

ՀՀ ԿՐԹՈՒԹՅԱՆ ԵՎ ԳԻՏՈՒԹՅԱՆ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ

ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

ԱՐՄԱՆ ՌԱՖԻԿԻ ՄԱՐՏԻՐՈՍՅԱՆ

ԻՆՏԵԳՐԱԼ ՍԽԵՄԱՆԵՐԻ ԱՎՏՈՄԱՏԱՑՎԱԾ ՖԻԶԻԿԱԿԱՆ
ՆԱԽԱԳԾՈՒՄԸ ՍԽԵՄԱՏԵԽՆԻԿԱԿԱՆ ԱՌԱՆՁՆԱՀԱՏԿՈՒԹՅՈՒՆՆԵՐԻ
ՀԱՇՎԱՌՄԱՄԲ

ԱՏԵՆԱԽՈՍՈՒԹՅՈՒՆ

Ե.13.02 «Ավտոմատացման համակարգեր» մասնագիտությամբ
տեխնիկական գիտությունների թեկնածուի գիտական աստիճանի համար

Գիտական ղեկավար՝ տ.գ.դ., դոցենտ Ա.Գ. Հարությունյան

ԵՐԵՎԱՆ 2019

Բովանդակություն

Ներածություն.....	5
ԳԼՈՒԽ 1. ԺԱՄԱՆԱԿԱԿԻՑ ՆԱՆՈՉԱՓԱԿԱՆ ԻՍ-ԵՐԻ ՖԻԶԻԿԱԿԱՆ ՆԱԽԱԳԾՄԱՆ ՍԱՐՏԱՀՐԱՎԵՐՆԵՐԸ ԵՎ ԴՐԱՆՑ ԱԶԴԵՑՈՒԹՅՈՒՆԸ ՍԽԵՄԱՏԵԽՆԻԿԱԿԱՆ ԵՎ ՖԻԶԻԿԱԿԱՆ ՆԱԽԱԳԾՄԱՆ ՓՈԽԿԱՊԱԿՑՎԱԾՈՒԹՅԱՆ ՎՐԱ	13
1.1. Ժամանակակից նանոչափական ԻՍ-երի ավտոմատացված ֆիզիկական նախագծման մարտահրավերն երբ.....	13
1.2. ԻՍ-երի մասշտաբավորումը և նախագծման ֆիզիկական ու սխեմատեխնիկական փուլերի միջև փոխկապվածության ուժեղացումը	23
1.3. ԻՍ-երի նախագծման ֆիզիկական և տրամաբանական մակարդակների փոխկապվածությունը.....	28
1.4. Խնդրի դրվածքը.....	39
Եզրակացություններ	42
ԳԼՈՒԽ 2. ՍԽԵՄԱՏԵԽՆԻԿԱԿԱՆ ԱՌԱՆՁՆԱՀԱՏԿՈՒԹՅՈՒՆՆԵՐԻ ՀԱՇՎԱՌՄԱՍԲ ԻՆՏԵԳՐԱԼ ՍԽԵՄԱՆԵՐԻ ԱՎՏՈՄԱՏԱՑՎԱԾ ՖԻԶԻԿԱԿԱՆ ՆԱԽԱԳԾՄԱՆ ՄԵԹՈԴՆԵՐԻ ՄՇԱԿՈՒՄԸ	43
2.1. ԻՍ-երի տարրերի ժամանակային ուղղորդմամբ տեղաբաշխումը՝ որպես սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ ֆիզիկական կառուցվածքի ստացման միջոց	43
2. 2. Տրամաբանական սխեմայի Verilog նկարագրից ազդանշանների հոսքի գրաֆի ստացման ավտոմատացված միջոցի մշակումը.....	46
2.3. Սխեմայի տարրերի ռանգավորումը՝ որպես սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ ֆիզիկական կառուցվածքի ստացման միջոց	50
2.4. Ինտեգրալ սխեմաների ֆիզիկական կառուցվածքների ներկայացումը «փափուկ» և «կոշտ» բլոկների տեսքով	55

2.5. Տրամաբանական տարրերի փոխանջատման ակտիվությունների միջոցով դիմամիկ հզորությունների գնահատումը՝ որպես սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ ֆիզիկական կառուցվածքի ստացման միջոց	62
Եզրակացություններ	64
ԳԼՈՒԽ 3. ՄԽԵՄԱՏԵԽՆԻԿԱԿԱՆ ԱՌԱՆՁՆԱՀԱՏԿՈՒԹՅՈՒՆՆԵՐԻ ՀԱՇՎԱՌՄԱՍԲ ԻՆՏԵԳՐԱԼ ՄԽԵՄԱՆԵՐԻ ՖԻԶԻԿԱԿԱՆ ՆԱԽԱԳԾՄԱՆ ԱՎՏՈՄԱՏԱՑՄԱՆ ԵՆԹԱՀԱՄԱԿԱՐԳԻ ԱԼԳՈՐԻԹՄԱԿԱՆ ԵՎ ԾՐԱԳՐԱՅԻՆ ԱՊԱՀՈՎՈՒՄԸ	
66	
3.1. Սխեմայի աշխատանքի տրամաբանության հաշվառմամբ ԻՍ-ի ավտոմատացված ֆիզիկական նախագծման հիմնական փուլերը	66
3.2. Տրամաբանական սխեմայի verilog նկարագրի փոխարկումը ազդանշանների հոսքի գրաֆի.....	68
3.3. Տրամաբանական սխեմայի ռանգավորումը	76
3.4. Թվային ինտեգրալ սխեմաների տրամաբանական նկարագրությունից «փափուկ» ֆիզիկական կառուցվածքի բլոկների սինթեզը	80
3.5. «Փափուկ» ֆիզիկական կառուցվածքի բլոկների փոխակերպումը «կոշտ» բլոկների.....	81
3. 6. Թվային ինտեգրալ սխեմաների տարրերի տեղաբաշխումը՝ դիմամիկ հզորությունների հաշվառմամբ	86
Եզրակացություններ	91
ԳԼՈՒԽ 4. ՄՇԱԿՎԱԾ ԱՎՏՈՄԱՏԱՑՎԱԾ ԵՆԹԱՀԱՄԱԿԱՐԳԻ ԾՐԱԳՐԱՅԻՆ ԻՐԱԳՈՐԾՈՒՄԸ ԵՎ ՀԵՏԱԶՈՏՈՒԹՅԱՆ ԱՐԴՅՈՒՆՔՆԵՐԸ	
93	
4.1. Սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ ինտեգրալ սխեմաների ֆիզիկական կառուցվածքի ավտոմատացված նախագծման ծրագրային միջոցի նկարագրությունը և փորձնական հետազոտության արդյունքները.....	93
Եզրակացություններ	107

ԷԶՐԱՀԱՆԳՈՒՄ	108
ՕԳՏԱԳՈՐԾՎԱԾ ԳՐԱԿԱՆՈՒԹՅԱՆ ՑԱՆԿ.....	110
ՀԱՎԵԼՎԱԾ 1.....	117
ՀԱՎԵԼՎԱԾ 2.....	118
ՀԱՎԵԼՎԱԾ 3.....	134
ՀԱՎԵԼՎԱԾ 4.....	138
ՀԱՎԵԼՎԱԾ 5.....	139

Ներածություն

Թեմայի արդիականությունը: Ինտեգրալ սխեմաների (ԻՍ) զարգացումը բնութագրվում է դրանց տեխնոլոգիական մասշտաբավորմամբ, որը հանգեցնում է ԻՍ-երի հիմնական պարամետրերի փոխկապվածությունների զգալի աճին: Դրա հետևանքով ԻՍ-երի նախագծման ժամանակ կարևորվում է սխեմատեխնիկական և ֆիզիկական պարամետրերի միջև փոխկապվածությունների հաշվի առնելը: Թվային ԻՍ-երի ֆիզիկական նախագծման փուլում, այդ փոխկապվածություններից, սխեմատեխնիկական առանձնահատկությունների հաշվի առնելու տեսանկյունից, առավել կարևորվում են միջմիացումներում առաջացող հապաղումներով պայմանավորված արագագործության անկումը և սխեմայի տրամաբանությամբ պայմանավորված տարրերի աշխատանքային հզորությունների ազդեցությունը ջերմային հուսալիության վրա:

Այսպիսով, միկրոէլեկտրոնային տեխնոլոգիայի զարգացմանը զուգընթաց՝ աճում է ԻՍ-երի ֆիզիկական նախագծման այնպիսի ավտոմատացված միջոցների մշակման կարևորությունը, որոնք կապահովեն սխեմատեխնիկական առանձնահատկությունները հաշվի առնող և սխեմային ներկայացվող ֆունկցիոնալ-տրամաբանական պահանջները բավարարող ֆիզիկական կառուցվածքների ստացումը:

Ինչպես հայտնի է, ԻՍ-երում տեղեկույթի մշակման ցիկլի ժամանակը ձևավորվում է սխեմայի տարրերում և միջմիացումներում հապաղումներից: Եթե տարրերի հապաղումները հայտնի են դառնում սխեմատեխնիկական նախագծման փուլում, ապա միջմիացումներում հապաղումները որոշվում են միայն ֆիզիկական նախագծման արդյունքում, այսինքն՝ ծրագծումից հետո: Սակայն, հաշվի առնելով, որ ծրագծման արդյունքները զգալիորեն կախված են տարրերի տեղաբաշխումից, ակնհայտ է դառնում տեղաբաշխման փուլում ազդանշանի հապաղումների հաշվառման կարևորությունը: Հապաղումները միջմիացումներում, առաջին մոտեցմամբ, որոշվում են կապի ուղու RC պարամետրերով, որոնք, իրենց հերթին, նախագծման ընթացքում հիմնականում կախված են միջմիացումների երկարություններից:

Միաժամանակ ենթադրվում է, որ տեղաբաշխվող տարրերի միջև մեծ տոպոլոգիական հեռավորությանը ծրագծման ժամանակ կհամապատասխանեն ավելի

երկար միջմիացումներ և, հետևաբար, ավելի մեծ հապաղումներ, և՛ հակառակը: Միջմիացումներում հապաղումների փոքրացման և դրանով ԻՍ-ի արագագործության մեծացման նպատակով ներկայումս ավտոմատացված նախագծման համակարգերում (ԱՆՀ) ներդրվում են տարրերի տեղաբաշխման միջոցներ, որոնք հաշվի են առնում հապաղումները շղթաներում: Տեղաբաշխման այդպիսի մոտեցումը ընդունված է անվանել ժամանակային ուղղորդմամբ տեղաբաշխում: Այսպիսով, տարրերի տեղաբաշխման ժամանակ միջմիացումներում հապաղումների հաշվառման խնդրի լուծումը հիմնված է սխեմայի մուտք-ելք ուղիների ժամանակային վերլուծության և ժամանակով ուղղորդված տեղաբաշխման մեթոդների կիրառման վրա: Ժամանակով ուղղորդված տեղաբաշխման գոյություն ունեցող մեթոդները կարող են խմբավորվել երկու դասով՝ ուղու վրա հիմնված և շղթայի վրա հիմնված:

Ուղու վրա հիմնված մոտեցումը ենթադրում է սխեմայի մուտք-ելք որոշիչ ուղիների հայտնաբերում և տարրերի այնպիսի տեղաբաշխում և վերատեղաբաշխում, որպեսզի որոշիչ ուղիների վրա գտնվող տարրերը տեղաբաշխվեն միմյանց հնարավորինս մոտ: Սա արդյունաբերական ԱՆՀ-երում ներկայումս օգտագործվող հիմնական մոտեցումն է: Այս մոտեցման թերությունն այն է, որ սխեմայի որոշիչ ուղիների վրա հապաղումների նվազարկումը հաճախ հանգեցնում է նոր որոշիչ ուղիների առաջացման՝ էլ ավելի մեծ հապաղումներով:

Շղթայի վրա հիմնված մեթոդները ենթադրում են յուրաքանչյուր շղթայի երկարության կառավարում անմիջականորեն տարրերի տեղաբաշխման ընթացքում՝ ըստ սխեմայի արագագործության վրա դրանց ազդեցության չափի: Այս մոտեցման թերությունը տեղաբաշխման փուլում շղթաների երկարությունների կառավարման և ծրագրային իրագործման բարդությունն է:

Նշված երկու դասի մեթոդների ընդհանուր թերությունն այն է, որ նրանք հաշվի չեն առնում թվային սխեմաների աշխատանքի ֆունկցիոնալ-տրամաբանական առանձնահատկությունը, ինչը առանձնապես կարևորվում է ժամանակակից նանոչափային ԻՍ-երի ֆիզիկական նախագծման ժամանակ:

Նշված թերության հաղթահարման նպատակով այս ատենախոսությունում առաջարկված և մշակված է տրամաբանական բլոկների ֆիզիկական կառուցվածքի

սինթեզի նոր մոտեցում, որը հիմնված է տրամաբանական սխեմայի ռանգավորման վրա և սխեմայի աշխատանքի տրամաբանության հաշվառման արդյունքը հանգեցնում է միջմիացումների երկարությունների և, հետևաբար, դրանցում հապաղումների նվազեցմանը՝ վերջնարդյունքում նպաստելով սխեմայի արագագործության բարձրացմանը:

Մյուս կողմից՝ սխեմատեխնիկական առանձնահատկությունների տեսանկյունից կարևորվում է ԻՄ-ի կիսահաղորդչային բյուրեղի վրա ցրվող տեսակարար հզորության զգալի աճը, որը հանգեցնում է տեղային բարձր ջերմաստիճանային գոտիների առաջացմանը և դրա հետևանքով՝ ԻՄ-ի տակդիրի ջերմաստիճանային գրադիենտի աճին: Ժամանակակից ԻՄ-երում կիսահաղորդչային բյուրեղի աշխատանքային ջերմաստիճանը կարող է հասնել ավելի քան 100°C , իսկ բյուրեղների տարբեր շերտերի միջև ջերմաստիճանի տարբերությունը գերազանցում է $10\text{--}20^{\circ}\text{C}$: Սա հանգեցնում է ԻՄ-ի ջերմային հուսալիության կտրուկ նվազմանը, քանի որ Բլեյքի օրենքի համաձայն՝ ջերմային հուսալիությունը էքսպոնենտային կախվածություն ունի ջերմաստիճանից:

Այս խնդրի լուծման առավել արդյունավետ եղանակներից են ԻՄ-ի ֆիզիկական նախագծման այնպիսի մեթոդների և ալգորիթմների մշակումն ու օգտագործումը, որոնք կապահովեն ԻՄ-ի տակդիրի ջերմային դաշտի համահարթեցումը՝ դրանով իջեցնելով բարձր ջերմաստիճանով տեղային հատվածների ջերմաստիճանը և բարձրացնելով ԻՄ-ի հուսալիությունը: Քանի որ այդ մեթոդներում վճռական դեր է խաղում տարրերի սպառած հզորությունը, ապա թվային սխեմաների նախագծման տրամաբանական և ֆիզիկական մակարդակների փոխադարձ կապը հաշվի առնելու տեսակետից կարևոր խնդիր է հանդիսանում տրամաբանական տարրերի սպառած աշխատանքային հզորության գնահատումը: Խնդիրն այն է, որ առանձին ինչ-որ տրամաբանական տարրի սպառած աշխատանքային հզորությունը կարող է փոփոխվել լայն սահմաններում՝ կախված այդ տրամաբանական սխեմայի արագագործությունից և կառուցվածքից, որը ներառում է տվյալ տարրը, այդ սխեմայում տվյալ տարրի տեղից և մշակվող տեղեկատվության բնույթից:

Ելնելով վերոհիշյալից՝ ԻՄ-ի տրամաբանական նախագծումից ֆիզիկականին անցնելիս, կարևոր է հաշվի առնել տրամաբանական տարրերի աշխատանքային

հզորությունների արժեքները ԻՄ-ի տակդիրի վրա դրանք տեղադրելու ժամանակ: Տրամաբանական տարրերի հզորության դինամիկ բաղադրիչը գնահատելիս անհրաժեշտ է հաշվի առնել տվյալ տարրի ոչ միայն տեսակն ու բնութագրող մեծությունները, այլ նաև սխեմայում տվյալ տարրի փոխանջատման արագությունը: Այսպիսով, ֆիզիկական նախագծման ընթացքում տրամաբանական սխեմայի աշխատանքի առանձնահատկությունների տեսանկյունից, տրամաբանական տարրերի աշխատանքային հզորություններից ելնելով, արդիական է դառնում դրանց տեղաբաշխման մեթոդների մշակումը:

Նշված խնդիրների հաղթահարման նպատակով առաջարկվում են տրամաբանական բլոկների ֆիզիկական կառուցվածքի սինթեզի նոր մոտեցումներ, որոնցից առաջինը հիմնված է տրամաբանական սխեմայի ռանգավորման ճանապարհով ստացված ֆիզիկական «փափուկ» բլոկից «կոշտ» բլոկի ձևափոխման վրա և միտված է միջմիացումների երկարությունների և, հետևաբար, դրանցում հապաղումների նվազեցմանը: Երկրորդ մոտեցումը, հիմնված է ԻՄ-երի տարրերի տեղաբաշխման փուլում դրանց դինամիկ հզորությունների գնահատման և հաշվառման վրա, որի համար օգտագործվել է միայն հզորությունների դինամիկ բաղադրիչը:

Հետազոտության առարկան: Սխեմատեխնիկական առանձնահատկությունների և նանոչափական տեխնոլոգիաներին բնորոշ մարտահրավերների հաշվառմամբ՝ ԻՄ-երի ավտոմատացված ֆիզիկական նախագծման մեթոդական, ալգորիթմական և ծրագրային ապահովման միջոցները:

Աշխատանքի նպատակը: Սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ (արագագործություն և դինամիկ հզորություն)՝ թվային ԻՄ-երի ֆիզիկական կառուցվածքների ավտոմատացված սինթեզի (տարրերի տեղաբաշխման) արդյունավետ մեթոդների, ալգորիթմների և ծրագրային գործիքների մշակումը և հետազոտումը:

Հետազոտման մեթոդները: Ատենախոսության կատարման ընթացքում օգտագործվել են թվային ԻՄ-երի տրամաբանական վերլուծության, ստատիկ

Ժամանակային վերլուծության, տարրերի ժամանակային և ջերմային տեղաբաշխման մեթոդները, գրաֆների և հավանականության տեսությունները, ինչպես նաև ավտոմատացված համակարգի ծրագրային ապահովման՝ օբյեկտակոդմնորոշված և QT գրադարանի միջոցով մոդելավորման մեթոդները:

Գիտական նորույթ: Կատարված համալիր հետազոտության ընթացքում ստացվել են հետևյալ արդյունքները, որոնք առանձնանում են գիտական նորույթով.

