միմյանցից բաժանված են ստորակետով
```

```
always @( sel, a, b) //սպասել sel, a, b փոփոխականներից որևէ մեկի փոփոխությանը  
y=sel ? a : b; // եթե sel=1, ապա y = a, իսկ եթե sel=0, ապա : b, y = b
```

Երբ զգայունության ցուցակում փոփոխականների թիվը մեծ է, ապա դժվար կլինի այն վերահսկել. որևէ փոփոխականի բացթողում այդ ցուցակից կհանգեցնի մոդելավորման և իրական սխեմայի տարբերության: Համակցական սխեմաների մոդելավորման ժամանակ կարելի է օգտվել **@\*** և **@(\*)** հատուկ սինվոլներից, որոնք լրիվ համարժեք են: Այդ սինվոլները զգայուն են դրանց հետևող հրամանի կամ հրամանների բլոկի մեջ մտնող բոլոր փոփոխականների փոփոխություններին: Այսինքն **@\*** և **@(\*)** հատուկ սինվոլների կիրառությունը համարժեք է **@(փոփոխականների լրիվ ցուցակը)**-ին: Դիտարկենք համակցական սխեմայի օրինակ.

```
//այս օրինակում բոլոր մուտքային փոփոխականները բացահայտ թվարկված են
```

```
//հավանականությունը մեծ է, որ որևէ մեկը բաց թողնվի
```

```
always @(a, b, c, d, e, f, g, h, p, m, x, y, u, v)
```

```
begin
```

```
  out1 = u ? a&b | c : d ^ e ^ f;
```

```
  out2 = v ? g&h | p : m ^ x ^ y;
```

```
end
```

```
//այս օրինակում օգտագործվում է @(*) սինվոլը, բոլոր փոփոխականները ավտոմատ
```

```
// ընդգրկված են զգայունության ցուցակում
```

```
always @(*)
```

```
begin
```

```
  out1 = u ? a&b | c : d ^ e ^ f;
```

```
  out2 = v ? g&h | p : m ^ x ^ y;
```

```
end
```

**Փոփոխականի մակարդակի նկատմամբ զգայուն ժամանակի կառավարում** (level-sensitive timing control): Մինչ այժմ մենք դիտարկեցինք ազդանշանի կամ պատահարի փոփոխությամբ ժամանակի կառավարումը, որը տրվում է **@** սինվոլով: Վերլուծը թույլ է տալիս ժամանակի կառավարումն իրականացնել, նաև, փոփոխականների մակարդակների (արժեքների) միջոցով՝ որևէ հրամանի կամ հրամանների բլոկի կատարումը թույլատրվում է որոշակի պայմանի բավարարման դեպքում՝ սպասում է այդ պայմանի ճշմարիտ լինելուն: Փոփոխականի մակարդակի նկատմամբ զգայուն ժամա-



նակի կառավարում տրվում է **wait** բանալի բառով:

**always**

```
wait (!enable) #10 a=b;
```

Այս օրինակում enable ազդանշանը հսկվում է անընդհատ: Եթե enable=1, ապա a=b հրամանը չի կատարվում, իսկ եթե enable=0 (!enable պայմանը տեղի ունի), ապա կատարվում է a=b հրամանը՝ 10 միավոր հապաղումից հետո:

#### 7.7.4. Ծրագրի ճյուղավորումների հրամաններ

Վերիլոգում ծրագրի ճյուղավորումների հրամանները նման են այլ ծրագրավորման լեզուների հրամաններին: Դրանք օգտագործվում են որոշակի պայմանների դեպքում ծրագրի ընթացքի ճյուղավորման որոշում կայացնելու համար: Վերիլոգում ճյուղավորումների հրամաններն են՝ պայմանական հրամանները և բազմաճյուղ ճյուղավորման հրամանները:

Պայմանական հրամաններն (կամ **if-else** հրամանները) օգտագործում են **if** և **else** բանալի բառերը: Գոյություն ունեն պայմանական հրամանների երեք տիպեր՝

**if** տիպի (առանց **else** մասի) հրամանի ձևաչափը հետևյալն է.

```
if (expression) //եթե expression պայմանը տեղի ունի,  
true_statement; //ապա true_statement հրամանը կատարվում է:  
//Եթե պայմանը տեղի չունի, հրամանը չի կատարվում
```

**if-else** տիպի հրամանի ձևաչափը հետևյալն է.

```
if (expression) //եթե պայմանը տեղի ունի, կատարվում է  
true_statement ; // true_statement-ը,  
else // հակառակ դեպքում՝  
false_statement ; // false_statement-ը:
```

**if-else-if** տիպի հրամանի ձևաչափը հետևյալն է.

```
if (expression1); //եթե <expression1>-ի արժեքը 0-ից տարբեր է, ապա  
true_statement1; //կատարվում է true_statement1-ը  
else if (expression2) //եթե <expression2>-ի արժեքը 0-ից տարբեր է, ապա  
true_statement2 ; //կատարվում է true_statement2-ը  
else if (expression3) //եթե <expression3>-ի արժեքը 0-ից տարբեր է, ապա  
true_statement3 ; //կատարվում է true_statement3-ը  
else //եթե <expression1>...<expression3> պայմաններից  
//ոչ մեկը տեղի չունի, ապա  
default_statement ; //կատարվում է default_statement –ը
```

Ստորև ցույց են տրված այդ կառուցվածքների կիրառության օրինակներ.

- if կառուցվածք (տրիգերը դնել 0 վիճակ reset ազդանշանով).

```
if(!reset == 1'b0)  
q=1'b0;
```

- if-else կառուցվածք (2:1 մուլտիպլեքսոր).

```
if(sel==0)  
q=in0;
```

```

else
    q=in1;

    • if-else-if կառուցվածք (4:1 մուլտիպլեքսոր).
if(sel1==0 && sel0==0)
    q=in0;
else if(sel1==0 && sel0==1)
    q=in1;
else if(sel1==1 && sel0==0)
    q=in2;
else
    q=in3;

```

Շատ դեպքերում ծրագրի ճյուղավորման պայմանների թիվը մեծ է և **if-else-if** կառուցվածքի կիրառությունը դժվարանում է: Վերիլոգն ունի հատուկ բազմատարբերակ ճյուղավորումների **case** հրաման, որն ստուգում, է թե տրված արտահայտությունը տրված ցուցակից որ արտահայտության հետ է համընկնում և կատարում է համապատասխան ճյուղավորումը: **case** հրամանի բլոկն ընդգրկվում է **case** և **endcase** բանալի բառերի միջև:

```

case (expression)
alt1: statement1;
alt2: statement2;
alt3: statement3;
...
...
default: default_statement;
endcase

```

statement1, statement2 ... հրամաններից յուրաքանչյուրը կարող է լինել մեկ առանձին հրաման կամ հրամանների բլոկ, որը պետք է ներառվի **begin** և **end** բանալի բառերի միջև: **case** բանալի բառին հաջորդող (expression) արտահայտության հաշվված արժեքը համեմատվում է alt1, alt2, .. այլընտրանքների հետ գրված հաջորդականությամբ: Կատարվում է այն հրամանը կամ հրամանների բլոկը, որի հետ առաջինն է համընկնում expression-ի արժեքը: Եթե expression-ի արժեքը այլընտրանքներից ոչ մեկի հետ չի համընկնում, կատարվում է default\_statement հրամանը, որի այլընտրանքը տրվում է **default** (ըստ նախնական պայմանավորվածության) բանալի բառով: default\_statement-ի առկայությունը **case** կառուցվածքում պարտադիր չէ, բայց նախընտրելի է: Մեկից ավելի default\_statement չի թույլատրվում: Բլոկները կարող են լինել իրար մեջ ներդրված:

**case** հրամանը օգտակար է օգտագործել վերծանիչների և մուլտիպլեքսորների նկարագրության համար: Ուսումնասիրենք **case** հրամանի չափաձևը և ուղղագրությունը հետևյալ 4:10 վերծանիչի օրինակի օգնությամբ:

```

reg[3:0] rega;      //վերծանիչի մուտքեր
reg [9:0] result;    //վերծանիչի ելքեր

case (rega)

```

```

4'd0: result = 10'b0000000001;
4'd1: result = 10'b0000000010;
4'd2: result = 10'b00000000100;
4'd3: result = 10'b00000001000;
4'd4: result = 10'b00000010000;
4'd5: result = 10'b00000100000;
4'd6: result = 10'b00001000000;
4'd7: result = 10'b00100000000;
4'd8: result = 10'b01000000000;
4'd9: result = 10'b10000000000;
default: result = 10'bx;

```

**endcase**

rega ռեգիստր-վեկտորով տրվում են վերժանիչի մուտքային (հասցեային) փոփոխականները, իսկ result ռեգիստր-վեկտորով նկարագրվում են վերժանիչի ելքերը: Եթե rega-ով տրվող քառաբիթ կոդը 9-ից մեծ է, կկատարվի **default** գործողությունը՝ ելքի գծերի վիճակը կմնա անորոշ:

**case** հրամանի օգտագործման ևս մեկ տիպային օրինակ՝ 4:1 մուլտիպլեքսոր.

```

module mux4_to_1 (q, in0, in1, in2, in3, s1, s0);
//մուլտիպլեք մատույցների հայտարարում
output q;
input in0, in1, in2, in3;
input s1, s0;
reg q; //ելքը հայտարարվում է որպես ռեգիստր, որպեսզի հնարավոր լինի
//կատարել վարքագծային վերագրում
always @(s1 or s0 or in0 or in1 or in2 or in3) //զգայունության ցուցակը
begin
case ({s1, s0}) // s1, s0-ն միակցվել են որպես երկբիթ վեկտոր
    2'b00: q = i0;
    2'b01: q = i1;
    2'b10: q = i2;
    2'b11: q = i3;
default: q = 1'bx; //իրականում սա երբեք չի հանդիպի
endcase
end
endmodule

```

**case** կառուցվածքը տարբերվում է **if-else-if** կառուցվածքից ոչ միայն ուղղագրությամբ.

1. **if-else-if** կառուցվածքն ավելի ընդհանուր է, քանի որ այն ստուգում է բազմաթիվ պայմաններ, այլ ոչ թե համեմատում մեկ առանձին արտահայտությունը բազմաթիվ այլընտրանքների հետ, ինչպես դա արվում է **case** կառուցվածքում:
2. **case** կառուցվածքը տալիս է որոշակի արդյունք մույնիսկ, երբ արտահայտության մեջ կան x և z արժեքներ:

**case** կառուցվածքում համարվում է, որ որևէ այլընտրանք համընկել է տրված

(expression) արտահայտության հետ, եթե դրանք հավասար են բիթ առ բիթ բոլոր հնարավոր արժեքներով՝ 0, 1, x և z: Բոլոր այլընտրանքների բիթերի թիվը պետք է հավասար լինի (expression)-ի բիթերի թվին: x և z արժեքների համեմատության սահմանումը թույլ է տալիս նվազեցնել անորոշությունը: Հետևյալ օրինակը ցույց է տալիս որոշակի արդյունքի ստացում x և z արժեքների դեպքում: Նշենք, որ **case** կառուցվածքում ընդգրկված նույն հրամանի համար կարելի է նշել ստորակետով բաժանված մի քանի այլընտրանքներ: Այդ հրամանը կատարվում է, եթե տրված (expression) արտահայտությունը համընկնում է թվարկված այլընտրանքներից որևէ մեկի հետ:

```
case (select[1:2])
    2'b00: result = 0;
    2'b01: result = flaga;
    2'b0x, 2'b0z: result = flaga?1'bx:0; //կատարվում է, եթե select[1:2] =2'b0x կամ 2'b0z
    2'b10: result = flagb;
    2'bx0, 2'bz0: result = flagb ? 1'bx :0; //կատարվում է, եթե select[1:2] =2'bx0 կամ 2'bz0
    default: result = 1'bx;
endcase
```

Այս օրինակում, եթե flaga=0 և select[1] =0, ապա անկախ select[2]-ի արժեքից result = 0: Հնարավոր անորոշությունը լուծվում է երրորդ հրամանով:

Հետևյալ օրինակում ստուգվում են x և z արժեքները.

```
case(sig)
    1'bz: $display("signal is floating");
    1'bx: $display("signal is unknown");
    default: $display("signal is %b", sig);
endcase
```

Արգելված (ոչ էական) բիթեր պարունակող հավաքածուներով տրված այլընտրանքների համեմատությունը կարելի է իրականացնել **casez** և **casex** կառուցվածքներով: **casez**-ում բարձր դիմադրության վիճակը (z արժեքը) համարվում է ոչ էական, իսկ **casex** –ում և՛ z-ը, և՛ անհայտ x արժեքը համարվում է ոչ էական: Ընդ որում z-ի փոխարեն կարելի է օգտագործել նաև հարցական նշանը (?):

Հետևյալ օրինակում դիտարկված է միկրոկոնտրոլերի հրամանների վերծանիչը, որտեղ հրամանի ir կոդի ավագ բիթերը ընտրում են կատարվող հրամանը: Եթե ir-ի ավագ բիթը 1 է, ապա կատարվում է instruction1 առաջադրանքը, անկախ մյուս բիթերի արժեքներից (առաջադրանքների մասին կխոսվի ավելի ուշ):

```
reg [7:0] ir;
casez (ir)
    8'b1??????? : instruction1(ir);
    8'b01??????? : instruction2(ir);
    8'b00010??? : instruction3(ir);
    8'b000001?? : instruction4(ir);
endcase
```

Հաջորդ օրինակը ցուցադրում է **case** կառուցվածքի կիրառությունը վերջավոր ավտոմատի «1-տաք» կոդավորված վիճակների վերժաման համար՝ միայն մեկ բիթ է օգտագործվում վիճակը վերժամելու համար, մնացած բիթերն անտեսվում են:

```
reg [3:0] encoding;
integer state;
case (encoding) //տրամաբանական x արժեքը ներկայացնում է ոչ էական բիթ
    4'b1xxx : next_state = 3;
    4'bx1xx : next_state = 2;
    4'bxx1x : next_state = 1;
    4'bxxx1 : next_state = 0;
    default : next_state = 0;
endcase
```

Օրինակ, encoding = 4'b10xz կոդի դեպքում կկատարվի next\_state = 3 հրամանը:

**case** կառուցվածքում կարող են օգտագործվել հաստատուն արտահայտություններ: Հաստատուն արժեքը համեմատվում է այլընտրանքների հետ: Օրինակ, դիտարկենք 3-բիթ առաջնահերթության կոդավորիչ:

```
reg [2:0] encode ;
case (1)
    encode[2] : $display("Select Line 2") ;
    encode[1] : $display("Select Line 1") ;
    encode[0] : $display("Select Line 0") ;
    default: $display("Error: One of the bits expected ON");
endcase
```

Այս օրինակում **case**-ի արտահայտությունը հաստատուն է՝ 1, իսկ այլընտրանքները փոփոխականներ են՝ ընտրվող բիթերն են, որոնք համեմատվում են հաստատունի հետ:

### 7.7.5 Ծրագրի օղակներ

Վերիլոգում կա ծրագրում օղակներ կազմելու չորս կառուցվածք՝ **while**, **for**, **repeat** և **forever**: Դրանք հնարավորություն են տալիս կառավարել հրամանի կամ հրամանների բլոկի կատարման (կրկնման) քանակը՝ ոչ մի անգամ, մեկ անգամ, մեկից ավելի անգամ: Օղակող կառուցվածքները կարող են հայտնվել միայն **initial** կամ **always** բլոկների ներսում:

**repeat** կառուցվածքում հրամանները կատարվում են տրված թվով անգամ: Կառուցվածքի ընդհանուր տեսքը հետևյալն է.

```
repeat(expression) statement
    կամ
repeat(expression)
begin
    statement1
    statement2
    .....
end
```

statement հրամանը կամ **begin** և **end** բլոկում զետեղված հրամանները կատարվում են (expression) արտայայտությամբ որոշվող թվով անգամ: Հետևյալ օրինակը ցուցադրում է **repeat** օղակի օգտագործումը գումարում և տեղաշարժ գործողություններով բազմաթիվ թվերի բազմապատկում իրականացնելու համար:

```
parameter size = 8, longsize = 16;
reg [size:1] opa, opb; //արտադրիչներ՝ opa-բազմապատկելի, opb-բազմապատկիչ
reg [longsize:1] result; //այստեղ պետք է ստացվի արտադրյալը
begin: mult //սա անվանումով բլոկ է: Անվանումով բլոկները կքքնարկվեն ավելի ուշ
    reg [longsize:1] shift_opa, shift_opb; //անվանումով բլոկում կարելի հայտարարել
    // փոփոխականներ

    shift_opa = opa;
    shift_opb = opb;
    result = 0;
    repeat(size)
        begin
            if (shift_opb[1]) result = result + shift_opa; //ստուգվում է բազմապատկիչի
            //հերթական բիթը

            shift_opa = shift_opa <<1;
            shift_opb = shift_opb >>1
        end
    end
end
```

**while** օղակում հրամանները կատարվում են, քանի դեռ տրված արտահայտությունը ճշմարիտ է: **while** օղակի գրառման ձևը հետևյալն է՝

```
while(expression) statement
կամ
while(expression)
    begin
        statement1
        statement2
        .....
    end
```

Հետևյալ օրինակում **while** օղակում հաշվվում է rega-ում 1-երի քանակը: Դա կատարվում է տվյալի բայթի հաջորդական տեղաշարժերով և կրտսեր բիթի ստուգումով. եթե այդ բիթը 1 է, արդյունքի հաշվիչին գումարվում է 1: rega-ի տվյալը կրկնօրինակվում է tempreg ռեգիստրում, որպեսզի մեկերի թվի հաշվումից հետո rega-ի սկզբնական տվյալը չաղավաղվի:

```
reg [7:0] rega;
begin: count1
    reg [7:0] tempreg;
    count = 0;
    tempreg = rega;
    while (tempreg)
        begin
            if (tempreg[0]) count = count + 1;
```

```

    tempreg = tempreg >> 1;
end
end

for կառուցվածքում ընդգրկված հրամանների կատարման կառավարումն իրականացվում է երեք քայլով՝
    ա. օղակի կատարման քանակը պահպանող ռեգիստրին՝ փոփոխականին, վերագրվում է սկզբնական արժեքը (initial_assignment):
    բ. հաշվվում է տրված կառավարող արտահայտության (condition) արժեքը, եթե այն զրո է, հետո դուրս գալ օղակից, եթե զրոյից տարբեր է, ապա կատարվում են օղակում զետեղված հրամանները, այնուհետև կատարվում է անցում (գ) քայլին:
    գ. կատարվում է օղակի կատարման քանակը պահպանող փոփոխականի արժեքը փոխող վերագրում (step_assignment), այնուհետև կրկնվում է (բ) քայլը:
for կառուցվածքի գրառման ձևը հետևյալն է.
for(initial_assignment; condition; step_assignment)
    statement
կամ
for(initial_assignment; condition; step_assignment)
begin
    statement1
    statement1
    .....
end

for օղակի կիրառությունը տալիս է նույն արդյունքը ինչ while օղակի վրա հիմնված հետևյալ փսևդոկոդը.

begin
    initial_assignment;
    while(condition)
        begin
            statement
            step_assignment;
        end
    end

end

for օղակը այս կառուցվածքն իրականացնում է երկու տողով, այն դեպքում, երբ while օղակը՝ 8 տողով:
    Դիտարկենք for օղակի կիրառության օրինակ. հիշողության բջիջներին վերագրել 0 սկզբնական արժեք.
begin: init_mem
    reg [7:0] tempi;
    for (tempi = 0; tempi < memsize; tempi = tempi + 1)
        memory[tempi] = 0;
end

```

tempo ռեգիստրն օգտագործվում է որպես հիշողության բջիջների ընթացիկ թվի հաշվիչ: Հիշողության բջիջների ընդհանուր թիվը տրված է memsize փոփոխականով կամ պարամետրով:

Դիտարկենք **for** օղակի կիրառության ևս մեկ օրինակ. բազմապատկիչ, որը վեկում արդեն կառուցվել է **repeat** օղակի կիրառությամբ:

```
parameter size = 8, longsize = 16;
reg [size:1] opa, opb;
reg [longsize:1] result;
begin: mult
    integer bindex;
    result = 0;
    for (bindex = 1; bindex <= size; bindex = bindex + 1)
        if (opb[bindex])
            result = result + (opa << (bindex-1));
end
```

Նկատենք, որ **for** օղակում քայլի փոփոխության արտահայտությունը կարող է լինել ավելի ընդհանուր, քան թվաբանական պրոգրեսիայով: Ստորև բերվում է քայլի փոփոխության մեկ այլ օրինակ. քայլի փոփոխություն տեղաշարժի օպերատորի միջոցով, որն օգտագործվում է rega ռեգիստրում 1-երի թվի հաշվման համար:

```
begin: count1
    reg [7:0] tempreg;
    count = 0;
    for (tempreg = rega; tempreg; tempreg = tempreg >> 1)
        if (tempreg[0]) count = count + 1;
end
```

**forever** օղակը չի պարունակում որևէ պայման, նրանում ընդգրկված հրամանները կատարվում են անվերջ: **forever** օղակի գրառման ձևը հետևյալն է՝

```
forever statement
կամ
forever
    begin
        statement1
        statement2
        ....
    end
```

**forever** օղակի կիրառության օրինակ. տակտային իմպուլսների գեներատորի մոդելավորում.

```
forever #20 clk=~clk; //իմպուլսել clk ռեգիստրի արժեքը 20միավոր հապաղումով,
// տակտային իմպուլսների պարբերությունը 40 միավոր է:
```



### 7.7.6 Հաջորդական և զուգահեռ բլոկներ

Բլոկները հրամանների խմբեր են, որոնց կատարման կառավարումն իրականացվում է այնպես, ինչպես մեկ առանձին հրամանի դեպքում: Վերիլրգում գոյություն ունի բլոկների երկու տիպ.

- Հաջորդակա բլոկներ կամ **begin-end** բլոկներ,
- Ջուգահեռ բլոկներ կամ **fork-join** բլոկներ:

Հաջորդական բլոկները սահմանվում են **begin** և **end** բանալի բառերով: Հաջորդական բլոկում հրամանները կատարվում են հաջորդաբար՝ տրված հաջորդականությամբ: Յուրաքանչյուր հրամանի համար նշված հապաղումը հաշվվում է նախորդ հրամանի կատարման ավարտի պահից: Ծրագրի կառավարումը բլոկից դուրս է փոխանցվում բլոկի վերջին հրամանի կատարման ավարտից հետո: Հաջորդական բլոկների օրինակներ՝

**Օրինակ 7. 20.** Բլոկում հրամանների հաջորդականությունը միարժեք որոշում է սպասվող արդյունքը.

```
begin  
areg = breg;  
creg = areg; // creg-ում կստացվի breg-ի սկզբնական արժեքը  
end
```

Առաջին հրամանի կատարումից հետո areg-ը կթարմացվի (այնտեղ կփոխանցվի breg-ի պարունակությունը) մինչև երկրորդ հրամանին անցնելը:

**Օրինակ 7.21.** Այս օրինակում հրամանների հաջորդականությամբ և հապաղումների կառավարման միջոցով ձևավորվում է ժամանակից կախված հաջորդականություն:

```
parameter d = 50; // d-ն հայտարարված է որպես պարամետր  
reg [7:0] r; // r-ը հայտարարված է որպես 8-բիթ reg
```

```
begin // ժամանակից կախված հաջորդականություն  
#d r = 'h35;  
#d r = 'hE2;  
#d r = 'h00;  
#d r = 'hF7;  
#d -> end_wave; //շրջել end_wave անունով պատահարը  
end
```

Ջուգահեռ բլոկները սահմանվում են **fork** և **join** բանալի բառերով: Ջուգահեռ բլոկում հրամանները կատարվում են զուգահեռ՝ միաժամանակ: Յուրաքանչյուր հրամանի համար նշված հապաղումը հաշվվում է նմանակման ժամանակ՝ տվյալ բլոկ մտնելու պահից սկսած: Հապաղումների միջոցով կարելի է կառավարել վերագրումների հաջորդականությունը: Ծրագրի կառավարումը բլոկից դուրս է փոխանցվում բլոկի վերագրումների հաջորդականությամբ վերջին հրամանի կատարման ավարտից հետո:

Հետևյալ բլոկում կոդավորվում է նախորդ օրինակի ժամանակային հաջորդականությունը.

```

fork
#50 r = 'h35;
#100 r = 'hE2;
#150 r = 'h00;
#200 r = 'hF7;
#250 -> end_wave;
join

```

### 7.7.7. Անվանումով բլոկներ

Բլոկներին կարելի է անուններ տալ. անունը տրվում է **begin** բանալի բառից հետո: Դա հնարավորություն է տալիս.

- անվանումով բլոկի ներսում հայտարարել լոկալ փոփոխականներ: Դրանց կարելի է դիմել նախագծի հիերարխիայի տարաբեր մակարդակներից՝ օգտագործելով հիերարխիկ անվանումը:
- անվանումով բլոկին կարելի դիմել տարբեր կառուցվածքներում, մասնավորապես բլոկի աշխատանքը արգելող կառուցվածքում:  
Անվանումով բլոկի օրինակ.

```

module top;
initial
    begin: block1      //բլոկի անվանումը block1 է
        integer i;     //integer տիպի i փոփոխականը լոկալ է block1 բլոկի ներսում
                        // դրան կարելի է դիմել նաև դրսից օգտագործելով հիերարխիկ
// անունը՝ top.block1.i
        ...
    end

```

**disable** բանալի բառն օգտագործվում է անվանումով բլոկի կատարումը դադարեցնելու համար: Դա կարող է արվել օրինակ՝ ծրագրային օղակներից դուրս գալու, ծրագրի կարգաբերման ժամանակ սխալի առաջացման պայմանների պարզաբանման, ծրագրի որևէ հատվածի կատարումը կառավարելի դարձնելու համար: Որևէ բլոկի կատարումը **disable** բանալի բառով դադարեցնելուց հետո ծրագրի կառավարումն անիջապես անցնում է այդ բլոկին հաջորդող հրամանին: Հետևյալ օրինակում block2 բլոկի կատարումը դադարեցվում է այդ նույն բլոկի ներսից

```

begin : block2
    rega = regb;
    disable block2
    regc = rega; // այս հրամանը երբեք չի կատարվի
end

```

### 7.7.8. Պարամետրացված մոդելների գեներացում՝ generate կառուցվածք

**generate** կառուցվածքը թույլ է տալիս կառուցել պարամետրացված մոդելներ, որոնց կողմից գեներացվում է ծրագրի կոմպիլացման ժամանակ: **generate** բլոկը հայտարարվում է **generate** և **endgenerate** բանալի բառերով, որոնք սահմանում են բլոկի իրավասության սահմանները: Այդպիսի բլոկը չի կարող պարունակել մատույցների և

պարամետրերի հայտարարություններ: **generate** կառուցվածքի միջոցով կարելի է մոդելի կառուցվածքը կառավարել պարամետրերի միջոցով, ինչը չափազանց հարմար է, մանավանդ, կրկնվող մասերով մոդելների դեպքում: Մոդուլի օրինակները կարելի է տեղադրել ռեկուրսիվ եղանակով: Օրինակ, կարելի է միաբիթ գումարիչի վրա կառուցվող բազմաբիթ գումարիչի մոդելում բիթերի թիվը տալ պարամետրով:

Վերիլոգում գոյություն ունի **generate** կառուցվածքների երկու տիպ՝ օղակով և պայմանական հրամաններով: Օղակով կառուցվածքը թույլ է տալիս որևէ գեներացվող բլոկ տեղադրել մոդուլում տրված թվով անգամ: Պայմանական կառուցվածքը տեղադրում է միայն մեկ (կամ ոչ մի հատ) գեներացվող բլոկ հնարավոր տարբերակների բազմությունից:

**generate** կառուցվածքը արժևորվում է (կամ վերլուծվում է) կոմպիլացման ժամանակ՝ մինչև նմանակումը կամ սինթեզը: Արժևորումը ներառում է մոդուլի օրինակների տեղադրում, պարամետրերի արժեքների հաշվում, այդ օրինակների միջմիացումների իրականացում, հիերարխիկ անվանումների ստեղծում: Այս արժևորման արդյունքում ստացվում է սովորական նմանակվող Վերիլոգ մոդուլ, ուր բացակայում է **generate** կառուցվածքը:

**generate** կառուցվածքը թույլ է տալիս ստեղծել համապիտանի հակիրճ սկզբնական կոդ, որը որոշակի կիրառության դեպքում ընդարձակվում է ըստ պարամետրերի տրված արժեքների: **generate** կառուցվածքի նկարագրության բոլոր արտահայտությունները պետք է պարունակեն միայն հաստատուն մեծություններ, որոնց արժեքները հայտնի են ծրագրի կոմպիլացման փուլում:

Օղակով **generate** կառուցվածքը հնարավորություն է տալիս գեներացվող օրինակը տեղադրել մոդուլում՝ բազմաթիվ անգամ օգտվելով **for** օղակի հնարավորություններից: Այդ նպատակով **genvar** բանալի բառով հայտարարվում է **integer** տիպի ցիկլի փոփոխական, որը գործում է միայն կոմպիլացման ժամանակ, բայց այն բացակայում է նմանակման ժամանակ: **for** կառուցվածքում գեներացվող բլոկները կարող են լինել անվանումով կամ առանց անվանման: Այդ բլոկները պետք է ներառվեն **begin/end** բանալի բառերի միջև, եթե նույնիսկ մեկ հրաման են պարունակում: Եթե բլոկն անվանումով է, ապա դրա հայտարարությունը համարժեք է բլոկների զանգվածի հայտարարության, որում ինդեքսը ստանում է կոմպիլացման ժամանակ **genvar**-ի գեներացվող արժեքները: Այդ բլոկներում փոփոխականների անվանումները նույնպես կրում են այդ ինդեքսի արժեքները: Հետևյալ օրինակում դիտարկում է Գրեյի կոդը երկուականի կերպափոխիչի պարամետրացված մոդելը: Այստեղ օգտագործվում է **for** օղակ՝ հաստատուն վերագրումներ գեներացնելու համար:

```
module gray2bin1 (bin, gray);
parameter SIZE = 16; // սա պարամետրացված մոդուլ է
output [SIZE-1:0] bin;
input [SIZE-1:0] gray;
genvar i;           // i-ն generate կառուցվածքի ցիկլի ինդեքսն է
generate
    for (i=0; i<SIZE; i=i+1)
```

```

        begin:bit
            assign bin[i] = ^gray[SIZE-1:i];
        end
    endgenerate
endmodule

```

Ստորև բերված է այս մոդուլի թեստավորման մոդուլը, որտեղ gray2bin1 մոդուլի dut օրինակի տեղադրման ժամանակ տրվում է գեներացվող բլոկների անհրաժեշտ քանակը (SIZE պարամետրի միջոցով) տվյալ կիրառության համար: SIZE պարամետրի արժեքը gray2bin1 մոդուլի ներսում փոխարինվում է տրվող արժեքով՝ 8-ով:

```

module test_gray2bin1;
parameter WIDTH = 8;
reg [WIDTH-1:0] gray=8'h55;
wire [WIDTH-1:0] bin;

gray2bin1 #(8) dut(bin, gray); //տեղադրել gray2bin1 մոդուլի օրինակ
                                //տալով պարամետրի արժեքը

initial
begin
    $monitor("gray=%b, bin=%b",gray, bin); // հսկել gray, bin փոփոխականների
                                              // երկուական արժեքները

    gray=8'h55;
    #10 gray=8'haa;
    #10 gray=8'hb6;
    #10 $finish;    //ավարտել նմանակումը
end
endmodule

```

Նմանակումից ստացված արդյունքները.

```

gray=01010101, bin=01100110
gray=10101010, bin=11001100
gray=10110110, bin=11011011

```

Նկատենք, որ տարբեր Վերիլոգ կոմպիլատորներ **generate** կառուցվածք պարունակող կոդերի ընկալման համար պահանջում են կոմպիլացման տարբեր օպցիաներ: Օրինակ, Սինոսիսի VCS նմանակիչի դեպքում պետք է օգտագործել +v2k օպցիան:

**Օրինակ 7.22.** Բազմաբիթ երկուական գումարիչի պարամետրացված մոդուլ: Այստեղ օգտագործվում է երկչափ ցանցի հայտարարություն **generate** կառուցվածքից դուրս: Այդ ցանցի միջոցով իրականացվում են գեներացվող տարրերի միջմիացումները:

```

module addergen (co, sum, a, b, ci);
parameter SIZE = 4;
output [SIZE-1:0] sum;
output co;
input [SIZE-1:0] a, b;
input ci;
wire [SIZE :0] c;
wire [SIZE-1:0] t [1:3]; //երկչափ ցանց, t [1:3]-ի չափսը տրվում է միաբիթ գումարիչի
                        //կառուցվածքով, այն կախված չէ կառուցվող գումարիչի
                        //բիթերի թվից՝ SIZE պարամետրի արժեքից:

```

```

genvar i;
assign c[0] = ci;
generate
for(i=0; i<SIZE; i=i+1)
begin:bit // տարրերի հիերարխիկ անվանումները կլինեն ըստ i-ի արժեքի
xor g1 ( t[1][i], a[i], b[i]); // xor տարրեր: bit[0].g1 bit[1].g1 bit[2].g1 bit[3].g1
xor g2 ( sum[i], t[1][i], c[i]); // xor տարրեր: bit[0].g2 bit[1].g2 bit[2].g2 bit[3].g2
and g3 ( t[2][i], a[i], b[i]); // and տարրեր: bit[0].g3 bit[1].g3 bit[2].g3 bit[3].g3
and g4 ( t[3][i], t[1][i], c[i]); // and տարրեր: bit[0].g4 bit[1].g4 bit[2].g4 bit[3].g4
or g5 ( c[i+1], t[2][i], t[3][i]); // or տարրեր: bit[0].g5 bit[1].g5 bit[2].g5 bit[3].g5
//գեներացված տարրերը միացվում են t[1][ SIZE-1:0],
// t[2][ SIZE-1:0], t[3][ SIZE-1:0]
//ցանցերով՝ ընդամենը 3* SIZE ցանց
end
endgenerate
assign co = c[SIZE];
endmodule

```

addergen մոդուլով գեներացված քառաբիթ գումարիչի սխեման ցույց է տրված նկ. 7.24-ում, իսկ թեստավորման մոդուլը հետևյալն է.

```

module test_addergen;
parameter SIZE = 4;
wire [SIZE-1:0] sum;
wire co;
reg [SIZE-1:0] a, b;
reg ci;
adder gen dut (co, sum, a, b, ci);

initial
begin
    $monitor("ci=%h, a= %h, b=%h, co=%h, sum=%h", ci,a,b,co,sum);
    ci=1'b0; a=4'ha; b=4'h5;
    #10 ci=1'b1; a=4'ha; b=4'h5;
    #10 ci=1'b1; a=4'hf; b=4'hf;
    #10 $finish;
end
endmodule

```

Նմանակման արդյունքները.

```

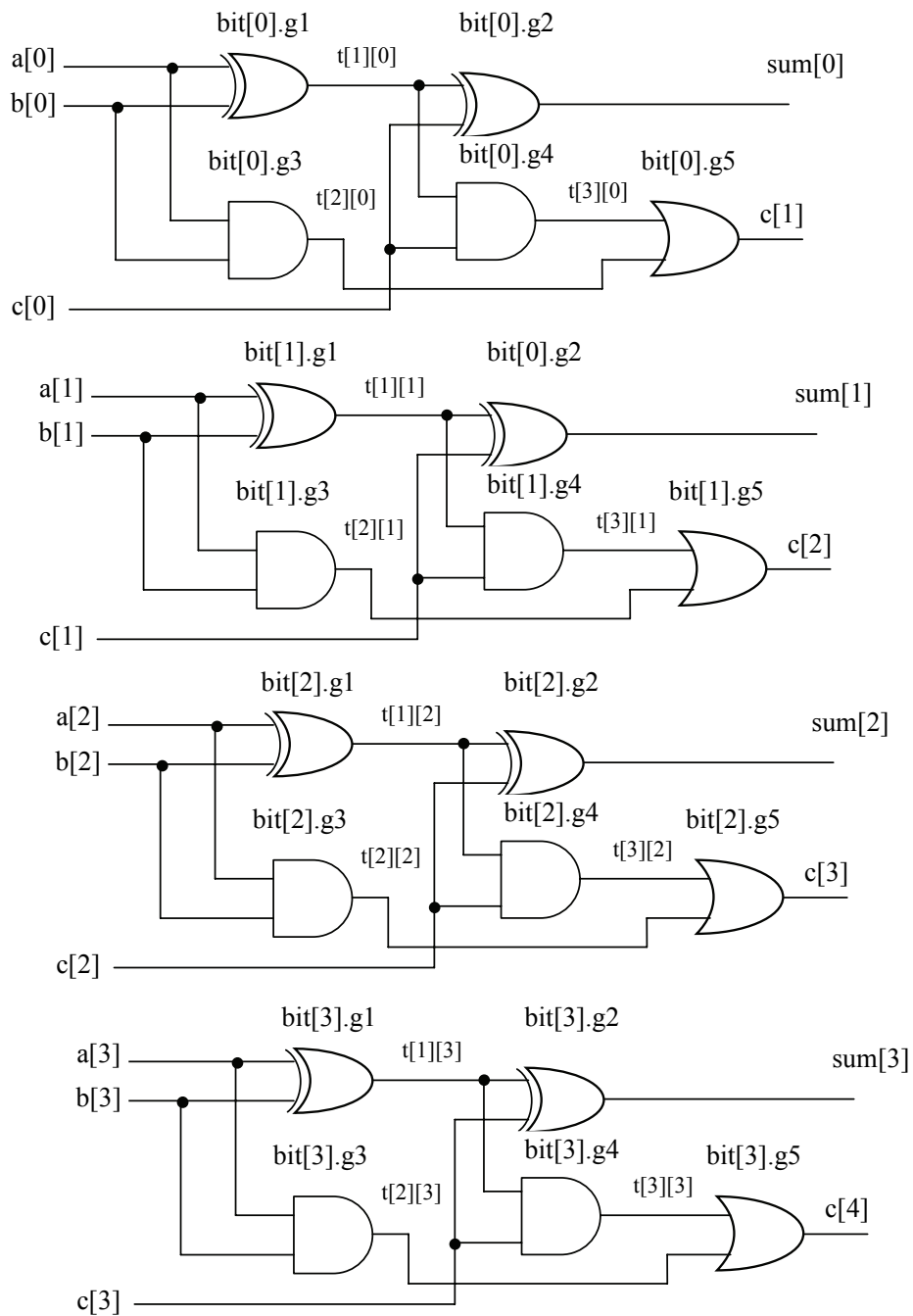
ci=0, a= a, b=5, co=0, sum=f
ci=1, a= a, b=5, co=1, sum=0
ci=1, a= f, b=f, co=1, sum=f

```

Պայմանական **generate** կառուցվածքը կարող է հիմնված լինել **if** և/կամ **case** հրամանների կիրառության վրա, այն ընտրում է մեկ գեներացվող բլոկ այլընտրանքային բլոկների բազմությունից: Այդ ընտրությունը կատարվում է ըստ կոմպիլացման ժամանակ հաշվվող հաստատուն արտահայտությունների: Ընտրված բլոկն էլ տեղադրվում է մոդելում:

Որպես օրինակ դիտարկենք պարամետրացված բազմապատկիչի մոդուլը, որտեղ տեղադրվող օրինակը գեներացվում է **if-else** հրամանով: Այս մոդուլում

օգտագործվում է `product_width` անունով լոկալ պարամետր, որի արժեքը որոշվում է `a_width` և `b_width` պարամետրերի արժեքներով: `a_width` և `b_width` պարամետրերով տրվում են արտադրիչների բիթերի թիվը, `product_width`-ով է արտադրյալինը: Լոկալ պարամետրի արժեքը արտաքինից հնարավոր չէ փոխել: Ընտրությունը կատարվում է CLA և WALLACE տիպի բազմապատկիչների միջև՝ ելնելով արտադրիչների բիթերի թվից:



Նկ. 7.24. Օրինակ 7.22-ի adder-ը մոդուլով գեներացված գումարիչի սխեման

CLA բազմապատկիչն ունի լավագույն ցուցանիշներ 8-ից փոքր կարգերի դեպքում, այն կառուցված է փոխանցման կանխագուշակմամբ գումարիչների վրա (carry-look-ahead adder): Մեծ կարգաթվերի դեպքում ավելի հարմար է Վալասի ծառի (Wallace tree) կառուցվածքով բազմապատկիչը: Արտաքինից տալով `a_width` և `b_width` պարամետրերի արժեքները՝ կարելի է կառուցել կամայական կարգայնությամբ բազմապատկիչ, որը կունենա լավագույն ցուցանիշներ: Համարվում է, որ `CLA_multiplier` և `WALLACE_multiplier` պարամետրացված մոդուլները մատչելի են:

```
module multiplier(a,b,product);
parameter a_width = 8, b_width = 8;
localparam product_width = a_width+b_width; // չի կարող փոխվել defparam հրամանով կամ
// մոդուլի օրինակի տեղադրման ժամանակ #-ով

input [a_width-1:0] a;
input [b_width-1:0] b;
output [product_width-1:0] product;
generate
    if((a_width < 8) || (b_width < 8))
        begin: mult
            CLA_multiplier #(a_width,b_width) u1(a, b, product); // տեղադրել CLA
            //բազմապատկիչ
        end

        else
            begin: mult
                WALLACE_multiplier #(a_width,b_width) u1(a, b, product); // տեղադրել
                // WALLACE բազմապատկիչ
            end
        endgenerate // տեղադրված բազմապատկիչի հիերարխիկ անունը mult.u1 է:
endmodule
```

Հետևյալ օրինակում ցույց է տրված `case` կառուցվածքի կիրառությունը **generate** կառուցվածքում: Այստեղ ընտրությունը կատարվում է երեք այլընտրանքների միջև:

```
generate
case (WIDTH)
1:    begin: adder // եթե WIDTH=1՝ ապա 1-բիթ գումարիչի իրագործում
        adder_1bit x1(co, sum, a, b, ci);
    end
2:    begin: adder // եթե WIDTH=2, ապա 2- բիթ գումարիչի իրագործում
        adder_2bit x1(co, sum, a, b, ci);
    end
default:
    begin: adder // մնացած դեպքերում իրագործվում է WIDTH-բիթ CLA գումարիչ
        adder_cla #(WIDTH) x1(co, sum, a, b, ci);
    end
endcase
endgenerate // գեներացված բլոկի հիերարխիկ անունը կլինի adder.x1
```

### 7.7.9. Վարքագծային մակարդակով մոդելավորման օրինակներ

Ստորև դիտարկվող օրինակներում ուսումնասիրվում են վարքագծային նկարագրության կառուցվածքների կիրառությունները: Օրինակները ավարտուն մոդուլներ են, որոնք կարելի է թեստավորել նմանակման միջոցով:

**Օրինակ 7.23.** Երթևեկության կարգավորման լուսակրի կառավարման սարք: Կառավարման խնդիրն է տրված հաջորդականությամբ և հապաղումներով փոխանջատել լուսակրի լույսերը: Հապաղումները ձևավորվում են՝ հաշվելով տակտային իմպուլսները:

```
module traffic_lights(clock, red, amber, green);
input clock;
output red, amber, green; //կարմիր, դեղին, կանաչ լույսերի միացման ելքեր
reg red, amber, green;
parameter
    on = 1,
    off = 0,
    red_tics = 350,    //կարմրի տևողությունը
    amber_tics = 30,  //դեղինի տևողությունը
    green_tics = 200; //կանաչի տևողությունը
// լույսերի կառավարման հաջորդականություն
always
    begin
        red = on;
        amber = off;
        green = off;
        repeat (red_tics) @(posedge clock);
        red = off;
        green = on;
        repeat (green_tics) @(posedge clock);
        green = off;
        amber = on;
        repeat (amber_tics) @(posedge clock);
    end
endmodule // traffic_lights

traffic_lights մոդուլի թեստավորման մոդուլը.
module test_traffic_lights;
reg clock;
wire red, amber, green;
traffic_lights i1(clock, red, amber, green); //տեղադրել traffic_lights մոդուլի օրինակը
// clock ազդանշանի ձևավորում
initial clock=1'b0;
always
    begin
```



```

        #100 clock = 0;
        #100 clock = 1;
    end
initial
    begin
        repeat (10) @ red; // երբ կարմիր լույսը 10 անգամ փոխանջատվել է,
        $finish;           //նմանակումն ավարտել
    end
// ցուցադրել ժամանակը և լույսերի փոխանջատումները
always @ ( red or amber or green )
    $display ("%d red=%b amber=%b green=%b", $time, red, amber, green);
endmodule // test_ traffic_lights

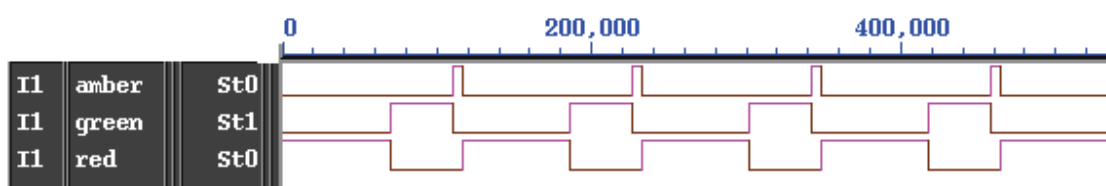
```

Նմանակման արդյունքները՝ գեներացված **\$display** ("%d red=%b amber=%b green=%b", \$time, red, amber, green) հրամանով.

```

0 red=1 amber=0 green=0 302000 red=0 amber=0 green=1
70000 red=0 amber=0 green=1 342000 red=0 amber=1 green=0
110000 red=0 amber=1 green=0 348000 red=1 amber=0 green=0
116000 red=1 amber=0 green=0 418000 red=0 amber=0 green=1
186000 red=0 amber=0 green=1 458000 red=0 amber=1 green=0
226000 red=0 amber=1 green=0 464000 red=1 amber=0 green=0
232000 red=1 amber=0 green=0

```



Նկ. 7.25. *traffic\_lights* մոդուլի թեստավորումից ստացված ժամանակային դիագրամները

**Օրինակ 7.24.** Այս օրինակում դիտարկվում է փոփոխական հապաղման կիրառությունը: Մոդուլում առկա է առաջնային տակտային ազդանշան՝ clock, որից ձևավորվում են մինյանցից 45° փուլով շեղված 2 սինքրոնացված ազդանշան՝ phase1 և phase2: Այդ ազդանշանների պարբերությունը հավասար է առաջնային clock ազդանշանի պարբերության կեսին: Նկատենք, որ phase1 և phase2 ազդանշանների ձևավորումն անկախ է նմանակման ժամանակից: 45° փուլային շեղումը տրվում է նմանակման ժամանակ հաշվվող d փոփոխականով, որի արժեքը վերահաշվվում է clock ազդանշանի յուրաքանչյուր պարբերությունը մեկ՝ դրական ճակատի պահին:

```

module synch_clocks;
    reg clock, phase1, phase2;
    time clock_time;
    initial clock_time = 0;
    always @ (posedge clock)
        begin :phase_gen //անվանումով բլոկ
            time d; //անվանումով բլոկում կարելի է հայտարարել փոփոխական

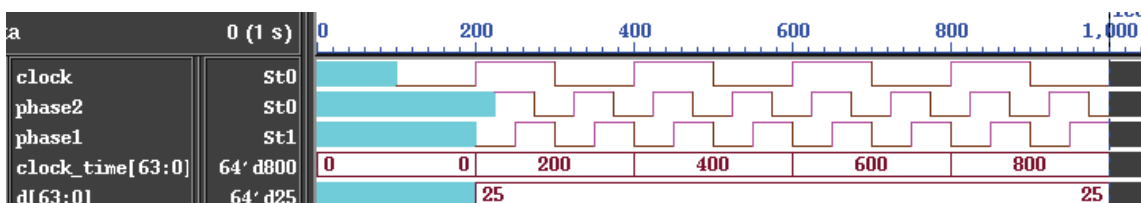
```

```

        d = ($time - clock_time) / 8; // այս հրամանը կատարվում է clock-ի
//յուրաքանչյուր պարբերությունը մեկ,
// d-արժեքը կլինի պարբերության 1/8-ը
        clock_time = $time; // clock_time փոփոխականին վերագրվում է
//նմանականման ընթացիկ ժամանակը
        phase1 = 0;
        #d phase2 = 1; //d-ն համապատասխանում է clock-ի պարբերության 1/8-ին
        #d phase1 = 1;
        #d phase2 = 0;
        #d phase1 = 0;
        #d phase2 = 1;
        #d phase1 = 1;
        #d phase2 = 0;
    end
// ձևավորել clock ազդանշանը
always
    begin
        #100 clock = 0;
        #100 clock = 1;
    end
initial #1000 $finish; // նմանակումն ավարտել 1000 պահին
always @ (phase1 or phase2) //ցուցադրել նմանական արդյունքները
    $display ($time, "clock = %b phase1 = %b phase2 = %b", clock, phase1, phase2);
endmodule

```

Նմանական արդյունքները ցուցադրված են նկ. 7.26–ում ժամանակային դիագրամների տեսքով:



Նկ. 7.26. *synch\_clocks* մոդուլի նմանական դիագրամները

Այս մոդուլը *synch\_clocks* կարելի է օգտագործել թվային համակարգերի մոդելների նմանական ժամանակ՝ որպես մուտքային տակտային ազդանշանների գեներատոր:

**Օրինակ 7.25.** Այս օրինակում ուսումնասիրվում է քառաբիթ սինքրոն հաշվիչ: Վերահաշվման գործակիցը կարելի է տալ 1...16 սահմաններում M պարամետրի միջոցով՝ վերահաշվման գործակիցը հավասար է M +1: Հաշվիչի վիճակը՝ q, պետք է լինի **reg** տիպի, որպեսզի հնարավոր լինի կատարել վարքագծային վերագրումներ: Որպեսզի out ելքը լինի ցանց տիպի, հաշվիչի վիճակը պետք է վերագրել ելքին անընդհատ վերագրման (**assign**) միջոցով: Նմանական տեսանկյունից դա ոչինչ չի փոխում, բայց էական է հիերարխիկ նախագծում միջմոդուլային կապերի տեսանկյունից:

```

module counter(clear, enable, clock, out);
parameter [3:0] M=4'd15;    // M=0...15, վերահաշվման գործակիցը՝ M+1
output [3:0] out;
input clear, enable, clock; //հաշվի գրոյացման, թույլտվության և տակտային ազդանշաններ
reg [3:0] q;    //հաշվիչի վիճակը հայտարարվում է ռեգիստր
always @( posedge clock or negedge clear )
begin
    if (!clear)
        q <= 4'd0; //ճակատով տակտավորվող տրիգերներ մոդելավորելու համար
    //պետք է օգտագործել չարգելափակող վերագրումներ else if (!enable)
        q <= q; //պահպանել վիճակը, եթե enable=0
    else if (q==M)
        q <= 4'b0; //հաշվիչը դնել սկզբնական վիճակ, եթե հասել է վերջին վիճակին
    else
        q <= q + 1; //հաշվել
end
assign out=q; //հաշվիչի ելքերին վերագրվում են տրիգերների վիճակները
endmodule

```

**Օրինակ 7.26.** Այս օրինակում ուսումնասիրվում է բազմաբիթ թվերի համակցական բաժանիչ: Բաժանելիի (divident) և բաժանարարի (divider), ինչպես նաև քանորդի (quotient) և մնացորդի (remainder) կարգաթիվը տրվում է «`**define** width 16» դիրեկտիվով: Այս ծրագրով կարելի է իրականացնել ցանկացած կարգաթիվ բաժանիչ, որի համար պետք է տալ կարգաթվի ու արժեքը «`**define** width ու» դիրեկտիվում: Բաժանման ալգորիթմը նկարագրված է 3.8 բաժնում:

```

`define width 16

module comb_divider(divbyzero, quotient, remainder, dividend, divider);
output divbyzero;
output [`width-1:0] quotient, remainder;
input [`width-1:0] dividend, divider;
parameter dwidth=2*`width;

reg divbyzero;    //գրոյի վրա բաժանման դրոշ
reg [`width-1:0] quotient_reg;
reg [dwidth-1:0] extended_divider, partial_remainder, partial_result; //բաժանարարի,
    //միջանկյալ մնացորդի և մասնակի արդյունքի 2width-բիթ ընդարձակված
    // ռեգիստրներ
wire [`width-1:0] quotient, remainder;

integer i;

always @(dividend or divider)
if (!divider) divbyzero=1;
else begin

```

```

divbyzero=0;
extended_divider = {1'b0, divider, {(`width-1){1'b0}}};
partial_remainder = {(`width){1'b0}}, dividend};
for(i = 0; i < `width; i = i + 1)
begin
    partial_result = partial_remainder - extended_divider;
    if(partial_result[dwidth-1 - i])
    begin
        quotient_reg[`width-1 - i] = 1'b0;
    end
    else begin
        quotient_reg[`width-1 - i] = 1'b1;
        partial_remainder = partial_result;
    end

    extended_divider = extended_divider >> 1;
end // for ghկլի վերջ
end // always բլոկի վերջ

assign remainder = partial_remainder[`width-1:0];
assign quotient = quotient_reg;

endmodule

```

Ստորև ցույց է տրված comb\_divider մոդուլի թեստավորման ծրագիրը, որով ստուգվում են 0–ից 255 թվերի բաժանումը 1–ից 255 թվերի վրա: Եթե որևէ բաժանման արդյունք սխալ է, ապա կարգվում է failure դրոշմ, տպվում է Failure! և բաժանելիի ու բաժանարարի արժեքները, որոնց դեպքում ստացվել է սխալ արդյունքը: Եթե բաժանելիի և բաժանարարի բոլոր արժեքների համար ստացվում են ճիշտ պատասխաններ, ապա ծրագրի կատարման ավարտին կտպվի Pass!

```

module test_comb_divider; // comb_divider մոդուլի թեստավորման մոդուլ
reg [`width-1:0] dividend, divider;
wire [`width-1:0] quotient, remainder;
wire divbyzero;
reg failure; // բաժանման սխալ արդյունքի դրոշմ
integer i,j;
comb_divider dut(divbyzero,quotient, remainder, dividend, divider);

initial
begin
    failure=0;
    for(i = 0; i < 256; i = i + 1)
    for(j = 1; j < 256; j = j + 1)
    begin
        dividend=i;
        divider=j;
    end
end

```

```

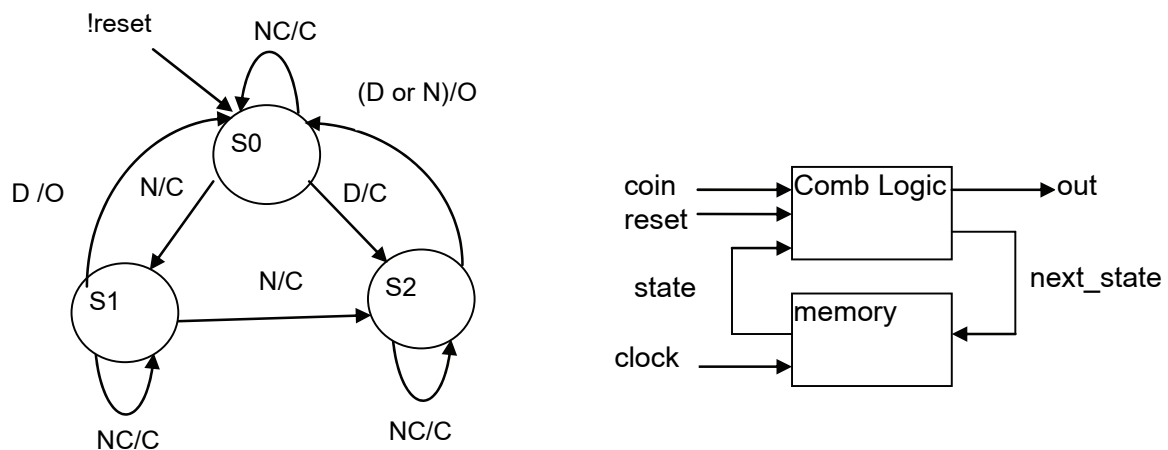
        if((quotient!=dividend/divider)||remainder!=dividend % divider)
        begin
            $display("Failure!: i= ", i, " j= ",j, " quotient= ",quotient, " remainder= ",remainder);
            failure=1;
        end
    end
    if(!failure) $display("Pass!");
#10 $finish;
end
endmodule

```

#### 7.7.10. Վերջավոր ավտոմատների Վերիլոգ կոդավորում

Որպես վերջավոր ավտոմատի Վերիլոգ կոդավորման օրինակ դիտարկենք 4-րդ բաժնում քննարկած վաճառող ավտոմատը: Ստորև բերվում են կոդավորման երեք տարբերակ՝ Միլի ավտոմատ համակցական ելքով, Միլի ավտոմատ սինքրոնացված ելքով, Մուրի ավտոմատ:

Համակցական ելքերով Միլի ավտոմատի կառուցվածքը ցույց է տրված նկ. 7.27-ում, որտեղ՝ coin-ը մետաղադրամի ներածման մուտքն է՝ N - 50 դրամ, D - 100 դրամ, NC-մետաղադրամ չի ներածվում, out-ը ելքն է՝ բաց O կամ փակ C արժեքներով: Անցումների գրաֆը ցույց է տրված նկ. 7.27-ում: Երբ մետաղադրամ չի ներածվում, ավտոմատը վիճակը չի փոխում: Ավտոմատը սկզբնական վիճակ է դրվում արտաքինից տրվող, ցածր ակտիվ մակարդակով reset ազդանշանով:



Նկ. 7.27. Վաճառող ավտոմատի անցումների գրաֆը՝ տրված Մուրի մոդելով և կառուցվածքը՝ համակցական ելքով տարբերակում

Վերջավոր ավտոմատների վարքագծի Վերիլոգ կոդավորումը հարմար է իրականացնել երկու **always** ընթացակարգերով՝ մեկում նկարագրել ավտոմատի համակցական մասը, որտեղ ձևավորվում է հաջորդ վիճակի փոփոխականը՝ next\_state, իսկ մյուսում՝ ներկա state վիճակի գրանցումը հիշողության ռեգիստր:

```

module vending_machine1(reset, clock, coin, out); //Միլի ավտոմատի մոդել
//պարամետրերի հայտարարում
parameter      N=2'b01, //մուտքի վիճակների կոդավորում
                  D=2'b10,
                  NC=2'b00,
                  O=1'b1,
                  C=1'b0;
parameter      S0=2'b00, //ավտոմատի ներքին վիճակների կոդավորում
                  S1=2'b01,
                  S2=2'b10;
//մուտք/ելք մատույցների հայտարարում
output out;
input reset, clock;
input [1:0] coin;
//ռեգիստր տիպի փոփոխականների հայտարարում
reg [1:0] state; //ներկա վիճակի, հետադարձ կապի ռեգիստրի ելքից
reg [1:0] next_state; //հաջորդ վիճակ, համակցական սխեմայի ելքից
reg out;
//հետադարձ կապի ռեգիստրի նկարագրություն
always @(posedge clock) //վիճակի գրանցում ռեգիստր՝ clock տակտային
state<=next_state; //ինպուլսի դրական ճակատով

//համակցական սխեմայի նկարագրություն (Comb Logic)
always @(reset, coin)
begin
    if(!reset) begin //եթե reset=0
        next_state=S0; //ավտոմատը դնել S0 վիճակ
        out=C; //ելքը փակել
    end

    else
    case(state) //հարմար է վիճակների փոփոխության տրամաբանությունը
        // նկարագրել case կառուցվածքով
    S0: if(coin==N) begin
        next_state=S1;
        out=C;
    end
    else if(coin==D) begin
        next_state=S2;
        out=C;
    end
    else next_state=S0;
    S1: if(coin==N) begin
        next_state=S2;
        out=C;
    end
    else if(coin==D) begin
        next_state=S0;
        out=O;
    end
    else next_state=S1;
    S2: if((coin==N) || (coin==D)) begin