- Մշակվել են ԻՄ-երի ավտոմատացված նախագծման գործընթացում թվային սխեմաների ֆիզիկական կառուցվածքի սինթեզի նոր մեթոդ և համապատասխան ծրագրային ապահովում՝ հիմնված տրամաբանական սխեմայի ռանգավորման, գրաֆներում ըստ լայնության փնտրման ճանապարհով ֆիզիկական «փափուկ» բլոկի ստացման և մոդիֆիկացված բարիկենտրոնային ալգորիթմի կիրառմամբ ֆիզիկական «կոշտ» բլոկի ձևափոխման վրա, ինչը հանգեցնում է ֆիզիկական նախագծման փուլում սխեմատեխնիկական առանձնահատկությունների հաշվառման արդյունավետության բարձրացմանը՝ միջմիացումների երկարությունների կրճատման, դրանցում հապաղումների նվազեցման և, հետևաբար, սխեմայի արագագործության բարձրացման շնորհիվ:
- Առաջարկվել է ԻՄ-երի ավտոմատացված նախագծման գործընթացում ֆիզիկական «կոշտ» բլոկների սինթեզի նոր մեթոդ՝ հիմնված շղթաների ժամանակային կարևորության և բարիկենտրոնային մոդիֆիկացված ալգորիթմի կիրառման վրա, ինչը նպաստում է սխեմայի աշխատանքի տրամաբանությանը համապատասխան ֆիզիկական կառուցվածքի ձևավորմանը:
- Առաջարկվել է տրամաբանական տարրերի աշխատանքային դինամիկ հզորությունների գնահատման և դրանց հիման վրա ջերմային տեղաբաշխման ավտոմատացման միջոց, որը թույլ է տվել, ի տարբերություն գոյություն ունեցողների, հաշվի առնել սխեմայի աշխատանքի տրամաբանությունը, որը, իր հերթին, հանգեցնում է սխեմայի ջերմային հուսալիության բարձրացմանը:

Պաշտպանության ներկայացվող դրույթները.

- ԻՄ-երի փականային մակարդակի Verilog նկարագրությունից ազդանշանների հոսքի գրաֆի (ԱՀԳ) սինթեզման ավտոմատացված միջոցը, որը հիմնված է տվյալների բինար որոնման բալանսավորված ծառի տեսքով ներկայացման և դրանում փնտրման կազմակերպման վրա:
- Թվային ԻՄ-երի տարրերի ռանգավորման ավտոմատացված միջոցը՝ հիմնված ԱՀԳ-ի բինար բալանսավորված ծառի օգտագործման վրա:
- ԻՄ-երի ֆիզիկական «փափուկ» բլոկների ստացման մեթոդ՝ հիմնված ԱՀԳ-ի մոդելի, ռանգավորման և ըստ լայնության փնտրման ալգորիթմների վրա:
- ԻՄ-երի ֆիզիկական «կոշտ» բլոկների ստացման մեթոդը և գործիքամիջոցը՝ հիմնված ֆիզիկական «փափուկ» բլոկի ձևափոխման և մոդիֆիկացված բարիկենսորոնային ալգորիթմի վրա:
- ԻՄ-երի ջերմային դաշտի համահարթեցման մեթոդը և գործիքամիջոցը՝ հիմնված տրամաբանական տարրերի աշխատանքային դինամիկ հզորությունների արժեքների գնահատման և օգտագործման վրա:
- Սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ ԻՄ-երի ֆիզիկական նախագծման ավտոմատացման ծրագրային համալիր, որը ներառում է երկու ծրագրային միջոցներ՝ Verilog նկարագրից ֆիզիկական կառուցվածքի «փափուկ» ու «կոշտ» բլոկների ավտոմատացված սինթեզի SHBM ծրագրային միջոցը և տրամաբանական տարրերի աշխատանքային դինամիկ հզորությունների գնահատման ու դրանց հիման վրա ջերմային տեղաբաշխման ծրագրային միջոցը:

Աշխատանքի գործնական արժեքը: Ատենախոսությունում առաջարկված հիմնական բոլոր մեթոդները և տեսական արդյունքներն իրագործվել են սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ՝ ԻՄ-երի ֆիզիկական նախագծման ավտոմատացման երկու ծրագրային գործիքների տեսքով: Դրանցից առաջին՝ SHBM ծրագրային գործիքը՝ հնարավորություն է տալիս ԻՄ-երի փականային մակարդակի Verilog նկարագրությունից ավտոմատացված ձևով սինթեզել ֆիզիկական «փափուկ» և «կոշտ» կառուցվածքով բլոկներ, որոնք հաշվառում են սխեմայում

ազդանշանային հոսքերի առանձնահատկությունները և նպաստում են սխեմայի արագագործության բարձրացմանը: Երկրորդ ծրագրային գործիքը նախատեսված է ԻՍ-երի տարրերի ջերմային տեղաբաշխման համար, որը հաշվի է առնում տարրերի աշխատանքային դինամիկ հզորությունները՝ կախված տարրերի տիպից և սխեմայում նրանց աշխատանքի տրամաբանությունից:

Մշակված ծրագրային գործիքները նախագծվել են C++17, Qt5.12 միջավայրերում և նախատեսված են GNU/Linux և Windows օպերացիոն համակարգերում օգտագործման համար:

Ստեղծված գործիքը կարող է օգտագործվել ինչպես տեղաբաշխման ինքնուրույն միջոցի տեսքով, այնպես էլ ինտեգրվել գոյություն ունեցող ԱՆՀ-երում՝ որպես տարրերի տեղաբաշխման ենթահամակարգ:

Գիտական դրույթների հավաստիությունը հիմնավորված է մշակված մեթոդների, ալգորիթմների և ավտոմատացման ծրագրային գործիքային միջոցների տեսական հիմնավորման և կիրառման արդյունքներով՝ մի շարք թեստային սխեմաների տարրերի տեղաբաշխման և դրանց արդյունքների հետագա համեմատությամբ ինչպես ավանդական մեթոդների, այնպես էլ կոմերցիոն ԱՆՀ - ով ստացված արդյունքների հետ:

Ներդրումը: Մշակված «Soft and Hard block maker» (SHBM) ծրագրային գործիքը ներդրված է «ՍԻՆՈՓՄԻՍ ԱՐՄԵՆԻԱ» ՓԲԸ-ում: Այն օգտագործվում է ԻՍ-երի ֆիզիկական նախագծման ընթացքում տարրերի տեղաբաշխման համար՝ սխեմատեխնիկական ժամանակային առանձնահատկությունների հաշվառմամբ:

Ատենախոսության փորձահավանությունը: Զեկուցումներ գիտաժողովներում.

- ՀԱՊՀ (Պոլիտեխնիկ) պրոֆեսորադասախոսական կազմի տարեկան գիտատեխնիկական գիտաժողովներ (Երևան, Հայաստան, 2016),
- ՀԱՊՀ (Պոլիտեխնիկ) պրոֆեսորադասախոսական կազմի տարեկան գիտատեխնիկական գիտաժողովներ (Երևան, Հայաստան, 2018),
- IEEE «Էլեկտրոնիկա եւ նանոտեխնոլոգիաներ» 39-րդ միջազգային գիտաժողով (ELNANO), (Կիև, Ուկրաինա, 2019)

Հրապարակումները: Ատենախոսության հիմնական դրույթները ներկայացված են 6 գիտական հրապարակումներում:

Ատենախոսության կառուցվածքը և ծավալը: Ատենախոսությունը բաղկացած է ներածությունից, չորս գլուխներից, հիմնական եզրահանգումից, 91 անուն գրականության ցանկից՝ հայերեն, ռուսերեն և անգլերեն լեզուներով, 5 հավելվածներից՝ (առաջին հավելվածում բերված է ներդրման ակտը, երկրորդում՝ ծրագրի կողը, երրորդում, չորրորդում և հինգերորդում՝ ատենախոսությունում պարունակվող նկարների, աղյուսակների և հապավումների ցանկերը): Ատենախոսությունը ներառում է 43 նկ. և 8 աղյուսակ: Հիմնական տեքստը կազմում է 116 էջ: Ատենախոսությունը գրված է հայերեն:

**ԳԼՈՒԽ 1. ԺԱՄԱՆԱԿԱԿԻՑ ՆԱՆՈՉԱՓԱԿԱՆ ԻՍ-ԵՐԻ ՖԻԶԻԿԱԿԱՆ
ՆԱԽԱԳԾՄԱՆ ՄԱՐՏԱՀՐԱՎԵՐՆԵՐԸ ԵՎ ԴՐԱՆՑ ԱԶԴԵՑՈՒԹՅՈՒՆԸ
ՍԽԵՄԱՏԵԽՆԻԿԱԿԱՆ ԵՎ ՖԻԶԻԿԱԿԱՆ ՆԱԽԱԳԾՄԱՆ
ՓՈԽԿԱՊԱԿՑՎԱԾՈՒԹՅԱՆ ՎՐԱ**

**1.1. Ժամանակակից նանոչափական ԻՍ-երի ավտոմատացված ֆիզիկական
նախագծման մարտահրավերն երբ**

Ինտեգրալ սխեմաների (ԻՍ) զարգացումը բնորոշվում է դրանց արտադրության տեխնոլոգիայի մասշտաբավորմամբ, ինչը ենթադրում է լիտոգրաֆիկական թույլատրելի չափսերի անընդհատ նվազեցում: Դրա հետևանքով տեղի է ունենում ԻՍ-երի ինտեգրման աստիճանի անընդհատ աճ: Տեխնոլոգիական մասշտաբավորումը զգալի ազդեցություն է գործում ԻՍ-երի հիմնական բնութագրերի վրա: ԻՍ-ի բարդության աճի և տեխնոլոգիական չափսերի փոքրացմանը զուգընթաց՝ ի հայտ են գալիս մի շարք նոր իրողություններ, որոնք պահանջում են նախագծման նոր և լավարկված այնպիսի միջոցների մշակում, որոնք կապահովեն ինչպես նախագծվող ԻՍ-երի որակի, այնպես էլ դրանց նախագծման գործընթացի տեխնիկա-տնտեսական ցուցանիշների անընդհատ բարելավում:

Միկրոէլեկտրոնային արդյունաբերության ասպարեզում գործող ընկերությունների զարգացումը և առաջընթացն ունի առանձնահատկություն. այն պայմանավորված է զարգացման մանրամասն մշակված պլանով, ինչը, մի կողմից՝ նպաստում է տարբեր ընկերությունների մրցունակության բարձրացմանը, իսկ մյուս կողմից՝ փոխադարձ լրացմանը: Սրան նպաստում է կիսահաղորդիչների միջազգային տեխնոլոգիական ճանապարհային քարտեզը, որը մշակվում է “Կիսահաղորդչային արդյունաբերության ասոցիացիա” միջազգային կազմակերպության կողմից [1,2]: Համաձայն կիսահաղորդիչների միջազգային ճանապարհային քարտեզի կանխատեսումների՝ ԻՍ-երի ֆիզիկական նախագծմանն առնչվող հիմնական պարամետրերի էվոյուցիան ներկայացված է ստորև բերված աղ. 1. 1-ում [2,3]:

ԻՄ-երի ֆիզիկական նախագծմանն առնչվող հիմնական պարամետրերի զարգացումը

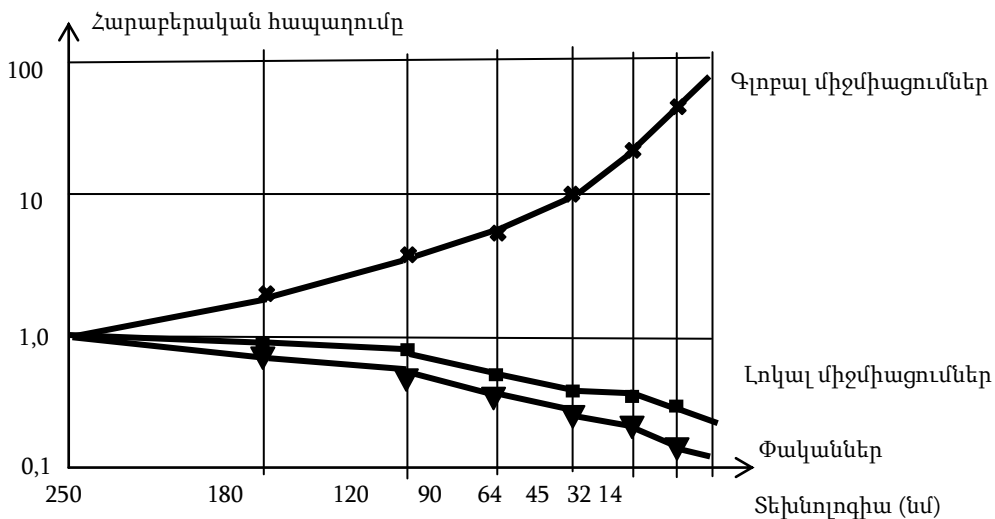
Տարի	2009	2012	2015	2018	2021	2024	2030
Փականի երկարությունը(նմ)	54	35	22	15,7	11,1	7,9	5
Բյուրեղի մակերեսը(մմ ²)	750	750	750	750	750	750	750
Միջին տեսակարար էներգասպառումը(Վտ/մմ ²)	0,45	0,6	0,75	0,9	1,05	1,2	1,4
Միջուկի սնուցման նվազագույն լարումը(Վ)	3,3	1,2	0,8	0,7	0,65	0,55	0,4
Արագագործությունը(ԳՀց)	5,45	6,82	8,52	10,65	13,32	15,41	18
Միջմիացման տեսակարար դիմադրությունը (Օմ/մկմ)	15	25	40	130	800	3000	3000
Միջմիացման տեսակարար ունակությունը (աՖ/մկմ)	215	210	200	190	180	180	180
դյուրակիր սարքերի դեպքում p-n անցման առավելագույն ջերմաստիճանը(°C)	105	105	105	105	105	105	105
Շահագործման առավելագույն ջերմաստիճանային միջակայքը(°C)	-40... 150	-40... 150	-40... 150	-40... 150	-40... 150	-40... 150	-40... 150
Օդի օպտիմալ հոսքի դեպքում ԻՄ-ի իրանի շրջապատի ջերմաստիճանը(°C)	45	45	45	45	45	45	45

Հաշվի առնելով աղյուսակում բերված տվյալները, ինչպես նաև ԻՄ-երի արդի իրողություններն արտացոլող գրականությունը, կարելի է առանձնացնել

Ժամանակակից նանոչափական ԻՄ-երի ավտոմատացված ֆիզիկական նախագծմանն առնչվող խոր ենթամիկրոնային տեխնոլոգիաներին բնորոշ հետևյալ հիմնական իրողությունները և դրանց հաղթահարմանը միտված ֆիզիկական նախագծման խնդիրները [2-6]:

1. Գերբարձր արագագործությունը և միջմիացումների աճող կարևորությունը

Ինչպես հայտնի է, ԻՄ-երի տեխնոլոգիական չափսերի փոքրացմանը և ինտեգրման մեծացմանը զուգընթաց՝ միջմիացումներով պայմանավորված հապաղումների մասն աճում է և ներկայումս հասնում է ընդհուպ մինչև ազդանշանի գումարային հապաղման 80...90% -ին: Այդ հապաղումները որոշվում են միջմիացումների RC պարամետրերով, որոնք, որոշակի տեխնոլոգիայի դեպքում, իրենց հերթին, հիմնականում կախված են միջմիացումների երկարություններից: Այդ երկարությունները որոշվում են միջմիացումների ծրագծումից հետո, որի արդյունքները զգալիորեն կանխորոշվում են տարրերի տեղաբաշխմամբ: Արագագործության ապահովման տեսանկյունից՝ ներկայումս ԻՄ-երի ֆիզիկական նախագծման կարևորագույն խնդիրներից է տարրերի ժամանակային ուղղորդմամբ տեղաբաշխումը, և ԻՄ-երի էլեկտրոնային նախագծման ավտոմատացման բոլոր զարգացած համակարգերում ներդրվում են համապատասխան գործիքային միջոցներ [3,7-9]: Այս երևույթը պատկերավոր ներկայացված է նկ. 1.1-ում [10]:



Նկ.1.1. ԻՄ-ի տրամաբանական փականներում և միջմիացումներում ազդանշանի հապաղումների էվոլյուցիան

Միջմիացումներում առաջացող հապաղումները կախված են նրանց պարագիտային պարամետրերից, իսկ վերջիններս, իրենց հերթին, կախում ունեն միջմիացումների չափերից՝ հիմնականում երկարություններից: Տարրերի տեղաբաշխմամբ կանխորոշվում են միջմիացումների ծրագծման արդյունքները, որոնցով որոշվում են նրանց երկարությունները:

2. Մեծ տեսակարար սպառող հզորությունը

Մեծ տեսակարար սպառող հզորությունը հանգեցնում է ԻՄ-երի աշխատանքային ջերմաստիճանի մինչև 100°C և ավելի բարձրացման, իսկ կիսահաղորդչային բյուրեղի տարբեր տեղամասերի միջև ջերմաստիճանների տարբերությունը հասնում է ավելի քան 20°C : Սա կտրուկ իջեցնում է ԻՄ-ի ջերմային հուսալիությունը, որը համաձայն Բլեքի օրենքի՝ ջերմաստիճանից ունի էքսպոնենտային կախվածություն [11-13]:

Ժամանակակից նանոչափային ԻՄ-երի էներգասպառումը կարող է հասնել մինչև 150 Վտ-ի, իսկ տեսակարար էներգասպառումը՝ մինչև 0,8 Վտ/մմ²-ի [1, 2]: Կանխատեսվում է տեսակարար էներգասպառման աճ՝ մինչև 1,35 Վտ/մմ² [6]: Այս պայմաններում ԻՄ-ի պատյանի ներսում ջերմաստիճանը կարող է հասնել 100°C և ավելի, ինչը հանգեցնում է հուսալիության կտրուկ անկման:

Այս խնդրի լուծման արդյունավետ ճանապարհներից մեկը ԻՄ-երի ֆիզիկական նախագծման այնպիսի մեթոդների և ալգորիթմների մշակումն ու օգտագործումն է, որոնք կապահովեն կիսահաղորդչային բյուրեղի ամբողջ մակերեսով ջերմային դաշտի համահարթեցումը՝ դրանով իսկ նվազեցնելով ամենատաք տեղամասերի ջերմաստիճանները և բարձրացնելով ԻՄ-երի հուսալիությունը [3,14,15]:

3. Միջմիացումների աճող մակերեսը: Գնահատված է, որ միկրոպրոցեսորի բյուրեղի մակերեսը ակտիվ սարքերով ծածկված է ընդամենը 60%-70%-ով [3,7]: Մնացած մակերեսը անհրաժեշտ է միջմիացումները տեղավորելու համար: Այսպիսի համամասնությունը հանգեցնում է արագագործության և արդյունավետության անկմանը: Վաղ ԻՄ-երում հաղորդակցվում էին մի քանի հարյուր տրանզիստորներ՝ օգտագործելով մետաղի մեկ շերտ: Քանի որ տրանզիստորների քանակն աճել է, ուստի միջմիացումների մակերեսն ավելացել է: Սակայն մետաղի երկրորդ շերտի

ներմուծմամբ մեկ շերտի մակերեսը նվազում է՝ մետաղական շերտերի միջև հաղորդակցման կազմակերպման անհրաժեշտությունից ելնելով: Սա միտում է նախագծման բարդության և մետաղե շերտերի քանակի միջև: Ժամանակակից նախագծերում, մոտավորապես հարյուր միլիոն տրանզիստորներով և չորսից վեց մետաղե շերտերով ԻՍ-երի կիսահաղորդչային բյուրեղներում, մակերեսի մոտավորապես 40%-ը բաժին է ընկնում հաղորդակցման կազմակերպմանը: Ավելի շատ մետաղե շերտերը նվազեցնում են բյուրեղի չափերը, և պետք է նշել, որ ավելի շատ մետաղե շերտերը (շերտերի որոշակի քանակից հետո) չեն նշանակում ավելի քիչ միջմիացումների մակերես: Սա լինում է ներքևի շերտերում միջշերտային միացումներով տարածություն զբաղեցնելու պատճառով:

4. Մետաղե շերտերի աճող քանակը: Միջմիացումների քանակի աճը բավարարելու համար աճում է նաև հաղորդակցման համար առկա մետաղե շերտերի քանակը: Ներկայումս համեմատաբար պարզ նախագծերի համար հիմնականում օգտագործվում է մետաղական եռաշերտ գործընթացը, իսկ համեմատաբար բարդ՝ միկրոպրոցեսորային նախագծերի համար օգտագործվում են չորս-վեց և ավելի շերտերով գործընթացները: Արդյունքում՝ ԻՍ-երի ստուգման, միջմիացումների մոդելավորման և պարազիտային երևույթների վերլուծության նպատակներով հարկ է լինում գործ ունենալ եռաչափ կառուցվածքների հետ[3,7]:

5. Հատակագծման պլանավորման աճող պահանջները: Միջմիացումների հապաղման վրա ամենամեծ ազդեցությունն ունենում են միջմիացումներին հատկացված բյուրեղի մակերեսը և մետաղե շերտերի մեծ քանակը, որոնք էապես կախված են կիսահաղորդչային բյուրեղի հատակագծման պլանավորման արդյունավետությունից[3,7]: Եթե հաշվի առնենք նաև այն հանգամանքը, որ բազմաշերտ մետաղական միջմիացումների ապագա երկարությունները մեծապես կախված են հատակագծման պլանավորումից և տեղաբաշխումից, ապա սրանից բխող ամենակարևոր եզրակացությունն այն է, որ տարրերի հարաբերական դիրքը միմյանց նկատմամբ դառնում է որոշիչ գործոն նրանցում առաջացող հապաղումների տեսանկյունից: Հետևաբար, ֆիզիկական նախագծման քննարկումները պետք է նախագիծ մուտք գործեն ավելի վաղ փուլերում: Այսպես, օրինակ, խիստ ցանկալի է, որ

ֆունկցիոնալ և տրամաբանական նախագծման փուլում արդեն հաշվի առնվեն հատակագծման պլանավորման հարցերը: Ընդ որում, պլանավորումը պետք է հասկանալ ինչպես բլոկների, այնպես էլ դրանց միջև հաղորդակցությունների պլանավորումը: Բլոկների պլանավորումը նշանակում է հիմնական ֆունկցիոնալ բլոկների ուրվագծերի և դիրքերի որոշումը: Հաղորդակցությունների պլանավորումը նշանակում է եռաչափ տարածության մեջ ազդանշանային միջմիացումների համակարգի կառուցվածքի որոշումը: Արդեն այս փուլում անհրաժեշտ է, որ գնահատվեն հիմնական ժամանակային պարամետրերը՝ չիպի պլանի վավերականությունը հաստատելու համար: Այս պլանը պետք է օգտագործվի նախագծի հետագա փուլերում ժամանակային սահմանափակումների սահմանման համար:

ԻՄ-երի տեխնոլոգիական չափսերի փոքրացմանը զուգընթաց, մի կողմից՝ մեծանում է առանձին իրողությունների ազդեցության աստիճանը, իսկ մյուս կողմից՝ դրանք դառնում են ավելի ու ավելի փոխկապակցված: Հետևաբար, կարևորվում է նաև առանձին խնդիրների համատեղ լուծումը: Տիպային համատեղված խնդիրներից են՝

- արագագործության և սպառող հզորության փոխկապակցվածությունը,
- ժամանակային և ազդանշանի ամբողջականության փոխկապակցվածությունը
- սպառող հզորության և հուսալիության փոխկապակցվածությունը:

Նանոչափական ԻՄ-երի ավտոմատացված ֆիզիկական նախագծման տեսանկյունից՝ նախագծային առաջատար ընկերությունները կարևորում են նաև համապատասխան տեխնոլոգիաներով նախագծվող տարրային հենքի մշակումը: Մասնավորապես նշվում է նոր սերնդի գրադարանային բջիջների մշակմանն առնչվող խնդիրների կարևորությունը[16]:

Նանոչափական տեխնոլոգիաներին բնորոշ իրողությունները նոր, ընդհանրացված պահանջներ են առաջացնում նաև ԻՄ-երի ֆիզիկական նախագծման գործնառաջի կազմակերպման արդյունավետության տեսանկյունից:

Էլեկտրոնային նախագծման ավտոմատացման հիմնական համաշխարհային առաջատար ընկերությունները (Սինոփսիս, Քեդենս, Մենթոր Գրաֆիկս) իրենց վերջին հետազոտություններում ԻՄ-երի ֆիզիկական նախագծման արդյունավետության

բարձրացման հիմնական ուղին տեսնում են նախագծման այնպիսի մեթոդների, ալգորիթմների և երթուղիների ստեղծման և կիրառման մեջ, որոնք կկրճատեն նախագծման բազմակրկնությունների քանակը [17,18]:

Այսպիսով, խոր ենթամիկրոնային տեխնոլոգիաներին բնորոշ իրողությունների հաշվառմամբ ԻՄ-երի ֆիզիկական նախագծումը պահանջում է այնպիսի նոր և լավարկված սկզբունքների, մեթոդների, ալգորիթմների և ծրագրային միջոցների մշակում և հետազոտում, որոնք նպատակաուղղված կլինեն նախագծվող ԻՄ-երի որակի և նախագծման գործընթացի արդյունավետության բարձրացմանը: Այդպիսի մեթոդների մշակումը պետք է հիմնված լինի ֆիզիկական նախագծմանն առնչվող հիմնական պարամետրերի հնարավորիս վաղ դիտարկման և հաշվառման վրա, ԻՄ-երի նախագծման երթուղում: ԻՄ-երի ֆիզիկական նախագծման խնդիրների վաղաժամ դիտարկման երթուղին բերված է նկ. 1.2-ում: Նկարում ցույց է տրված ԻՄ-երի նախագծման երթուղին, որում ֆիզիկական նախագծումը նախագծման ընդհանուր շրջափուլում շատ վաղ է սկսվում, որպեսզի ապահովվի ԻՄ-ի աշխատունակությունը նախագծի ավարտական փուլերում: Ֆիզիկական նախագծման սկզբնական գործողությունները հանգեցնում են ավելի ու ավելի բարելավված պլանավորման, քանի որ յուրաքանչյուր բլոկ վերամշակվում է: Սա նաև թույլ է տալիս կիսահաղորդչային բյուրեղի մակերեսի ավելի լավ օգտագործում՝ եռաչափ միջմիացումները բաշխելու համար: Այս բաշխումը նպաստում է բյուրեղի չափսերի փոքրացմանը, աշխատանքի բարելավմանը և ծախսերի նվազեցմանը: Ըստ էության, ԻՄ-ի նախագծման շրջափուլը տալիս է ԻՄ-ի ավելի ու ավելի լավ որակներ: Յուրաքանչյուր շրջափուլի արդյունքները ստուգվում են, և եթե ստուգման արդյունքները չեն բավարարում պահանջներին, ապա փուլը կրկնվում է:

Տրամաբանական մակարդակից ֆիզիկականին անցումը: Այս գործընթացը փոխարկում է բլոկի նկարագիրը սարքավորումների նկարագրման լեզվից (HDL) նրա ֆիզիկական կառուցվածքին: Սրա իրագործման համար անհրաժեշտ է բլոկի տրամաբանական ներկայացումը տեխնոլոգիական տարրերի և նրանց միջև փոխկապակցությունների տեսքով: Այսպիսի նկարագրության օրինակ է տրամաբանական սխեմայի ներկայացումը փականային մակարդակի Verilog լեզվով, որտեղ որպես փականներ հանդես են գալիս ստանդարտ գրադարանային բջիջները:

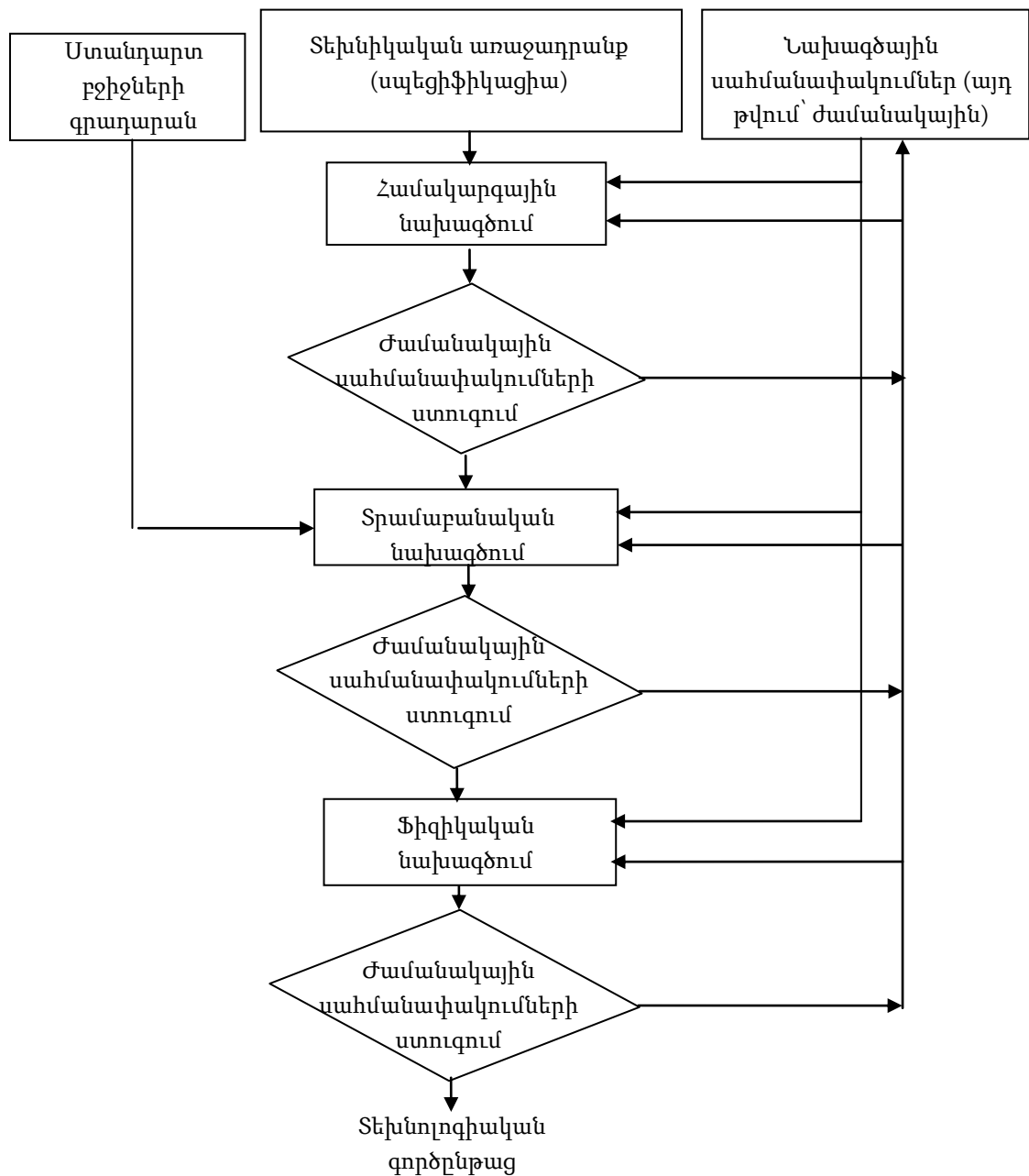
Բարձր մակարդակից ֆիզիկականին անցումը: Այս գործընթացը փոխում է ֆունկցիոնալ կամ միկրո-ճարտարապետական նկարագիրը տոպոլոգիական կամ RTL նկարագրի[3,7]: Բարձր մակարդակի սինթեզում մուտքային տվյալները ներկայացնում են նկարագիր, որն արտացոլում է համակարգի միայն վարքագծային ասպեկտները:

Ներկայումս բարձր մակարդակից ֆիզիկականին անցման գործընթացն ունի գործնական իրագործում սահմանափակ թվով կանոնավոր կառուցվածքների համար՝ ազդանշանների թվային մշակման համակարգեր, հիշող սարքեր, մոդուլի գեներատորներ և այլն: Այս սինթեզի համակարգերը երբեմն կոչվում են սիլիկոնե կոմպիլյատորներ:

Ինչպես երևաց վերը շարադրվածից, ԻՄ-երի ֆիզիկական սինթեզի փուլի կապը նախորդող տրամաբանական նախագծման փուլի հետ առաջին հերթին պայմանավորված է սխեմայի արագագործության ապահովմամբ՝ պայմանավորված միջմիացումներում հապաղումների գերակշռող բնույթով:

Ընդհանրացնելով ԻՄ-երի նախագծման մեթոդների հետագա զարգացման տեսանկյունից առավել կարևոր իրողությունների և դրանց լուծմանը նպաստող ֆիզիկական նախագծման խնդիրների վերը բերված վերլուծությունը՝ կարող ենք ասել, որ ԻՄ-երի ֆիզիկական նախագծման խնդիրների կապը դրան նախորդող սխեմատեխնիկական փուլերի հետ առաջին հերթին արտահայտվում է ժամանակային պարամետրերով, որն էլ իր հերթին մեծապես կախված է միջմիացումների հապաղումներից: Վերջինս, իր հերթին, պայմանավորված է հատակագծման և տեղաբաշխման խնդիրների արդյունավետ լուծմամբ: Նկ. 1.3-ում ներկայացված է ԻՄ-

երի նախագծման գործընթացի պարզեցված երթուղին՝ ժամանակային սահմանափակումների հաշվառմամբ:



Նկ. 1.3. ԻՄ-երի ֆիզիկական նախագծման խնդիրների վաղաժամ դիտարկման երթուղին

Այսպիսով՝ ԻՄ-երի ֆիզիկական նախագծման փուլում սխեմատիկական առանձնահատկությունների հաշվառման տեսանկյունից առավել կարևորվում են՝

- գերբարձր արագագործությունը և ֆիզիկական նախագծման փուլում դրա ապահովման համար միջմիացումների աճող կարևորությունը,

- մեծ տեսակարար սպառող հզորությունը և ֆիզիկական նախագծման փուլում դրա դեմ պայքարի համար տարրերի ջերմային տեղաբաշխումը:

1.2. ԻՄ-երի մասշտաբավորումը և նախագծման ֆիզիկական ու սխեմատեխնիկական փուլերի միջև փոխկապվածության ուժեղացումը

Նախորդ բաժնում բերված նանոչափական ԻՄ-երի ավտոմատացված ֆիզիկական նախագծման մարտահրավերների վերլուծությունը, ինչպես նաև ԻՄ-երի մասշտաբավորմանն առնչվող գրականության ուսումնասիրումը ցույց են տալիս, որ դրանց ավտոմատացված նախագծման ֆիզիկական ու սխեմատեխնիկական փուլերի միջև փոխկապվածության ուժեղացումը հիմնականում պայմանավորված է ֆիզիկական նախագծման փուլում արագագործության և ջերմային ռեժիմի ապահովմամբ [1-3,5,6]:

ԻՄ-երի ավտոմատացված նախագծման ֆիզիկական ու սխեմատեխնիկական փուլերի միջև փոխկապվածության ուժեղացումը՝ պայմանավորված սխեմայի արագագործությամբ:

Թվային սխեմաներում տեղեկույթի մշակման ընդհանրացված ուղին կարելի է ներկայացնել համակցական. հիշողության սխեմաների հաջորդականությամբ, որոնց ամբողջականությունը կազմում է հաջորդական սխեմա: Հաջորդական սխեմայի աշխատանքի ժամանակ համակցական սխեմաները փոխարկում են տվյալները որոշակի տրամաբանական կախվածությամբ, իսկ հիշողության սխեմաները դրանք ընդունում է անցումային պրոցեսների ավարտից հետո: Համակցական սխեմաներում գոյություն ունեն մուտքերից մինչև ելքեր տարբեր ճանապարհներ, որոնք կարող են ունենալ տարբեր հապաղումներ: Հաջորդական սխեմաներում ազդանշանների ժամանակային մրցության վերացման հիմնական մեթոդը դրանց համաժամանակեցումն է: Համաժամանակեցումն իրականացվում է սինքրոզեներատորով, որի ազդանշանները տրվում են հիշողության սխեմաներին, որոնք, իրենց հերթին, տեղադրված են համակցական սխեմաների միջև և թույլատրում են տվյալների ընդունումը դրանց կողմից [19]:

Այսպիսով, հաջորդական սխեմայի արագագործությունը որոշվում է սինքրոզեներատորից տրվող սինքրոիմպուլսների պարբերությամբ, որն էլ, իր հերթին,

որոշվում է համակցական սխեմաների մուտք-ելք ուղիներում ազդանշանների հապաղումներով [3]: Գործնականում գոյություն ունեն համակցական սխեմաների սկզբնական մուտքերից մինչև ելքեր ազդանշանի թույլատրելի հապաղման (T_{ph}) ներքին (T_{in}) և վերին (T_{vl}) սահմաններ, որոնց միջակայքում սխեման ճիշտ է գործում՝

$$T_{in} \leq T_{ph} \leq T_{vl} : \quad (1.1)$$

Այսպիսով, հաջորդական սխեմայի ընդհանուր արագագործությունը որոշվում է այդ սխեմայի համակցական մասերի T_{in} և T_{vl} արժեքներով: Հետևաբար, սխեմայի արագագործության մեծացմանը կարելի է հասնել՝ հնարավորինս կրճատելով T_{vl} և T_{in} սահմանների տարբերությունը՝ առանց T_{vl} -ի մեծացման: Համակցական սխեմայի մուտք-ելք որևէ ուղու հապաղումը կարելի է ներկայացնել երկու գումարելիների տեսքով՝

$$T_{hամ} = T_{տ} + T_{մմ} , \quad (1.2)$$

որտեղ $T_{տ}$, $T_{տ}$ և $T_{մմ}$ –ն համապատասխանաբար ուղու և այդ ուղու վրա առկա տարրերի ու միջմիացումների հապաղումներն են:

Ինչպես արդեն նշվել է, ԻՍ-երի մասշտաբավորմանը զուգընթաց աճում է միջմիացումներում ազդանշանների հապաղման մասնաբաժինը, և հետևաբար՝ գնալով ավելի է կարևորվում միջմիացումներում հապաղումների դերը սխեմայի արագագործության տեսանկյունից:

Եթե ուղիների T_{in} և T_{vl} ընդհանուր հապաղումներից հանվի տարրերի հապաղումներով պայմանավորված մասը, ապա միայն շղթաներով պայմանավորված հապաղումների համար կարող ենք գրել՝

$$T_{2in} \leq T_{2ph} \leq T_{2vl} , \quad (1.3)$$

որտեղ T_{2in} , T_{2vl} և T_{2ph} համապատասխանաբար սխեմայի մուտքից ելք շղթաներով պայմանավորված հապաղման ներքին, վերին և թույլատրելի մեծություններն են:

Եթե որպես տարրային հենք հանդես է գալիս թվային ստանդարտ բջիջների որոշակի գրադարան, ապա տրամաբանական սինթեզի արդյունքում տարրերի

հապաղումները դառնում են հայտնի, իսկ որոշակի տեխնոլոգիայի դեպքում շղթաների հապաղումները հիմնականում կորոշվեն դրանցում առկա միջմիացումների երկարություններով: Միջմիացումների երկարությունները որոշվում են ֆիզիկական նախագծման արդյունքում: Եթե հաշվի առնենք, որ ֆիզիկական նախագծման վերջին խնդրի՝ ծրագծման արդյունքները մեծաաամբ կախված են տարրերի տեղաբաշխման արդյունքից, ապա ակնհայտ է դառնում տեղաբաշխման փուլում ապագա միջմիացումներում հապաղումները հաշվի առնելու կարևորությունը: Այսինքն՝ կարևորվում է նախագծման սխեմատեխնիկական և ֆիզիկական փուլերի փոխկապվածությունը:

Ենթամիկրոնային տեխնոլոգիաները առաջադրում են նոր պահանջներ ԻՄ-երի սխեմատեխնիկական և ֆիզիկական նախագծման փուլերի միջև փոխկապվածության ուժեղացման տեսանկյունից: Այս պահանջները առաջին հերթին պայմանավորված են միջմիացումներում ազդանշանի հապաղումներով: Եթե հաշվի առնենք, որ ժամանակակից թվային ԻՄ- երում ազդանշանների հապաղումը միջմիացումներում մի քանի հարյուր անգամ գերազանցում է տրամաբանական փականների ազդանշանի հապաղմանը և կազմում է ԻՄ- ի ընդհանուր հապաղումների 80-ից 90 տոկոսը, ապա կարելի է ասել, որ սխեմատեխնիկական առանձնահատկությունների հաշվի առնելը ավտոմատացված ֆիզիկական նախագծման փուլում կարևորագույն խնդիր է, և դրա նշանակությունը աճում է տեխնոլոգիայի զարգացմանը զուգընթաց [3,4]: Քանի որ միջմիացումների երկարությունները առաջին մոտավորությամբ պայմանավորված են դրանցով միացվող տարրերի տեղաբաշխումով, ապա վերը նշված իրողությունները սկզբունքորեն նոր պահանջներ են առաջադրում տեղաբաշխման խնդրին, և կարևորվում է դրա լուծման այնպիսի մեթոդների և ալգորիթմների մշակումը, որոնք հաշվի կառնեն սխեմայի աշխատանքի տրամաբանությունը, դրանով իսկ նպաստելով միջմիացումներով պայմանավորված արագագործության բարձրացմանը: Այս նպատակով, սկսած այն պահից, երբ միջմիացումներում հապաղումները սկսեցին գերակայել տրանզիստորների հապաղումների համեմատ, կարևորվել և խորապես ուսումնասիրվել է ԻՄ-երի տարրերի ժամանակային ուղղորդմամբ տեղաբաշխման խնդիրը [3,7,20]:

ԻՄ-երի ավտոմատացված նախագծման ֆիզիկական ու սխեմատիկական փուլերի միջև փոխկապվածության ուժեղացումը՝ պայմանավորված հզորության աճով և ջերմային ռեժիմով:

Մեծ տեսակարար էներգասպառումը, արագագործության աճի հետ համատեղ, պատճառ է հանդիսանում կիսահաղորդչային բյուրեղի վրա, ինչպես տեղային ջերմաստիճանների բարձրացման, այնպես էլ տարբեր տեղամասերի ջերմաստիճանների տարբերության, այսինքն՝ ջերմային գրադիենտի մեծացման [6,21]:

Ֆիզիկական նախագծման տեսանկյունից սա պայմանավորված է այն փաստով, որ կոմպլեմենտար մետաղ - օքսիդ - կիսահաղորդիչ (ԿՄՕԿ) կառուցվածքների հիման վրա կառուցված սխեմաներում ընդհանուր սպառվող հզորությունը (P_{Σ}) կազմված է դինամիկ (P_d) և ստատիկ ($P_{\text{ստ}}$) հզորությունների բաղադրիչներից՝

$$P_{\Sigma} = P_d + P_{\text{ստ}}, \quad (1.4)$$

որտեղ դինամիկ բաղադրիչը, իր հերթին, կախված է արագագործությունից [22,23]:

Եթե հաշվի առնենք, որ սպառվող հզորության ստատիկ բաղադրիչը պայմանավորված է ԿՄՕԿ տրանզիստորի կորստային հոսանքով, կարող ենք ասել, որ տրված տարրային հենքի դեպքում ԻՄ-երի ավտոմատացված նախագծման ֆիզիկական ու սխեմատիկական փուլերի միջև հզորությունով պայմանավորված փոխկապվածությունը հիմնականում արտահայտվում է հզորության դինամիկ բաղադրիչով:

Ժամանակակից ԻՄ-երի տարրային հենքում օգտագործվող ԿՄՕԿ կառուցվածքների համար սխեմայի դինամիկ հզորության որոշման արտահայտությունն ունի հետևյալ տեսքը [23]՝

$$P_{\text{դին}} = C_L * V_{DD}^2 * P_{\text{փոխ}} * f_{\text{սինք}}, \quad (1.5)$$

որտեղ C_L -ը բեռի ունակությունն է, V_{DD} -ն՝ սնուցման լարումը, $P_{\text{փոխ}}$ -ը՝ տրամաբանական տարրի փոխանջատման հավանականությունը, իսկ $f_{\text{սինք}}$ -ը՝ սինքրոադանշանի հաճախականությունը:

Դինամիկ հզորության (1.5) բանաձևը հիմնված է այն ենթադրության վրա, որ տրամաբանական տարրերը սինքրոադանշանի յուրաքանչյուր տակտում կատարում

են ընդամենը մեկ փոխանջատում: Սակայն իրականում տրամաբանական տարրերի փոխանջատումների քանակը միավոր ժամանակում կախված է ինչպես սխեմայի տրամաբանական կառուցվածքից, այնպես էլ սխեմայի մուտքերին տրվող ազդանշանային հավաքներից: Հետևաբար, ցանկացած տրամաբանական տարրի փոխանջատման հաճախականությունը, որպես կանոն, փոքր է լինում սինքրոնազդանշանի հաճախականությունից: Այս երևույթը հաշվի առնելու նպատակով, տրամաբանական տարրի աշխատանքային հզորության գնահատման համար, օգտագործվում է փոխանջատման ակտիվություն հասկացությունը, որը կախված է ինչպես տարրի տիպից, այնպես էլ սխեմայի տրամաբանությունից և մուտքային հավաքից: Որևէ տրամաբանական տարրի փոխանջատման ակտիվությունը ցույց է տալիս տակտային ազդանշանի մեկ պարբերությունում տվյալ տարրի ելքային ազդանշանի փոխանջատումների թիվը: Մյուս կողմից՝ որպես ԻՄ-երի տարրային հենք օգտագործվող ստանդարտ բջիջների գրադարանների համար տրված են լինում գրադարանում առկա յուրաքանչյուր տրամաբանական բջջի դինամիկ հզորությունը տվյալ գրադարանի նոմինալ հաճախականությանը համապատասխանող մեկ տակտի համար: Այսպիսով, տրված ստանդարտ թվային գրադարանային տարրային հենքի վրա կառուցված սխեմայում ցանկացած տարրի աշխատանքային դինամիկ սպառող հզորությունը ($P_{առդ}$) որոշվում է հետևյալ կերպ [3];՝

$$P_{առդ} = P_{առգ} K F, \quad (1.6)$$

որտեղ $P_{առգ}$ – ն տվյալ տարրի տեսակարար դինամիկ հզորությունն է գրադարանի նոմինալ արագագործության դեպքում, K -ն տվյալ տարրի փոխանջատումների ակտիվությունը դիտարկվող սխեմայում, F -ը թվային ստանդարտ բջիջների գրադարանի աշխատանքային հաճախությունը:

Հաշվի առնելով (1.4) և (1.6)-ը, վերը շարադրվածից հետևում է, որ եթե ֆիզիկական նախագծման արդյունքում կիսահաղորդչային բյուրեղի որոշակի լոկալ տեղամասում տեղաբաշխվեն մեծ փոխանջատումների ակտիվությամբ տարրեր, ապա այդ տեղամասում կառաջանա մեծ տեսակարար էներգասպառում, որն իր հերթին կհանգեցնի տեղային բարձր ջերմաստիճանների: Ամբողջական կիսահաղորդչային բյուրեղի տիրույթում այսպիսի տեղամասերի առկայությունը կհանգեցնի մեծ ջերմային

գրադիենտի: Մյուս կողմից, համաձայն Բլեյքի օրենքի, ԻՄ-ի հուսալիությունը ջերմաստիճանից կախված ընկնում է էքսպոնենցիալ ձևով՝

$$T_{\text{միջ}}(\theta) = AJ^{-m} \exp\left(\frac{E}{k\theta}\right), \quad (1.7)$$

որտեղ $T_{\text{միջ}}(\theta)$ -ն ԻՄ-ի անխափան աշխատանքի միջին ժամանակն է, A -ն հաստատուն, որը կախված է տեխնոլոգիական միկրոկառուցվածքից, J -ն՝ հոսանքի խտությունը, իսկ m ցուցիչը $J=(0,2\text{-ից } 2,0)106 \text{ Ա/սմ}^2$ սահմաններում ընդունում է $m=1\text{-ից } 2$ արժեքը, E -ն՝ ակտիվացման էներգիան, k -ն՝ Բոլցմանի հաստատունը, իսկ θ -ն՝ ԻՄ-ի բացարձակ ջերմաստիճանը [13,24]:

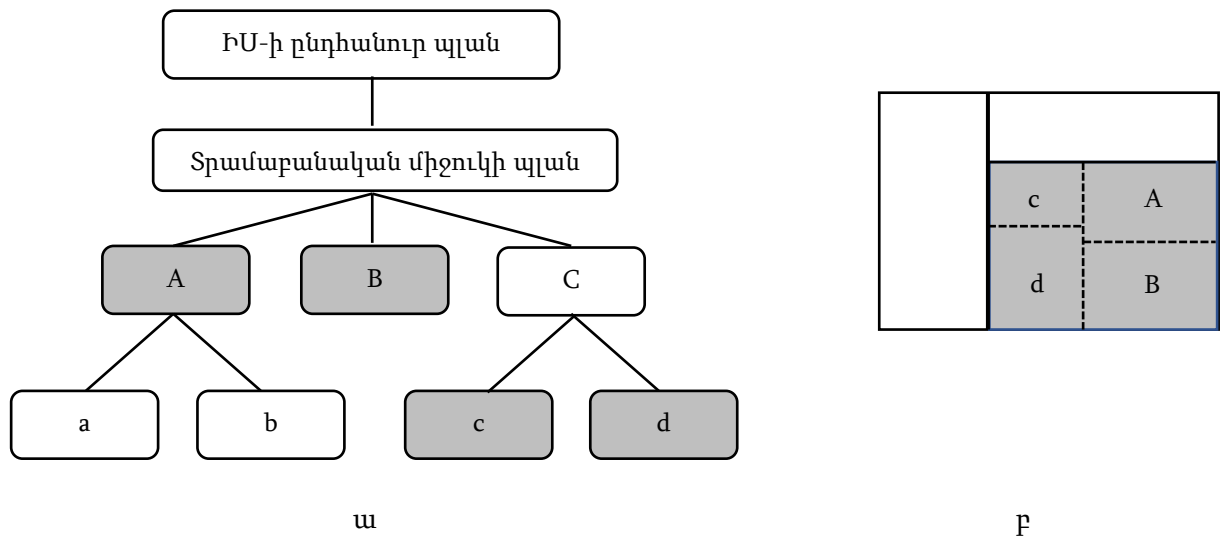
Այսպիսով, տեսակարար հզորության աճը, հանգեցնում է ԻՄ-ի հուսալիության զգալի նվազման, իսկ ֆիզիկական նախագծման փուլում որպես դրա դեմ պայքարի միջոց կարող է ծառայել տարրերի այնպիսի տեղաբաշխումը, որը կապահովի կիսահաղորդչային բյուրեղի վրա ջերմային դաշտի համահարթեցումը և դրանով իսկ՝ առավել բարձր ջերմաստիճանային տիրույթներում ջերմաստիճանի իջեցմանը [3,25]:

Սկսած այն պահից, երբ միջմիացումներում հապաղումները սկսեցին գերակայել տրանզիստորների հապաղումների համեմատ, կարևորվել է ԻՄ-երի տարրերի ժամանակային ուղղորդմամբ տեղաբաշխման խնդիրը: Այս իրողությունը սկզբունքորեն նոր պահանջներ է առաջադրում տեղաբաշխման խնդրին, և կարևորվում է դրա լուծման այնպիսի մեթոդների և ալգորիթմների մշակումը, որոնք հաշվի կառնեն սխեմայի աշխատանքի տրամաբանությունը, դրանով իսկ նպաստելով միջմիացումներով պայմանավորված արագագործության բարձրացմանը:

1.3. ԻՄ-երի նախագծման ֆիզիկական և տրամաբանական մակարդակների փոխկապվածությունը

Ընդհանուր դեպքում, տրամաբանական և ֆիզիկական մակարդակներում նախագծվող սխեմայի հիերարխիկ ներկայացումը կարող է տարբեր լինել: Այս դեպքում տրամաբանական հիերարխիայում տրամաբանական այնպիսի հանգույցների համախումբը, որոնք ֆիզիկական կառուցվածքի իմաստով պետք է գտնվեն իրար մոտ, ֆիզիկական մակարդակում անհրաժեշտ է ներկայացնել ֆիզիկական բլոկի տեսքով:

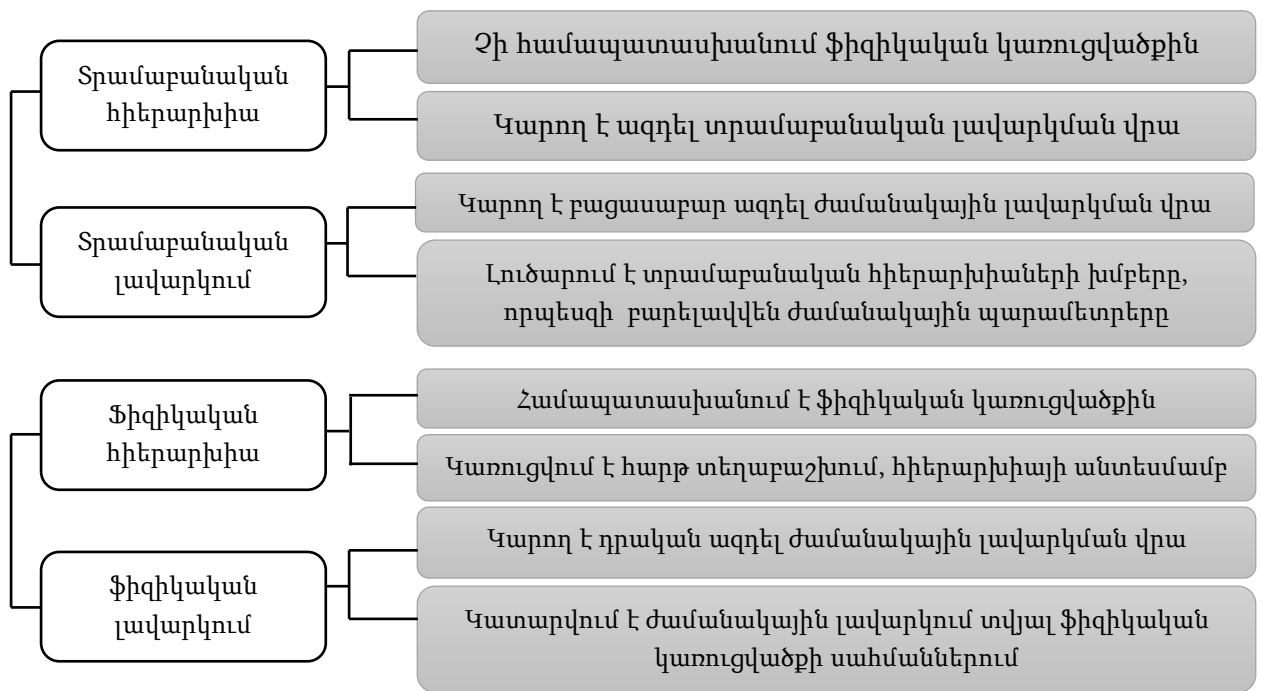
Այդպիսի տարրերի խումբը կոչվում է պլանային խումբ[26]։ Այդպիսի պլանային խմբի ձևավորման օրինակ բերված է նկ.1.4-ում։