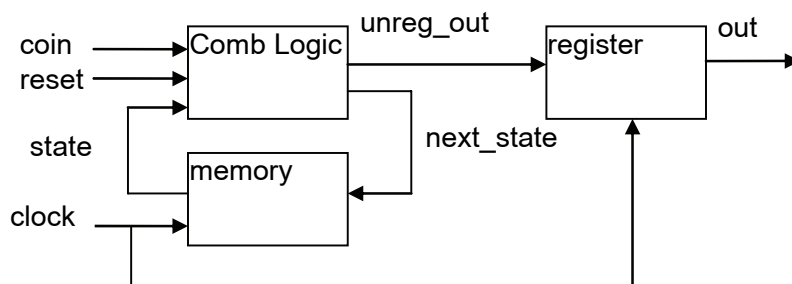
```

```

                                next_state=S0;
                                out=O;
                                end
    else        next_state=S2;
default: begin next_state=S0;//ըստ նախանական պայմանավորվածության
                out=C;
            end
        endcase
    end
endmodule

```

Սինթրոնացված ելքով Միլի ավտոմատի կառուցվածքը ցույց է տրված նկ. 7.28–ում: Ելքի ստորբացման համար օգտագործվում է տակտավորվող լրացուցիչ ռեգիստր՝ register: Համակցական սխեմայից ստացված ելքային ազդանշանը նշանակված է unreg\_out, ստորբացումից հետո՝ out:



Նկ. 7.28. Սինթրոնացված ելքով Միլի ավտոմատ

```

module vending_machine2(reset, clock, coin, out);// Սինթրոնացված ելքով Միլի
                                                // ավտոմատ

```

```

parameter      N=2'b01,
                D=2'b10,
                NC=2'b00,
                O=1'b1,
                C=1'b0;
parameter      S0=2'b00,
                S1=2'b01,
                S2=2'b10;

```

```

output out;
input reset, clock;
input [1:0] coin;
reg [1:0] state, next_state;
reg unreg_out, out; // unreg_out- համակցական ելք, out-ստորբացված ելք

```

```

always @(posedge clock)
begin
    state<=next_state;
    out<= unreg_out; // սինթրոնացնել ելքը
end

```

```

always @(reset, coin)

```

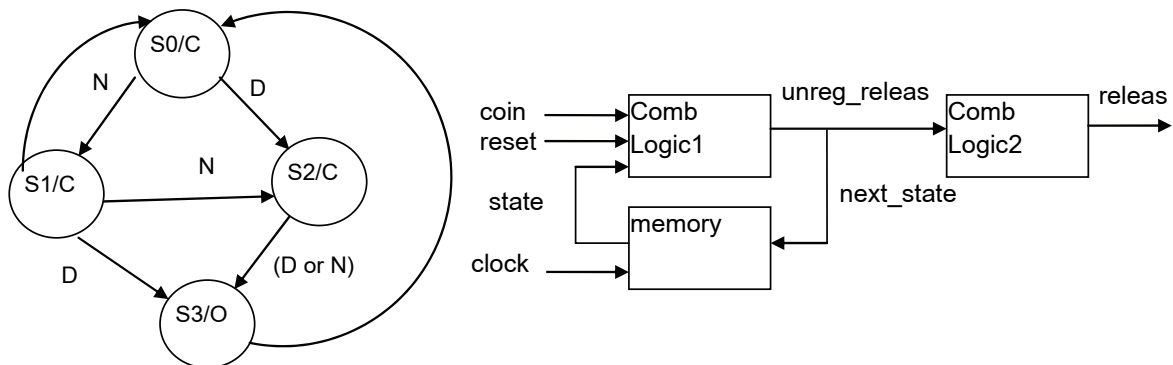
```

begin
  if(!reset)
    begin
      next_state=S0;
      unreg_out=C;
    end

  else
    case(state)
      S0: if(coin==N) begin
          next_state=S1;
          unreg_out=C;
        end
      else if(coin==D) begin
          next_state=S2;
          unreg_out=C;
        end
      else next_state=S0;
      S1: if(coin==N) begin
          next_state=S2;
          unreg_out=C;
        end
      else if(coin==D) begin
          next_state=S0;
          unreg_out=O;
        end
      else next_state=S1;
      S2: if((coin==N) || (coin==D)) begin
          next_state=S0;
          unreg_out=O;
        end
      else next_state=S2;
    default: begin next_state=S0;
                  unreg_out=C;
                end
    endcase
  end
end
endmodule

```

Մուրի ավտոմատի անցումների գրաֆը ցույց է տրված նկ. 7.29–ում:



Նկ. 7.29. Մուրի ավտոմատի անցումների գրաֆը և կառուցվածքային սխեման



Միլի ավտոմատի նկատմամբ ավելացված է ևս մեկ վիճակ՝ S3, որում հայտնվելիս out=O, այսինքն՝ բացվում է վաճառող ավտոմատի թողարկման պատուհանը: Մուրի ավտոմատի կառուցվածքային սխեման ցույց է տրված նկ. 7.29-ում:

```

module vending_machine3(reset, clock, coin, out); //Մուրի ավտոմատ
parameter      N=2'b01,
                  D=2'b10,
                  NC=2'b00,
                  O=1'b1,
                  C=1'b0;
parameter      S0=2'b00,
                  S1=2'b01,
                  S2=2'b10,
                  S3=2'b11;

output out;
input reset, clock;
input [1:0] coin;
reg [1:0] next_state, state;
reg out;

always @(posedge clock)
    state<=next_state;
always @(reset or coin or state)
begin
    if(!reset) next_state=S0;

else
case(state)
S0: if(coin==N)      next_state=S1;
    else if(coin==D) next_state=S2;
        else         next_state=S0;
S1: if(coin==N)      next_state=S2;
    else if(coin==D) next_state=S3;
        else         next_state=S1;
S2: if((coin==N) || (coin==D)) next_state=S3;
    else             next_state=S2;
S3:                  next_state=S0; // S3 վիճակից առանց որևէ պայմանի
                                //անցնել S0 վիճակ
default:           next_state=S0;
endcase
end

//ձևավորել ելքի ազդանշանը (Comb Logic2)
always @(state)
case(state)
    S0: out=C;
    S1: out=C;
    S2: out=C;
    S3: out=O;
endcase
endmodule

```

Ստորև բերված է վաճառող ավտոմատի Վերիլոգ մոդելի թեստավորման մոդուլը:

```

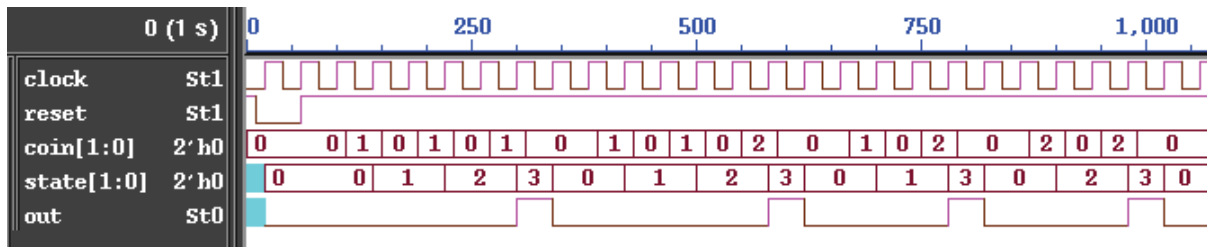
module test_vending_machine;
  parameter      N=2'b01,
                  D=2'b10,
                  NC=2'b00,
                  O=1'b1,
                  C=1'b0;

  reg [1:0] coin; //ռեգիստր տիպի պետք է հայտարարել բոլոր այն փոփոխականները,
  reg reset, clock; //որոնց արժեքներ են վերագրվում
  wire out;        //սա տեղադրվող մոդուլի օրինակի ելք է, պետք է լինի ցանց տիպի

  vending_machine3 dut(reset, clock, coin, out); //այստեղ տեղադրվում է Մուրի մոդելը:
                                                //կարելի է տեղադրել դիտարկված երեք մոդելներից ցանկացածը
  initial          //մնանակման համար մոտքային ազդանշանների ձևավորում
    begin
      $monitor($time, " reset=%b coin=%d state=%d out=%b \n", reset, coin, dut.state,
               out);
      clock=1'b0;
      coin=NC;
      reset=1'b1; //ձևավորվում է սկզբնական վիճակի կարգման 1-0-1
      #10 reset=1'b0; // reset հաջորդականություն
      #50 reset=1'b1;
      #50 coin=N; //մետաղադրամների ներածման հաջորդականություն
      #40 coin=NC;
      #40 coin=N;
      #40 coin=NC;
      #40 coin=N;
      #40 coin=NC;
      #40 coin=NC;
      #40 coin=N;
      #40 coin=NC;
      #40 coin=N;
      #40 coin=NC;
      #40 coin=D;
      #40 coin=NC;
      #40 coin=NC;
      #40 coin=N;
      #40 coin=NC;
      #40 coin=D;
      #40 coin=NC;
      #40 coin=NC;
      #40 coin=D;
      #40 coin=NC;
      #40 coin=NC;
      #40 coin=D;
      #40 coin=NC;
      #40 coin=NC;
      #40 coin=D;
      #40 coin=NC;
      #40 coin=NC;
      #40 $finish;
    end
  always #20 clock=~clock; //ձևավորել տակտային իմպուլսներ
endmodule

```

Նմանակման արդյունքները ցույց են տրված նկ. 7.30–ի ժամանակային դիագրամներով:



Նկ. 7.30. Վաճառող ավտոմատի նմանակման ժամանակային դիագրամները

Միկրոծրագրային ավտոմատի կոդավորումն ալգորիթմի գրաֆ-սխեմայից: Նկարագրենք երկուական թվերի բազմապատկիչը, որի ալգորիթմի գրաֆ-սխեման ցույց է տրված նկ. 6.10-ում: Բազմապատկումն սկսվում է start ինպուլսով, երբ արդյունքը պատրաստ է, կարգվում է ready դրոշմը:

```

module multiplier(start, clk, multiplier, multiplicand, product, ready);
parameter SIZE=4;
parameter    Y1=2'b00, //վիճակները
               Y2=2'b01,
               Y3=2'b10,
               Y4=2'b11;
input [SIZE-1:0] multiplier, multiplicand;
input start, clk;
output [2*SIZE-1:0] product;
output ready;          //բազմապատկման ավարտի ազդանշան
reg [SIZE-1:0] a, b, q; //արդյունքը ձևավորվում է a և q ռեգիստրներում
reg [SIZE-1:0] p;      //բիթերի հաշվիչ
reg e;                //գումարիչի բարձր կարգից անցման բիթը
reg ready;
reg [1:0] state, next_state; //նախորդ և հաջորդ վիճակի փոփոխականները

always @(posedge clk or posedge start) //վիճակի ռեգիստրի տակտավորում
if(start) state<=Y1;
else state<=next_state;

always @(start or a or b or q or state) //հաջորդ վիճակի և արտադրյալի հաշվում
if(start) next_state=Y1;
else
case(state)
    Y1: begin
        ready=1'b0;
        e=1'b0;
        a=0;
        q= multiplier;
        b= multiplicand;
        p= SIZE;
        next_state=Y2;
    end

```

```

Y2: begin
    p=p-1;
    if(!q[0]) next_state=Y4;
    else next_state=Y3;
end
Y3: begin
    {e, a}=a+b;
    next_state=Y4;
end
Y4: begin
    {e,a,q}={e,a,q}>>1;
    if(p) next_state=Y2;
    else ready=1'b1;
end
default: next_state=2'bx;
endcase
assign product={a,q}; //ստացված արդյունքը վերագրել product ելքին
endmodule

module test_multiplier;

parameter SIZE=4;
reg [SIZE-1:0] multiplier, multiplicand;
reg start, clk;
wire [2*SIZE-1:0] product;

multiplier    dut(start, clk, multiplier, multiplicand, product, ready);

initial
begin
    start =1'b0;
    clk=1'b0;
    multiplier=4'd12;
    multiplicand=4'd5;
    #50 start =1'b1;
    #50 start =1'b0;
    #200 $finish;
end
always #10 clk=~clk;
always @ (ready) //ցուցադրել նմանակման արդյունքները
    $display ("multiplier = %d multiplicand = %d product = %d", multiplier,
multiplicand,product);

endmodule

```

### 7.11. Խնդիրներ

**Խնդիր 7.17.** Հայտարարել clock անունով ռեգիստր, սկզբնական վիճակը կարգել 0: Այնուհետև ստիպել, որ այն փոխի վիճակը յուրաքանչյուր 20 նվ-ը մեկ՝ իմպուլսների պարբերությունը կլինի 40 նվ: (ա) Օգտագործել **always** հրամանը: (բ) օգտագործել **forever** օղակը:

**Խնդիր 7.18.** Մոդելավորել տակտային իմպուլսների գեներատոր՝ 120 նվ պարբերությամբ և 30% լցման գործակցով: Օգտագործել **always** հրամանը: Սկզբնական վիճակը կարգել 0 (**initial** հրամանով): Իմպուլսի և դադարի տևողությունները տալ պարամետրերով, որոնք հաշվվում են պարբերությունը սահմանող պարամետրերից:

**Խնդիր 7.19.** Որոշել ստորև նկարագրված **initial** բլոկում յուրաքանչյուր վերագրման կատարման ժամանակը և փոփոխականների միջանկյալ ու վերջնական արժեքները:

```
initial
begin
  a = 1'b0;
  b = #10 1'b1;
  c = #5 1'b0;
  d = #20 {a, b, c};
end
```

**Խնդիր 7.20.** Կրկնել նախորդ խնդիրը չարգելափակող վերագրումների համար:

**Խնդիր 7.21.** Մշակել տակտային իմպուլսի դրական ճակատով փոխանջատվող D տրիգերի մոդուլ: Տրիգերը պետք է ունենա 1 և 0 կարգելու ասինքրոն մուտքեր (set, clear): Մշակել տրիգերի աշխատանքը թեստավորող մոդուլ:

**Խնդիր 7.22.** Մշակել տակտային իմպուլսի բացասական ճակատով փոխանջատվող JK տրիգերի մոդուլ: Տրիգերը պետք է ունենա 1 և 0 կարգելու ասինքրոն մուտքեր: Մշակել տրիգերի աշխատանքը թեստավորող մոդուլ:

**Խնդիր 7.23.** Մշակել տակտային իմպուլսի մակարդակով փոխանջատվող D տրիգերի (սևեռիչի) մոդուլ:

**Խնդիր 7.24.** Մշակել միանիշ տասական թվերի գումարիչ՝ օգտագործելով վարքագծային հրամաններ:

**Խնդիր 7.25.** Նկարագրել (3.48) հավասարումներով և նկ.3.40-ով տրված Հենինգի (7,4) կոդի վերծանիչ:

**Խնդիր 7.26.** Մշակել թվաբանական-տրամաբանական սարքի (ԹՏՍ, ALU) Վերիլոգ մոդուլ՝ օգտագործելով **case** կառուցվածքը: ԹՏՍ-ն իրագործում է աղյուսակ 7.9-ում բերված գործողությունները առանց նշանի մուտքային 8-բիթ a և b փոփոխականների հետ: Ելքերը տրվում են 8-բիթ արդյունքով և carry (անցում բարձր կարգ) ու z (զրոյի հայտանիշ) դրոշմներով:

Գործողության կոդը	Գործողությունը	Մեկնաբանություն
0001	$a + b$	
0010	$a - b$	
0011	$a > b$ , համեմատել	(a-b), carry, z
0100	$a \& b$	
0101	$a   b$	
0110	$a \wedge b$	
0111	$a \gg b$	
1001	$a \ll b$	
1010	$a \ggg b$	
1011	$a \lll b$	

**Խնդիր 7.27.** Կազմել երկուական տվյալների հոսքից 010 կամ 110 եռաբիթ հաջորդականությունները հայտնաբերող ավտոմատի վարքագծային նկարագրությունը (տե՛ս օրինակ 4.6): Ավտոմատի անցումները նկարագրված են աղյուսակ 4.26-ում: Կազմել ավտոմատի աշխատանքն ստուգող թեստավորման մոդուլը:

**Խնդիր 7.28.** Կազմել  $M=10$  վերահաշվման գործակցով սինքրոն հաշվիչի վարքագծային նկարագրությունը: Հաշվիչը պետք է ունենա սկզբնական վիճակի ասինքրոն բեռնավորման հատկություն: Գրել հաշվիչի աշխատանքն ստուգող թեստավորման մոդուլը:

**Խնդիր 7.29.** Նկարագրել 8-բիթ թվերի համեմատման մոդուլ՝ զենեռացնելով միաբիթ թվերի համեմատման բլոկները ըստ (3.37) և (3.38) ռեկուրսիվ բանաձևերի:

## 7.8. Ֆունկցիաներ և առաջադրանքներ (task)

### 7.8.1. Վերլուգ առաջադրանքներ

Ֆունկցիաները և առաջադրանքները (task) Վերլուգում հնարավորություն են տալիս միևնույն ընթացակարգը կատարել ծրագրի տարբեր տեղերից: Դրանք նաև հնարավորություն են տալիս մեծ գործընթացները բաժանել ավելի փոքր մասերի, ինչը պարզեցնում է ծրագրի ընթեռնելիությունը և կարգաբերումը (ինչպես ենթածրագրերը): Մուտք/ելք արգումենտների միջոցով տվյալները կարելի է փոխանցել ընթացակարգին և այդտեղից հետ փոխանցել կանչող ծրագրին:

Առաջադրանքները և ֆունկցիաները տարբերվում են հետևյալ կանոններով.

- Ֆունկցիան պետք է կատարվի նմանակման մեկ ժամանակային միավորում, առաջադրանքը կարող է պարունակել ժամանակի կառավարման կառուցվածքներ:
- Ֆունկցիան կարող է կանչել ֆունկցիա, չի կարող կանչել առաջադրանք: Առաջադրանքը կարող է կանչել այլ առաջադրանքներ և ֆունկցիաներ:
- Ֆունկցիան պետք է ունենա ամենաքիչը մեկ մուտքի արգումենտ: Առաջադրանքը կարող է ունենալ ցանկացած թվով արգումենտներ կամ կարող է արգումենտ չունենալ:

- Ֆունկցիան վերադարձնում է մեկ արժեք: Առաջադրանքն արժեքներ չի վերադարձնում:

Ֆունկցիայի կիրառության նպատակն է պատասխանել մուտքային արժեքին՝ վերադարձնելով մեկ արժեք: Առաջադրանքը կարող է ստանալ բազմաթիվ մուտքային արժեքներ և հաշվել ելքի բազմաթիվ արժեքներ: Սակայն առաջադրանքից տվյալները կարող են հետ փոխանցվել կանչող ծրագրին միայն **output** կամ **inout** տիպի արգումենտների միջոցով: Ֆունկցիան կարող է լինել որևէ արտահայտության օպերանդ, այդ օպերանդի արժեքը արտահայտության մեջ ֆունկցիայով վերադարձվող արժեքն է:

Օրինակ, ենթադրենք նախագծում հաճախ անհրաժեշտ է 16-աբիթ տվյալի բառի բայթերի տեղերը փոխել: Դա կարելի է իրականացնել առաջադրանքի կամ ֆունկցիայի միջոցով և, այնուհետև, կանչել ֆունկցիան կամ առաջադրանքը ծրագրի ցանկացած կետից:

Առաջադրանքը հետ է փոխանցում փոխատեղված տվյալի բառը ելքի արգումենտով: Այս առաջադրանքի կանչը հիմնական ծրագրից ունի հետևյալ տեսքը՝

```
switch_bytes (old_word, new_word);
```

`switch_bytes` առաջադրանքը ստանում է `old_word` բառը մուտքի արգումենտով, փոխատեղում է բայթերը՝ շրջում է բառը, շրջված `new_word` բառը հետ է ուղարկում հիմնական ծրագիր ելքի արգումենտով:

Այդ նույն ֆունկցիոնալությունն ունեցող ֆունկցիան, բայց շրջված բառը հետ է փոխանցում որպես ֆունկցիայի վերադարձի արժեք: `switch_bytes` ֆունկցիայի կանչը կարող է ունենալ հետևյալ տեսքը՝

```
new_word = switch_bytes (old_word);
```

Առաջադրանքը կանչվում է, այսինքն՝ առաջադրանքի կատարումը թույլատրվում է, առաջադրանքի անունը կրող հրամանով, որը որոշում է արգումենտների արժեքները, որոնք պետք է փոխանցվեն առաջադրանքին և փոփոխականները, որտեղ պետք է ընդունվեն արդյունքները: Ծրագրի կատարման կառավարումը հետ է փոխանցվում կանչող ծրագրին առաջադրանքի ավարտից հետո: Ուստի, եթե առաջադրանքում կան ժամանակի կառավարման հրամաններ, ապա դա կփոխի առաջադրանքի ավարտի պահը: Առաջադրանքը կարող է կանչել այլ առաջադրանքներ: Անկախ այն բանից, թե քանի առաջադրանք է կանչվել, կառավարումը հետ չի փոխանցվում հիմնական ծրագրին մինչև բոլոր առաջադրանքների ավարտը:

Առաջադրանքի սահմանումը սկսվում է **task** բանալի բառով, որին կարող է հետևել ոչ պարտադիր **automatic** բանալի բառը, որին հաջորդում է առաջադրանքի անվանումը և կետ-ստորակետ: Առաջադրանքի սահմանումն ավարտվում է **endtask** բանալի բառով: **automatic** բանալի բառով հայտարարվում է վերականգնող առաջադրանք, որի բոլոր կանչերի տվյալները հիշողությունում պահվում են դիմամիկ կերպով: Առաջադրանքի սահմանումը կարող է պարունակել հետևյալ հայտարարությունները՝

— **input** արգումենտներ

— **output** արգումենտներ

— **inout** արգումենտներ

— ընթացակարգային բլոկում հայտարարվող բոլոր տիպի տվյալներ:

Այս հայտարարություններին հաջորդում է առաջադրանքի մարմինը՝ հրամանների հաջորդականությունը: Առաջադրանքն ավարտվում է **endtask** բանալի բառով:

Առանց **automatic** բանալի բառի առաջադրանքը ստատիկ է՝ բոլոր հայտարարված տվյալները հիշողությունում պահվում են ստատիկ կերպով: Այդ տվյալները կօգտագործվեն առաջադրանքի բոլոր միաժամանակ կատարվող կանչերում: **automatic** առաջադրանքի տվյալները հիշողությունում տեղաբաշխվում են դինամիկ կերպով՝ յուրաքանչյուր կանչի համար: Այն առաջադրանքները, որոնք վերականգնվող են, պետք է հայտարարվեն որպես **automatic**, որպեսզի առաջադրանքի միաժամանակյա կանչերը չաղավաղեն միմյանց տվյալները: **automatic** առաջադրանքը պետք է կանչվի իր հիերարխիկ անվանումով:

Առաջադրանքը կանչող հրամանը պետք է մուտք/ելք արգումենտները փոխանցի փակագծերում ստորակետով բաժանված արտահայտությունների ցուցակի տեսքով: Առաջադրանքի կանչի ձևաչափը հետևյալն է.

```
<name_of_task> ( <expression1>,<expression2>,...) ;
```

որտեղ **<name\_of\_task>** -ը առաջադրանքի անվանումն է, **<expression1>**, **<expression2>**, ... մուտք/ելք արգումենտներն են:

Եթե առաջադրանքի սահմանումը արգումենտներ չունի, այն կանչվում է մշտնջեն միայն անունը.

```
<name_of_task> ;
```

Արգումենտների ցուցակը պետք լինի նույն հաջորդման կարգով, ինչպես դրանք հայտարարվել են առաջադրանքի սահմանման մեջ:

Հետևյալ օրինակը ցուցադրում է առաջադրանքի սահմանման և կանչի կազմակերպումը.

```
task my_task;
```

```
input a, b;
```

```
inout c;
```

```
output d, e;
```

```
begin
```

```
// այստեղ տեղադրվում են առաջադրանքի աշխատանքն իրականացնող հրամանները
```

```
    c = foo1;                // ելքի արգումենտներին վերագրել
```

```
    d = foo2;                // առաջադրանքի աշխատանքի արդյունքները
```

```
    e = foo3;
```

```
end
```

```
endtask
```

Հետևյալ հրամանով կանչվում է **my\_task** առաջադրանքը՝

```
my_task (v, w, x, y, z);
```



Կանչի (v, w, x, y, z) արգումենտները համապատասխանում են (a, b, c, d, e) մուտք/ելք արգումենտներին, որոնք հայտարարվել են առաջադրանքի սահմանման մեջ: Առաջադրանքի կանչի ժամանակ input և inout տիպի (a, b, c) արգումենտները ստանում են v, w, x փոփոխականների արժեքները: Այսինքն՝ առաջադրանքի կանչը կատարում է հետևյալ վերագրումները.

```
a = v;
b = w;
c = x;
```

Առաջադրանքի կատարման արդյունքները տեղադրվում են output և inout տիպի c, d, e փոփոխականներում: Երբ առաջադրանքի կատարումն ավարտվում է, ծրագրի կառավարումը վերադարձվում է կանչող ծրագրին, որը և c, d, e փոփոխականներով ստացված արդյունքները վերագրում է կանչող ծրագրի x, y, z փոփոխականներին՝

```
x = c;
y = d;
z = e;
```

**Օրինակ 7.27.** Այս օրինակում դիտարկվում է առաջադրանքի կիրառությունը խաչմերուկի լուսակրի կառավարման ծրագրում, որն արդեն քննարկվել է օրինակ 7.23-ում:

```
module traffic_lights;
reg clock, red, amber, green; //տակտային իմպուլսների, կարմիր, դեղին և կանաչ գույների
//լույսերի վիճակների ռեգիստրներ
parameter on = 1, off = 0; // on-միացած վիճակ, off-անջատված վիճակ
parameter red_tics = 350, amber_tics = 30, green_tics = 200; //լույսերի տևողությունը
//սահմանող պարամետրեր
initial red = off; // տալ լույսերի սկզբնական վիճակները
initial amber = off;
initial green = off;
always
    begin // լույսերի կառավարման հաջորդականությունը.
        red = on; // միացնել կարմիր լույսը
        light(red, red_tics); //սպասել՝ կանչել light առաջադրանքը
        green = on; // միացնել կանաչ լույսը
        light(green, green_tics); // սպասել
        amber = on; // միացնել դեղին լույսը
        light(amber, amber_tics); // սպասել
    end
// light առաջադրանքի սահմանում: Այս առաջադրանքով ձևավորվում է ժամանակային
//հապաղում, որի տևողությունը հավասար է clock իմպուլսների tics թվով
//պարբերությունների: Այդ հապաղման ավարտին լույսն անջատվում է:
task light;
output color;
input [31:0] tics;
begin
    repeat (tics) //կրկնել tics անգամ
        @(posedge clock);
        color = off; // անջատել լույսը
    end
end
```

```
endtask
```

```
always                                // clock տակտային ազդանշանի ձևավորում
    begin
        #100 clock = 0;
        #100 clock = 1;
    end
endmodule // traffic_lights.
```

### 7.8.2. Վերիլոգ ֆունկցիաներ

Ֆունկցիան վերադարձնում է մեկ արժեք, որը կարող է օգտագործվել արտահայտություններում: Ֆունկցիան հայտարարվում է **function** և **endfunction** բանալի բառերով: Ֆունկցիան ունի ավելի սահմանափակ հնարավորություններ, քան առաջադրանքը: Ֆունկցիան պետք է բավարարի հետևյալ կանոններին.

1. Ֆունկցիան չի կարող պարունակել ժամանակի կառավարման որևէ հրաման: Այսինքն ֆունկցիայում չեն կարող օգտագործվել՝ **#**, **@**, կամ **wait** սիմվոլներ կամ կառուցվածքներ:
2. Ֆունկցիան չի կարող կանչել առաջադրանք, այն կարող է կանչել այլ ֆունկցիաներ:
3. Ֆունկցիան պետք է ունենա առնվազն մեկ մուտքի արգումենտ՝ **input** տիպի:
4. Ֆունկցիայի սահմանումը պետք է ներառի ֆունկցիայի արդյունքը ֆունկցիայի անունը կրող ներքին փոփոխականին վերագրման հրաման:

**Ֆունկցիայի սահմանման** ընդհանուր կառուցվածքը հետևյալն է.

```
function <range_or_type> <name_of_function> ;
<tf_declaration>    //փոփոխականների հայտարարում
<statement>         //ֆունկցիայի աշխատանքը որոշող հրամանները
endfunction
```

Այստեղ՝ **<range\_or\_type>** դաշտում տրվում է ֆունկցիայով վերադարձվող փոփոխականի բիթերի տիրույթը ([n:m] տեսքով) և տիպը, եթե այն **integer** կամ է **real** է: Եթե **<range\_or\_type>** դաշտը բացակայում է, ապա ֆունկցիայով վերադարձվող փոփոխականը 1-բիթ ռեգիստր է: **<name\_of\_function>** ֆունկցիայի անունն է՝ իդենտիֆիկատորը: Օրինակ,

```
function [7:0] adder ; // adder անվանումով 8-բիթ ռեգիստր տիպի:
```

**<tf\_declaration>** ընդգրկում է մուտքի և ներքին փոփոխականների, պատահարների և պարամետրերի հայտարարում

```
<parameter_declaration>;
<input_declaration> ;
<output_declaration> ;
<inout_declaration> ;
<reg_declaration>;
<time_declaration>;
<integer_declaration>;
```

```
<real_declaration>;
<event_declaration>;
```

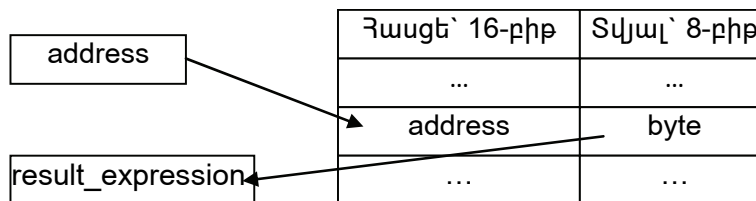
Այստեղ թվարկվածներից պարտադիր է միայն <input\_declaration>-ը:

Հետևյալ օրինակում սահմանվում է getbyte անվանումով ֆունկցիա.

```
function [7:0] getbyte;
input [15:0] address;
begin
<statements> // հրամաններ, որոնցով address հասցեից տվյալի
              // բայթը ընթերցվում է հիշողությունից (տե՛ս նկ. 7.31)
getbyte = result_expression; //արդյունքը վերագրվում է ֆունկցիայի
//անունը կրող փոփոխականին
end
endfunction
```

Ֆունկցիայի սահմանումը անբացահայտ կերպով հայտարարում է ներքին ռեգիստր, որն ունի նույն անվանումը, ինչ ֆունկցիան: Այս ռեգիստրի չափսը 1-բիթ է կամ տրվում է ֆունկցիայի հայտարարման <range\_or\_type> դաշտում: Ֆունկցիայի սահմանման մեջ պետք է լինի ֆունկցիայի աշխատանքի արդյունքը ֆունկցիայի անունը կրող ներքին փոփոխականին վերագրելու հրաման՝

```
getbyte = result_expression;
```



Նկ. 7.30. getbyte ֆունկցիայի <statements> հրամանների աշխատանքի մեկնաբանություն

**Ֆունկցիայի կանչն** իրականացվում է հետևյալ կառուցվածքով.

```
<name_of_function> ( <expression>,<expression>,... )
```

Ֆունկցիայի կանչը կարող է լինել որևէ արտահայտության օպերատոր: Հետևյալ օրինակում word արտահայտությունում getbyte ֆունկցիայի երկու կանչերից ձևավորվում է 16-բիթ բառ՝ բայթերի միակցումով.

```
word = control ? {getbyte(msbyte), getbyte(lsbyte)} : 0; // msbyte և lsbyte արդյունարար
//բառի ավագ և կրտսեր բայթերի հասցեներն են:
```

Ստորև դիտարկվում են ֆունկցիայի կիրառության օրինակներ:

**Օրինակ. 7.28.** 64-բիթ տվյալում զույգության հայտանիշի ստուգում ֆունկցիայի կիրառությամբ:

```
//Սահմանել մոդուլ, որը պարունակում է even_parity անվանումով ֆունկցիա
module check_parity;
```

```
...
```

```

reg [63:0] data;
reg parity;

//հաշվել 64-բիթ տվյալում զույգության հայտանիշը, երբ այդ տվյալը փոխվում է
always @(data)
    begin
        parity = even_parity(data); //կանչել even_parity ֆունկցիան
        $display("Parity status = %b", parity);
    end
...
...
//սահմանել even_parity ֆունկցիան
function even_parity; //ֆունկցիայի աշխատանքի արդյունքը վերադարձվում է even_parity
    // անբացահայտ հայտարարվող ռեգիստրով
input [63:0] data;
begin
    even_parity = ^ data; //կատարել սեղմման xor գործողություն data-ի
    //բոլոր բիթերի նկատմամբ
end
endfunction //ֆունկցիայի սահմանման ավարտ
...
...
endmodule // check_parity մոդուլի ավարտ

```

Ֆունկցիաները սովորաբար օգտագործվում են ոչ ռեկուրսիվ եղանակով: Եթե ֆունկցիան կանչվում է միաժամանակ երկու տեղից, արդյունքը կարող է լինել անկանխատեսելի, քանի որ երկու կանչերն էլ օգտագործում են փոփոխականների արժեքները նույն տարածքից: Այս խնդիրը կարելի է լուծել ֆունկցիայի սահմանման մեջ օգտագործելով **automatic** բանալի բառը՝ ինչպես առաջադրանքի սահմանման ժամանակ: **automatic** բանալի բառով հայտարարվում է ռեկուրսիվ (ինքն իրեն կանչող) ֆունկցիա, որի յուրաքանչյուր կանչի դեպքում նույն փոփոխականների համար տրամադրվում են առանձին տարածքներ: **automatic** ֆունկցիային հիերարխիկ դիմելու համար պետք է կանչի դեպքում օգտագործել նրա հիերարխիկ անվանումը:

**Օրինակ 7.29.** Այս օրինակում **automatic** ֆունկցիայով հաշվվում է տրված թվի ֆակտորիալը՝ ֆունկցիայի ռեկուրսիվ կանչերով: Ֆունկցիան արդյունքը վերադարձնում է **integer** տիպի փոփոխականով, որն անբացահայտ կրում է ֆունկցիայի անունը:

```

module tryfact; //այս մոդուլում օգտագործվում ֆակտորիալ հաշվելու ֆունկցիա
function automatic integer factorial; // սահմանել factorial անվանումով ֆունկցիան
input [31:0] operand;
integer i;
if (operand >= 2)
    factorial = factorial (operand - 1) * operand;
else
    factorial = 1;
endfunction

reg [31:0] result; / //ստուգել factorial ֆունկցիան

```

```

integer n;
initial
    begin
        for (n = 0; n <= 7; n = n+1) //կրկնել 0-ից 7 թվերի համար
            begin
                result = factorial(n);
                $display("%0d factorial=%0d", n, result);
            end
        end
    endmodule // tryfact

```

Նմանական արդյունքներն են.

```

0 factorial=1
1 factorial=1
2 factorial=2
3 factorial=6
4 factorial=24
5 factorial=120
6 factorial=720
7 factorial=5040

```

**Հաստատուն ֆունկցիան** Վերիլոգի սովորական ֆունկցիա է, բայց որոշակի սահմանափակումներով: Այս ֆունկցիաներն օգտագործվում են ծրագրում այն հաստատունների որոշման համար, որոնք պահանջում են բարդ հաշվարկներ: Այս ֆունկցիաները կատարվում են ծրագրի կոմպիլացման ժամանակ, և ոչ թե նմանական ընթացքում: Ֆունկցիայի հաշված արժեքն օգտագործվում է իբրև հաստատուն պարամետր Վերիլոգ նմանական ընթացքում: Հաստատուն ֆունկցիան կարող է կանչվել միայն այն մոդուլի ներսում, որտեղ այն սահմանվել է՝ հիերարխիկ կանչ չի թույլատրվում: Ֆունկցիայում կարելի է օգտագործել միայն այն պարամետրերի արժեքները, որոնք սահմանվել են մինչև ֆունկցիայի կանչը: Հետևյալ օրինակը սահմանում է clogb2 անունով հաստատուն ֆունկցիա, որը վերադարձնում է տրված հաստատունի երկու հիմքով լոգարիթմի՝ դեպի մոտակա ամբողջ թիվը կլորացրած արժեքը: Այդ ֆունկցիան օգտագործվում է հիշողության մոդելի մոդուլում հասցեային փոփոխականների անհրաժեշտ թիվը հաշվելու համար:

```

module ram_model (address, write, chip_select, data);
parameter data_width = 8;           //հիշողությունում բառի բիթերի թիվը
parameter ram_depth = 256;         //հիշողության բջիջների թիվը
localparam addr_width = clogb2(ram_depth); //հաստատուն ֆունկցիայով վերադարձվող
                                     //արժեքը համարվում է լոկալ պարամետր՝
                                     //հաստատուն

input [addr_width - 1:0] address;
input write, chip_select;           //հիշողության ԻՍ-ի կառավարման մուտքեր
inout [data_width - 1:0] data;      // հիշողության տվյալների մուտք/ելք դուղ

function integer clogb2;             //սահմանել clogb2 ֆունկցիան
input [31:0] value;
begin
    value = value - 1;

```

```

        for (clogb2 = 0; value > 0; clogb2 = clogb2 + 1)
            value = value >> 1;
    end
endfunction
reg [data_width - 1:0] data_store[0:ram_depth - 1]; //հիշողության զանգվածի հայտարարում
//հիշողության մոդուլի նկարագրության մնացած հրամանները
endmodule // ram_model

```

Մոդուլի կանչի ժամանակ կարելի է տալ մոդուլում որոշված պարամետրերի նոր արժեքները՝ փոխել մոդուլում տրված արժեքները: Բայց դա չի կարելի անել **localparam** բանալի բառով հայտարարված լոկալ պարամետրերի դեպքում: Հետևյալ հրամանով ստեղծվում է ram\_model մոդուլի օրինակ, որում #(32,421) կառուցվածքով փոխվում են մոդուլի մեջ սահմանված data\_width և ram\_depth պարամետրերի արժեքները համապատասխանաբար՝ 32 և 421 արժեքներով:

```

ram_model #(32,421) ram_a0(a_addr,a_wr,a_cs,a_data); // ram_model մոդուլի ram_a0
                //օրինակի տեղադրման ժամանակ տրվում են պարամետրերի նոր
                //արժեքները

```

**Օրինակ 7.30.** Կազմել Վերիլոգ մոդուլ՝ տվյալների զտման համար: Սահմանել ֆունկցիա՝ հետևյալ երկրորդ կարգի զտիչի իրականացման համար:

$y(n)=0.25x(n)+0.5x(n-1)+0.25x(n-2)$ :

Մուտքային x և ելքային y տվյալները 8-բիթ են:

```

module fir3(clk,x,y);
input clk;
input [7:0] x;
output [7:0] y;
reg [7:0] x1, x2;

always @(posedge clk)
begin
    x1<=x;           // x տվյալի հապաղում մեկ տակտով՝ x1=x(n-1)
    x2<=x1;          //x2=x(n-2)
end

assign y=fir3(x, x1, x2);

function [7:0] fir3;
input [7:0] a, b, c;
    assign fir3= (a >> 2) + (b >> 1) + (c >> 2) ;
endfunction

endmodule

```

### 7.8.3.Խնդիրներ

**Խնդիր 7.30.** Սահմանել ֆունկցիա, որը բազմապատկում է երկու քառաբիթ թվեր: Ելք-ը 8-բիթ թիվ է: Ստուգել ֆունկցիան՝ տալով մուտքի արժեքներ և կանչելով ֆունկցիան:

**Խնդիր 7.31.** Սահմանել ֆունկցիա, որով նկարագրվում է թվաբանական-տրամաբանական սարք, որում իրականացվող գործողությունն ընտրվում է 3-բիթ sel ազդանշանով (տե՛ս աղյուսակ 7.10-ը),  $a$  և  $b$  օպերանդները քառաբիթ են, արդյունքը հետ է փոխանցվում 5-բիթ փոփոխականով: Անտեսել գերլցման բիթերը:

*Աղյուսակ 7.10*

sel ազդանշանները	ֆունկցիայի ելքը
3'b000	$a+1$
3'b001	$a+b$
3'b010	$a-b$
3'b011	$a*b$
3'b100	$a\%b$ (մնացորդը)
3'b101	$a<<1$
3'b110	$a>>1$
3'b111	$a>b$ (համեմատել թվերը)

## Գլուխ 8. Թվային համակարգերի ավտոմատացված սինթեզ

### 8.1. Սինթեզի ընթացակարգը

Տրամաբանական սինթեզի ժամանակակից ծրագրային գործիքների համար որպես նախագծի ներմուծվող նկարագրությունն օգտագործվում է ՌՓՄ Վերիլոգ (կամ մեկ այլ ՍՆԼ, օրինակ, VHDL) կոդը: Սինթեզի արդյունքը տրվում է տրամաբանական տարրերի մակարդակով սխեմային միացումների ցուցակով: Թվային համակարգերի ավտոմատացված նախագծման ընթացքը ցույց է տրված նկ. 7.1–ում, որտեղ ներկայացված է նաև տրամաբանական սինթեզի փուլը:

Նախագծի սկզբնական նկարագրությունը ավտոմատացված նախագծման համար տրվում է Վերիլոգ կամ VHDL լեզվով: Նախագծի ներկայացման համար կա վերացարկման երեք մակարդակ՝ վարքագծային, ՌՓՄ և կառուցվածքային: Վարքագծային նկարագրությունը վերացարկման բարձրագույն մակարդակն է, որը ծառայում է՝ նախագծի ալգորիթմը կամ ճարտարապետությունը նմանակվող նկարագրության թարգմանելու համար: Նմանակման միջոցով ստուգվում է նախագծի ալգորիթմի և ճարտարապետության ճշմարտացիությունը և իրագործելիությունը: ՌՓՄ նկարագրությունն օգտագործվում է նախագծի ֆունցիոնալության և կառուցվածքի նկարագրության համար, որը ավտոմատացված սինթեզի գործիքի համար ծառայում է նախագծի մուտքային նկարագրություն: Սինթեզի ընթացքում կատարվում է ՌՓՄ կոդից անցում օպտիմալացված կառուցվածքային նկարագրության՝ տեխնոլոգիական գրադարանի տարրերի միացումների ցուցակի (netlist) տեսքով:

Նախագծման յուրաքանչյուր փուլում կատարվում է ստացված արդյունքի ֆունկցիոնալ համադրում մուտքային նկարագրության հետ (ռեգրեսիա, regression): Դա կատարվում է դինամիկ նմանակումների միջոցով՝ վարքագծային կոդի նմանակման համադրություն ալգորիթմի աշխատանքի և ճարտարապետության հետ, ՌՓՄ կոդի նմանակման արդյունքի համադրություն վարքագծայինի հետ, տարրերի միացումների Վերիլոգ ցուցակի նմանական համադրություն ՌՓՄ-ի հետ: Նախագծի նմանակումը կատարվում է թեստավորման Վերիլոգ մոդուլի միջոցով, որը գեներացնում է մուտքային ազդեցությունները և ցուցադրում կամ գրանցում է ստացված արդյունքները: Մուտքային ազդեցությունների վեկտորների ընտրությունը պետք է բավարար լինի նմանակման արդյունքների հիման վրա նախագծի ճշմարտացիության մեջ վստահ լինելու համար: Թեստավորման մոդուլը սովորաբար գրվում է վարքագծային մակարդակով, այն դեպքում, երբ ստուգվող նախագիծը կարող է լինել կոդավորված ցանկացած մակարդակով՝ վարքագծային, ՌՓՄ, կառուցվածքային տարրերի միացումների ցուցակ:

Ավտոմատացված սինթեզի գործիքներն ընձեռում են հետևյալ հնարավորությունները.

- Նախագծի նկարագրությունը վերացարկման բարձր մակարդակ՝ զգալի նվազեցնում է մարդու կողմից թույլ տրվող սխալների և վրիպումների հավանականությունը:
- Անցումը բարձր մակարդակի նկարագրությունից տրամաբանական տարրերի մակարդակով՝ սխեմային՝ միացումների ցուցակին, կատարվում է շատ արագ: Եթե



նախագծողին անհրաժեշտ է մտցնել փոփոխություններ սարքի աշխատանքում, ուրեմն պետք է միայն շտկել ՌՓՄ Վերիլոգ կողը և կրկնել սինթեզի փուլը:

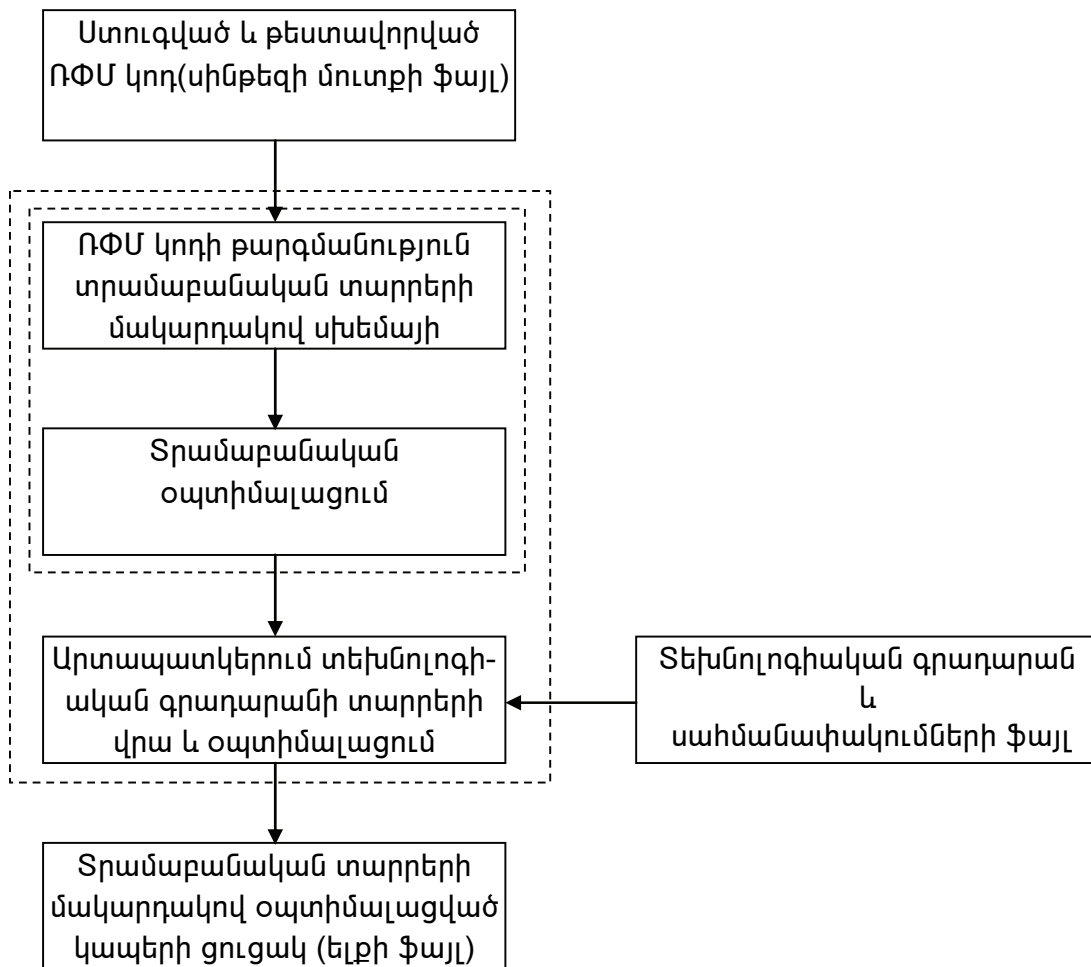
- Տրամաբանական սինթեզը կատարվում է արդյունարար սխեմայի իրականացման օպտիմալացմամբ ըստ տրված չափանիշի՝ արագագործություն, էներգասպառում, չափսեր:
- Նախագծի մուտքային ՌՓՄ Վերիլոգ նկարագրությունը կախված չէ սարքի իրականացման տեխնոլոգիայից: Այդ նկարագրությունը համապիտանի է և կարող է օգտագործվել բազմաթիվ անգամ: Յուրաքանչյուր կոնկրետ իրականացման դեպքում պետք է միայն փոխել օգտագործվող տարրերի տեխնոլոգիական գրադարանները:

Տրամաբանական սինթեզի փուլում ՌՓՄ նկարագրությունից ստեղծվում է նախագծվող համակարգի նկարագրությունը՝ տրված տեխնոլոգիայով կառուցվածքային տարրերի կապերի ցուցակի տեսքով, որը համարժեք է էլեկտրական սխեմային: Կառուցվածքային տարրերի տեխնոլոգիական գրադարանը կարող է տրվել կամ օգտագործվող ծրագրավորվող տրամաբանական սարքերի (հիմնականում ԿԾՓԶ, FPGA) տիպով, երբ համակարգն իրագործվում է ծրագրավորվող ԻՍ-երի վրա, կամ ստանդարտ թվային բջիջների գրադարանի տեսքով, երբ համակարգն իրագործվում է կիրառությանը յուրահատուկ ԻՍ-ի միջոցով:

Նախագծի ՌՓՄ Վերիլոգ կողը մշակելուց հետո այն պետք է մանրամասն ստուգվի Վերիլոգ նմանակումով, որպեսզի վստահություն լինի սինթեզվող սարքավորումով իրականացվող ալգորիթմի և նախագծին ներկայացված պահանջների համապատասխանության մեջ: Այդ գործընթացը կոչվում է ֆունկցիոնալ թեստավորում և ստուգում:

Նմանակումները պահանջում են երկար ժամանակ և զգալի ռեսուրսներ: Ավտոմատացված նախագծման ժամանակակից գործիքները թույլ են տալիս տարբեր մակարդակների նկարագրությունների համադրությունը կատարել ձևականացված եղանակով՝ առանց նմանակման: Նման հնրավորություն է ընձեռնում, օրինակ, Սինօֆսիսի Formality գործիքը: Այն կարող է համեմատել ՌՓՄ նկարագրությունը ՌՓՄ նկարագրության հետ, ՌՓՄ նկարագրությունը կառուցվածքային տարրերի միացումների ցուցակ հետ, միացումների մեկ ցուցակի համեմատություն մեկ այլ ցուցակի հետ: Համեմատման ժամանակ ստուգվում է տրամաբանական համարժեքությունը մաթեմատիկական մեթոդներով: Սակայն մշենք, որ ձևական համեմատությունը ստուգում է միայն նախագծի տարբեր նկարագրությունների տրամաբանական համարժեքությունը, բայց ոչ դրանց ժամանակային պարամետրերը: Ժամանակային սահմանափակումներին բավարարելու հատկության ստուգման համար պետք է օգտվել դինամիկ նմանակումից: Բայց այդ նմանակումն արդեն անհրաժեշտ կլինի կատարել մուտքային ազդեցությունների միայն այն վելտորների համար, որոնք համապատասխանում են ժամանակային սահմանափակումներին բավարարելու տեսանկյունից կրիտիկական ուղիներին, օրինակ՝ տրամաբանական տարրերի ամենաերկար շղթային:

Ավտոմատացված նախագծման տրամաբանական սինթեզի փուլն ավելի մանրամասնված է նկ. 8.1-ում: Տրամաբանական սինթեզի գործընթացն սկսվում է Վերիլոգ կոդը տրամաբանական տարրերի մակարդակով սխեմայի ձևափոխմամբ՝ թարգմանությամբ: Այս թարգմանությունը կատարվում է Վերիլոգ կոդի որոշակի կառուցվածքների մեկնաբանությամբ և դրանք համապատասխանության մեջ դնելով տրամաբանական տարրերի և տիպային թվային հանգույցների (ֆունկցիաների) հետ: Այս փուլում հաշվի չեն առնվում կառուցվող սարքին ներկայացվող արագագազործության, չափսերի և հզորության ծախսի սահմանափակումներն ու օգտագործվող տեխնոլոգիան:



Նկ. 8.1. Տրամաբանական սինթեզի ընթացակարգը

Վերիլոգ ՌԲՄ կոդի թարգմանությունից ստացվում է միջանկյալ՝ դեռևս ոչ օպտիմալ թվային շղթա՝ կառուցված ընդհանուր բնույթի (generic) տրամաբանական տարրերի և տիպային թվային հանգույցների վրա: Այդ շղթան, սովորաբար, ավելցուկային է: Այնուհետև, սինթեզի գործիքն անցնում է շղթայի տրամաբանական օպտիմալացմանը՝ համակցական շղթաների տարրերի թվի նվազարկմանը և վերջավոր ավտոմատների հիշողության նվազարկմանը: Արդյունքում ստացվում է օպտիմալացված

նախագիծ ընդհանուր՝ (generic) տիպային տարրերի վրա: Ստացված նախագիծը կախված չէ տեխնոլոգիայից: Հաջորդ քայլում սինթեզի գործիքը ընդհանուր (generic) տարրերը փոխարինում է տրված տեխնոլոգիական գրադարանի տարրերով և կատարում մակերեսի, հզորության և ժամանակային պարամետրերի օպտիմալացում ըստ տրված սահմանափակումների:

Սինթեզի համար մուտքեր են հանդիսանում համակարգի ՌՓՄ նկարագրությունը, կառուցվածքային տարրերի տեխնոլոգիական գրադարանը և նախագծման միջավայրի ու ժամանակային սահմանափակումների նկարագրությունը: Նախագծման միջավայրը տրվում է սինթեզի գործիքի համապատասխան օպցիաների ընտրությամբ: Ժամանակային սահմանափակումները նկարագրում են մուտքային ազդանշանների և տակտային ազդանշանի ժամանակային հարաբերակցությունները: Այդ սահմանափակումների ուղղագրությունը կարող է տարբեր լինել՝ կախված սինթեզի գործիքից: Ստորև բերված են ժամանակային սահմանափակումների օրինակներ՝ Սինոփսիսի DC սինթեզի գործիքի համար.

```
/* սահմանել 50% լցման գործակցով clk անունով տակտային ազդանշան 10 պարբերությամբ */
create_clock clk -period 10
/* տալ clk ազդանշանի պարբերության կամ ճակատի դիրքի անկայունությունը */
set_clock_uncertainty 0.2 clk
/* տալ մուտքային DATA_IN ազդանշանի ուշացումը clk ազդանշանի ճակատի նկատմամբ */
set_input_delay 20 -clock clk DATA_IN
/* տալ ելքային DATA_OUT ազդանշանի ուշացումը clk ազդանշանի ճակատի նկատմամբ */
set_output_delay 15 -clock clk DATA_OUT
```

Տեխնոլոգիական գրադարանում տրվում են կառուցվածքային տարրերի (տրամաբանական տարրերի և տիպային հանգույցների) տրամաբանական նկարագրությունը, ժամանակային պարամետրերը (հապաղումներ, տեղակայման և պահման ժամանակներ, ինպուլսի նվազագույն թույլատրելի տևողություն), սպառվող էներգիան, մուտքային մատույցների ունակությունները, ելքի ճյուղավորման թույլատրելի առավելագույն արժեքը և այլն: Տեխնոլոգիական գրադարանը տրվում է ստանդարտ դարձած Սինոփսիսի liberty չափաձևով (գրադարանի ֆայլի ընդարձակումը .lib է): Ստորև ցույց է տրված այդպիսի գրադարանի ֆայլի մեկ հատված.

```
cell(and2a3) {
    area : 6.7896;
    cell_leakage_power : 179.132 ;
    pin(A) {
        direction : input;
        capacitance : 0.001233;
    }
    pin(Y) {
        direction : output;
    }
    function : "B*A"; /* բջջով իրականացվող տրամաբանական ֆունկցիան */
    timing() {
        related_pin : "A"; /* ելքի ժամանակային պարամետրերը A մուտքի նկատմամբ */
        cell_rise( variable_1 : input_net_transition; /*A մուտքից ելք հապաղումը, երբ ելքը
աճում է*/
```

/\* հապաղումը ֆունկցիա է մուտքային ազդանշանի աճի տևողությունից և բջջի ելքը բեռնավորող ունակությունից, ֆՖ: Այդ ֆունկցիան տրվում է աղյուսակով, որի սյուների ինդեքսը բեռի ունակությունն է, իսկ տողերինը՝ մուտքային ազդանշանի աճի տևողությունը\*/

```
index_1("0.020, 0.033000,0.055800,0.103200"); /*տողերի ինդեքս, նվ*/
variable_2 : total_output_net_capacitance;
index_2("0.001580,0.004095,0.009488,0.020221"); /*սյուների ինդեքս, ֆՖ*/
values( " 0.102229, 0.114042, 0.135150, 0.170952",\
        " 0.106622, 0.118408, 0.139497, 0.175300",\
        " 0.115107, 0.126864, 0.147964, 0.183761",\
        " 0.132946, 0.144674, 0.165701, 0.201500");
rise_transition(..... /* ելքի աճի տևողությունը*/
cell_fall(...../*A մուտքից ելք հապաղումը, երբ ելքը նվազում է*/
fall_transition(.... /* ելքի անկման տևողությունը*/
}
timing() {
    related_pin : "B"; /* ելքի ժամանակային պարամետրերը B մուտքի նկատմամբ */
```

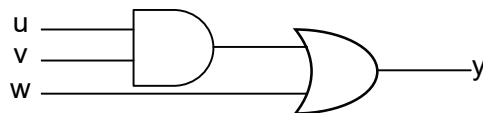
## 8.2. ՌՓՄ նկարագրության թարգմանությունը կառուցվածքային նկարագրության

Ստորև կքննարկենք, թե ինչպես սինթեզի գործիքները Վերիլոգ կառուցվածքները փոխակերպում են տրամաբանական տարրերի և տիպային թվային հանգույցների:

Շարունակական վերագրումը (**assign**) համակցական շղթաների ՌՓՄ նկարագրության հիմնական կառուցվածքն է: Եթե վերագրման արտահայտությունը պարունակում է բիթառօրին տրամաբանական օպերատորներ, ապա այն համարժեք է թվային շղթան նկարագրող տրամաբանական ֆունկցիայի բանաձևին: Սինթեզի գործիքը վերագրման արտահայտության օպերատորները փոխարինում է համապատասխան տրամաբանական տարրերով: Հետևյալ վերագրման արտահայտությունը՝

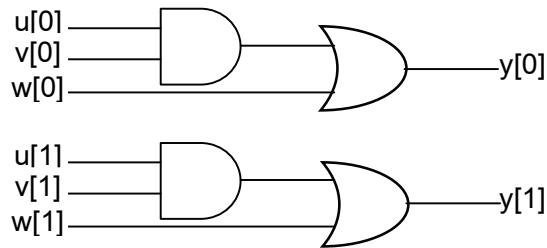
**assign**  $y = (u \& v) \mid w$ ;

կփոխակերպվի նկ. 8.3-ի տրամաբանական տարրերի մակարդակով սխեմայի:



Նկ. 8.3 **assign**  $y = (u \& v) \mid w$ ; արտահայտությունից սինթեզվող սխեման

Եթե  $u$ ,  $v$ ,  $w$  և  $y$ -ը 2-բիթ վեկտորներ են [1:0], ապա այդ նույն **assign** վերագրման արտահայտությունը կթարգմանվի վեկտորների բիթերին համապատասխանող երկու նույնատիպ սխեմայի (նկ. 8.4):

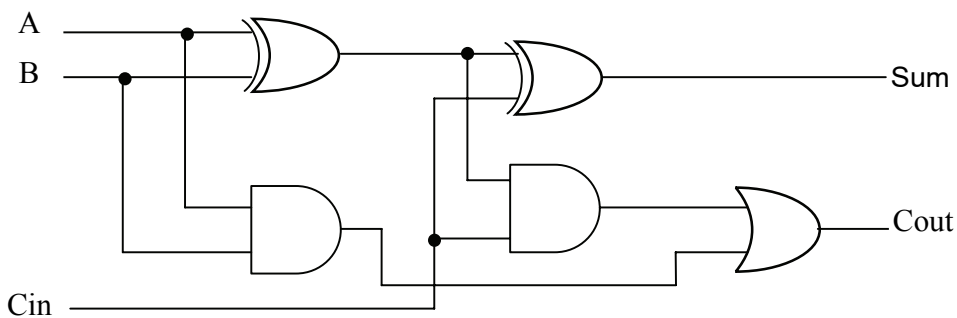


Նկ. 8.4. **assign**  $y = (u \& v) / w$ ; արտահայտությունից սինթեզվող սխեման, երբ  $u$ ,  $v$ ,  $w$  և  $y$ -ը 2-բիթ վեկտորներ են [1:0]

Եթե **assign** վերագրման արտահայտությունում օգտագործվում են թվաբանական օպերատորներ, ապա յուրաքանչյուր օպերատոր իրականացվում է համարժեք տրամաբանական սխեմայով, որն առկա է տրամաբանական սինթեզի գործիքում: 1-բիթ լրիվ գումարիչը տրվում է հետևյալ վերագրմամբ.

**assign** {Cout, Sum} = A + B + Cin;

Սովորաբար սինթեզի գործիքի կողմից դա ընկալվում է նկ. 8.5-ում ցույց տրված սխեմայի տեսքով:



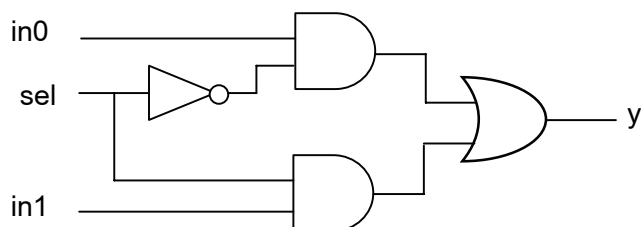
Նկ. 8.5. Միաբիթ երկուական գումարիչ՝ կառուցված **assign** {c\_out, sum} = a + b + c\_in; վերագրումից

Եթե սինթեզվում է բազմաբիթ գումարիչ, սինթեզի գործիքը այն կօպտիմալացնի, և արդյունքը կարող է ստացվել նկ. 7.20-ից տարբեր:

Պայմանական օպերատորի կիրառության դեպքում սինթեզի գործիքն այն ընկալում է իբրև մուլտիպլեքսոր.

**assign**  $y = (sel) ? in1 : in0$ ;

Այս կառուցվածքը կթարգմանվի նկ. 8.6-ի սխեմայի:



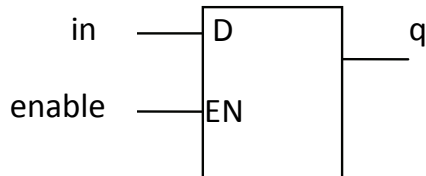
Նկ. 8.6. **assign**  $y = (sel) ? in1 : in0$ ; կառուցվածքին համապատասխանող սխեման

Վարքագծային կառուցվածքներին համապատասխանում են ինչպես համակցական, այնպես էլ հաջորդական գործողության շղթաներ:

**if** կառուցվածքը թարգմանվում է սևեռիչի (հաջորդական շղթայի): Օրինակ.

```
if(enable)    q = in;
```

կառուցվածքին համապատասխանում է enable թույլտվության մուտքով և in տվյալի մուտքով D սևեռիչ (նկ. 8.7):

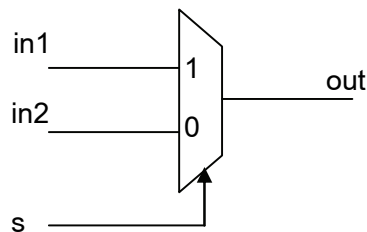


Նկ. 8.7. **if(enable) q = in;** կառուցվածքին համապատասխանող սևեռիչի շղթա

**if-else** կառուցվածքը թարգմանվում է մուլտիպլեքսորի: Օրինակ,

```
if(s)
    out = in1;
else
    out = in0;
```

կառուցվածքը սինթեզի գործիքը թարգմանում է նկ. 8.8-ում ցույց տրված 2:1 մուլտիպլեքսորի:



Նկ. 8.8. **if-else** կառուցվածքի թարգմանությունը մուլտիպլեքսորի

Վերիլոգ **case** կառուցվածքը

```
case (s)
    1'b0 : out = in0;
    1'b1 : out = in1;
endcase
```

նույնպես սինթեզի ժամանակ բերվում է նկ. 8.8-ի մուլտիպլեքսորի:

ո փոփոխականներով զգայունության ցուցակով և  $2^n$  տողերով **case** կառուցվածքին համապատասխանում է  $2^n$  ինֆորմացիոն և ո ընտրության մուտքերով մուլտիպլեքսոր: Օրինակ, հետևյալ կոդով կառուցվում է 8:1 մուլտիպլեքսոր.

```
case({s0, s1, s2})
    3'b000: out=in0;
    3'b001: out=in1;
```

```
....
3'b111: out=in7;
endcase
```

**for** օղակն օգտագործվում է կասկադացված համակցական սխեմաներ կառուցելու համար: Օրինակ, հետևյալ կոդով կառուցվում է 8-բիթ գումարիչ.

```
c = cin;
for(i=0; i <=7; i = i + 1)
    {c, sum[i]} = a[i] + b[i] + c; // կառուցվում է 8-բիթ գումարիչ
cout = c;
```

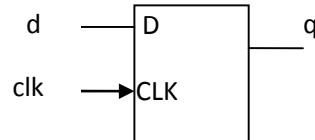
**always** կառուցվածքը կարող է օգտագործվել ինչպես համակցական, այնպես էլ հաջորդական տրամաբանական շղթաների սինթեզի համար: Հետևյալ կոդով գեներացվում է համակցական շղթա.

```
always @(a or b or c or d)
begin
    out1 = (a & b) | c;
    out2 = (a & b) | (c & d);
end
```

Տակտային իմպուլսի ճակատով տակտավորվող հաջորդական շղթա ստանալու համար անհրաժեշտ է **always** կառուցվածքը կառավարել տակտային clk ազդանշանի փոփոխությամբ.

```
always @(posedge clk)
    q <= d;
```

Սինթեզի գործիքն այս կառուցվածքը թարգմանում է clk ազդանշանի դրական ճակատով տակտավորվող D տրիգերի (նկ. 8.9):



Նկ. 8.9. **always @(posedge clk)**  $q <= d$ ; կառուցվածքի թարգմանությունը D տրիգերի

Եթե **always** կառուցվածքը կառավարվի տակտային ազդանշանի մակարդակով, ապա այն կթարգմանվի սևեռիչի.

```
always @(clk or d)
if (clk)
    q <= d;
```

Մեկ ելքով համակցական շղթաներ կառուցելու համար հարմար է օգտագործել Վերիլոգ ֆունկցիաներ: Ելքը կարող է լինել սկալյար կամ վեկտոր: Հետևյալ ֆունկցիան կառուցում է 8-բիթ գումարիչ.

```
function [8:0] fulladd;
input [7:0] a, b;
input cin;
begin
    fulladd = a + b + cin;
```

end  
endfunction

Վերիլոգ ՌՓՄ կոդի թարգմանությունից ստացվում է միջանկյալ՝ չօպտիմալացված թվային շղթա՝ կառուցված տրամաբանական տարրերի և տիպային թվային հանգույցների վրա: Այդ շղթան, սովորաբար, ավելցուկային է: Այնուհետև, սինթեզի գործիքն անցնում է շղթայի տրամաբանական օպտիմալացմանը՝ համակցական շղթաների նվազարկմանը և վերջավոր ավտոմատների հիշողության նվազարկմանը: Արդյունքում ստացվում է օպտիմալացված նախագիծ ընդհանուր (generic) տիպային տարրերի վրա: Ստացված նախագիծն ընդհանուր է՝ կախված չէ տեխնոլոգիայից: Հաջորդ քայլում սինթեզի գործիքն ընդհանուր տարրերը փոխարինում է տրված տեխնոլոգիական գրադարանի տարրերով և կատարում մակերեսի, հզորության և ժամանակային պարամետրերի օպտիմալացում ըստ տրված սահմանափակումների:

### 8.3. Սինթեզվող Վերիլոգ կոդին ներկայացվող սահմանափակումները

Սինթեզի գործիքը ՌՓՄ նկարագրությունը վերածում է տրամաբանական տարրի մակարդակով օպտիմալացված սխեմայի կապերի ցուցակի: ՌՓՄ նկարագրությունը բաղկացած է տվյալների հոսքի և վարքագծային նկարագրության կառուցվածքներից: Սակայն պետք է հաշվի առնել, որ ոչ բոլոր վարքագծային կառուցվածքները կարող են օգտագործվել տրամաբանական սինթեզում: Սովորաբար, տրամաբանական սինթեզի համար Վերիլոգ կոդը պետք է գրված լինի տակտից տակտ կատարվող վարքագծային բլոկներով, որոնք իրենց աշխատանքով ինտուիտիվ նման են իրական թվային համակարգերի աշխատանքին: Սահմանափակումները կարող են կախված լինել նաև, ինչ-որ չափով, օգտագործվող սինթեզի գործիքից: Սակայն այդ տարբերությունները սովորաբար չնչին են: Աղյուսակ 8.1-ում բերված Վերիլոգ կառուցվածքները ընդունելի են սինթեզի բոլոր հայտնի գործիքների կողմից:

Աղյուսակ 8.1

Կառուցվածքի տիպը	Նկարագրությունը (բանալի բառը)	Մեկնաբանություն
մատույցներ	<b>input, inout, output</b>	
պարամետրեր	<b>parameter</b>	
մոդուլի սահմանում	<b>module</b>	
ազդանշաններ և փոփոխականներ	<b>wire, reg, tri</b>	թույլատրվում է վեկտորներ
օրինակի տեղադրում	<b>module, primitive</b>	օրինակ, mux21 i1(out, i0, i1, s); nand (out, a, b);
ֆունկցիաներ և առաջադրանքներ	<b>function, task</b>	Ժամանակի կառավարման կառուցվածքներն անտեսվում են
ընթացակարգեր	<b>always, if, then, else, case, casex, casez</b>	<b>initial</b> չի թույլատրվում
ընթացակարգային բլոկներ	<b>begin, end, named blocks, disable</b>	թույլատրվում է անունով բլոկների արգելում
տվյալների հոսքուղի	<b>assign</b>	հապաղումներն անտեսվում են
օղակներ	<b>for, while, forever,</b>	<b>while</b> և <b>forever</b> օղակները պետք է պարունակեն <b>@(posedge clk)</b> կամ <b>@(negedge clk)</b>



Քանի որ սինթեզի համար թվային համակարգի նկարագրությունը պետք կառուցվի տակտից տակտ կատարվող վարքագծային բլոկներով, ուստի վարքագծային բլոկներ կառուցելիս պետք հետևել որոշակի սահմանափակումների: Օրինակ, **while** և **forever** օղակները պետք է ընդհատվեն **@ (posedge clock)** կամ **@(negedge clock)** կառուցվածքներով, որպեսզի պահպանվի տակտից տակտ վարքագիծը և կանխվեն համակցական հետադարձ կապերը: Մյուս կարևոր սահմանափակումն այն է, որ սինթեզի ժամանակ անտեսվում են **#<delay>** կառուցվածքով տրված հապաղումները: Հետևաբար, նախասինթեզի և հետսինթեզի Վերիլոգ նմանակումները կարող են տարբերվել: Նախագծողը պետք է օգտագործի Վերիլոգ նկարագրության այնպիսի ոճ, որը կբացառի այդպիսի տարբերությունները: Կարևոր է նշել նաև, որ սինթեզվող կոդում պետք է չլինեն **initial** կառուցվածքներ: Նախագծողը պետք է սինթեզվող Վերիլոգ կոդում օգտագործի շղթաների նախնական վիճակի կարգման (**reset**) մեխանիզմներ: Խորհուրդ է տրվում ազդանշանների վեկտորներում բացահայտ նշել բիթերի թիվը: Առանց չափսի վեկտորների դեպքում սինթեզված շղթան կարող է ունենալ ավելցուկություն:

Վերիլոգի համարյա բոլոր օպերատորները, որոնք թվարկված են աղյուսակ 7.5-ում, թույլատրվում են տրամաբանական սինթեզում: Չեն թույլատրվում միայն **===** և **!=** օպերատորները, որոնք առնչվում են ազդանշանների **x** և **z** արժեքների հետ և որոնք սինթեզում որևէ իմաստ չունեն: Արտահայտությունները գրելիս խորհուրդ է տրվում օգտագործել փակագծեր, որպեսզի սինթեզից ստացվի ակնկալվող կառուցվածքը: Եթե սինթեզի գործիքը շղթան կառուցի՝ հիմնվելով օպերատորների համար ընդունված առաջնահերթությունների վրա, ապա հնարավոր է սինթեզից ստանալ անցանկալի կառուցվածք:

ՌԺՄ կոդի գրման ոճը կարևոր ազդեցություն ունի սինթեզի արդյունքի օպտիմալության վրա: Օպտիմալ սինթեզված համակարգ ունենալու համար պետք է հետևել ՌԺՄ կոդավորման օրինակելի ոճին: Խորհուրդ տրվող կոդավորման ոճը ինչ-որ չափով կախված է, նաև, սինթեզի գործիքի առանձնահատկություններից: Այստեղ կքննարկվեն միայն այնպիսի դրույթներ, որոնք ընդհանուր են՝ արդյունավետ սինթեզվող ՌԺՄ կոդ գրելու համար:

Սինթեզի ընթացքի կառավարման համար մտցվել է Վերիլոգ լեզվի նոր կառուցվածք՝ **attribute**, որով սինթեզի գործիքին հայտնվում է Վերիլոգ կոդից սխեմային անցնելու ցանկալի եղանակը: **attribute** կառուցվածքն ունի հետևյալ ուղղագրությունը՝ (\*ատրիբուտի անվանումը և պարամետրը\*): Օրինակ, բազմաթիվ երկուական գումարիչ կառուցելու դեպքում

```
a = b + (* mode = "cla" *) c;
```

նշելով (\* mode = "cla" \*) ատրիբուտը, հայտնվում է, որ գումարիչը պետք է կառուցվի փոխանցման կանխագուշակմամբ:

Սինթեզի և նախագծի ընթացքի արդյունավետ կառավարման համար խորհուրդ է տրվում առաջնորդվել Վերիլոգ կոդավորման հետևյալ սկզբունքներով:

Օգտագործել իմաստ ունեցող անվանումներ (իդենտիֆիկատորներ): Դա կողը դարձնում է հեշտ ընթերցվող և նվազեցնում է մեկնաբանությունների անհրաժեշտությունը: Օգտագործել վերջածանցներ՝ իդենտիֆիկատորներն ավելի իմաստալից դարձնելու համար: Օրինակ, `_clk` վերջածանց տակտային ազդանշանների համար, `_next` վերջածանց վիճակի փոփոխականների համար՝ նախքան ռեգիստրում տակտավորումը, `_n` ակտիվ ցածր մակարդակով ազդանշանների համար, `_z` եռավիճակ ելքին միացված ազդանշանների համար:

Խուսափել նույն նախագծում տակտային ազդանշանի դրական և բացասական ճակատներով տակտավորումից: Դա կրթերի մեծ թվով շրջիչների օգտագործման տակտային ազդանշանի տարածման ծառու: Միայն մեկ ճակատով տակտավորման դեպքում համակցական շղթաներով ազդանշանների տարածման համար տրամադրվում տակտի մեկ լրիվ պարբերություն:

Մուլտիպլեքսորների կառուցվածքի հանգելու համար օգտագործել **if-else** և **case** լրիվ նկարագրված կառուցվածքներ: Թերի կառուցվածքները կարող են հանգել սևեռիչների կառուցվածքների: Եթե **case** կառուցվածքում կան ոչ էական պայմաններ, ապա մուլտիպլեքսոր ստանալու համար նկարագրությանն ավելացնել **default** պայմանը:

Համակցական տրամաբանության կառուցվածքի օպտիմալացման համար կողում օգտագործել փակագծեր: Փակագծերը թույլ են տալիս ուղղորդել սինթեզը և, միաժամանակ, կողը դարձնել ընթերցելի: Օրինակ,

```
out = a + b + c + d;
```

այս կողը թարգմանվում է երեք հաջորդաբար միացված գումարիչների: Այն դեպքում, երբ

```
out = (a + b) + (c + d) ;
```

կողը թարգմանվում է երկու զուգահեռ միացված գումարիչների և մեկ վերջնական գումարիչի, որը գումարում է զուգահեռ գումարիչներից ստացված արդյունքները: Այս կառուցվածքն ունի նախորդից ավելի փոքր հապաղում:

Խուսափել միևնույն փոփոխականին վերագրումներ կատարել տարբեր բլոկներում, դա կարող է հանգեցնել անցանկալի սխեմատիկ իրականացման: Օրինակ, հետևյալ երկու բլոկներում միևնույն *q* փոփոխականին վերագրումներ են կատարվում երկու տարբեր **always** բլոկներում.

```
always @(posedge clk)
```

```
    if(load1) q <= a1;
```

```
always @(posedge clk)
```

```
    if(load2) q <= a2;
```

Այստեղ հնարավոր է, որ երկրորդ վերագրումը գրվի առաջինի վրա, որը համարժեք է առաջինի չլինելուն: Սինթեզի գործիքը կարող է նաև այդ կառուցվածքները թարգմանել երկու տրիգերների և դրանց ելքերը միացնի ԵՎ տարրով կամ ուղղակի այդ տրիգերների ելքերը իրար միացնի, որպեսզի ստացվի *q* ելքը: Նախագծողը պետք

է զգույշ լինի նման իրավիճակներից խուսափելու համար:

Տակտային իմպուլսի ճակատով գործող տրիգերի հանգեցնելու համար օգտագործել

```
always @(posedge clk)
    q <= d;
```

կառուցվածքը, իսկ իմպուլսի մակարդակով գործող տրիգերի սկեռիչի հանգեցնելու համար՝

```
always @(clk, d)
    q <= d;
```

կառուցվածքը:

Վերջավոր ավտոմատների կողավորման համար վիճակի ռեգիստրի փոփոխությունը և համակցական շղթաները նկարագրել տարբեր **always** բլոկներում.

```
always @(posedge clk)
state<=next_state //վիճակի ռեգիստրի տակտավորում
```

```
always @(inputs or state)
//համակցական շղթայի նկարագրություն
```

Հետևյալ օպերատորները՝ **+**, **\***, **/**, **%** (մնացորդ) սինթեզի ժամանակ հանգում են մեծաթիվ տարրերով համակցական շղթայի, որը դժվար է օպտիմալացվում: Այդպիսի դեպքերում կարելի է օգտվել գոյություն ունեցող օպտիմալացված շղթաներից, որոնք առկա են սինթետիկ գրադարաններում: Օրինակ, Սինոփսիսի DC գործիքով սինթեզի ժամանակ կարելի է նշել այդպիսի բաղադրիչը կոմպիլատորի դիրեկտիվի միջոցով: Հետևյալ օրինակում 32-բիթ գումարիչի համար օգտագործվում է Սինոփսիսի DW01\_add սինթետիկ մոդուլը՝ փոխանցման կանխագուշակմամբ 32-բիթ գումարիչ:

```
module add32 (a, b, cin, sum, cout);
input [31:0] a, b;
input cin;
output [31:0] sum;
output cout;
reg [33:0] temp;
always @(a or b or cin)
begin : add1
//Սինթետիկ մոդուլի ընդգրկման դիրեկտիվ՝ այն ընդգրկվում է /*..*/ կառուցվածքում և պետք է
//սկսվի synopsys բառով: Ուշադրություն դարձրեք, որ Վերիլոգ կոմպիլատորի համար սա
//մեկնաբանություն է, իսկ սինթեզի DC գործիքն այն ընկալում է որպես դիրեկտիվ
/* synopsys resource r0:
   ops = "A1",
   map_to_module = "DW01_add",
   implementation = "cla"; */
temp = ({1'b0, a, cin} + // synopsys label A1
{1'b0, b, 1'b1});
end
```

```
assign {cout, sum} = temp[33:1];
endmodule
```

Նախագծի ԿԾՓԶ-ով իրականացման դեպքում կարելի է օգտվել ԿԾՓԶ-ում առկա մեգաֆունկցիաներից՝ օպտիմալացված ավարտուն տիպային թվային հանգույցներից:

#### 8.4. ՌՓՄ-ից տրամաբանական տարրերի կապերի ցուցակի անցման օրինակ

Որպես օրինակ դիտարկենք քառաբիթ սինքրոն հաշվիչի սինթեզը տրված տեխնոլոգիական գրադարանի տարրերի վրա: Հաշվիչի սինթեզվող ՌՓՄ նկարագրությունը բերված է ստորև.

```
module counter(clear, enable, clock, Q);
output [3:0] Q;
input clear, enable, clock;
reg [3:0] Q; //Ելքը սահմանված է ռեգիստր տիպի
always @(posedge clock or negedge clear )
begin
    if (!clear)
        Q <= 4'd0; // հաջորդական շղթաների դեպքում հարմար է օգտագործել
                //չարգելափակող վերագրումներ
    else if(enable)
        Q <= Q + 1; // վերահաշվման գործակիցը 16 է, քանի որ չի ստուգվում վիճակը
                //տրված արժեքին հասնելը: 4-բիթ վիճակն ավտոմատ կերպով 15-ից
                // անցնում է 0-ի:
    else
        Q <= Q;
end
endmodule
```

Այս ՌՓՄ կոդի ստուգման համար օգտագործվում է հետևյալ թեստավորող մոդուլը.

```
module test_counter;
reg clear, enable, clock;
wire [3:0] Q;
counter dut(clear, enable, clock, Q);
initial
begin
    $monitor($time, " clear=%b enable=%b clock=%b Q=%h \n", clear, enable,
clock, Q);
    clock=1'b0;
    enable=1'b0;
    clear=1'b1;
    #10 clear=1'b0;
    #25 clear=1'b1;
    #20 enable=1'b1;
    #100 enable=1'b0;
    #100 enable=1'b1;
```

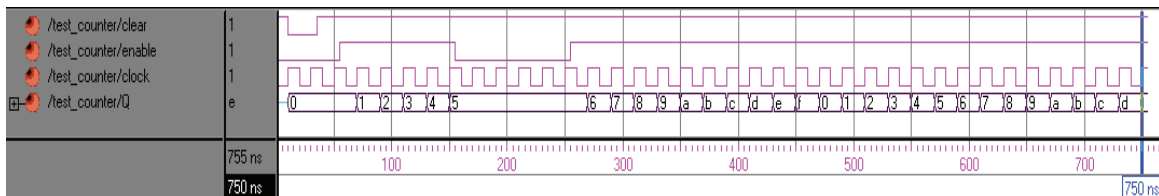
```

        #500 $finish;
    end

    always #10 clock=~clock;    //տակտային իմպուլսների գեներացում
endmodule

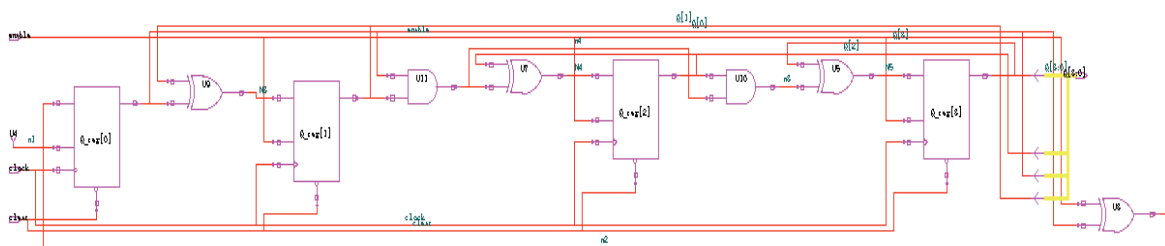
```

ՌԲՄ կոդի նմանակումից ստացված ժամանակային դիագրամները ցույց են տրված նկ. 8.10–ում:



Նկ. 8.10. ՌԲՄ կոդի նմանակումից ստացված ժամանակային դիագրամները

Սինթոսիսի DC գործիքով counter մոդուլի սինթեզից ստացվում է նկ.8.11 –ում ցույց տրված թվային շղթան՝ կառուցված տրված տեխնոլոգիական գրադարանի տարրերի վրա:



Նկ. 8.11. Սինթոսիսի DC գործիքով counter մոդուլի սինթեզից ստացված սխեման

DC-ից ստացված տրամաբանական տարրերի մակարդակով Վերիլոգ նկարագրումը՝ կապերի ցուցակն (netlist) ունի հետևյալ տեսքը.

```

module counter ( Q, clock, clear, enable );
    output [3:0] Q;
    input clock, clear, enable;
    wire N3, N4, N5, n2, n3, n4;
    fdef2a1 Q_reg0 ( .D(n2), .E(1'b1), .CLK(clock), .CLR(clear), .Q(Q[0]) );
    fdef2a1 Q_reg1 ( .D(N3), .E(enable), .CLK(clock), .CLR(clear), .Q(Q[1]) );
    fdef2a1 Q_reg2 ( .D(N4), .E(enable), .CLK(clock), .CLR(clear), .Q(Q[2]) );
    fdef2a1 Q_reg3 ( .D(N5), .E(enable), .CLK(clock), .CLR(clear), .Q(Q[3]) );
    xor2a6 U3 ( .A(enable), .B(Q[0]), .Y(n2) );
    xor2a6 U5 ( .A(Q[3]), .B(n3), .Y(N5) );
    xor2a6 U7 ( .A(Q[2]), .B(n4), .Y(N4) );
    xor2a6 U9 ( .A(Q[1]), .B(Q[0]), .Y(N3) );
    and2a3 U10 ( .A(Q[2]), .B(n4), .Y(n3) );
    and2a3 U11 ( .A(Q[0]), .B(Q[1]), .Y(n4) );
endmodule

```

Այստեղ տեխնոլոգիական գրադարանից օգտագործվել են fdef2a1 D-տրիգերը, որը բացի D տվյալի մուտքից, ունի զրո վիճակ կարգման CLR մուտքը՝ ակտիվ ցածր մակարդակով, թույլտվության E մուտքը: Բացի տրիգերից, օգտագործվել են հետևյալ տրամաբանական տարրերը՝ xor2a6, երկմուտք Բացառող-ԿԱՄ տարրեր, and2a3, երկմուտք ԵՎ տարրեր:

Սինթեզի արդյունքում ստացված Վերիլոգ կապերի ցուցակը պետք է նմանակվի՝ օգտվելով գրադարանային տարրերի Վերիլոգ նմանակման մոդելներից և տարրերի ժամանակային պարամետրերից, որոնք տրված են տեխնոլոգիական գրադարանում (\*.lib): Սինթեզն ավարտվելուց հետո սինթեզի գործիքից պետք է դուրս գրվի օգտագործված գրադարանային տարրերի ժամանակային պարամետրերը տվյալ նախագծում, որոնք տրվում են ստանդարտ հապաղումների չափաձևով (standard delay format, SDF): SDF ֆայլը գեներացվում է ըստ սինթեզից ստացված նախագծի տարրերի կապերի ցուցակի և այդ տարրերի տեխնոլոգիական գրադարանում տրված ժամանակային պարամետրերի:

Նմանական մոդելները կարող են ունենալ հետևյալ տեսքը.

```
`timescale 1ns/1ps
```

```
`celldefine
```

```
module xor2a6(A, B, Y); //Ելքի 6-րդ տանող ուժով 2Բացառող-ԿԱՄ տարր
```

```
input A ;
```

```
input B ;
```

```
output Y ;
```

```
xor (Y, A, B);
```

```
//ստորև բերվող specify բլոկում տրվում են ժամանակային պարամետրերը: Վերիլոգ մոդելում
```

```
//տրվում են այդ պարամետրերի սկզբնական արժեքները, որոնք նմանակման ժամանակ
```

```
//կարող են փոխարինվել սինթեզի գործիքի կողմից գեներացված SDF ֆայլից վերցրած
```

```
//արժեքներով: SDF ֆայլը գեներացվում է ընթացիկ կապերի ցուցակի և տեխնոլոգիական
```

```
//գրադարանում տրված ժամանակային պարամետրերից:
```

```
specify
```

```
// հապաղումների պարամետրերը
```

```
specparam // ժամանակային պարամետրերի սկզբնական (default) արժեքները
```

```
    tpIIh$A$Y = 0.0758481:0.0758481:0.0758481,
```

```
    tpIhI$A$Y = 0.0987435:0.0987435:0.0987435,
```

```
    tpIIh$B$Y = 0.107761:0.107761:0.107761,
```

```
    tpIhI$B$Y = 0.117729:0.117729:0.117729;
```

```
// path delays
```

```
(A *> Y) = (tpIIh$A$Y, tpIhI$A$Y); //եթե SDF ֆայլը առկա է, ապա tpIIh$A$Y, tpIhI$A$Y
```

```
    //հապաղումների արժեքները նմանակման գործիքը կվերցնի
```

```
    // SDF ֆայլից, եթե ոչ, ապա կվերցնի վերևում տրված արժեքները
```

```
(B *> Y) = (tpIIh$B$Y, tpIhI$B$Y);
```

```
endspecify
```

```
endmodule
```

```

`endcelldefine

`timescale 1ns/1ps
`celldefine
module and2a3(A,B,Y); //Ելքի երրորդ տանող ուժով 2ԵՎ տարր
input A,B;
output Y;

and (Y,A,B);

specify
specparam
    tpllh$A$Y = 0.110671:0.110671:0.110671,
    tphhl$A$Y = 0.121153:0.121153:0.121153,
    tpllh$B$Y = 0.115751:0.115751:0.115751,
    tphhl$B$Y = 0.130234:0.130234:0.130234;

    // path delays
    (A *> Y) = (tpllh$A$Y, tphhl$A$Y);
    (B *> Y) = (tpllh$B$Y, tphhl$B$Y);
endspecify

endmodule
`endcelldefine

`timescale 1ns/1ps
`celldefine
module fdef2a1(D,E,CLK,CLR,Q); //Ելքի առաջին տանող ուժով D տրիգեր՝ մուտքի E
    //թույլտվությանը և Ելքի զրոյացման ասինքրոն CLR ազդանշանով
input D,E,CLK,CLR;
output Q;
reg Q;
reg notifier ;//այս փոփոխականով նմանակիչին հայտնվում է ստուգվող ժամանակային
    //պայմանի խախտման մասին
always @(posedge CLK or negedge CLR) //սովորաբար տրիգերների նմանական մոդելները
    // կառուցվում են աղյուսակային նկարագրությամբ`
    //օգտագործողի սահմանած պրիմիտիվների տեսքով
    //(user defined primitive, UDP), որոնք այս գրքի
    //շրջանակներից դուրս են:

if(!CLR) Q<=1'b0;
    else if(E) Q <= D;
        else Q <= Q;
//ստորև բերվում են տրիգերի ժամանակային պարամետրերը, որոնք այստեղ չեն քննարկվի
specify

```

```

// delay parameters
specparam
  tp1lh$CLK$Q = 0.745:0.745:0.745,
  tp1hl$CLK$Q = 0.778:0.778:0.778,
  trec$CLR = -0.186:-0.186:-0.186,
  trem$CLR = 0.836:0.836:0.836,
  tsetup_posedge$D$CLK = 0.173:0.173:0.173,
  thold_posedge$D$CLK = 0.145:0.145:0.145,
  tsetup_negedge$D$CLK = 0.186:0.186:0.186,
  thold_negedge$D$CLK = 0.198:0.198:0.198,
  tsetup_posedge$E$CLK = 0.182:0.182:0.183,
  thold_posedge$E$CLK = 0.145:0.145:0.145,
  tsetup_negedge$E$CLK = 0.186:0.186:0.186,
  thold_negedge$E$CLK = 0.198:0.198:0.198,
  tminpwh$CLK = 0.335:0.335:0.335,
  tminpwl$CLK = 0.342:0.342:0.342,
  tminpwl$CLR = 0.584:0.584:0.584;

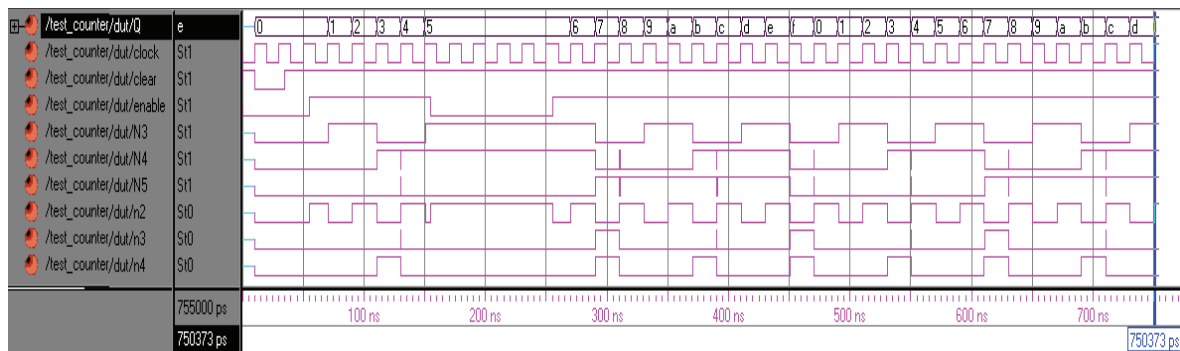
// path delays
(posedge CLK *> (Q : (D))) = (tp1lh$CLK$Q, tp1hl$CLK$Q);
$setuphold(posedge CLK, posedge D, tsetup_negedge$D$CLK, thold_negedge$D$CLK,
notifier ); //notifier փոփոխականով հայտնվում է ժամանակային պայմանի խախտման մասին
$setuphold(posedge CLK, negedge D, tsetup_posedge$D$CLK, thold_posedge$D$CLK,
notifier );
$setuphold(posedge CLK, posedge E ,tsetup_negedge$E$CLK, thold_negedge$E$CLK,
notifier);
$setuphold(posedge CLK, negedge E ,tsetup_posedge$E$CLK, thold_posedge$E$CLK,
notifier);
$recrem(posedge CLR, posedge CLK, trec$CLR, trem$CLR, notifier, notifier);
$width(posedge CLK, tminpwh$CLK notifier,);
$width(negedge CLK, tminpwl$CLK, notifier);
$width(negedge CLR, tminpwl$CLR, notifier);
endspecify
endmodule
`endcelldefine

```

Սինթեզից ստացված կապերի ցուցակի նմանակումն իրականացվել է վերևում բերված տարրերի Վերիլոգ մոդելների ընդգրկմամբ, որոնք պարունակում են նաև տարրերի ժամանակային պարամետրերը **specify** բլոկներում: Նմանակման համար օգտագործվել է նույն թեստավորման մոդուլը, ինչ վարքագծային կամ ՌՓՄ մոդելավորման ժամանակ: Այս նմանակման արդյունքները պետք է համեմատել ՌՓՄ կոդի նմանակման արդյունքների հետ (օրինակ, **\$monitor** համակարգային առաջադրանքով գեներացված աղյուսակները): Նմանակումից ստացված ժամանակային դիագրամները ցույց են տրված նկ. 8.12–ում: Ինչպես կարելի է տեսնել նկ. 8.10–ի և 8.12–ի դիագրամ–



ների համեմատումից, արդյունքները չեն տարբերվում: Միաժամանակ, նախագծի ներքին հանգույցների ժամանակային դիագրամներում կարելի է տեսնել կարճատև իմպուլսներ, որոնք առաջանում են տարրերի հապաղումների պատճառով: Տվյալ դեպքում դրանք էական ազդեցություն չունեն՝ հաշվիչն աշխատում է անխափան: Բայց դրանք կարող են բերել նախագծի ժամանակային սահմանափակումների խախտման (մասնավորապես տրիգերների տեղակայման և պահման պայմանների խախտման), եթե նմանական հաճախականությունը գերազանցի սինթեզի սահմանափակումներով առաջադրված հաճախականությանը:



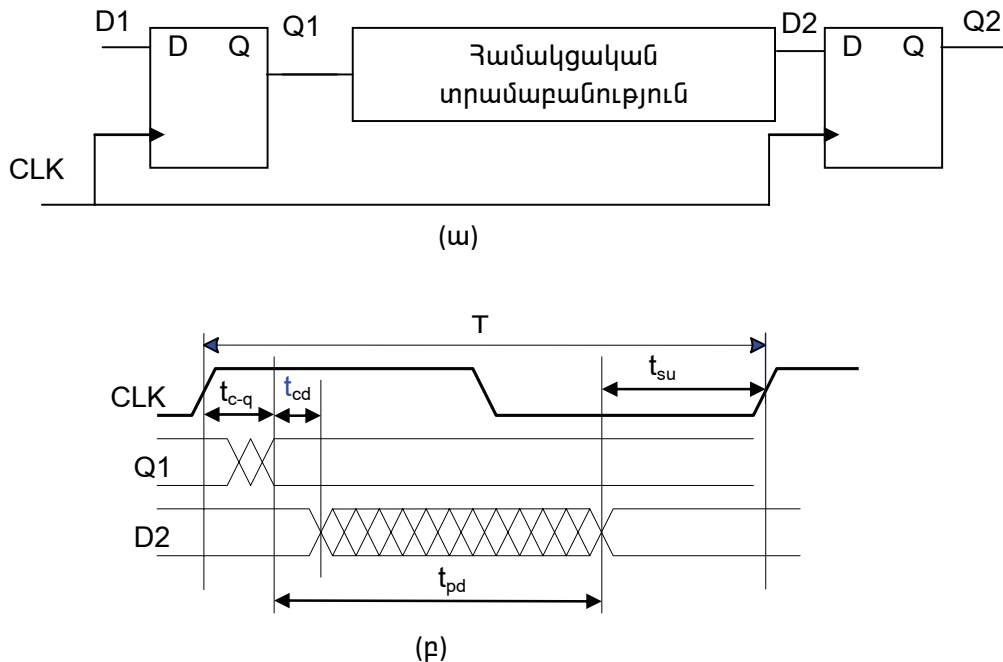
Նկ. 8.12. Սինթեզված նախագծի նմանակում գրադարանային մոդելներով և ժամանակային պարամետրերով

## 8.5. Թվային համակարգերի անսխալ սինթրոնացման պայմանների ստուգում

### 8.5.1. Թվային համակարգի ժամանակային սահմանափակումները իդեալական տակտային իմպուլսների դեպքում

Տիպային թվային համակարգը կառուցվում է վերջավոր ավտոմատի սկզբունքով, որը բաղկացած է համակցական տրամաբանական բլոկներից և հիշողության տարրերից (սևեռիչներ և տրիգերներ): Սինթրոն համակարգերում պատահարների ճշգրիտ հաջորդականությունն ապահովելու համար պետք է ապահովել սևեռիչների և տրիգերների ճշգրիտ տակտավորում: Տրիգերների ժամանակային պարամետրերը և թվային հմակարգում տրիգերների անսխալ տակտավորման պայմանները քննարկվել են 4.10 բաժնում:

Պարզագույն թվային համակարգը կարելի է ներկայացնել նկ. 8.13-ում ցույց տրված տեսքով՝ բաղկացած համակցական տրամաբանությունից և այդ տրամաբանության մուտքային և ելքային տվյալները սևեռող հաջորդական տարրերից (տրիգերներից): Համակարգի տակտավորումը կատարվում է տակտային իմպուլսի դրական (աճող) ճակատով:



Նկ. 8.13. Պարզագույն թվային համակարգի կառուցվածքը (ա) և ժամանակային դիագրամները (բ)

Համակցական շղթան, սովորաբար, պարունակում է մուտքից ելք տանող բազմաթիվ ուղիներ և բնութագրվում է վատագույն ուղով, որն ունի առավելագույն հապաղում ու լավագույն ուղով, որն ունի նվազագույն հապաղում: Համակցական տրամաբանության մուտքում ազդանշանի նոր արժեքի հաստատման պահից մինչև նրա ելքում ազդանշանի փոփոխության սկիզբն ընկած ժամանակահատվածը համապատասխանում է նվազագույն հապաղմանը և կոչվում է սկզբնական հապաղում՝  $t_{cd}$  (contamination delay): Իսկ համակցական տրամաբանության մուտքում ազդանշանի նոր արժեքի հաստատման պահից մինչև նրա ելքում ազդանշանի հաստատման պահն ընկած ժամանակահատվածը համապատասխանում է առավելագույն հապաղմանը և կոչվում է տարածման հապաղում՝  $t_{pd}$  (propagation delay): Նույն ձևով կարելի է սահմանել նշված ժամանակային պարամետրերը նաև բացասական (ընկնող) ճակատով տակտավորման դեպքում:

Նկ. 8.13-ում ցույց տրված ժամանակային դիագրամները համապատասխանում են իդեալական դեպքին, երբ տակտային իմպուլսները բոլոր տրիգերներին հասնում են ժամանակի ճիշտ նույն պահին: Իհարկե, իրականում դա այդպես չէ, բայց պարզության համար մենք կոդիտարկենք հենց այդ դեպքը: Թվային համակարգի տակտավորման պայմանների ճշգրիտ գնահատման համար պետք է հաշվի առնել տակտային ազդանշանի ուշացումները և աղավաղումները:

Համակարգի ճշգրիտ աշխատանքի համար պետք է պահպանել որոշակի պայմաններ տակտային իմպուլսների պարբերության, համակցական տրամաբանության հապաղումների և տրիգերների ժամանակային պարամետրերի միջև, որոնք կոչվում են ժամանակային պայմաններ (timing): Հերթական տակտային իմպուլսով (նկ. 8.13)

առաջին տրիգերի ելքից  $t_{c-q}$  ժամանակ (այն նշանակվում է նաև  $t_{c2q}$ ,  $t_{dFF}$ ) հետո նոր տվյալներ են փոխանցվում համակցական տրամաբանությանը: Այդ տվյալները պետք է երկրորդ տրիգերի մուտքին հասնեն հաջորդ տակտային իմպուլսի տրման պահից  $t_{su}$  ժամանակ առաջ: Նկ. 8.13-ից կարելի ստանալ համակցական տրամաբանության առավելագույն հապաղմանը ներկայացվող սահմանափակումը՝

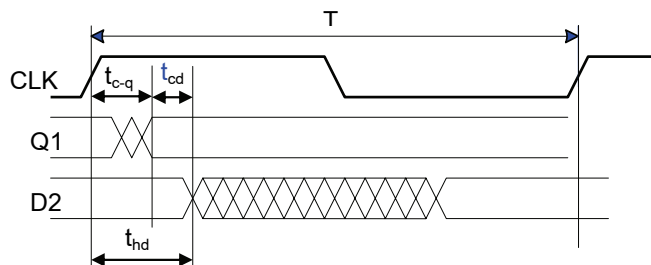
$$t_{pd} \leq T - (t_{su} + t_{c-q}) : \quad (8.1)$$

Դա նշանակում է, որ D2 ազդանշանը պետք է հաստատվի տրիգերի անհրաժեշտ տեղակայման ժամանակից ոչ ուշ, այլապես տակտային իմպուլսով երկրորդ տրիգերում կգրանցվի չհաստատված ազդանշան՝ նախատեսվածից տարբեր տրամաբանական մակարդակ: Դա կբերի համակարգի աշխատանքի խափանման: Երբ թվային շղթան չի բավարարում (8.1) պայմանին, ասում են, որ շղթայում առկա է տեղակայման ժամանակի խախտում: Ակնհայտ է, որ այդ խախտումը կարելի է վերացնել մեծացնելով տակտի տևողությունը՝  $T$ : Սակայն դա նշանակում է համակարգի արագագործության նվազեցում, որը շատ դեպքերում կարող է անթույլատրելի լինել:

Համակցական տրամաբանության նվազագույն հապաղմանը ներկայացվող սահմանափակումը կարելի է ստանալ նկ. 8.14-ում բերված ժամանակային դիագրամներից: Տվյալ տակտային իմպուլսի ազդեցության ժամանակ երկրորդ տրիգերի մուտքում D2 ազդանշանը պետք է մնա կայուն երկրորդ տրիգերի  $t_{hd}$  պահպանման ժամանակից ոչ փոքր ժամանակահատվածում:

$$t_{cd} \geq t_{hd} - t_{c-q} , \quad (8.2)$$

այսինքն Q1-ի փոփոխությունից D2-ի մակարդակը չպետք է սկսի փոխվել տրիգերի պահպանման ժամանակից ավելի շուտ, այլապես երկրորդ տրիգերում կգրանցվի ոչ թե մինչև տակտային իմպուլսի կիրառումը եղած կայուն մակարդակը, այլ տակտային իմպուլսի կիրառման պատճառով արդեն փոփոխվող D2 ազդանշանը: Եթե (8.2) պայմանը չի բավարարվում, ապա շղթայում առկա է պահպանման ժամանակի խախտում: Նկատենք, որ այս խախտումը հնարավոր չէ վերացնել տակտի տևողությունը մեծացնելով: Քանի որ  $t_{hd}$ -ն և  $t_{c-q}$ -ն տրիգերի պարամետրերն են, ապա (8.2) պայմանի խախտումը վերացնելու միակ միջոցը համակցական տրամաբանության նվազագույն հապաղման մեծացումն է:



Նկ. 8.14. Համակցական տրամաբանության (նկ. 8.13) նվազագույն հապաղմանը ներկայացվող սահմանափակումը պարզաբանող ժամանակային դիագրամները

ԻՍ-ում համակցական շղթաների հապաղումները փոխվում են մեծ սահմաններում կախված ՊԼՁ-ից (պրոցես-լարում-ջերմաստիճան): Ուստի պետք է ապահովել ժամանակային ճիշտ առնչությունները ՊԼՁ-ի բոլոր սահմանային դեպքերի համար: Այդ պատճառով սինթեզի համար նախատեսված տիպային թվային տարրերի տեխնոլոգիական գրադարանները գեներացվում են տարբեր ՊԼՁ-ների համար: Առավելագույն հապաղումը ստացվում է վատագույն դեպքում՝ դանդաղ պրոցես, ցածր լարում, բարձր ջերմաստիճան: Հետևաբար, (8.1) պայմանը պետք է ստուգել վատագույն դեպքի տրամաբանական հապաղումների և տրիգերների ժամանակային պարամետրերի համար: Նվազագույն հապաղումը ստացվում է լավագույն դեպքում՝ արագ պրոցես, բարձր լարում, ցածր ջերմաստիճան: Ուստի, (8.2) պայմանը պետք է ստուգել լավագույն դեպքի տրամաբանական հապաղումների և տրիգերների ժամանակային պարամետրերի համար:

Սինթրոն թվային համակարգի աշխատունակության ստուգումը բերվում է տրիգերների տեղակայման և պահպանման ժամանակների խախտման դեպքերի հայտնաբերմանը: Համակարգն աշխատունակ է, եթե նրանում չկա և ոչ մի տրիգերի տեղակայման կամ պահպանման ժամանակի խախտում:

Ժամանակակից թվային համակարգերում տրիգերների թիվը շատ մեծ է, իսկ տրիգերների միջև ազդանշանները փոխանցվում են՝ անցնելով մեծ թվով տրամաբանական ուղիներով: Այդ պայմանների ստուգման համար նախագծված են համապատասխան ծրագրային գործիքներ (օրինակ, Սինոփսիսի PimeTime-ը):

(8.1), (8.2) պայմանների ստուգման արդյունքները ներկայացվում են ժամանակային պայմանների խախտման (slack) արժեքներով.

$$Slack_{su} = T - t_{pd} - t_{su} - t_{c-q} \quad (8.3)$$

$$Slack_{hd} = t_{cd} + t_{c-q} - t_{hd} \quad (8.4)$$

Համակարգն աշխատունակ է, եթե հաշվված  $Slack_{su}$  և  $Slack_{hd}$  արժեքները դրական են:

Բացասական  $Slack_{hd}$ -ի դեպքում, ավտոմատացված տեղաբաշխման և ծրման գործիքը, սովորաբար, կարողանում է շտկել այն՝ ավելացնելով լրացուցիչ տրամաբանական բուֆեր տրիգերի մուտքի շղթային, որը չի փոխում շղթայի տրամաբանական ֆունկցիան, բայց  $t_{cd}$ -ին ավելացնում է հապաղում:

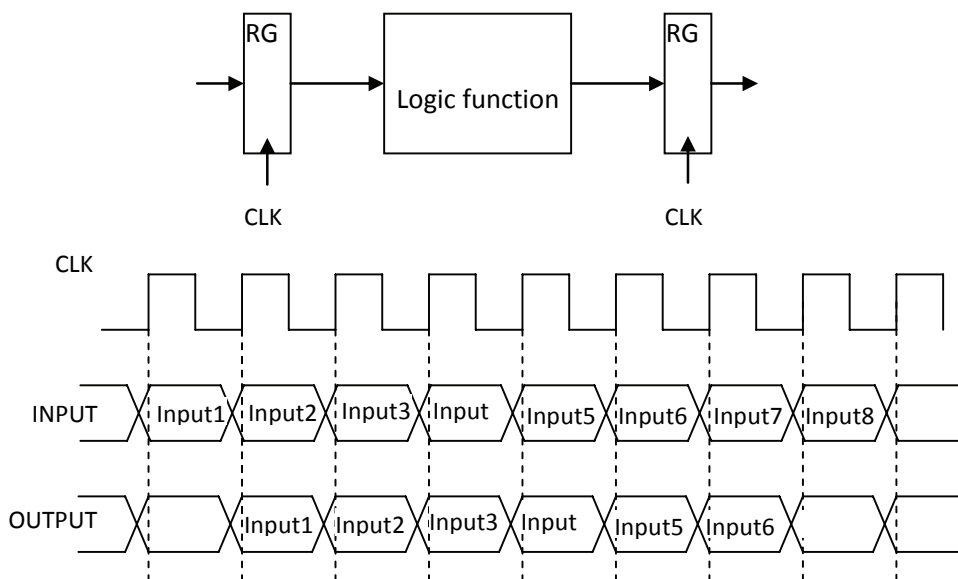
Խնդիրն ավելի բարդ է, երբ ստացվում է բացասական  $Slack_{su}$ : Այս դեպքում պետք է փոքրացնել  $t_{pd}$ -ն: Եթե բացասական  $Slack_{su}$ -ը հնարավոր չէ շտկել ավելի արագագործ տարրեր օգտագործելով, ապա պետք է կարճացնել ազդանշանների տարածման ուղիները՝ փոքրացնել դրանցում տրամաբանական տարրերի թիվը: Ակնհայտ է, որ դրա համար պետք է փոխվի սկզբնական ՌՄՓ մակարդակի կողը և կրկնել նախագծման փուլը: Կարճեցված տրամաբանական ուղիներով ՌՄՓ մակարդակի կողավորման արդյունավետ միջոց է համակցական տրամաբանության տրոհումը կոմպլեյերացման աստիճանների:

### 8.5.2. Ազդանշանների կոնվերտացված մշակում

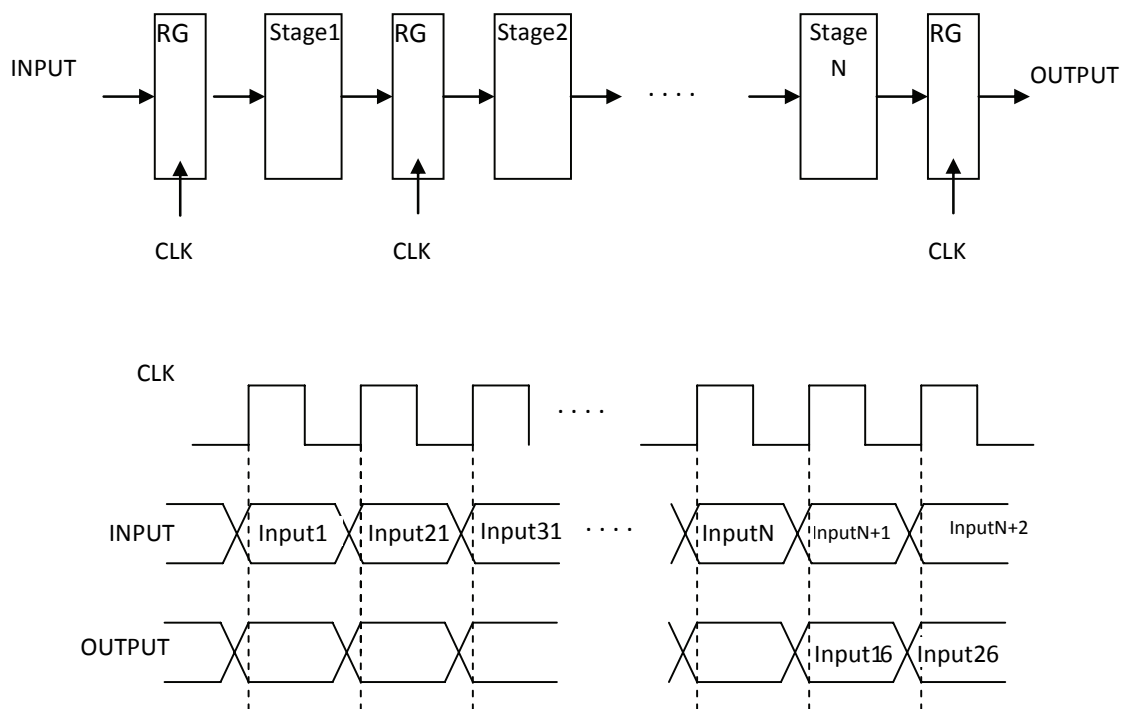
Նկ. 8.15-ում ցույց տրված սկզբնական թվային բլոկը ելքում իրականացնում է  $F(\text{INPUT})$  տրամաբանական ֆունկցիան՝ կախված մուտքային փոփոխականների INPUT վեկտորից: Մուտքային և ելքային ազդանշանները տակտային CLK ազդանշանով գրանցվում են ռեգիստրներում: Յուրաքանչյուր նոր մուտքային վեկտոր տակտային ազդանշանով գրանցվում է մուտքային ռեգիստր, որին համապատասխանող ելքային տվյալը պիտանի է դառնում տակտի մեկ պարբերության չափով ուշացումից հետո: Որպեսզի բլոկն անսխալ աշխատի  $F=1/T$  տակտային հաճախականությամբ, մուտքից ելք ամենամեծ հապաղումով ուղու հապաղումը պետք է բավարարի (8.1) պայմանին:

Այժմ դիտարկենք նկ. 8.16-ում ցույց տրված  $N$ -աստիճան կոնվերտացված կառուցվածք, որն իրականացնում է նույն տրամաբանական ֆունկցիան:  $F(\text{INPUT})$  ֆունկցիան տրոհված է  $N$  հաջորդական աստիճանների, որոնցից յուրաքանչյուրի ելքը գրանցվում է միջանկյալ ռեգիստրում՝ ընդհանուր  $(N-1)$  ռեգիստրներ են ավելացված: Բոլոր ռեգիստրները տակտավորվում են նույն սկզբնական CLK ազդանշանով: Եթե համարենք, որ տրոհված ֆունկցիայի բոլոր աստիճաններն ունեն հավասար հապաղումներ, մեկ աստիճանի հապաղումը կփոքրանա  $N$  անգամ:

$$t_{pd1} = \frac{t_{pd}}{N}, \quad (8.5)$$



Նկ. 8.15. Մեկ աստիճանով իրականացված թվային բլոկը և նրա ժամանակային դիագրամները



Նկ. 8.16. N-աստիճան կոնվեյերացված կառուցվածք, որն իրականացնում է նույն ֆունկցիան, ինչ նկ. 8.15-ի կառուցվածքը

իսկ անսախալ աշխատանքի (8.1) պայմանը կձևափոխվի հետևյալին՝

$$t_{pd1} = \frac{t_{pd}}{N} \leq T - t_{su} - t_{c-q} \quad (8.6)$$

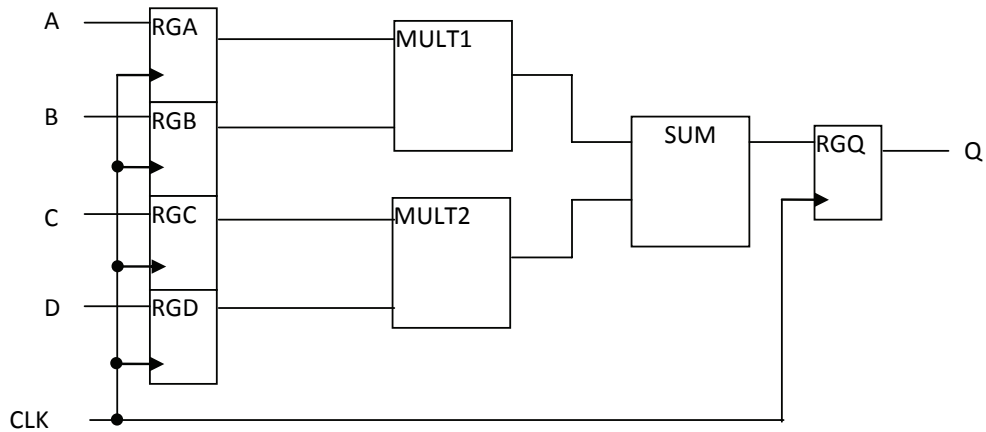
Կոնվեյերացման աստիճանների անհրաժեշտ թիվը՝ N կարելի է որոշել այս պայմանից:

Կոնվեյերացումն օգտագործում է լրացուցիչ (N-1) ռեգիստր: Այս մոտեցումը մեծացնում է մուտքից ելք տվյալների փոխանցման ուշացումը՝ մեկ տակտից մինչև N տակտ: Շատ կիրառություններում, ինչպիսիք են տվյալների կոդավորումը և մշակումը, ուշացումը զգալի անհարմարություն չէ: Կոնվեյերացումը կարող է նաև մեծացնել թվային բլոկի զբաղեցրած մակերեսը և սպառվող հզորությունը:

Որպես օրինակ դիտարկենք տվյալների մշակման հոսքուղին, որը նկարագրվում է հետևյալ արտահայտությամբ՝

$$Q = A \cdot B + C \cdot D, \quad (8.7)$$

որտեղ A, B, C, D-ն n-բիթ երկուական տվյալներ են: Այս գործողություններն իրականացնող թվային հանգույցի կառուցվածքային սխեման ցույց է տրված նկ. 8.17-ում: Այն բաղկացած է երկու n-բիթ բազմապատկիչներից (MULT1, MULT2) և մեկ 2n-բիթ գումարիչից (SUM):



Նկ. 8.17. Առանց կոմպլեյերացման տվյալների մշակման հանգույց

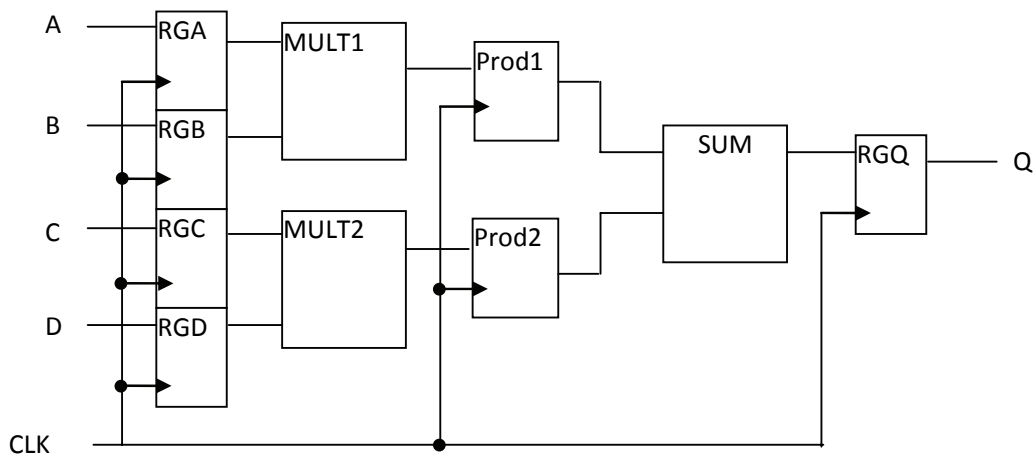
Տակտային ազդանշանի դրական ճակատով մուտքային A, B, C, D տվյալները գրանցվում են RA, RB, RC, RD ռեգիստրներ, իսկ գումարիչի ելքում ձևավորված վերջնական արդյունքը գրանցվում է RGQ ռեգիստր: Ռեգիստրից ռեգիստր հապաղման չափը կլինի՝

$$t_{pdR-R} = t_{pds m} + t_{p d m l t}, \quad (8.8)$$

որտեղ  $t_{pds m}$  և  $t_{p d m l t}$ -ն գումարիչի և բազմապատկիչի հապաղումներն են համապատասխանաբար: Այս թվային հանգույցում տեղակայման ժամանակը չխախտելու համար պետք է ապահովել հետևյալ պայմանը.

$$t_{pdR-R} \leq T - (t_{su} + t_{c-q}): \quad (8.9)$$

Դիտարկենք նկ. 8.18-ում ցույց տրված նույն ֆունկցիան իրականացնող թվային հանգույցի երկաստիճան կոմպլեյերացված իրականացումը, որտեղ առկա են երկու միջանկյալ ռեգիստրներ՝ Prod1, Prod2, որոնցում գրանցվում են A·B և C·D արտադրյալները:



Նկ. 8.18. Կոմպլեյերացմամբ տվյալների մշակման հանգույց

Այս թվային հանգույցում տեղակայման ժամանակը չխախտելու համար պետք է ապահովել հետևյալ պայմանները.

$$t_{pdmult} \leq T - (t_{su} + t_{c-q}), \quad (8.10)$$

$$t_{pdsm} \leq T - (t_{su} + t_{c-q}): \quad (8.11)$$

Ակնհայտ է, որ նույն  $T$ -ի դեպքում (8.10) և (8.11) պայմաններին ավելի հեշտ է բավարարել, քան (8.9) պայմանին, քանի որ  $t_{pdsm}, t_{pdmult} < t_{pdsm} + t_{pdmult}$ :

Ստորև բերված Վերիլոգ կոդով կարելի է սինթեզել նկ. 8.17 և նկ. 8.18-ում ցույց տրված կառուցվածքները.

- Եթե **`define** pipeline տողը փակված չէ, ապա կկոմպիլացվի **`ifdef** pipeline-ի և **`else**-ի միջև ընկած կոդը, այինքն՝ կսինթեզվի կոմպլեյերացված սարք:
- Իսկ եթե **`define** pipeline տողը փակված է (**//`define pipeline**), ապա կկոմպիլացվի **`else**-ի և **`endif**-ի միջև ընկած հատվածը, այինքն կսինթեզվի չկոմպլեյերացված սարք:

```
`define pipeline
`timescale 1ns/1ns

module sum_of_products(clk,a,b,c,d,q);
parameter width=16;
parameter pwidth=2*width;

output q;
input a,b,c,d;
input clk;

wire [width-1:0] a,b,c,d;
reg [width-1:0] ra,rb,rc,rd;
reg [pwidth:0] q, next_q;
reg [pwidth-1:0] prod1, prod1_next, prod2, prod2_next;

`ifdef pipeline
always @(posedge clk)
begin
    q<=next_q;
    ra<=a;
    rb<=b;
    rc<=c;
    rd<=d;

    prod1<=prod1_next;
    prod2<=prod2_next;
end
```



```

always @(ra or rb or rc or rd or prod1 or prod2)
begin
    prod1_next=a*b;
    prod2_next=c*d;
    next_q=prod1+prod2;
end
`else
always @(posedge clk)
begin
    q<=next_q;
    ra<=a;
    rb<=b;
    rc<=c;
    rd<=d;
end
always @(ra or rb or rc or rd)
begin
    next_q=ra*rb+rc*rd;
end
`endif
endmodule

```

Նկ. 8.19 և նկ. 8.20-ում ցույց են տրված չկոմպլեյերացված և կոմպլեյերացված տարբերակների նմանակումից ստացված ժամանակային դիագրամները: Նմանակումը կատարվել է հետևյալ թեստ-ձրագրի միջոցով:

```

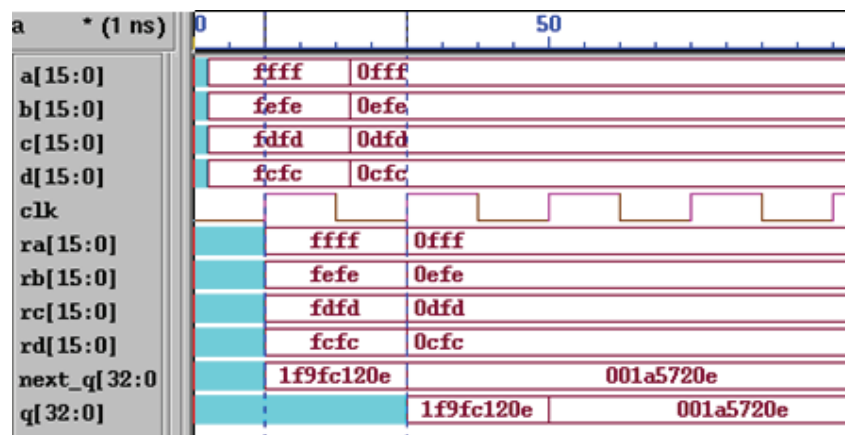
module test();
parameter width=16;
parameter pwidth=2*width;
reg clk;
reg [width-1:0] a,b,c,d;
wire [pwidth:0] q;
sum_of_products dut(clk,a,b,c,d,q);
initial
begin
    clk = 1'b0;
    #12 a =16'hffff;
    b=16'hfefe;
    c= 16'hfdfd;
    d=16'hfcfc;
    #20 a =16'h0fff;

```

```

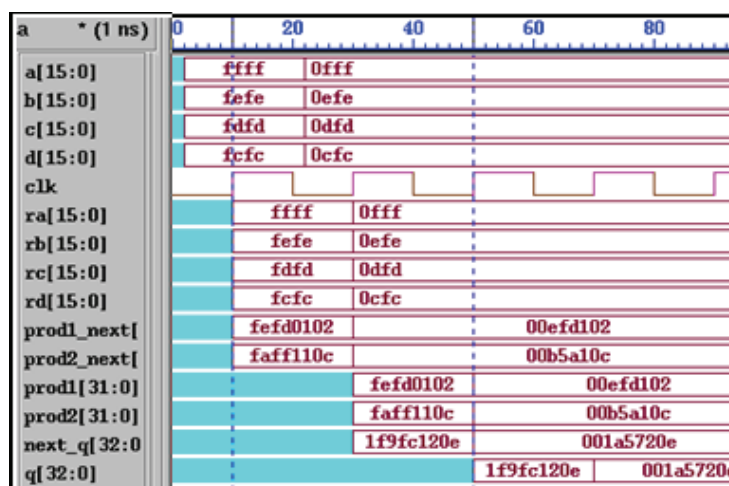
b=16'h0efe;
c= 16'h0dfd;
d=16'h0cfc;
#40 $display("a= %h",a," b= %h", b, " c= %h" , c, " d= %h", d, " q= %h",q);
#40 $finish;
end
always #10 clk=~clk;
endmodule

```



Նկ. 8.19. Առանց կոնվերացման տարբերակի նմանակումից ստացված ժամանակային դիագրամները

Առանց կոնվերացման տարբերակում մուտքային a,b,c,d տվյալները ra,rb,rc,rd ռեգիստրներում գրանցող տակտային իմպուլսի դրական ճակատից հետո հաջորդ տակտային իմպուլսի դրական ճակատով ելքային տվյալները գրանցվում են ելքային q ռեգիստր: Այսինքն՝ ելքային ազդանշանի ուշացումը մուտքայինի նկատմամբ կազմում է տակտային իմպուլսի մեկ պարբերություն:



Նկ. 8.20. Կոնվերացմամբ տարբերակի նմանակումից ստացված ժամանակային դիագրամները

Կոնվեյերացմամբ տարբերակում մուտքային a,b,c,d տվյալները ra,rb,rc,rd ռեգիստրներում գրանցող տակտային իմպուլսի դրական ճակատից հետո հաջորդ տակտային իմպուլսի դրական ճակատով միջանկյալ prod1\_next և prod2\_next տվյալները գրանցվում են prod1, prod2 ռեգիստրներ, իսկ դրան հաջորդող իմպուլսի դրական ճակատով ելքային տվյալները գրանցվում են ելքային q ռեգիստր, այսինքն՝ ելքային ազդանշանի ուշացումը մուտքայինի նկատմամբ կազմում է տակտային իմպուլսի երկու պարբերություն:

Սինթեզի համար ժամանակային սահմանափակումներից կառաջադրենք միայն տակտային իմպուլսների պարբերությունը՝ Tclkmin=9.5ns (105MHz):

Սինթեզն իրականացվել է Սինոփսիսի DC ծրագրային գործիքով՝ օգտագործելով Սինոփսիս Արմենիա ընկերության ուսումնական դեպարտամենտում մշակված SAED թվային գրադարանը (90նմ տեխնոլոգիայով):

Տվյալ օրինակում մեզ հետաքրքրում է սինթեզվող կառուցվածքների արագագործությունը, որը որոշվում է օգտագործված տրիգերների տեղակայման ժամանակների խախտումով (8.3): Հիշենք, որ տեղակայման ժամանակները պետք է ստուգվեն թվային գրադարանի վատագույն արագության պայմանից: Դրա համար սինթեզի համար օգտագործվել է գրադարանի վատագույն պրոցես/լարում/ջերմաստիճան (ՊԼՋ, PVT) պայմաններին համապատասխանող ժամանակային ֆայլը (saed90\_ss.lib).

Operating Conditions: WORST Library: saed90\_ss

Սինթեզի և օպտիմալացման արդյունքում չկոնվեյերացված տարբերակի համար ստացվել է ժամանակային պայմանների ստուգման հետևյալ հաշվետվությունը (ցույց են տրված միայն առաջին երկու տողը).

Slack, Startpoint Pin Name, Endpoint Pin Name, Arrival, Required

-1.40350, rc\_reg[0]/CK, q\_reg[30]/D, 10.66407, 9.26057

-1.40263, rc\_reg[0]/CK, q\_reg[29]/D, 10.65721, 9.25458

Այստեղ՝ Slack –ը տեղակայման ժամանակի խախտման արժեքն է, Startpoint Pin Name և Endpoint Pin Name-ը դիտարկվող ուղու սկզբնական և վերջնական հանգույցներն են, Arrival –ը վերջնական հանգույցում ազդանշանի հասնելու իրական պահն է՝ հաշված տակտային իմպուլսի դրական ճակատից, Required-ը վերջնական հանգույցում ազդանշանի հասնելու պահանջվող պահն է՝ հաշված տեղակայման ժամանակի խախտում չունենալու պայմանից ( $Required = T - t_{su}$ ): Օրինակ, առաջին տողում rc\_reg[0] սկզբնական հանգույցից (rc ռեգիստրի 0–րդ տրիգերից) ազդանշանը դուրս է մղվում CK տակտային ազդանշանի ճակատով և վերջնական q\_reg[30]/D հանգույց (q ռեգիստրի 30–րդ տրիգերի D մուտք) է հասնում 10.66407 նվ հետո (այն ներառում է rc\_reg[0] տրիգերի  $t_{c-q}$  ժամանակը և այդ ուղում համակցական հապաղումը՝  $t_{pd}$ ), իսկ պահանջվող ժամանակը 9.26057 նվ է: Հետևաբար, առկա է  $Slack = Required - Arrival = -1.40350$  նվ տեղակայման ժամանակի խախտում:

Այսպիսով, չկոնվեյերացված տարբերակը անաշխատունակ է պահանջվող 105MHz հաճախականության համար:

Այժմ դիտարկենք նույն պայմաններում սինթեզված կոնվեյերացված տարբերակի ժամանակային պայմանների ստուգման հաշվետվությունը.

Slack, Startpoint Pin Name, Endpoint Pin Name, Arrival, Required

0.00166, rb\_reg[5]/CK, prod1\_reg[28]/D, 9.12071, 9.12236

0.08899, prod1\_reg[1]/CK, q\_reg[31]/D, 9.04173, 9.13071

Պարզ է, որ սինթեզված թվային սարքում բացասական ժամանակային խախտումներ չկան: Սարքը կարող է անխափան աշխատել 105MHz հաճախականությամբ: Կոնվեյերացումը լուծեց նախագծվող հանգույցի արագագործության խնդիրը:

## 8.6. Խնդիրներ

**Խնդիր 8.1.** Կազմել (3.12)-(3.16) արտահայտություններով նկարագրվող փոխանցման կանխագուշակմամբ քառաբիթ գումարիչի ՌՓՄ կողը՝ մուտքերը և ելքերը տակտային ազդանշանի ճակատով սևեռել ռեգիստրներում: Սինթեզել գումարիչը՝ օգտվելով առկա տեխնոլոգիական գրադարանից: Օպտիմալացնել հնարավոր ամենաարագ աշխատանքի համար: Նմանակման միջոցով ստուգել աշխատունակությունը՝ ՌՓՄ-ի և սինթեզված նախագծի համար՝ օգտագործելով նույն թեստավորող մոդուլը:

**Խնդիր 8.2.** Կառուցել երկկարգ տասական թվերի գումարիչ (տե՛ս բաժին 3.7)՝ գրել ՌՓՄ կողը և սինթեզել, մուտքերը և ելքերը տակտային ազդանշանի ճակատով սևեռել ռեգիստրներում: Սինթեզել գումարիչը՝ օգտվելով առկա տեխնոլոգիական գրադարանից: Օպտիմալացնել հնարավոր ամենաարագ աշխատանքի համար:

**Խնդիր 8.3.** Կազմել Դադայի բազմապատկիչի ՌՓՄ կողը (տե՛ս բաժին 3.8), մուտքերը և ելքերը տակտային ազդանշանի ճակատով սևեռել ռեգիստրներում: Սինթեզել բազմապատկիչը՝ օգտվելով առկա տեխնոլոգիական գրադարանից: Օպտիմալացնել հնարավոր ամենաարագ աշխատանքի համար:

**Խնդիր 8.4.** Կառուցել քառաբիթ դարձափոխելի հաշվիչ՝  $M=16$ , որը կունենա K561IE11 հաշվիչի ֆունկցիոնալությունը: Սինթեզել հաշվիչը՝ օգտվելով առկա տեխնոլոգիական գրադարանից: Նմանակման միջոցով ստուգել աշխատունակությունը:

**Խնդիր 8.5.** Կառուցել քառաբիթ դարձափոխելի հաշվիչ՝  $M=16$  ու  $M=10$ , որը կունենա K561IE14 հաշվիչի ֆունկցիոնալությունը: Սինթեզել հաշվիչը՝ օգտվելով առկա տեխնոլոգիական գրադարանից: Օպտիմալացնել հնարավոր ամենաարագ աշխատանքի համար: Նմանակման միջոցով ստուգել աշխատունակությունը:

**Խնդիր 8.6.** Կազմել նկ. 5.38-ով և աղյուսակ 5.6-ով նկարագրվող համապիտանի քառաբիթ ռեգիստրի ՌՓՄ կողը: Սինթեզել ռեգիստրը՝ օգտվելով առկա տեխնոլոգիական գրադարանից: Օպտիմալացնել հնարավոր ամենաարագ աշխատանքի համար:

**Խնդիր 8.7.** Կազմել բազմաբիթ թվերի հաջորդական բազմապատկիչի ՌՓՄ կողը ըստ նկ. 6.10-ի ԱԳՍ-ի: Ավելացնել արտաքինից տրվող *reset* ազդանշան՝ ավտոմատը սկիզբ վիճակ կարգելու համար: Սինթեզել բազմապատկիչը՝ օգտվելով առկա տեխնոլոգիական գրադարանից: Նմանակման միջոցով ստուգել աշխատունակությունը:

**Խնդիր 8.8.** Կազմել բազմաբիթ թվերի հաջորդական բաժանիչի ՌՓՄ կողը ըստ նկ. 6.13-ի ԱԳՍ-ի: Ավելացնել արտաքինից տրվող *reset* ազդանշան՝ ավտոմատը սկիզբ

վիճակ կարգելու համար: Սինթեզել բաժանիչը օգտվելով առկա տեխնոլոգիական գրադարանից:

**Խնդիր 8.9.** Նկ. 8.21-ում ցույց տրված համակարգում՝  $t_{pdmult} = 14$  նվ,  $t_{pdadder} = 12$  նվ,  $t_{pdmux} = 4$  նվ,  $t_{c2q} = 3$  նվ,  $t_{su} = 1$  նվ,  $t_{hd} = 2$  նվ:

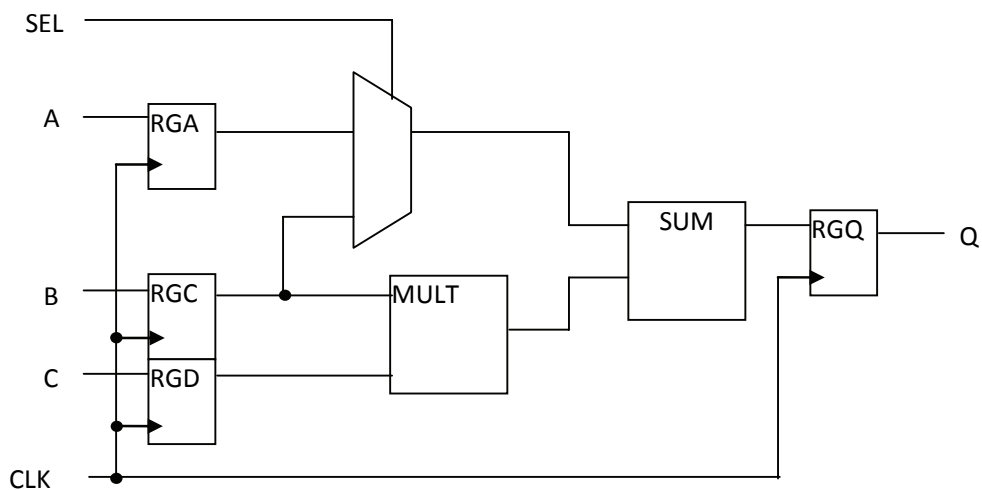
ա) Որոշել ռեգիստրից-ռեգիստր առավելագույն հապաղումը: Որոշել տակտավորման առավելագույն հաճախականությունը:

բ) Ավելացնել կոմպլեյերացման մեկ աստիճան՝ ռեգիստրից ռեգիստր հապաղումը առավելագույնս կրճատելու համար: Որոշել տակտավորման առավելագույն հաճախականությունը:

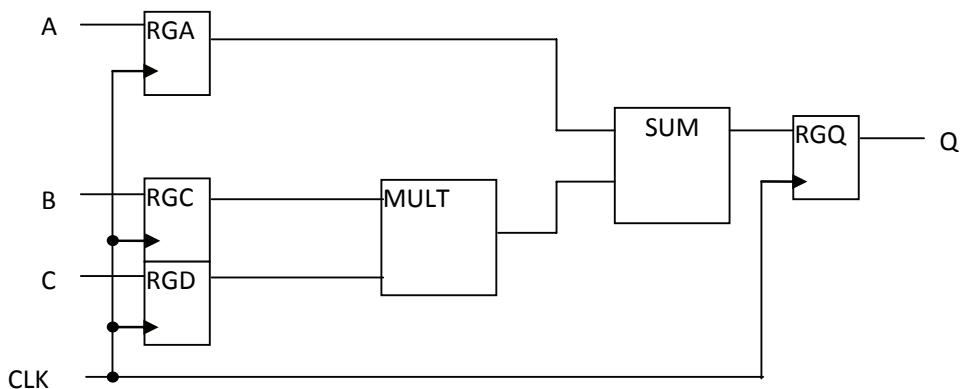
**Խնդիր 8.10.** Նկ. 8.22-ում ցույց տրված համակարգում՝  $t_{pdmult} = 18$  նվ,  $t_{pdsum} = 7$  նվ,  $t_{c2q} = 3$  նվ,  $t_{su} = 1$  նվ,  $t_{hd} = 2$  նվ:

ա) Որոշել ռեգիստրից ռեգիստր առավելագույն հապաղումը: Որոշել տակտավորման առավելագույն հաճախականությունը:

բ) Ավելացնել կոմպլեյերացման մեկ աստիճան՝ ռեգիստրից ռեգիստր հապաղումը առավելագույնս կրճատելու համար: Որոշել տակտավորման առավելագույն հաճախականությունը:



Նկ. 8.21. Խնդիր 8.9-ին վերաբերող շղթան



Նկ. 8.22. Խնդիր 8.10-ին վերաբերող շղթան

## ԽՈՒՆՈՒՄՆԵՐԻ ԼՈՒՑՈՒՄՆԵՐ և պատասխաններ

### Գլուխ 1

1.2. ա)  $\bar{x} + y\bar{z}$ , բ)  $\bar{x}\bar{z} + x\bar{y}z$ , գ)  $x\bar{y} + wxz$ :

1.4. ա)  $y$ , բ)  $\bar{w}yz$ , գ)  $0$

1.5. ա)  $(ab + c + d)ef$ , բ)  $x$ , գ)  $w\bar{x}z$ , դ)  $x$ , ե)  $yz + xz$ , զ)  $wy\bar{z}$

1.6. ա)  $n$

1.8. ա) ա)  $n$ , բ) ա)  $n$ , գ)  $n$  չէ:

1.11.  $\bar{x}yz + \bar{x}\bar{y}\bar{z}$

1.12.  $\bar{y} + xz$

1.15. ա)  $x\bar{z} + \bar{x}z$ , բ)  $\bar{x}z + x\bar{y}\bar{z}$ , գ)  $x + \bar{y} + z$

1.16.  $\bar{z}$

1.17. ա)  $\bar{x}\bar{z} + \bar{w}z + w\bar{y}\bar{z}$ , բ)  $x + \bar{y}$

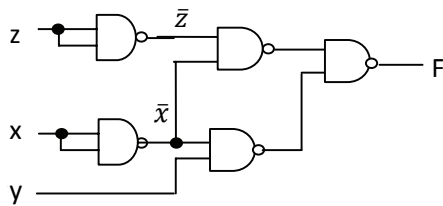
1.18. ա)  $\bar{w}\bar{z} + wyz + \bar{x}\bar{y}\bar{z}$ , բ)  $w\bar{y} + \bar{y}z + \bar{w}xy + \bar{x}y\bar{z}$ , գ)  $\bar{x}\bar{z} + \bar{w}\bar{x} + wx + \bar{w}y$

1.20. ա)  $z + xy$ , բ)  $z$ , գ)  $wx + xz + wyz$ , դ)  $x + \bar{w}yz$

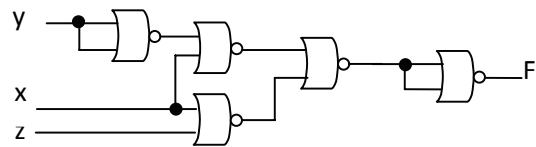
### Գլուխ 2

2.1. ա)  $\bar{x}\bar{z} + \bar{x}y$ , բ)  $\bar{z}$ , գ)  $x\bar{z} + \bar{y}$

2.2. ա)



բ)



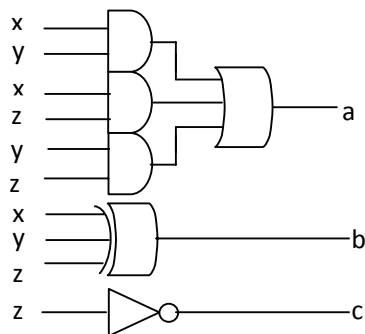
2.8.  $F = xy + xz + yz$

2.9.  $F = x\bar{y} + \bar{x}z + y\bar{z}$

2.10.  $F = wxy + wxz + wyz + xyz$

2.11.

xyz	abc
000	001
001	010
010	011
011	100
100	011
101	100
110	101
111	110



$$a = xy + xz + yz, b = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz = x \oplus y \oplus z, c = \bar{z}$$

2.14.  $\bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$

2.17.

xyz	F
000	0
001	0
010	1
011	1
100	1
101	1
110	1
111	1

2.18. Առանձին նվազարկված

$$F_1 = \bar{a}\bar{d} + \bar{a}bc + a\bar{b}d, F_2 = \bar{a}\bar{c}\bar{d} + \bar{b}c\bar{d} + a\bar{c}d + abc$$

$$C = C(F_1) + C(F_2) = 15 + 21 = 36$$

Համատեղ նվազարկված՝

$$F_1 = \bar{a}\bar{c}\bar{d} + \bar{b}c\bar{d} + \bar{a}bc + a\bar{b}d, F_2 = \bar{a}\bar{c}\bar{d} + \bar{b}c\bar{d} + a\bar{c}d + abc$$

$$C = 12 + 23 = 35$$

2.19. Առանձին նվազարկված.

$$F_1 = \bar{a}\bar{b}c\bar{d} + \bar{a}\bar{b}d\bar{e} + \bar{a}bd + abde + ac\bar{d}\bar{e}, F_{2min} = \bar{c}\bar{e} + \bar{a}\bar{b}\bar{d} + \bar{a}bd + bde + a\bar{d}\bar{e}$$

$$C = C(F_1) + C(F_2) = 30 + 25 = 55$$

Համատեղ նվազարկված՝

$$F_1 = \bar{a}\bar{b}c\bar{d} + \bar{a}\bar{b}d\bar{e} + \bar{a}bd + abde + ac\bar{d}\bar{e}, F_2 = F_1 + \bar{c}\bar{e}$$

$$C = 30 + 6 = 36$$

2.20. ա)  $\bar{b}\bar{c} + \bar{a}\bar{b} + \bar{a}\bar{c}$ , բ)  $\bar{a}\bar{b}cd + \bar{a}\bar{b}c + b\bar{c} + b\bar{d}$ , գ)  $\bar{a}\bar{b} + cd + \bar{b}\bar{d}$ ,

դ)  $\bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}c\bar{d} + \bar{a}b\bar{c}\bar{d} + \bar{a}bcd + ab\bar{c}\bar{d} + abc\bar{d} + a\bar{b}c\bar{d} + a\bar{b}cd$

2.21. ա)  $\bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}d + \bar{a}\bar{c}d + abc$ , բ)  $\bar{a}\bar{b} + \bar{b}\bar{c} + \bar{b}d$ , գ)  $\bar{a}b + \bar{c}d$

2.22. ա)  $\bar{a}\bar{b}c + \bar{a}\bar{c}$ , բ)  $\bar{a}\bar{b}c + a\bar{c}$ , գ)  $\bar{a}\bar{b}c + \bar{c}$

2.23. ա)  $\bar{a}ce + \bar{a}bd + bce + \bar{a}\bar{b}\bar{d}\bar{e}$ , բ)  $\bar{b}\bar{c}\bar{d}\bar{e} + \bar{a}b\bar{d} + cde + acd$ , գ)  $\bar{a}\bar{b}\bar{c} + \bar{b}c\bar{d} + bd\bar{e} + ab\bar{d} + b\bar{c}d$ ,

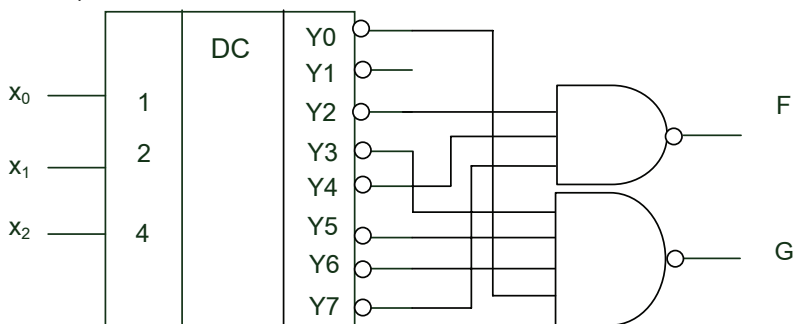
դ)  $\bar{b}\bar{c}\bar{e} + \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}cd + bc\bar{d}e + bcd\bar{e} + \bar{a}b\bar{c}d + \bar{a}\bar{d}\bar{e}$

2.25.  $\bar{a}c + c\bar{e} + be$

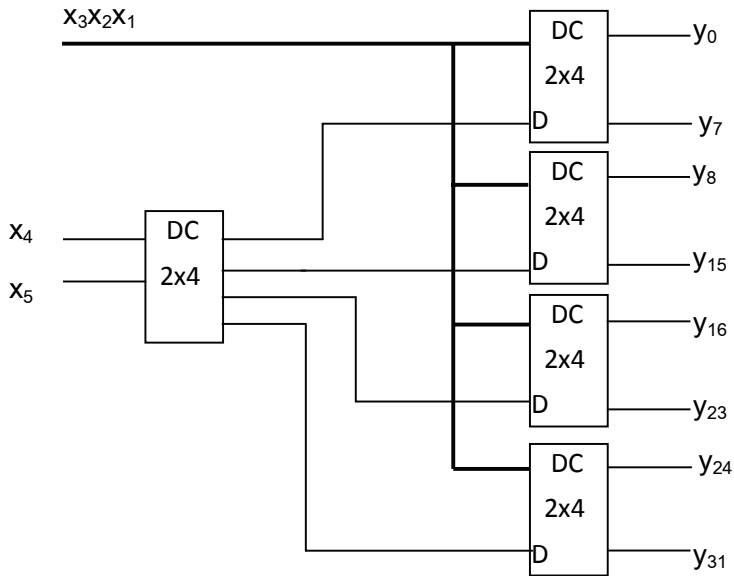
2.26. ա)  $\bar{a}\bar{b}\bar{e}f + \bar{a}bdf + d\bar{e}f$ , բ)  $\bar{d}\bar{e}\bar{f} + a\bar{c}\bar{d}\bar{f} + \bar{a}\bar{b}\bar{c}df + \bar{b}\bar{c}\bar{e}\bar{f}$ , գ)  $\bar{b}\bar{c}\bar{e}\bar{f} + \bar{a}b\bar{c}e + b\bar{c}\bar{d}\bar{f} + \bar{a}de + \bar{a}bd$

2.27. 12.5

2.28. ա)

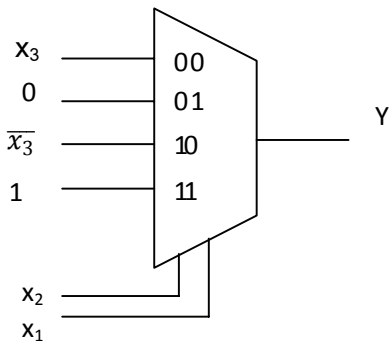


**2.31.**



**2.33.**  $A_0 = D_1 \bar{D}_0 + \bar{D}_2 \bar{D}_0$ ,  $A_1 = \bar{D}_1 \bar{D}_0$ ,  $G = \overline{D_0 + D_1 + D_2 + D_3}$

**2.35.**  $l_0 = x_3$ ,  $l_1 = 0$ ,  $l_2 = \bar{x}_3$ ,  $l_3 = 1$



**2.36.** у)  $l_0=0, l_1=\bar{x}_4, l_2=0, l_3=x_4, l_4=1, l_5=1, l_6=1, l_7=0$

p)  $l_0=1, l_1=\bar{x}_4, l_2=1, l_3=1, l_4=0, l_5=0, l_6=0, l_7=1$

q)  $l_0 = x_4, l_1 = \bar{x}_4, l_2 = \bar{x}_4, l_3 = x_4, l_4 = \bar{x}_4, l_5 = x_4, l_6 = x_4, l_7 = \bar{x}_4$

### ԳԼՈՒԽ 3

**3.1.** ω) 6a, ρ) b6, q) 152, η) 266, t) 10001101100111, q) 1101010111011101, t) 1111111100011001, ρ) 1000100010001, ρ) 3f4f, d) 3f1e, h) 125715, j) 37575:

**3.2.** ω) 1111111, ρ) 111010101111, q) 1112, η) 7315, ε) 11d8, q) 10524:

**3.3. ω) 106, ρ) 182, q) 9063, η) 54749, t) 32537, q) 16253:**

**3.4.** ω)  $0.5765=0.447$ , ρ)  $12.307=14.2351$ , q)  $0.5765=0.10010011$ , η)  $12.307=1100.010011101$ :

**3.5.** у) 100100111, р) 001101111001, қ) 001110000010, ң) 0010000000111000,  
б) 0011100101000100, ж) 0010001000010000, т) 000100000110, п) 000110000010:

**3.6. ω) 1001100, ρ) 100000010, q) 11110000:**

**3.7. ω) 11110, ρ) 1101100, q) 11100110:**



3.8. ա) 1\_10011010, բ) 0\_1000011001, գ) 1\_100101, դ) 1\_1010001100, ե) 0\_110011100:

3.9. ա) +213, բ) -237, գ) -13, դ) +42:

3.10. ա) 43, բ) 322, գ) 6044, դ) 54.63, ե) 0.2325:

3.11. ա) 42, բ) 677, գ) 3955, դ) 44.36, ե) 0.7674:

3.12. ա) 00101010, բ) 00010010, գ) 01110010, դ) 11010101:

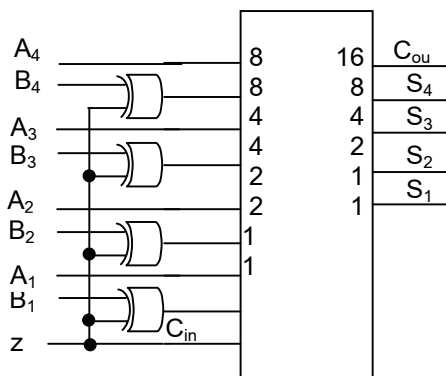
3.13. ա) 00101011, բ) 00010011, գ) 01110011, դ) 11010110:

3.14. ա) 01011011, բ) 00110001, գ) 11101111, դ) 01111100 (գերլցում),  
ե) 10010111 (գերլցում):

3.15. ա) 7-ական և բարձր, բ) 4-ական և բարձր, գ) 4-ական, դ) 8-ական

ե) 5-ական, գ) 6-ական

3.19.

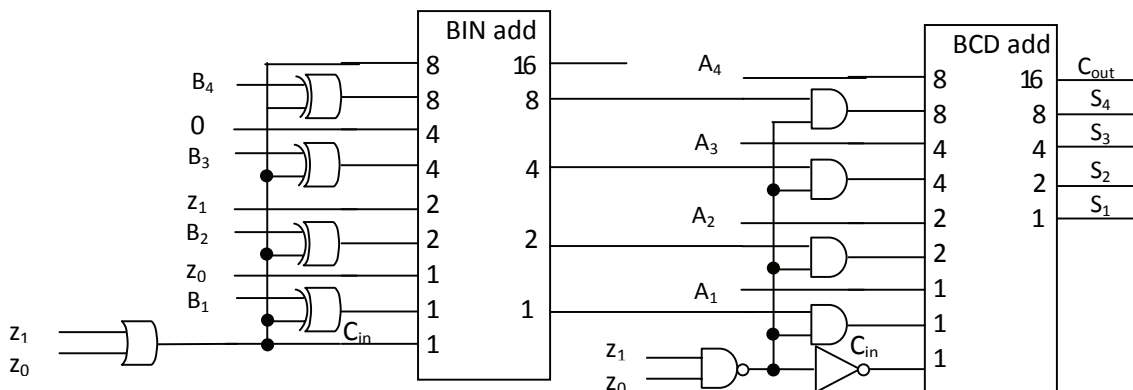


3.20. Գումարիչի սխեման ներկայացված է նկ.3.7-ում: ա) 4նվ, բ) 10նվ

3.21. Խնդիր 3.18-ի պատասխանում եԿՏ թիվը կիրառել B մուտքերին և վերցնել A=9:

3.22. Խնդիր 3.18-ի պատասխանում եԿՏ թիվը կիրառել B մուտքերին և վերցնել A=1010:

3.23.



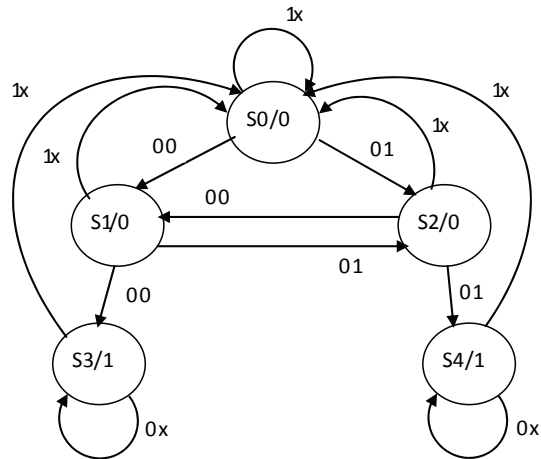
3.24. bcd0=bin0, bcd7=0, մնացած bcd բիթերը ծրագրավորվում են ՀՀՍ-ում: ՀՀՍ-ի հասցեային մուտքերին տրվում են bin1... bin5:

3.25. ա) 256x8, բ) 32x9, գ) 198x9

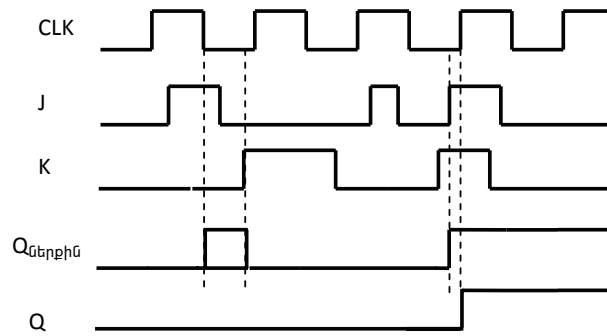
3.28. Խնդիր 3.18-ի պատասխանում Cout ցույց կտա համեմատման արդյունքը, երբ z=1:

## Գլուխ 4

4.1.

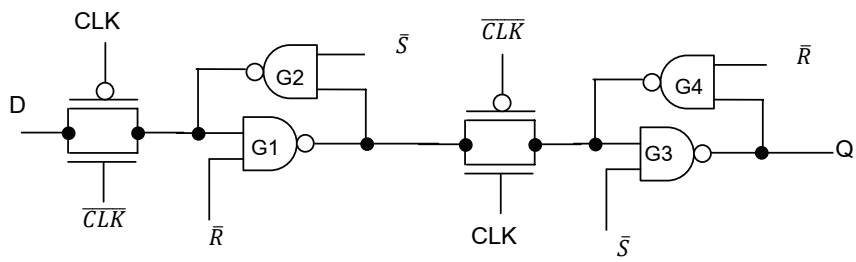


4.2.



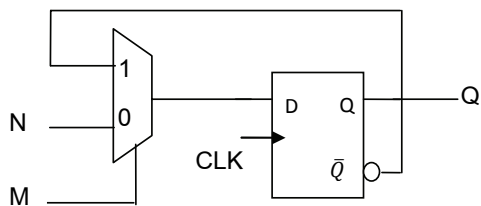
4.3.  $Q_1Q_2=10$

4.4.



ա)  $G_1=0, G_2=1, G_3=0, G_4=1$ , բ)  $G_1=1, G_2=1, G_3=0, G_4=1$ , գ)  $G_1=0, G_2=1, G_3=0, G_4=1$

4.5.



4.8. 760 սվ

4.12. ա. 0001, 1000, 1100, 0110, 1011, 0101, 1010, 1101, 1110, 1111, 0111, 0011:

4.13.

Ներկա վիճակ	Հաջորդ վիճակ		Ելք	
	X=0	X=1	X=0	X=1
A	F	B	0	0
B	D	A	0	0
D	G	A	1	0
F	F	B	1	1
G	G	D	0	1

4.14.  $A^+ = (xB + \bar{y}\bar{B})\bar{A} + \overline{x\bar{y}\bar{B}A}$ ,  $B^+ = x\bar{A}\bar{B} + \overline{x\bar{y}} + \bar{A}B$

4.15.  $T_0 = x$ ,  $T_1 = \bar{Q}_1Q_0 + \bar{x}Q_2 + xQ_0 + \bar{x}Q_1\bar{Q}_0$ ,  $T_2 = Q_2 + xQ_1Q_0$

4.17.  $D_0 = x$ ,  $D_1 = \bar{x}Q_1 + \bar{x}Q_0 + xQ_1Q_0$

4.19.  $D_0 = y$ ,  $D_1 = \bar{x}yQ_0$ ,  $z = \bar{Q}_1\bar{x}$

Ներկա վիճակ $Q_1Q_0$	Հաջորդ վիճակ				Ելք, z			
	xy=00	xy=01	xy=10	xy=11	xy=00	xy=01	xy=10	xy=11
00	00	01	00	01	1	1	0	0
01	00	11	00	01	1	1	0	0
10	00	01	00	01	0	0	0	0
11	00	11	00	01	0	0	0	0

4.20.  $D_0 = \bar{Q}_1\bar{Q}_0 + x\bar{Q}_1 + \bar{x}\bar{Q}_0$ ,  $D_1 = \bar{x}\bar{Q}_1\bar{Q}_0 + \bar{x}\bar{Q}_1Q_0 + xQ_1Q_0$ ,  $Y = Q_1\bar{Q}_0$

## Գլուխ 5

5.1.  $T_0 = 1$ ,  $T_1 = \bar{Q}_0$

5.2.  $J_0 = \bar{Q}_1$ ,  $K_0 = 1$ ,  $J_1 = Q_0$ ,  $K_1 = 1$ ,

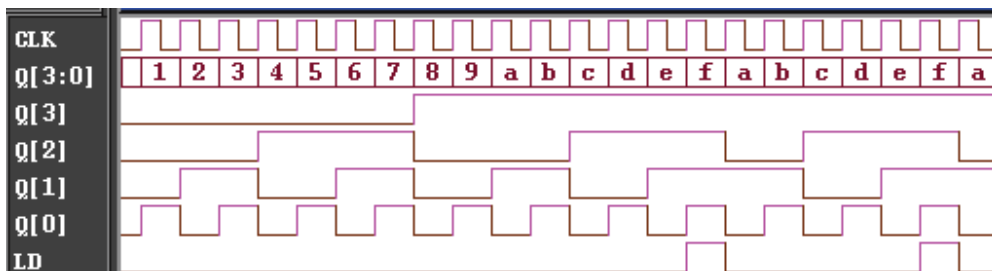
5.3.  $R_2 = Q_1Q_0$ ,  $S_2 = Q_1\bar{Q}_0$ ,  $R_1 = Q_2Q_1$ ,  $S_1 = \bar{Q}_1Q_0$ ,  $R_2 = Q_1Q_0$ ,  $S_0 = \bar{Q}_1$ ,  $R_0 = Q_1$

5.4.  $T_2 = \bar{Q}_1 + Q_2$ ,  $T_1 = Q_2 + Q_1$ ,  $T_0 = \bar{Q}_2Q_1$

5.6. Հուշում՝ տե՛ս նկ. 5.19 աղյուսակ 5.4:

5.7. 4294967

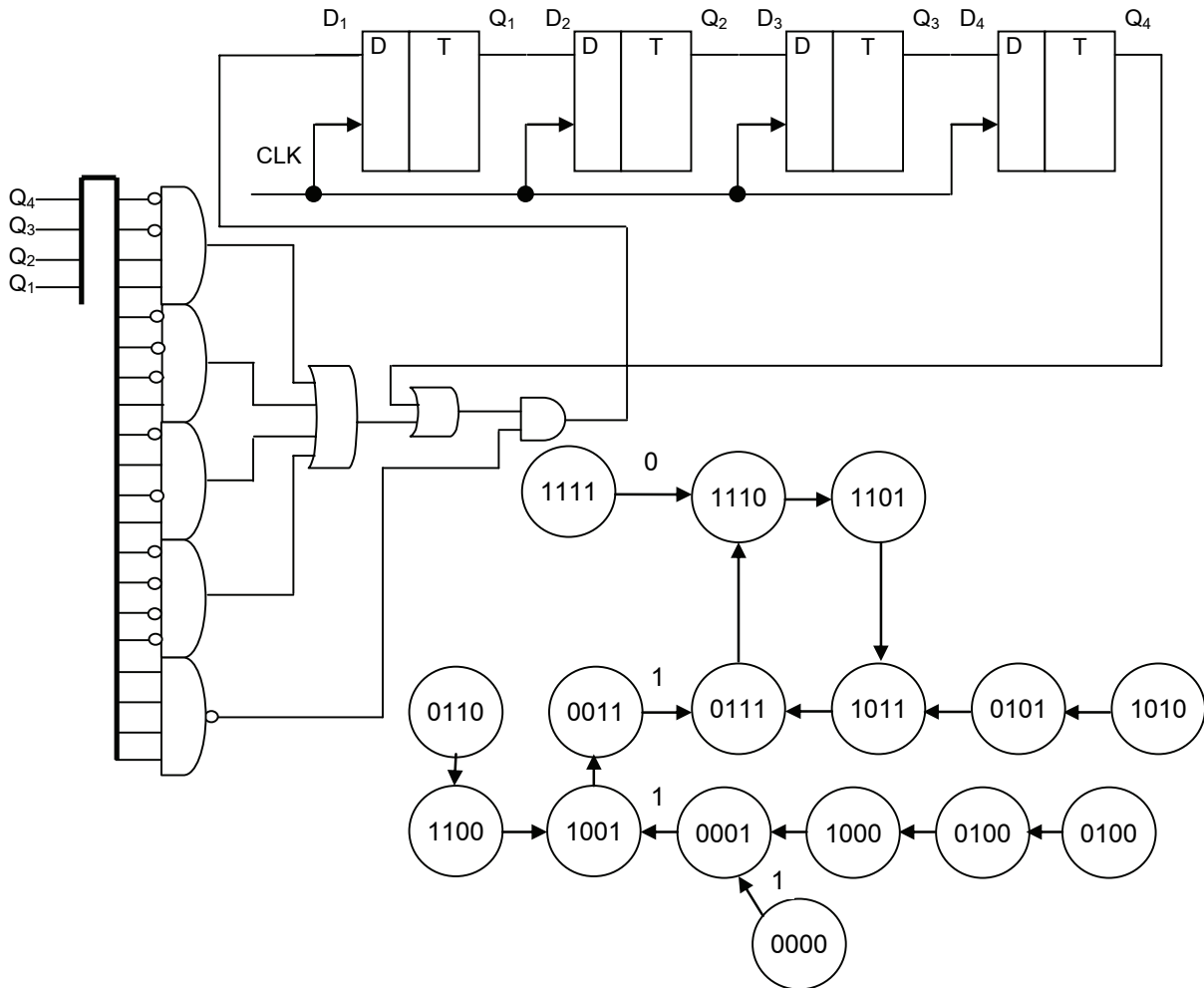
5.8.  $M=2^4-D=16-10=6$ . Հաշվիչը հաջորդաբար անցնում է a,b,c,d,e,f վիճակներով: Ժամանակային դիագրամները բերված են ստորև:



5.11.  $t_{C-Q} + (n-1)t_{and} + t_{su}$

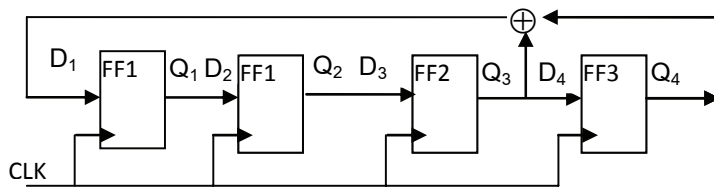
5.14. 1011

**5.15.**



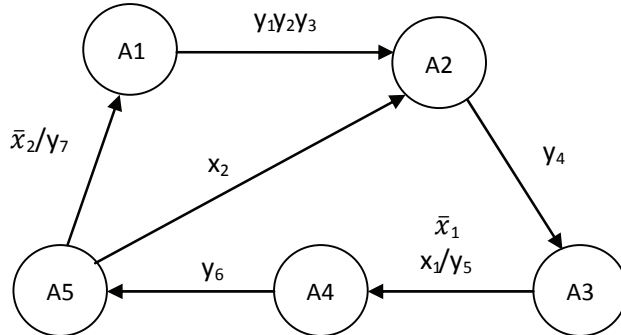
Աղեղները նշված են D1-ի այն արժեքով, որի դեպքում հաշվիչը վերադառնում է հիմնական աշխատանքային ցիկլին:

**5.17.**

$$Q_1Q_2Q_3Q_4=1000,0100,0010,1001,1100,0110,1011,0101,1010,1101,1110,1111,0111,0011,0001$$


## Գլուխ 6

### 6.2.



### 6.4. Տե՛ս խնդիր 6.2-ի պատասխանը:

Տողի #	$a_m$	$K(a_m) = Q_3 Q_2 Q_1$	$a_s$	$K(a_s) = Q_3^+ Q_2^+ Q_1^+$	$X(a_m, a_s)$	$Y(a_m, a_s)$	$I(a_m, a_s)$
1	$a_1$	000	$a_2$	001	1	$y_1, y_2, y_3$	$D_1$
2	$a_2$	001	$a_3$	010	1	$y_4$	$D_2$
3	$a_3$	010	$a_4$	011	$\bar{x}_1$	-	$D_2 D_1$
4	$a_4$	011	$a_5$	100	$x_1$	$y_5$	
5	$a_5$	100	$a_2$	001	$x_2$	-	$D_1$
6	$a_5$	100	$a_1$	000	$\bar{x}_2$	$y_7$	-

$$D_1 = \bar{Q}_3 \bar{Q}_2 \bar{Q}_1 + \bar{x}_1 \bar{Q}_3 Q_2 \bar{Q}_1 + x_2 Q_3 \bar{Q}_2 \bar{Q}_1, D_2 = \bar{Q}_3 \bar{Q}_2 Q_1 + \bar{x}_1 \bar{Q}_3 Q_2 \bar{Q}_1, D_3 = \bar{Q}_3 Q_2 Q_1$$

$$y_1 = y_2 = y_3 = \bar{Q}_3 \bar{Q}_2 \bar{Q}_1, y_4 = \bar{Q}_3 \bar{Q}_2 Q_1, y_5 = x_1 \bar{Q}_3 Q_2 \bar{Q}_1, y_6 = \bar{Q}_3 Q_2 Q_1, y_7 = \bar{x}_2 Q_3 \bar{Q}_2 \bar{Q}_1$$

### 6.6. Տե՛ս խնդիր 6.4-ի պատասխանը:

#	$x_2$	$x_1$	$Q_3$	$Q_2$	$Q_1$	$y_7$	$y_6$	$y_5$	$y_4$	$y_3$	$y_2$	$y_1$	$D_3$	$D_2$	$D_1$
1	-	-	0	0	0	-	-	-	-	1	1	1	-	-	1
2	-	-	0	0	1	-	-	-	1	-	-	-	-	1	-
3	-	0	0	1	0	-	-	-	-	-	-	-	-	1	1
4	-	1	0	1	0	-	-	1	-	-	-	-	-	-	-
5	-	-	0	1	1	-	1	-	-	-	-	-	1	-	-
6	1	-	1	0	0	-	-	-	-	-	-	-	-	-	1
7	0	-	1	0	0	1	-	-	-	-	-	-	-	-	-

## Գլուխ 7

### 7.2.

```
module one_bit_comparator(); //միաբիթ համեմատիչի մոդուլ
    մուտք/ելք մատույցների հայտարարում
    համեմատիչի ֆունկցիան նկարագրող արտահայտություններ
endmodule

module four_bit_comparator(); //քառաբիթ համեմատիչի մոդուլ՝ բարձր մակարդակի մոդուլ
    մուտք/ելք մատույցների հայտարարում
        one_bit_comparator i0(); // one_bit_comparator մոդուլի չորս օրինակների տեղադրում
        one_bit_comparator i1();
        one_bit_comparator i2();
        one_bit_comparator i3();
endmodule
```

7.3. ա) 8b'01111001, բ) 8'hzz, գ) -4'd2=4'1101:

7.4. ա) սխալ է, ճիշտը՝ %, բ) ճիշտ է, գ) ճիշտ է, դ) սխալ է, ճիշտը՝ \:

7.5. ա) ճիշտ է, բ) սխալ է՝ չի կարող սկսվել թվանշանով, գ) ճիշտ է:

7.6. ա) wire [9:0] bus; բ) reg [63:0] data; գ) integer count; դ) time delay;  
ե) integer count[0:19]; զ) reg [63:0] stack[0:255];

7.7. ա) The current value of latch =

1100

բ) <ընթացիկ ժամանակ> In register value =

010

գ) The maximum memory size is 1000000000

7.10. ա)

```
module shift_reg4(clk, din, qout);
    input clk;
    input [3:0] din;
    output [3:0] qout;
endmodule

բ)

module test;
    reg CLK;
    reg [3:0] REG_IN;
    wire [3:0] REG_OUT;
    shift_reg4 sr1(CLK, REG_IN, REG_OUT);
endmodule

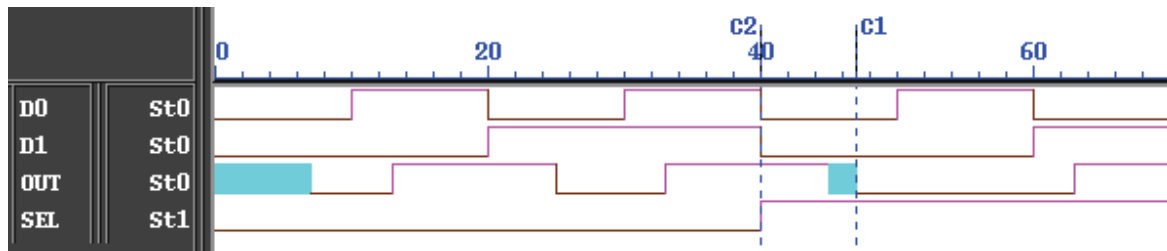
գ) shift_reg4 sr1(.qout(REG_OUT), .clk(CLK), .din(REG_IN));
դ) test.sr1, test.sr1.clk, test.sr1.din:
```

```

7.11. module my_and2(Y, A, B);
    output Y;
input A, B;
    wire n1;
        nand i1( n1, A, B);
        nand i2(Y, n1, n1);
    endmodule

```

7.13.



```

7.15. module hamming_coder_4_7(b,a);
    input [4:1] b;
    output [7:1] a;
        assign a[3] = b[1], a[5]=b[2], a[6]=b[3], a[7]=b[4],
            a[1] = a[3] ^a[5]^a[7],
            a[2] = a[3]^a[6]^a[7],
            a[4] = a[5]^a[6]^a[7];
    endmodule

```

```

7.18. module clock;
    parameter period=120;
    parameter pulse_width=period*30/100;
    parameter pause=period*70/100;
    reg clk;
    initial clk=1'b0;
    always
    begin
        #pause clk=1'b1;
        #pulse_width clk=1'b0;
    end
endmodule

```

```

7.19. 0 a=0 b=x, c=x, d=xxx
10 a=0 b=1, c=x, d=xxx
15 a=0 b=1, c=0, d=xxx
35 a=0 b=1, c=0, d=010

```

```

7.20. 0 a=0 b=x, c=x, d=xxx
5 a=0 b=x, c=0, d=xxx
10 a=0 b=1, c=0, d=xxx
20 a=0 b=1, c=0, d=010

```

```

7.21. module d_ff(clear, set, clk, d, q);
    input clear, set, clk, d;
    output q;
    reg q;

```

```

always @(posedge clk or posedge clear or posedge set)
    if(clear) q=1'b0;
    else if(set) q=1'b1;
    else q <= d;

endmodule

7.24. module bcd_adder(cin, a, b, sum, cout);
    input cin;
    input [3:0] a, b;
    output reg [3:0] sum;
    output reg cout;
    reg [4:0] bin_sum;

    always @(cin, a, b)
    begin
        bin_sum=a+b+cin;
        if(bin_sum < 5'd10) {cout, sum} = bin_sum;
        else {cout, sum} = bin_sum+6;
    end
endmodule

7.27. module hamming_decoder_4_7(a,b);
    input [7:1] a;
    output [4:1] b;
    reg [4:1] b;
    reg [2:0] s;
    always @(a)
    begin
        s[0] = a[1] ^ a[3] ^ a[5] ^ a[7];
        s[1] = a[2] ^ a[3] ^ a[6] ^ a[7];
        s[2] = a[4] ^ a[5] ^ a[6] ^ a[7];
    end
    always @(s or a)
    case(s)
        3'b000, 3'b001, 3'b010, 3'b100: b={a[7:5], a[3]};
        3'b011: b={a[7:5], ~a[3]};
        3'b101: b={a[7:6], ~a[5], a[3]};
        3'b110: b={a[7], ~a[6], a[5], a[3]};
        3'b111: b={~a[7], a[6], a[5], a[3]};
    endcase
endmodule

7. 32. module multiply;
    function [7:0] mult4;
    input [3:0] multiplier, multiplicand;
    begin
        mult4=multiplier*multiplicand;
    end
endfunction

initial $display ("product=%d", mult4(4'd12,4'd8));
endmodule

```



## Գլուխ 8

### 8.8.

```
module divider(reset, start, clk, divisor, dividend, quotient, remainder, ready);
parameter SIZE=4;    //կարելի է փոխել մոդուլի օրինակը տեղադրելիս
parameter Y0=3'b000, Y1=3'b001, Y2=3'b010, Y3=3'b011, Y4=3'b100;
input [SIZE-1:0] divisor, dividend;
input reset, start, clk;
output [SIZE-1:0] quotient;
output [SIZE-1:0] remainder;
output ready;

reg [SIZE-1:0] a, b, q, r;
reg [SIZE-1:0] p; //բիթերի հաշվիչ
reg e;           //կարգվում է 0, երբ  $r < b$ , հակառակ դեպքում՝ 1
reg ready;       //արդյունքի պատրաստ լինելու ազդանշան
reg [2:0] state, next_state;

always @(posedge clk or posedge reset)
if(reset) state<=Y0;
else state<=next_state;

always @(state or start)
case(state)
Y0: if(start) next_state=Y1;
    else next_state=Y0;
Y1: begin
    ready=1'b0;
    r=0;
    a=dividend;
    q= 0;
    b= divisor;
    p= SIZE;
    next_state=Y2;
end
Y2: begin
    {r, a}={r, a}<<1;
    p=p-1;
    e=1'b0;
    if(r < b) next_state=Y4;
    else next_state=Y3;
end
Y3: begin
    r=r-b;
    e=1'b1;
    next_state=Y4;
end
end
```

```

        Y4: begin
            {q,e}={q,e}<<1;
            if(p) next_state=Y2;
            else begin
                ready=1'b1;
                next_state=Y0;
            end
        end
    end
    default: next_state=2'bx;
endcase
    assign remainder=r,
        quotient=q;
endmodule

module test_divider;
parameter SIZE=8;
reg [SIZE-1:0] divisor, dividend;
reg reset, start, clk;
wire [SIZE-1:0] quotient, remainder;
divider #8 dut(reset, start, clk, divisor, dividend, quotient, remainder, ready);
initial
    begin
        start=0;
        reset=0;
        start =1'b0;
        clk=1'b0;
        dividend=140;
        divisor=9;
        #5 reset=1;
        #20 reset=0;
        #20 start =1'b1;
        #500 $finish;
    end
always #10 clk=~clk;
endmodule

```

8.9. ա)  $T_{clk} > t_{c2q} + t_{m1t} + t_{sm} + t_{su} = 3 + 14 + 12 + 1 = 30 \text{ ns}$

բ) ավելացնել ռեգիստր բազմապատկիչի և գումարիչի միջև.

$T_{clk} > t_{c2q} + t_{mx} + t_{sm} + t_{su} = 3 + 4 + 12 + 1 = 20 \text{ ns}$

## ՕԳՏԱԳՈՐԾՎԱԾ ԳՐԱԿԱՆՈՒԹՅԱՆ ՑՈՒՑԱԿ

1. Угрюмов Е.П. Цифровая схемотехника. 2-е издание. Издательство: БХВ-Петербург. 2006, 524 с.
2. Р. Дж. Точки, Н.С. Уидмер. Цифровые системы. Теория и практика Издательский дом "Вульямс", Москва, Санкт-Петербург, Киев 2004, 1024 с.
3. В.В. Соловьев. Проектирование цифровых систем на основе программируемых логических интегральных схем. Искд.-во Горячая линия — Телеком. 2007, 636 с.
4. Поляков А.К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры. М: Солон-Пресс. 2003, 305 с.
5. Дж. Ф. Уейкерли. Проектирование цифровых устройств. Том 1,2.-М.: Постмаркет, 2002. -528с.
6. Баранов С. И, Склад В.А. Цифровые устройства на программируемых БИС с матричной структурой. — М.: Радио и связь, 1986, 270с.
7. Mano M. Moris. Digital Desgn. Prentice Hall; 4<sup>th</sup> edition, 2006, pp.624.
8. S. Brown, Z. Vranesic. Fundamentals of Digital Logic with Verilog Design. 2<sup>nd</sup> edition, McGraw-Hill Higher Education, 2008, 935p.
9. M. J. S. Smith. Application-Specific Integrated Circuits. Addison Wesley Longman, Inc. 1997. pp. 1040.
10. S. Palnitkar. Verilog HDL: A Guide to Digital Design & Synthes. Second Edition. Prentice Hall PTR. 2003. pp. 496.
11. Verilog Hardware Description Language Reference Manual, Release 1.0, November, 1991. pp. 296.
12. IEEE Std. 1364-1995. IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language. 1996. pp. 653.
13. IEEE Std. 1364-2001(E). Verilog Hardware Description Language. 2004. pp. 862.
14. IEEE Std. 1364-2005. IEEE Standard for Verilog Hardware Description Language. 2006. pp. 590.
15. H. Bhatnagar. Advanced ASIC Chip Synthesis Using Synopsys Design Compiler Physical Compiler And Primetime. Kluwer Academic Publishers, second Edition, 2002, pp 341.
16. Վ. Ս. Մովսիսյան. Էլեկտրոնային արդյունաբերական սարքավորումների տրամաբանական նախագծում: Ուսումնական ձեռնարկ: ԵՐՊԻ հրատարակչություն, 1989թ., 103 էջ:
17. Վ. Ս. Մովսիսյան. Թվային էլեկտրոնային սարքավորումների տրամաբանական նախագծում: Ուսումնական ձեռնարկ, մաս 2: ԵՐՊԻ հրատարակչություն, 1994թ., 90 էջ:

Վիլյամ Մկրտչի Մովսիսյան

## Թվային համակարգերի տրամաբանական նախագծում

Դասագիրք

Вильям Мкртычевич Мовсисян

## Логическое проектирование цифровых систем

Учебник

Խմբագիր՝ Ն. Խաչատրյան

Ստորագրված է տպագրության՝ 17.04.2012

Թուղթը՝ «օֆսեթ»: Տպագրությունը՝ ռիզո, Փորմատ՝ (60×84) 1/8:

Շարվածքը՝ համակարգչային:

Տառատեսակը՝ Arial Armenian: 58.5 տպ. մամ.:

Պատվեր՝ 73 Տպաքանակ՝ 200

*Հայաստանի Պետական*

*Ճարտարագիտական*

*Համալսարանի տպարան*

Երևան, Տերյան 105 Հեռ. 52-03-56

*Типография Государственного Инженерного*

*Университета Армении*

Ереван, ул. Теряна 105 Тел.: 52-03-56