Նկ. 1.4. ա-տրամաբանական հիերարխիա, որտեղ A,B,c,d տարրերը ֆիզիկական նախագծման ժամանակ պետք է պահվեն միասին, բ-ֆիզիկական հիերարխիա, որտեղ A,B,c,d տարրերը ներկայացված են մեկ ֆիզիկական բլոկի մեջ։

Տրամաբանական հիերարխիայի բոլոր տարրերը, որոնք պատկանում են մեկ պլանային խմբի, ֆիզիկական հիերարխիայում պետք է տեղադրվեն որոշակի ֆիզիկական դաշտում։ Հետագայում մինևույն տարածքում այլ տարրի տեղադրումը արգելվում է։ Պլանային խմբերը տրամաբանական հիերարխիայում կարող են պարունակել հիերարխիայի ցանկացած մակարդակի տրամաբանական տարրեր։ Ցանկացած պլանային խումբ ֆիզիկական հիերարխիայում ներկայացնում է մեկ հիերարխիա։ Ներկայացված օրինակում (նկ. 1.4) A, B, c, d -ն պատկանում են տրամաբանական հիերարխիայի տարբեր մակարդակների, բայց ֆիզիկական հիերարխիայում նրանք տեղակայված են նույն ֆիզիկական պլանային խմբի մեջ։

Տրամաբանական և ֆիզիկական հիերարխիաներում լավարկման առանձնահատկությունները բերված են նկ. 1.5 – ում:



Նկ. 1.5. Տրամաբանական և ֆիզիկական հիերարխիաների կապը լավարկման հետ

Տրամաբանական հիերարխիան չի առնչվում տեղաբաշխման որակին, այն կարող է միայն ազդել սխեմայի աշխատանքի տրամաբանական որակների վրա: Տրամաբանական մակարդակի ոչ օպտիմալ հիերարխիայի կառուցումը, առաջին հերթին, կարող է բացասական ազդեցություն ունենալ սխեմայի ժամանակային պարամետրերի վրա: Հաճախ, ժամանակային պարամետրերի բարելավման համար, տրամաբանական մակարդակում հարկ է լինում տրամաբանական հիերարխիան ենթարկել հիմնային փոփոխության, այսինքն՝ փոփոխել սխեմայի ճարտարապետությունը:

Ֆիզիկական հիերարխիան համապատասխանում է տեղաբաշխմանը այն դեպքում, երբ տեղաբաշխումը հարթ է, այսինքն՝ ֆիզիկական սինթեզման ծրագիրը տեղաբաշխում է բոլոր հիերարխիկ տարրերը մեկ հարթության վրա՝ ամբողջ տեղաբաշխման տիրույթով, և դրա ժամանակ հիերարխիան անտեսվում է: Բայց ֆիզիկական պլանային խմբերի մասնատման ձևը կարող է զգալի ազդեցություն ունենալ ժամանակային լավարկման վրա: Ինչպես արդեն վերը նշվել է, պայմանավորված

միջմիացումների վրա հապաղումների մասնաբաժնի աճով, տեղաբաշխումը կարող է էականորեն ազդել ժամանակային պարամետրերի վրա: Այսինքն, եթե նախկինում ժամանակային սահմանափակումները հաշվի էին առնվում միայն սխեմատիկական նախագծման փուլում, ապա ներկայումս դրանք դարձել են ֆիզիկական նախագծման փուլի ամենակարևոր սահմանափակումներից մեկը: Ժամանակային տեսակետից որոշիչ ուղիների վրա ընկած տարրերը դրվում են միմյանց մոտիկ, որը հանգեցնում է դրանք միացնող միջմիացումների երկարությունների կրճատման: Սա հանգեցնում է այդ ուղիների հապաղումների փոքրացմանը, դրանով իսկ նպաստելով սխեմայի աշխատանքի բարելավմանը: Տրամաբանական հիերարխիայից ֆիզիկականին անցման ժամանակ, ժամանակային պարամետրերի լավարկման համար, կարող են ստեղծվել նոր ֆիզիկական հիերարխիկ կառուցվածքներ: Ստեղծված ֆիզիկական հիերարխիան ունի հետևյալ առանձնահատկությունները՝

- վիրտուալ հարթ նախագիծը կարող է փոխակերպվել բազմամակարդակ հիերարխիկ նախագծի,
- տրամաբանական մակարդակի բջիջներից կարող են սինթեզվել ֆիզիկական մակրոսներ, որոնք իրագործում են մեկից ավելի բջիջների համատեղված ֆունկցիաներ,
- սինթեզվող մակրոսները սկզբում ունենում են «փափուկ» բլոկների տեսք, որոնց ֆիզիկական չափսերը ֆիքսված չեն,
- «փափուկ» մակրոսների միջև ստեղծվում են կապող շղթաներ, որոնք հիմնված են տրամաբանական հիերարխիայից բխող ֆունկցիոնալության պահպանման վրա,
- որոշվում են «փափուկ» բլոկների սնուցման կազմակերպման կառուցվածքային հարցերը,
- որոշվում են «փափուկ» բլոկների տեղաբաշխման և ծրագծման կազմակերպման կառուցվածքային հարցերը:

Ֆիզիկական հիերարխիան ստեղծելուց հետո, ֆիզիկական նախագծման կանոնների և սահմանափակումների հաշվի առնելու արդյունքում, «փափուկ»

մակրոսները վերածվում են «կոշտ» բլոկների, որոնք ունեն որոշակի ֆիքսված ձև և չափսեր:

Տրամաբանական նախագծման արդյունքում ստացված ժամանակային պարամետրերը, որոնք բավարարում են նախագծային սահմանափակումները, հաճախ ֆիզիկական նախագծման արդյունքում կարող են չբավարարել այդ սահմանափակումները: Հետևաբար, կարևորվում է ֆիզիկական նախագծման այնպիսի միջոցների մշակումը, որոնք թույլ կտան հաշվի առնել այդ սահմանափակումները:

Ինչպես արդեն նշվել է, ԻՄ-երի տեխնոլոգիաների զարգացումը հանգեցնում է ֆիզիկական և սխեմատեխնիկական պարամետրերի փոխկապվածության աճին: Այս տեսակետից առավել կարևորվում է միջմիացումների հապաղումներով պայմանավորված արագագործության ապահովումը ֆիզիկական նախագծման փուլում: Ֆիզիկական սինթեզը սկսվում է նախագծվող սխեմայի շղթաների ցուցակից, որը գեներացվում է տրամաբանական սինթեզի արդյունքում: Ընդ որում, տրամաբանական սինթեզի արդյունքը նախօրոք ենթարկվում է տեխնոլոգիական քարտազման և ներկայացվում է կոնկրետ ստանդարդ բջիջների գրադարանի հենքի վրա կառուցված փականային մակարդակի շղթաների ցուցակով, սարքավորումների նկարագրման որևէ լեզվով: Նշենք, որ ընդհանուր դեպքում շղթաների ցուցակը գրադարանային ստանդարտ բջիջներից բացի կարող է պարունակել նաև մտավոր սեփականության տարբեր բլոկներ: Ներկայումս լայնորեն տարածված է տրամաբանական սինթեզի արդյունքների՝ Verilog լեզվով ներկայացումը: Ընդհանուր առմամբ, ֆիզիկական սինթեզի նպատակն է այնպիսի ֆիզիկական կառուցվածքների ձևավորումը, որոնք կբավարարվեն նախագծին ներկայացվող արագագործության, հզորության և մակերեսային պահանջներն ու սահմանափակումները: Ինչպես արդեն նշվել է, ֆիզիկական նախագծման փուլում այս նպատակին հասնելու հիմնական միջոցը տարրերի տեղաբաշխման խնդիրն է: Այդ նպատակով ներկայումս սխեմայի ժամանակային պարամետրերը, որոնք ստուգվել և գնահատվել են տրամաբանական սինթեզի փուլում, որպես կանոն, տեղաբաշխումից հետո նույնպես ստուգվում են[27]: Այսպիսի ստուգման արդյունքում հաճախ հայտնաբերվում են անընդունելի ժամանակային շեղումներ, որոնք վերացվում են միջմիացումների բազմակի վերատեղաբաշխումների միջոցով ապագա

միջմիացումների երկարությունների կառավարման, բուֆերացման, իսկ, անհրաժեշտության դեպքում, նաև տրամաբանական սինեզի փուլ վերադարձի և տրամաբանական դեկոմպոզիցիայի փոփոխման ճանապարհով [28]:

ԻՄ-երի ավտոմատացված ֆիզիկական նախագծման փուլում սխեմատեխնիկական առանձնահատկությունները հաշվի առնելու տեսանկյունից կարևորվում է սխեմայի շղթաների այնպիսի ժամանակային պարամետրերի գնահատումը, որոնք թույլ կտան կառավարել այդ շղթաների երկարությունները տեղաբաշխման փուլում: Հաշվի առնելով այն հանգամանքը, որ մի կողմից՝ սխեմայի արագագործության մեջ զգալի ազդեցություն ունեն միջմիացումներով պայմանավորված հապաղումները, որոնք որոշվում են դրանց երկարությամբ, իսկ մյուս կողմից՝ միջմիացումների երկարությունները որոշվում են դրանցով միացվող տարրերի տեղաբաշխմամբ, որպես այդպիսի պարամետրեր կարող են հանդես գալ նախորդ բաժնում դիտարկված սխեմայի մուտքից ելք շղթաներով պայմանավորված հապաղման ներքին (T_{2n}) և վերին (T_{2v}) սահմանները: Այս պարամետրերի տարբերությունը, որը կոչվում է ժամանակի պահուստ, տարրերի տեղաբաշխման փուլում կարող է ծառայել որպես միջոց՝ շղթաների կարևորությունների գնահատման համար:

Այս պարամետրերի գնահատման համար սխեմատեխնիկական նախագծման փուլում օգտագործվում են ստատիկ ժամանակային վերլուծության մեթոդները [29-33]:

Ստատիկ ժամանակի վերլուծության խնդիրը սովորաբար լուծվում է սխեմայի դիտարկման երկու անցումների հիման վրա: Առաջին անցումը կատարվում է հիմնական մուտքերից մինչև հիմնական ելքեր, իսկ երկրորդ անցումը՝ հակառակ ուղղությամբ: Սխեմայի ուղիղ անցման ժամանակ հաշվարկվում է յուրաքանչյուր շղթայի հապաղման ներքին սահմանը:

Հիմնվելով ասվածի վրա, սխեմայի ցանկացած i -րդ շղթայի համար ժամանակի ներքին սահմանը կարելի է որոշել հետևյալ ընդհանրացված բանաձևով[3]՝

$$T_{\text{in}} = \max[T_{\text{in}(i-1)} + T_{(i-1),i}]; i = 1, 2, \dots, n, \quad (1.8)$$

որտեղ T_{ii} և $T_{i(i-1)}$ -ը համապատասխանաբար i -րդ և $(i-1)$ -րդ շղթաների ժամանակի ներքին սահմաններն են; $T_{(i-1),i}$ -ն՝ $(i-1)$ և i շղթաները կապող տարրի հապաղումը; n -ը՝ շղթաների թիվը: Ընդ որում, $(i-1)$ -ից i անցումները կատարվում են սխեմայի տարրերի աշխատանքի նախորդ-հաջորդ տրամաբանությամբ: Այսինքն՝ i -րդ շղթայի համար $(i-1)$ -րդ շղթաներ են բոլոր այն շղթաները, որոնք միացված են $(i-1)$ և i շղթաները կապող տարրի մուտքերին: Սխեմայի առաջնային մուտքերի համար ընդունված է $T_0 = 0$:

Որևէ շղթայում ճիշտ ազդանշանի ձևավորման վերին սահմանը սխեմայի առաջնային մուտքերին ազդանշանը տալու պահից մինչև տվյալ շղթայում ճիշտ ազդանշանի ձևավորումը ընկած այն առավելագույն ժամանակն է, որը դեռևս չի հանգեցնում սխեմայի ելքերում ճիշտ ազդանշանի հայտնվելու ժամանակի մեծացմանը: Այն որոշվում է հետևյալ բանաձևով՝

$$T_{qi} = \max [T_{q(i+1)} - T_{(i+1),i}]; i = 1, 2, \dots, n, \quad (1.9)$$

որտեղ T_{qi} և $T_{q(i+1)}$ -ը համապատասխանաբար i -րդ և $(i+1)$ - րդ շղթաների ժամանակի վերին սահմաններն են; $T_{(i+1),i}$ -ն՝ այն տարրի հապաղումն է, որի համար $(i+1)$ -րդ շղթան ելքային է, իսկ i -րդ շղթան՝ մուտքային; n -ը՝ շղթաների թիվը: Ընդ որում, $(i+1)$ -ից i անցումները կատարվում են սխեմայի տարրերի աշխատանքի հաջորդ - նախորդ տրամաբանությամբ: Այսինքն՝ i -րդ շղթայի համար $(i+1)$ -րդ շղթաներ են համարվում այն բոլոր շղթաները, որոնք միացված են i և $(i+1)$ շղթաները կապող տարրի ելքին: Այս դեպքում սխեմայի բոլոր ելքերի համար որպես հաշվարկի սկիզբ է ծառայում դրանցում ճիշտ ազդանշանի ձևավորման ժամանակների վերին սահմաններից առավելագույնի արժեքը, քանի որ տվյալ համակցական սխեմայից ելքային ազդանշանների փոխանցումը հաջորդ համակցական սխեմային կարող է կատարվել այդ ժամանակից ոչ ավելի շուտ:

Որևէ i -րդ շղթայի ժամանակի պահուստը ներկայացնում է տվյալ շղթայում ճիշտ ազդանշանի ձևավորման ժամանակների վերին և ներքին սահմանների տարբերությունը՝

$$R_i = T_{qi} - T_{ii}; i = 1, \dots, n: \quad (1.10)$$

Մխենայի արագագործության տեսանկյունից՝ (1.3) և (1.10)-ից հետևում է, որ շղթայի ժամանակի պահուստը բնութագրում է շղթայի կարևորությունը: Որքան փոքր է շղթայի ժամանակի պահուստը, այնքան տվյալ շղթայում առաջացող հապաղումը ավելի է ազդում սխեմայի արագագործության վրա: Հետևաբար, սխեմայում այս կամ այն շղթայի ժամանակի պահուստը ֆիզիկական նախագծման փուլում կարող է ծառայել որպես ֆիզիկական կառուցվածքում շղթաների երկարության կառավարման միջոց: Հաշվի առնելով այն հանգամանքը, որ ֆիզիկական կառուցվածքում շղթաների երկարությունները կանխորոշվում են տարրերի տեղաբաշխման ժամանակ, կարող ենք հանգել կարևոր եզրակացության. շղթայի ժամանակի պահուստը կարող է ծառայել, որպես ժամանակային տեղաբաշխման որակի չափանիշ: Տեղաբաշխման ավանդական ալգորիթմներում որպես որակի չափանիշ օգտագործվում է միջմիացումների գումարային երկարության չափանիշը [3]՝

$$F = \min_{\Omega_N} \sum_{i=1}^N \sum_{\substack{j=1 \\ i \neq j}}^N r_{ij} d_{ij}, \quad (1.11)$$

որտեղ r_{ij} -i-րդ և j-րդ տարրերի միջև կապերի թիվն է, d_{ij} -ն՝ տեղաբաշխման արդյունքում i-րդ և j-րդ տարրերի միջև եղած հեռավորությունը, N-ը՝ տեղաբաշխվող տարրերի թիվը, Ω_N -ը՝ տարրերի տեղաբաշխման բոլոր հնարավոր տարբերակների թիվը:

Այսպիսի մոտեցման դեպքում արագագործության տեսանկյունից բոլոր շղթաներն ընդունվում են հավասարաթեք կարևորությամբ և արդյունքում նվազարկվում է նրանց գումարային երկարությունը:

Մխենայի աշխատանքի առանձնահատկությունները հաշվի առնելու համար տարրերի միջև առկա ցանկացած կապի դեպքում անհրաժեշտ է հաշվի առնել այդ կապի կարևորությունը ժամանակի պահուստի տեսակետից: Այսինքն՝ անհրաժեշտ է յուրաքանչյուր կապի գծին վերագրել որոշակի կշռային գործակից:

Ինչպես արդեն նշվել է, ժամանակային տեղաբաշխման մեթոդները երկուսն են՝ ուղու վրա հիմնված և շղթայի վրա հիմնված մեթոդներ: Արդյունաբերական էլեկտրոնային նախագծման ավտոմատացման համակարգերում կիրառվող ժամանակային

տեղաբաշխման ալգորիթմները, որպես կանոն, աշխատում են ուղու վրա հիմնված մեթոդներին համապատասխան՝ իրագործելով բազմակրկնություններ, երբ փորձ է արվում փոքր ժամանակային պահուստով շղթաներից կազմված ուղիների վրայի տարրերի վերատեղաբաշխումների ճանապարհով փոքրացնել դրանք կապող շղթաների երկարությունները՝ դրանով իսկ կրճատելով այդ ուղիների հապաղումները [34-37]:

Շղթաների ժամանակային պահուստների անհրաժեշտ արժեքները ստացվում են ստատիկ ժամանակային վերլուծության ծրագրերի միջոցով [38]:

Առկա արդյունաբերական ծրագրային գործիքները գործնականում չեն օգտագործում ժամանակային տեղաբաշխման եղանակները նախնական տեղաբաշխման դեպքում, քանի որ ժամանակային նախնական տեղաբաշխման ժամանակ անհրաժեշտ է տարրերի հաջորդական տեղաբաշխման ընթացքում կառավարել յուրաքանչյուր շղթայի երկարությունը, որը պահանջում տեղաբաշխման որակի համապատասխան չափանիշի առկայություն: Արդյունքում՝ տարրերի վերատեղաբաշխմամբ որոշիչ ուղիների բարելավումը հաճախ հանգեցնում է նոր որոշիչ ուղիների առաջացման, որոնցում հապաղումները կարող են գերազանցել նախորդներին:

Չնայած ուղու վրա հիմնված մեթոդներն ավելի ճշգրիտ են քան շղթաների եղանակը, բայց, քանի որ սխեմաներում ճանապարհների քանակը կարող է էքսպոնենցիալ աճել տարրերի քանակի հետ միասին, այն գործնականում բավարար չափով արդյունավետ չէ:

Շղթաների վրա հիմնված մոտեցման դեպքում, որպեսզի հաշվի առնվի ժամանակը, տեղաբաշխման որակի չափանիշում անհրաժեշտ է ունենալ շղթաների կշիռներ, որոնք պետք է ապահովեն որոշիչ շղթաների համեմատաբար փոքր երկարությունները:

Այս նպատակով (1.10) չափանիշում r_{ij} -ն կարելի է փոխարինել i -րդ և j -րդ տարրերի միջև առկա բոլոր շղթաների կարևորության գումարային որոշակի գործակցով՝ $\lambda_{ij} = \sum_{k=1}^{n(ij)} \lambda_k$, որտեղ λ_k -ն k -րդ շղթայի կարևորության գործակիցն է, իսկ $n(ij)$ -ն i -րդ և j -րդ տարրերի միջև առկա շղթաների քանակը:

Տարրերի ժամանակային տեղաբաշխման նպատակով հաշվի առնելով վերը շարադրվածը՝ (1.11) չափանիշը կընդունի հետևյալ տեսքը՝

$$F = \min_{\Omega_N} \sum_{i=1}^N \sum_{\substack{j=1 \\ i \neq j}}^N d_{ij} \sum_{k=1}^{n(ij)} \lambda_k: \quad (1.12)$$

Որևէ k -րդ շղթայի կարևորության գործակիցը որոշվում է՝ ելնելով այն նկատառումից, որ դրա և k -րդ շղթայի R_k ժամանակային պահուստի միջև կապի արտահայտման համար անհրաժեշտ է ապահովել հակադարձ համեմատական կապվածություն: Այս նպատակով կարելի է կատարել λ_k -ի նորմավորում՝ ելնելով պայմաններից, որի մեծ արժեքներին պետք է համապատասխանեն R_k -ի փոքր արժեքներ և հակառակը [3]:

Մյուս կողմից՝ տեղաբաշխման ալգորիթմի հաշվարկային տեսանկյունից հարմար է, որ R -ը նորմավորվի $[0, 1]$ միջակայքում, որը անհրաժեշտ է սխեմայի բարդության և հապաղման չափողականության փոփոխությունները հաշվի առնելու համար: Այդ նպատակով λ_k -ի որոշման համար կարելի է օգտվել նորմավորման հետևյալ բանաձևից՝

$$\tilde{R}_k = \frac{R_k - R_{min}}{R_{max} - R_{min}}, \quad (1.13)$$

որտեղ $\tilde{R}_k$ և R_k - ն k -րդ շղթայի ժամանակի պահուստի համապատասխանաբար նորմավորված և ընթացիկ արժեքներն են, իսկ R_{max} և R_{min} -ը՝ դիտարկվող սխեմայի շղթաների ժամանակի պահուստների համապատասխանաբար առավելագույն և նվազագույն արժեքները: Այս դեպքում k -րդ շղթայի λ_k կարևորության գործակիցը կարող է որոշվել հետևյալ կերպ՝

$$\lambda_k = 1 - \tilde{R}_k: \quad (1.14)$$

Շղթաների կշռային գործակիցների կիրառմամբ այլընտրանքային մոտեցում է հապաղման բյուջետավորումը, ինչը նշանակում է հաշվի առնել միայն ժամանակի վերին սահմանները [39]:

Այս դեպքում շղթայում հապաղման չափի կամ նրա երկարության վրա դրվում են սահմանափակումներ՝ ելնելով տվյալ շղթայում ճիշտ ազդանշանի ձևավորման վերին սահմանից: Այս եղանակը, համեմատած շղթայի կշիռներով եղանակի հետ, ավելի պարզ է: Սակայն այս մոտեցումը հարմար չէ նախնական տեղաբաշխման հաջորդական ալգորիթմներում (1.11) և (1.12) տիպի չափանիշների կիրառման տեսանկյունից, ինչպես նաև, սխեմաների մասշտաբավորմանը զուգընթաց, ավելանում են դրանց

օգտագործման դժվարությունները: Այսպես, օրինակ, մաշտաբավորման ապահովման համար շղթայի սահմանափակումները պետք է գեներացվեն այնպես, որ դրանք չափից շատ չսահմանափակեն լուծումների տիրույթը կամ լուծումների ընդհանուր քանակ՝ այդ կերպ վնասելով լուծման որակը: Այս թերությունը կարող է հաղթահարվել սահմանափակումների դինամիկ ճշգրտումների ճանապարհով, ինչը, վերջին հաշվով, հանգեցնում է ուղու վրա հիմնված մեթոդների նմանությանը և առաջ է բերում համանման թերություններ:

Ինչպես արդեն նշվել է 1.2 բաժնում, ֆիզիկական և տրամաբանական մակարդակների միջև փոխկապվածության մյուս կարևոր բաղադրիչը մեծ տեսակարար էներգասպառմամբ պայմանավորված ջերմային գրադիենտի մեծացումն է կիսահաղորդչային բյուրեղի վրա, որի դեմ պայքարի միջոց կարող է ծառայել կիսահաղորդչային բյուրեղի վրա ջերմային դաշտի համահարթեցումը տարրերի տեղաբաշխման փուլում [3, 25]: Այս խնդիրը առավել կարևորություն է ձեռք բերում եռաչափ ԻՍ-երի նախագծման դեպքում, որոնք թույլ են տալիս իրագործել ավելի բարդ ֆունկցիոնալությամբ համակարգեր՝ առանց մեծացնելու առանձին բյուրեղների չափսերը [25]: Այս դեպքում, որպես կիսահաղորդչային բյուրեղի վրա տարրերի տեղաբաշխման մեթոդ, կարող է ծառայել է տեղաբաշխման հաջորդական ալգորիթմը՝ հզորությունները հաշվի առնելով: Որպես տեղաբաշխման չափանիշ կարող է հանդես գալ բյուրեղի ջերմային դաշտի տոպոլոգիական համահարթեցման չափանիշը [3, 25]՝

$$f = \sum_{i=1}^N \sum_{j=1, j \neq i}^N |P_i - P_j| d_{ij} \rightarrow \min, \quad (1.15)$$

որտեղ P_i և P_j -ն i -րդ և j -րդ տարրերի հզորություններն են; d_{ij} -ն i -րդ և j -րդ տարրերի դիրքերի միջև եղած հեռավորությունը; N – ը՝ տեղաբաշխվող տարրերի քանակը:

(1.15) պայմանը նախագծողի տեսանկյունից նշանակում է, որ հզորության մեծ տարբերություն ունեցող տարրերն անհրաժեշտ է հնարավորինս իրար մոտ տեղադրել և հակառակը, ինչը կհանգեցնի կիսահաղորդչային բյուրեղի ջերմային դաշտի հավասարակշռմանը և, հետևաբար, կնվազի հուսալիության տեսանկյունից ամենավտանգավոր տեղամասերի ջերմաստիճանը:

Նախագծման ֆիզիկական և տրամաբանական մակարդակների փոխկապվածության տեսանկյունից՝ (1.15)-ում կարևորվում է տարրերի աշխատանքային հզորությունների գնահատման խնդիրը: Եթե ընդունենք, որ հզորությունների գնահատման (1.4) արտահայտությունում հզորության ստատիկ բաղադրիչը գործնականում կախված չէ տարրերի փոխադարձ դասավորությունից, ապա տարրերի տեղաբաշխման փուլում հզորությունների գնահատման համար կարելի է օգտագործել միայն դրանց դինամիկ բաղադրիչը: Այս դեպքում, եթե աշխատանքային հզորությունների գնահատման (1.5) արտահայտության մեջ առանձին տարրերի տեսակարար դինամիկ հզորությունները և թվային ստանդարտ բջիջների գրադարանի աշխատանքային հաճախությունը հայտնի են դառնում գրադարանի ընտրության արդյունքում և կախված չեն տրամաբանական սխեմայից, ապա տարրերի փոխանջատումների ակտիվությունները անմիջականորեն կախված են սխեմայի տրամաբանությունից: Հետևաբար, ֆիզիկական նախագծման դեպքում կարևորվում է այնպիսի միջոցների մշակումը, որոնք թույլ կտան ֆիզիկական կառուցվածքների ստացման ժամանակ հաշվի առնել տրամաբանական սխեմայում տարրերի ակտիվությունները: Ասվածից հետևում է, որ որպես դրա իրագործման արդյունավետ միջոց կարող է ծառայել տարրերի ջերմային տեղաբաշխումը:

Այսպիսով, միկրոէլեկտրոնային տեխնոլոգիաների մասշտաբավորմանը զուգընթաց, անհրաժեշտ է լինում անընդհատ կատարելագործել ԻՄ-երի ավտոմատացված ֆիզիկական նախագծման մեթոդները և մշակել այնպիսի նոր մեթոդներ, որոնք հնարավորինս հաշվի կառնեն դրանց սխեմատեխնիկական առանձնահատկությունները:

1.4. Խնդրի դրվածքը

Տրված է՝

1. Նախագծվող սխեմայի նկարագրությունը փականային մակարդակի շղթաների ցուցակի տեսքով՝ $G(C, W)$, որտեղ $C = \{c_1, c_2, \dots, c_m\}$ –ը տարրերի բազմությունն է, $W = \{w_1, w_2, \dots, w_n\}$ –ը շղթաների բազմությունը:

Աշխատանքում օգտագործված է տրամաբանական սխեմայի փականային մակարդակի Verilog նկարագիրը, որը պարունակում է՝

- մուտքային շղթաների բազմությունը. $W_i, i = \overline{1, k}$; k -ն մուտքային շղթաների քանակն է,
- ելքային շղթաների բազմությունը. $W_o, O = \overline{k + 1, q}$; q -ն ելքային և մուտքային շղթաների գումարային քանակն է,
- ներքին շղթաների բազմությունը. $W_i, i = \overline{q + 1, n}$; n -ը բոլոր շղթաների քանակն է,
- տարրերը. $C_i, i = \overline{1, m}$; m -ը տարրերի քանակն է,
- յուրաքանչյուր տարրի մուտքային և ելքային շղթաները:

2. Թվային ստանդարտ բջիջների գրադարանը, որի հենքով պետք է սինթեզվի նախագծվող սխեմայի ֆիզիկական կառուցվածքը:

Ստանդարտ բջիջների գրադարանի տվյալներից օգտագործվում են՝

- տարրերի երկրաչափական չափսերը՝ $\{a_i, h_i\}, i = \overline{1, m}$, որտեղ a_i և h_i համապատասխանաբար c_i տարրի երկարությունը և բարձրությունն են,
- տարրերի տեսակարար դինամիկ հզորությունները գրադարանի անվանական արագագործության դեպքում՝ $P_{i\text{տղգ}}, i = \overline{1, m}$,
- թվային ստանդարտ բջիջների գրադարանի աշխատանքային հաճախությունը:

3. Շղթաների ժամանակի պահուստները՝ $R_i, i = \overline{1, n}$, որոնց արժեքները վերցվում են սխեմայի ստատիկ ժամանակային վերլուծության արդյունքում:

4. Սինթեզվող բլոկի կողերի չափսերի հարաբերակցությունը:

Պահանջվում է՝

մշակել տրված սխեմայի ֆիզիկական կառուցվածքի ավտոմատացված սինթեզի միջոց, որը պետք է՝

- հաշվի առնի սխեմայի աշխատանքի տրամաբանական առանձնահատկությունները,
- նպաստի ԻՄ-ի միջմիացումներով պայմանավորված արագագործության բարձրացմանը,
- նպաստի ԻՄ-ի ջերմային հուսալիության բարձրացմանը:

Խնդրի դրվածքից բխող ենթախնդիրները ներկայացված են ստորև:

1. Սխեմայի Verilog նկարագրի փոխարկումը ազդանշանների հոսքի գրաֆի:
2. Սխեմայի տարրերի ռանգավորումը՝ ըստ աշխատանքի տրամաբանության հաջորդականության:
3. Ֆիզիկական «փափուկ» կառուցվածքով բլոկի ավտոմատացված սինթեզը՝ հաշվի առնելով սխեմայի աշխատանքի տրամաբանությունը:
4. Շղթաների ժամանակի պահուստներից դրանց կարևորության գործակիցների ստացումը՝ համաձայն (1.12) և (1.13)–ի:
5. Ֆիզիկական «կոշտ» կառուցվածքով բլոկի ավտոմատացված սինթեզը՝ հաշվի առնելով շղթաների կարևորության գործակիցները:
6. Ֆիզիկական «կոշտ» կառուցվածքով բլոկի ավտոմատացված սինթեզը՝ հաշվի առնելով ջերմային ռեժիմը:
7. Սխեմայի տարրերի փոխանջատման ակտիվությունների՝ $k_i, i = \overline{1, m}$, n , գնահատումը:
8. Սխեմայի տարրերի աշխատանքային դինամիկ հզորությունների գնահատումը:
9. Տրամաբանական տարրերի դինամիկ հզորության հաշվառմամբ՝ տարրերի ջերմային տեղաբաշխումը:

Այսպիսով, ԻՄ-երի ավտոմատացված ֆիզիկական նախագծման փուլում սխեմատեխնիկական առանձնահատկությունները հաշվի առնելու հիմնական միջոց կարող է ծառայել տարրերի տեղաբաշխման խնդրի լուծման այնպիսի միջոցների մշակումը, որոնց դեպքում հաշվի կառնվեն սխեմայի ֆունկցիավորման վրա ազդող առավել կարևոր պարամետրերը:

Եզրակացություններ

1. ԻՄ-երի մասշտաբավորմանը զուգընթաց՝ աճում են դրանց սխեմատեխնիկական և ֆիզիկական պարամետրերի միջև փոխկապվածությունները:
2. Ժամանակակից նանոչափական ԻՄ-երի որոշ ֆիզիկական պարամետրերի ազդեցությունը դրանց աշխատանքի վրա դարձել է որոշիչ: Այդ պարամետրերից իրենց կարևորությամբ առանձնանում են միջմիացումներով պայմանավորված հապաղումները և տեսակարար հզորության աճով պայմանավորված ջերմային ռեժիմները:
3. Նշված խնդրի առկա լուծումները բավարար արդյունքների չեն հանգեցնում և կարիք ունեն զարգացման:
4. Միկրոէլեկտրոնային տեխնոլոգիաների մասշտաբավորմանը զուգընթաց, անհրաժեշտություն է առաջանում անընդհատ կատարելագործել ԻՄ-երի ավտոմատացված ֆիզիկական նախագծման մեթոդները և մշակել այնպիսի նոր մեթոդները, որոնց դեպքում հնարավորինս հաշվի կառնվեն դրանց սխեմատեխնիկական առանձնահատկությունները:
5. ԻՄ-երի ավտոմատացված ֆիզիկական նախագծման փուլում սխեմատեխնիկական առանձնահատկությունները հաշվի առնելու հիմնական միջոց կարող է ծառայել տարրերի տեղաբաշխման խնդրի լուծման այնպիսի միջոցների մշակումը, որոնց դեպքում հաշվի կառնվեն սխեմայի ֆունկցիավորման վրա ազդող առավել կարևոր պարամետրերը:

**ԳԼՈՒԽ 2. ՄԽԵՄԱՏԵԽՆԻԿԱԿԱՆ ԱՌԱՆՁՆԱՀԱՏԿՈՒԹՅՈՒՆՆԵՐԻ
ՀԱՇՎԱՌՄԱՄԲ ԻՆՏԵԳՐԱԼ ՄԽԵՄԱՆԵՐԻ ԱՎՏՈՄԱՏԱՑՎԱԾ
ՖԻԶԻԿԱԿԱՆ ՆԱԽԱԳԾՄԱՆ ՄԵԹՈԴՆԵՐԻ ՄՇԱԿՈՒՄԸ**

**2.1. ԻՄ-երի տարրերի ժամանակային ուղղորդմամբ տեղաբաշխումը՝ որպես
սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ ֆիզիկական
կառուցվածքի ստացման միջոց**

ԻՄ- երի տեխնոլոգիական չափերի նվազեցման և դրանց ինտեգրման մեծացմանը զուգընթաց միջմիացումները դառնում են թվային ԻՄ-երի արագությունը որոշող գերիշխող գործոն: Նանոչափային թվային ԻՄ-երում ազդանշանի տարածման գումարային հապաղումը միջմիացումներում կարող է հասնել տասնյակ և հարյուրավոր տրամաբանական փականների ընդհանուր հապաղմանը [6]: Եթե տրամաբանական բջիջներում հապաղումները հայտնի են դառնում, երբ ընտրվում է ստանդարտ բջիջների որոշակի գրադարան, որի հենքով կատարվում է տրամաբանական սինթեզը, ապա միջմիացումներում հապաղումները որոշվում են միայն ֆիզիկական նախագծման արդյունքում, այսինքն՝ տարրերի տեղաբաշխումից և միջմիացումների ծրագծումից հետո:

Առաջին մոտավորությամբ՝ միջմիացումների հապաղումները որոշվում են RC կապի գծերի պարամետրերով, որոնք, իրենց հերթին, հիմնականում կախված են դրանց երկարությունից: Հաշվի առնելով, որ միջմիացումների ծրագծման արդյունքները, որոնք որոշում են դրանց վերջնական երկարությունները, մեծապես կախված են բջիջների տեղաբաշխման արդյունավետությունից, ակնհայտ է դառնում, որ ծրագծման հաջողությունը և միջմիացման երկարությունը հիմնականում կախված են տեղաբաշխման արդյունքներից: Հետևաբար, տեղաբաշխումը կարող է մեծապես նպաստել սխեմայի արագագործության ապահովմանը: Այդ պատճառով, արագագործության ապահովման տեսանկյունից, ներկայումս ԻՄ- երի ֆիզիկական նախագծման կարևորագույն խնդիրներից մեկը տարրերի տեղաբաշխման ժամանակի հապաղումների հաշվառումն է, այսինքն՝ զլոբալ իմաստով՝ ֆիզիկական նախագծման

փուլում ԻՄ-երի աշխատանքի սխեմատեխնիկական առանձնահատկությունների հաշվառումը:

ԻՄ-երի տարրերի տեղաբաշխման ժամանակ միջմիացումների արագագործության հաշվման խնդիրը սովորաբար անվանում են ժամանակային ուղղորդմամբ տեղաբաշխում [19]: Ներկայումս էլեկտրոնային նախագծման ավտոմատացման համակարգերում ներդրվում են համապատասխան գործիքներ, որոնք նպաստում են ժամանակային ուղղորդմամբ տեղաբաշխմանը: Գոյություն ունեցող ժամանակային ուղղորդմամբ տեղաբաշխման մեթոդները կարող են խմբավորվել ըստ երկու դասերի՝ ուղու վրա հիմնված և շղթայի վրա հիմնված [3,20,40]:

Այս պայմաններում ԻՄ-երի նախագծման մեթոդներն գործիքներն նոր պահանջներ են առաջադրվում: Նախևառաջ՝ սխեմատեխնիկական և ֆիզիկական նախագծման փուլերի միջև փոխկապվածության ամրապնդումը: Դա պայմանավորված է այն հանգամանքով, որ տեխնոլոգիական մասշտաբավորումների հետ միասին էլեկտրական և ֆունկցիոնալ բնութագրերի վրա ֆիզիկական պարամետրերի ազդեցությունը մեծանում է: Ինչպես նշվել է նախորդ բաժնում, ֆիզիկական նախագծման փուլում սխեմատեխնիկական առանձնահատկությունների հաշվառման տեսակետից գլխավոր խնդիրը միջմիացումների հապաղումներով պայմանավորված արագագործության ապահովումն է նախագծվող սխեմայի ֆիզիկական կառուցվածքի սինթեզի ժամանակ: Այս խնդրի լուծումը կատարվում է սխեմայի տարրերի ժամանակային ուղղորդմամբ տեղաբաշխման ճանապարհով:

Ժամանակային ուղղորդմամբ տեղաբաշխման առկա մեթոդները կարելի է դասակարգել երկու խմբի [3,7,20,41]՝

- Ուղու վրա հիմնված մեթոդներ, որոնք հաշվի են առնում սխեմայի առաջնային մուտքերից մինչև ելքերը տանող ուղիներում ազդանշանի հապաղումները:
- Շղթայի վրա հիմնված մեթոդներ, որոնք հաշվի են առնում ազդանշանի հապաղումը սխեմայի յուրաքանչյուր շղթայում:

Էլեկտրոնային նախագծման ավտոմատացման արդյունաբերական համակարգերում, որպես կանոն, օգտագործվում է առաջին մոտեցումը, որը ենթադրում է նախագծվող սխեմայի որոշիչ «մուտք-ելք» ուղիների գնահատումը՝ հետագայում այս

ուղիների հապաղումների նվազեցմամբ [8]: Այս մոտեցման թերությունն այն է, որ որոշիչ ուղիներում հապաղումների նվազեցումը կարող է հանգեցնել նոր որոշիչ ուղիների առաջացման, իսկ նոր ուղիների հապաղումները հաճախ գերազանցում են նախորդ ուղիների հապաղումներին: [41] աշխատանքում ուղու վրա հիմնված մոտեցմանը ցուցաբերած է յուրահատուկ մոտեցում, որը 1.3 բաժնում բերված ժամանակային պարամետրերի հաշվարկ տարրերի ռանգավորմամբ փոխարինելն է: Սակայն այն չի տալիս տարրերի վերջնական տեղաբաշխման մեխանիզմը: Ներկայումս այս խնդիրը լուծվում է բարելավման բազմակրկնակի ալգորիթմների կիրառմամբ, որոնք հիմնված են միջմիացումների որոշիչ ուղիների երկարությունների աստիճանաբար նվազեցումով տարրերի ռեկուրսիվ տեղաբաշխման վրա՝ մինչև հապաղումների ընդունելի արժեքներին հասնելը: Բազմակրկնակի ալգորիթմների հիմնական թերությունները համակարգչային մեծ ժամանակի անհրաժեշտությունը և նախնական տեղաբաշխման արդյունքներից խնդրի լուծման արդյունավետության կախվածությունն են:

Շղթայի վրա հիմնված մոտեցման դեպքում, տարրերի տեղաբաշխման գործընթացին զուգահեռ, իրականացվում է որոշիչ ուղիների վրա գտնվող անհատական շղթաների երկարությունների հարաբերական նվազեցում: Այս մոտեցման հիմնական թերությունն այն է, որ անհատական շղթաներում դժվար է սահմանել հապաղումների թույլատրելի սահմանները, և կիրառել դրանք տեղաբաշխման խնդրի լուծման ժամանակ՝ միջմիացումների երկարությունները վերահսկելու համար:

Ուղու վրա հիմնված մեթոդների առավելությունն է գլոբալ օպտիմումին հասնելու պոտենցիալ հնարավորությունը, իսկ թերությունն է մեծ ալգորիթմական բարդությունը:

Հակառակ դրան՝ շղթայի վրա հիմնված մեթոդների առավելությունն է փոքր ալգորիթմական բարդությունը՝ $O(n \log(n))$, իսկ թերությունը՝ լոկալ օպտիմումի առկայությունը:

Վերոհիշյալ երկու մոտեցումներն ունեն մեկ ընդհանուր թերություն, այն է, որ դրանք հաշվի չեն առնում թվային սխեմաների տրամաբանությունը սխեմայի տարրերի աշխատանքի տրամաբանական հաջորդականությունը տեղաբաշխելիս: Հետևաբար, երկու դեպքում էլ թույլ է արտահայտված տեղաբաշխման արդյունքի և սխեմայի աշխատանքի տրամաբանական կապը:

Նշված թերության հաղթահարման նպատակով այս ատենախոսությունում առաջարկված և մշակված է տրամաբանական բլոկների ֆիզիկական կառուցվածքի սինթեզի նոր մոտեցում, որը հիմնված է տրամաբանական սխեմայի ռանգավորման վրա և միտված է միջմիացումների երկարության և, հետևաբար, դրանցում հապաղումների նվազեցմանը:

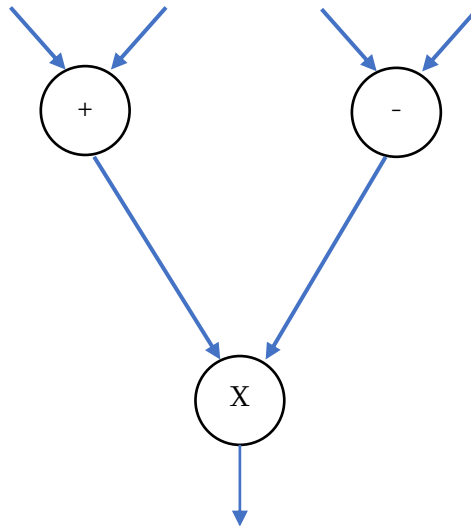
ԻՄ ֆիզիկական նախագծման ժամանակ սխեմայի ֆունկցիոնալ-տրամաբանական առանձնահատկությունների հաշվառման առաջարկվող մոտեցումը ենթադրում է հետևյալ խնդիրների հաջորդական լուծում.

- նախագծվող սխեմայի տրամաբանական մակարդակի փականային Verilog նկարագրության փոխակերպում ազդանշանների հոսքի գրաֆի (ԱՀԳ-ի),
- սխեմայի տարրերի ռանգավորում,
- տարրերի տեղաբաշխումը՝ համաձայն դրանց ռանգավորման, այսինքն՝ «փափուկ» բլոկի ստացում,
- ստացված «փափուկ» բլոկի տարրերի վերատեղաբաշխում՝ հաշվի առնելով կոնստրուկտիվ սահմանափակումները, այսինքն՝ «կոշտ» բլոկի ստացում:

Այսպիսով, կարևորվում է ԻՄ-երում տարրերի ժամանակային ուղղորդմամբ տեղաբաշխումը՝ որպես սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ ֆիզիկական կառուցվածքի ստացման միջոց: Ներկայումս գոյություն ունեցող ժամանակային ուղղորդմամբ տեղաբաշխման մեթոդները[3,20,40] ունեն որոշակի թերություններ: Առաջարկվում է նոր մոտեցում՝ ԻՄ ֆիզիկական նախագծման ժամանակ սխեմայի ֆունկցիոնալ-տրամաբանական առանձնահատկությունների հաշվառման համար:

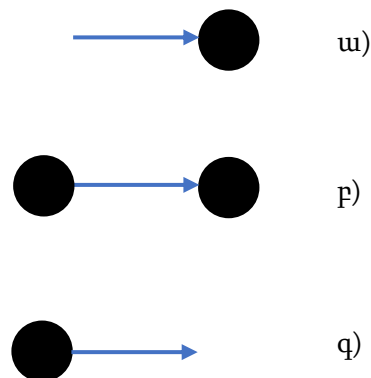
2. 2. Տրամաբանական սխեմայի Verilog նկարագրից ազդանշանների հոսքի գրաֆի ստացման ավտոմատացված միջոցի մշակումը

Ազդանշանների հոսքի գրաֆը բաղկացած է հանգույցներից (գագաթներից) և ուղղորդված կողերից, որտեղ ուղղորդված կողերը ներկայացնում են տվյալների կախվածությունը: Նկ. 2. 1-ում պատկերված է (ԱՀԳ-ի) պարզագույն մոդել:



Նկ. 2.1. ԱՀԳ-ի պարզագույն օրինակ

ԱՀԳ-ում հանգույցները ներկայացնում են տվյալների գործողություններ, օրինակ՝ հաշվողական գործողություններ [42,43]: Որոշ կողեր վերջանում են ինչ-որ հանգույցում, բայց կարող են չսկսվել որևէ հանգույցից և հակառակը: Հետևաբար, դրանց կարելի է



Նկ. 2.2. ԱՀԳ-ում կողերի դասակարգումը՝ ա) դասակարգել 3 տեսակի՝ մուտքային, միջանկյալ, գ) ելքային

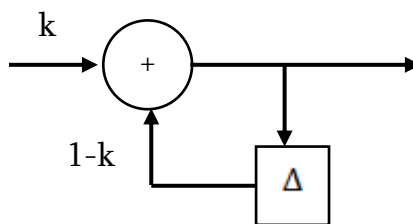
դասակարգել 3 տեսակի՝ մուտքային, միջանկյալ և ելքային (նկ. 2. 2):

ԱՀԳ-երը կարող են ներկայացվել ծառերի տեսքով, այսինքն՝ ացիկլիկ գրաֆի տեսքով: Դա թույլ է տալիս ավելի հեշտ վերլուծել գրաֆը, բայց սահմանափակում է դրանց օգտագործումը: Հիմնականում տվյալների հոսքի գրաֆն օգտագործում են կոմպիլյատորներում: Անկախ այն բանից՝ կատարվում են այդ բոլոր գործողությունները հատ առ հատ, թե մի քանիսը միաժամանակ, գործողությունների

ցանկացած հերթականություն ընդունելի է, եթե դրանք բավարարում են տվյալների կախվածության պահպանմանը: Նկատենք նաև, որ ՏՀԳ-ն ինքն իրենով վերջավոր է. այն ոչ միայն ունի վերջավոր քանակով հանգույցներ, այլև չունի ցիկլեր, հետևաբար՝ ՏՀԳ-ում ցանկացած տվյալների ճանապարհ ունի վերջավոր երկարություն: Տվյալների հոսքի գրաֆի ավելի կոնկրետացված եղանակ է ազդանշանների հոսքի գրաֆը (ԱՀԳ), որը սովորաբար օգտագործվում է ազդանշանների մշակման (signal processing) համար[44]: ԱՀԳ-ն ցիկլիկ ուղղորդված գրաֆ է: Նկ. 2.3 -ում ցույց է տրված, որ ԱՀԳ-ում ավելանում է նոր հանգույցի տեսակ՝ հապաղում, որը Δ սիմվոլով հանգույցն է: Եթե ընդունենք, որ այս հապաղումը t_1 է մուտքային S ազդանշանի համար, ապա գործողության արդյունքը ընդհանուր հապաղումով կլինի՝

$$D(t) = S(t - t_1) \quad (2.1)$$

ԻՄ-ների ժամանակային տեղաբաշխում կատարելու դեպքում նշված հապաղման հանգույցները չեն կիրառվել, բայց փոխարենը նրանք հաշվի են առնվում միջմիացումների երկարությունների հետ միասին:



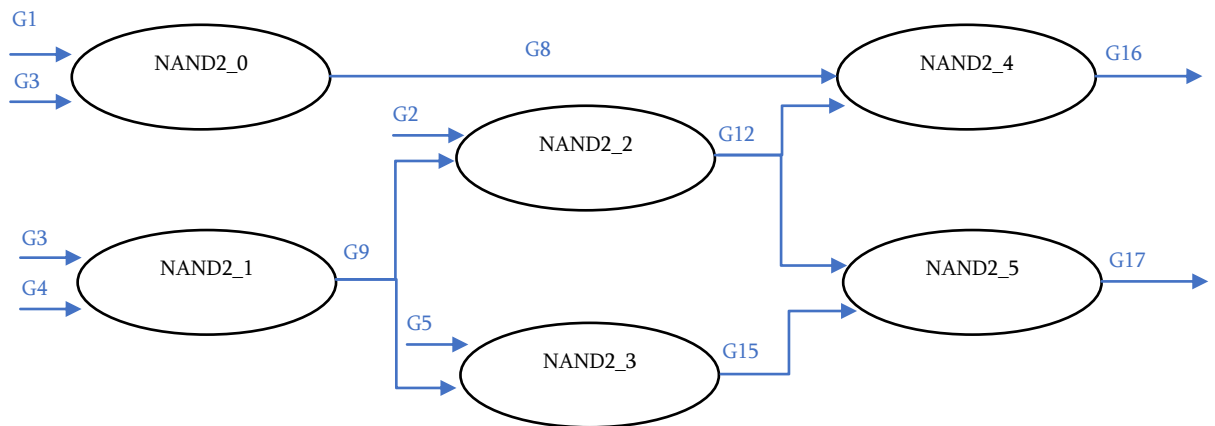
Նկ. 2.3. Ազդանշանի հոսքի գրաֆի հապաղման հանգույցով օրինակ

Դինամիկ համակարգերի մաթեմատիկական մոդելավորման ծրագրերը վերաբերում են գրաֆային միջավայրի հիերարխիկ կառուցվածքային բարձր մակարդակի ծրագրերի ստեղծմանը և դրանց մի մասը հիմնված է ղեկավարման հոսքային մոդելի վրա[45]: Ղեկավարման հոսքային մոդելը ծրագրավորման մոդել է, որտեղ հրամանները, պրոցեսորներն ու ֆունկցիաները կատարվում են միայն այն ժամանակ, երբ բոլոր մուտքային տվյալները (փոփոխականներ, արգումենտներ) պատրաստ են: Ծրագրեր գրելու (մոդելների ստեղծման) համար օգտագործվում են գրաֆիկական լեզուներ, որոնց միջոցով իրականացվում է տվյալների գործընթացների՝ ֆունկցիոնալ սխեմաների, բլոկ

սխեմաների, տեսքի վերափոխման նկարագրությունը: Դիտարկենք տրամաբանական մակարդակի նկարագրության վերափոխման օրինակ, հիմնված ISCAS85 գրադարանի c17 verilog նկարագրության վրա [46]: Ունենք՝

```
nand NAND2_1 (N10, N1, N3);
nand NAND2_2 (N11, N3, N6);
nand NAND2_3 (N16, N2, N11);
nand NAND2_4 (N19, N11, N7);
nand NAND2_5 (N22, N10, N16);
nand NAND2_6 (N23, N16, N19);
```

Ազդանշանների հոսքի գրաֆը պատկերված է նկ.2.4-ում:



Նկ. 2.4. ISCAS85 գրադարանի c17 verilog տրամաբանական նկարագրության ազդանշանների հոսքի գրաֆը

Վերլուծելով նկ. 2. 4-ը, կարելի է նկատել, որ ցանկացած տեղեկատվական հոսքում (G1,G8,G16...) տվյալները տարածվում են ազդանշանային աղբյուրներից դեպի ընդունիչները (NAND2_0, NAND2_4...): Ակնհայտ է, որ մեկ հոսքում կարող են լինել ճյուղավորումներ, զուգահեռ ալիքներ և հակադարձ կապեր: Զուգահեռ հոսքերը կարող են լինել մեկը մյուսից՝ կախված և անկախ: Ազդանշանների հոսքի գրաֆը այսպիսի տեսքով ներկայացնելու համար օգտագործվում են մի քանի տեսակ բլոկներ: Դրանցից են՝ ազդանշանների աղբյուրները, ազդանշանների ընդունիչները և միջանկյալ բլոկները, որոնք հանդիսանում են առաջին երկուսի համախումբը: Այդ բլոկներն իրենց հերթին լինում են մաթեմատիկական գործողություններ՝ հաշվողական, տրամաբանական, մատրիցային, ոչ գծային և այլն: Verilog նկարագիր սինթեզելու համար պետք է օգտվել

SAED90 [47] կամ այլ գրադարանից, որտեղ նշված են ԻՄ-երի տրամաբանական նկարագրության բոլոր տեսակի բլոկները՝ NAND, AND, NOT, OR, NOR և այլն:

Գոյություն ունեն տարբեր մեթոդներ՝ սինթեզելու համար սխեմայի Verilog նկարագիրը: Դրանցից ամենաարդի եղանակներից են մատրիցային և գրաֆային նկարագրությունները: Մատրիցային եղանակը, ի տարբերություն գրաֆայինի, օգտագործում է շատ մեծ հիշողություն, բայց նաև թույլ է տալիս հաստատուն ժամանակում կարդալ ցանկալի տարրում գրված ինֆորմացիան: Գրաֆայինը, իր հերթին, կարող է ներկայացվել հոսքային տեսքով՝ տվյալների հոսքի, ազդանշանների հոսքի, գրաֆներով: Փականային մակարդակի Verilog նկարագրի փոխակերպման համար կիրառվել է այս երկու մոտեցումների համախումբը, որով հնարավոր է դարձել խուսափել անցանկալի մեծ հիշողության օգտագործումից և ստանալ ցանկալի մեծ արագագործություն (ալգորիթմական լուծումը ներկայացված է 3-րդ գլխում):

Այսպիսով, նկարագրվել է ազդանշանների հոսքի գրաֆի աշխատանքի սկզբունքը, և կարևորվել է սխեմայի Verilog նկարագրից ԱԳՀ-ի ստացման այնպիսի մոտեցման մշակումը, որով շնորհիվ հնարավոր կլինի խուսափել անցանկալի մեծ հիշողության օգտագործումից և ստանալ ավտոմատացված միջոցի ցանկալի մեծ արագագործություն:

2.3. Սխեմայի տարրերի ռանգավորումը՝ որպես սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ ֆիզիկական կառուցվածքի ստացման միջոց

Տրամաբանական մոդելավորման ժամանակ օգտագործվում են ֆունկցիոնալ մոդելների բաղադրիչներ՝ տարրեր: Ֆունկցիոնալ մոդելը հանդես է գալիս «սև արկղի» տեսքով, որի համար մուտքային և ելքային ազդանշանների միջև կապը տրվում է բուլյան հավասարումներով կամ ճշտության աղյուսակներով [48]: Այդպիսի մոդելներում ներքին կառուցվածքը չի հաշվի առնվում: Կոմբինացիոն տրամաբանական տարրական մոդելներից են՝ AND, OR, NOT, NAND և այլն, որոնք համապատասխանում են բուլյան ֆունկցիաներին: Ժամանակային հապաղումները հաշվի առնող մոդելներում ստեղծվում են ազդանշանների հապաղումների մեծություններ՝ տարրերի մեկից մյուսն անցում կատարելիս:

Տրամաբանական մոդելավորման մեթոդներ: Տրամաբանական մոդելավորման ժամանակ լուծվում են հետևյալ խնդիրները՝ սխեմայի աշխատանքի տրամաբանության ստուգում, անցումային գործընթացների վերլուծում, սխեմայի աշխատանքի հուսալիության որոշում՝ կախված տարրերի պարամետրերի բաշխումից: Մոդելավորման եղանակն ընտրվում է՝ հաշվի առնելով խնդրի դրվածքը: Եղանակների հիմնական առանձնահատկություններից են՝ սխեմայում ազդանշանի ժամանակի տարածումը հաշվի առնելու և համակարգչով մոդելն իրականացնելու հնարավորությունները [48]: Ազդանշանի ժամանակային հոսքի եղանակից կախված մեթոդները լինում են սինխրոն (առանց սխեմայի տարրերում ժամանակային հապաղումները հաշվի առնելու) և ասինխրոն (հապաղումների հաշվառմամբ) [49]: Նկ.2.5-ում բերված սխեմայի տարրերի մաթեմատիկական մոդելը կլինի՝

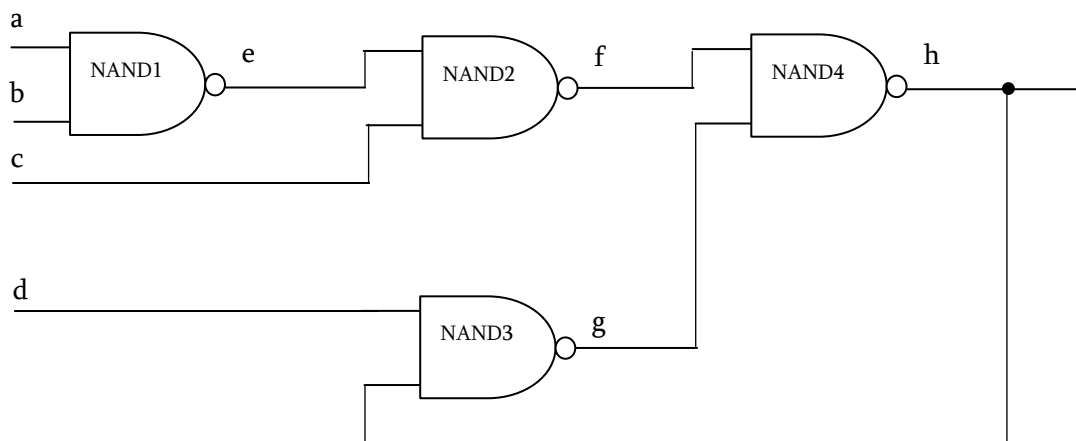
$$y(t) = \overline{x_1(t) \cap x_2(t)}: \quad (2.2)$$

Այդ դեպքում տրված սխեման (նկ. 2. 5) ունի հետևյալ տրամաբանական հավասարումների համախումբը՝

$$e = \overline{a \cap b}; f = \overline{e \cap c}; h = \overline{f \cap g}; g = \overline{d \cap h} \quad (2.3)$$

(2.3)-ում ներկայացված արտահայտությունները ԻՄ-ի փականային մակարդակի verilog նկարագրության համար կլինեն՝

```
nand NAND1(e,a,b); nand NAND2(f,e,c);
nand NAND3(g,d,h); nand NAND4(h,f,g);
```



Նկ. 2.5. Տրամաբանական սխեմայի

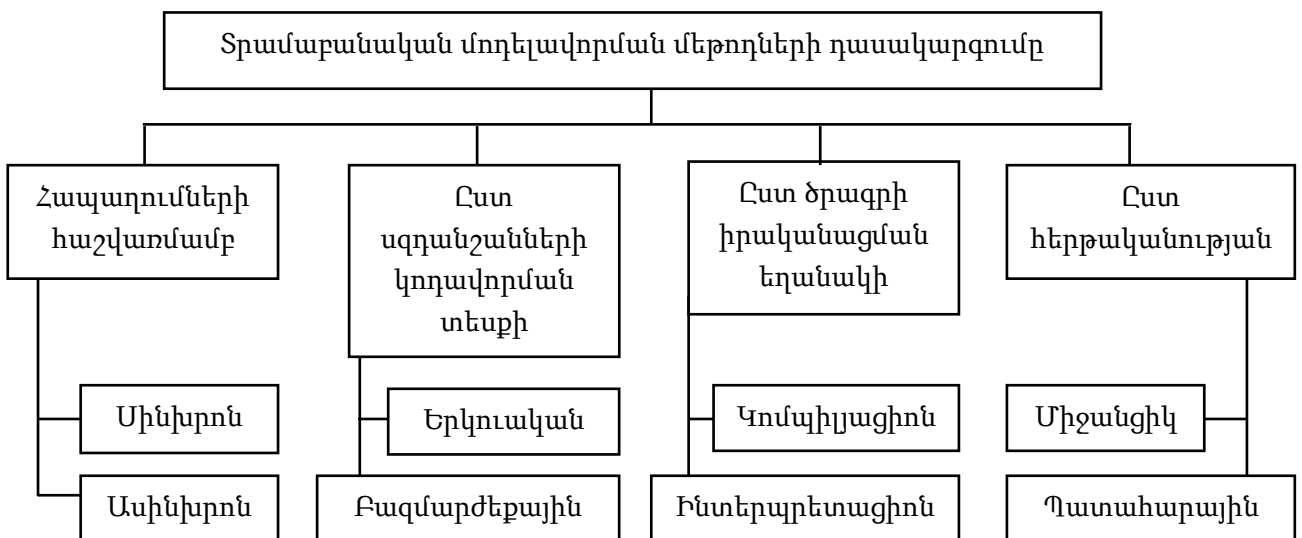
Ավելի ընդհանուր մաթեմատիկական մոդելը կարելի է ներկայացնել հետևյալ կերպ՝

$$V' = F(V, U), \quad (2.4)$$

որտեղ U -ն մուտքային փոփոխականների վեկտորն է, V -ն՝ ելքային և միջանկյալ փոփոխականների վեկտորը ժամանակի t պահին, V' -ն՝ սխեմայի մուտքային և ելքային փոփոխականների վեկտորը ժամանակի՝ $t' = t + \Delta t$ պահին, F -ը՝ դիսկրետ փոփոխականների փոխակերպման օպերատորը [48]: Փոփոխականները՝ U և V վոկտորներում դիսկրետ մեծություններ են: Տրամաբանական մոդելավորման մեջ կիրառվում են մոդելներ՝ փոփոխականների հնարավոր տարբեր քանակի արժեքներով: Երկուական համակարգի մոդելներում U և V -ն բուլյան փոփոխականներ են:

Տրամաբանական սինխրոն մոդելավորում: Սինխրոն մոդելավորման դեպքում, տարրերի մոդելները ներկայացվում են առանց ազդանշանների հապաղումների հաշվառման: Դա կիրառվում է՝ գնահատելու համար տրամաբանական ֆունկցիոնալության ճշտությունը: Սխեմայի աշխատանքի մոդելավորումը սխեմայի տրամաբանական տարրերում հանգեցվում է՝ տրված մուտքային ազդանշաններով հաշվարկելու ելքային ազդանշանների արժեքները հաշվարկելուն:

Նկ. 2. 6 -ում բերված է տրամաբանական մոդելավորման մեթոդների դասակարգումը ԻՍ-ի վերլուծության համար բերված մեթոդներից, ամենահարմար տարբերակը սինխրոնային մոդելավորման կիրառումն է [48,50]:



Նկ. 2. 6. Տրամաբանական մոդելավորման մեթոդների դասակարգումը

Կախված ազդանշանների տարածման հապաղման հաշվառման մեթոդից՝ տրամաբանական մոդելավորման մեթոդները լինում են սինխրոն, երբ հաշվի չեն առնվում տրամաբանական տարրերի հապաղումները, և ասինխրոն, երբ դրանք հաշվի չեն առնվում:

Այս դեպքում մեդելավորման արդյունքը ճիշտ համընկնում է իրական սխեմայի հետ: Սխեմայի մաթեմատիկական մոդելը կունենա հետևյալ տեսքը՝

$$V = F(V, U): \quad (2.5)$$

Եթե ինտեգրալ սխեման չունենա հետադարձ կապեր, ապա հարաբերակցությունը կլինի՝

$$V = F(U): \quad (2.6)$$

Սխեմայի ամեն տարր նկարագրվում է հետևյալ տրամաբանական արտահայտությամբ՝

$$Y = f(x_1, x_2, \dots, x_n), \quad (2.7)$$

որտեղ Y -ը տարրի ելքային, իսկ $x_1, x_2, \dots, x_n$ -ը՝ մուտքային ազդանշաններն են: Արդյունքում՝ ամբողջ սխեման նկարագրվում է հետևյալ հարաբերակցությամբ: Սինխրոն մոդելավորումը հանգում է դրանց՝ մեկը մյուսից հետո հաջորդաբար հաշվարկին: Տրամաբանական արտահայտության հաշվարկումը ներկայացնում է Y -ի տրամաբանական արժեքի հաշվարկումը՝ հայտնի $x_1, x_2, \dots, x_n$ տրամաբանական արժեքներով: Սինխրոն մոդելավորման առանձնահատկություններից է տրամաբանական արտահայտությունների հաշվարկումը որոշակի հերթականությամբ, որը համապատասխանում է սխեմայում տարրերի ազդանշանների հոսքի հերթականությանը: Այդ հերթականությունը որոշելու համար սխեման անհրաժեշտ է ռանգավորել (նկ. 2. 6), որպեսզի այն պահին, երբ տեղի ունենա ամեն Y -ի հաշվարկը՝ ըստ (2.7) արտահայտության, բոլոր $x_1, x_2, \dots, x_n$ -երի արժեքները լինեն հայտնի: Սա նաև թույլ կտա, որ իրար հետ խիստ կապված տարրերը դրվեն իրար մոտ, ինչն էլ, իր հերթին, կնպաստի միջմիացումների երկարությունների զգալի կրճատմանը:

Ռանգավորման ժամանակ հաշվի է առնվում, որ սխեմայի մոդելավորումը սկսելու պահին դրա բոլոր մուտքային ազդանշանները հայտնի են, և դրանց բոլորին տրվում է 0 ռանգ $r = 0$: Օրինակ, նկ. 2. 6-ում բերված սխեմայի ազդանշանների որոշման հերթականությունը կլինի հետևյալը՝

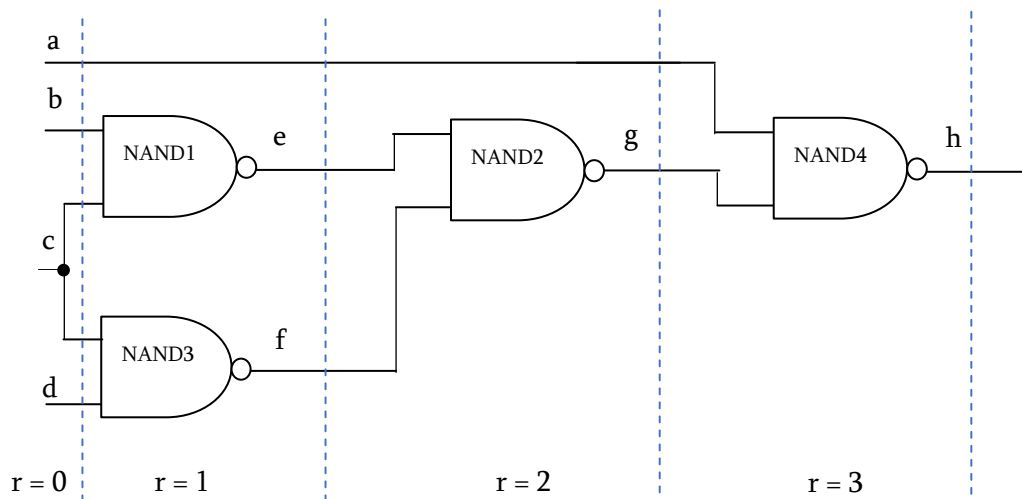
$$1) e = b \cap c; f = c \cap d; r = 1,$$

$$2) g = e \cap f; r = 2,$$

$$3) h = a \cap g; r = 3,$$

որտեղ a, b, c, d -ն սխեմայի մուտքային ազդանշաններն են, e, f, g, h -ը տարրերի ելքային ազդանշաններն են (սխեմայի ելքային և ներքին փոփոխականներն են), r -ը տարրերի ռանգը:

Ռանգավորումը սկսվում է այն բանից, երբ բոլոր մուտքային փոփոխականներին տրվում է 0 ռանգ, որից հետո որոշվում են 1-ին ռանգի տարրերը, 1-ին ռանգի շղթայի փոփոխականները, 2-րդ ռանգի տարրերը և այդպես շարունակ (ալգորիթմի աշխատանքը նկարագրված է 3-րդ գլխում): Արդյունքում՝ սխեմայի բոլոր տարրերը տեղաբաշխվում են ռանգերի աճման կարգով (նկ. 2. 7):



Նկ. 2.7. Տրամաբանական սխեմայի ռանգավորված օրինակ

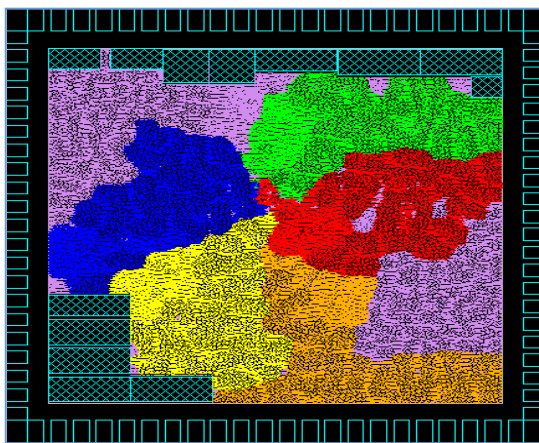
Ռանգավորումն ամեն ինչ ավելի բարդանում է, երբ առկա է հետադարձ կապով սխեմա (նկ. 2. 5): Այս դեպքում տեղի է ունենում արհեստական հետադարձ կապի խզում, և այն տարրի մուտքին, որին կպած էր հետադարձ կապը, տրվում է 0 ռանգ:

Այսպիսով, նկարագրվել և կարևորվել է սխեմայի տարրերի ռանգավորումը՝ որպես սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ ֆիզիկական կառուցվածքի ստացման միջոց:

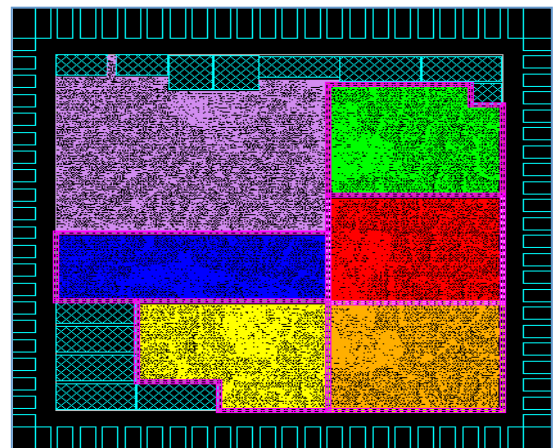
2.4. Ինտեգրալ սխեմաների ֆիզիկական կառուցվածքների ներկայացումը «փափուկ» և «կոշտ» բլոկների տեսքով

Ռանգավորումից հետո լուծվում է տարրերի տեղաբաշխման խնդիրը՝ սկզբում «փափուկ» և այնուհետև «կոշտ» բլոկների տեսքով:

«Կոշտ» և «փափուկ» բլոկները, որոնք հաճախ անվանվում են նաև մակրոսներ, առավելապես կիրառվում են որպես մտավոր սեփականության (ՄՍ) (IP - Intellectual property) բլոկների ներկայացման ձևեր: Դրանք պարամետրացվում են իրենց ֆիզիկական չափերով, հզորությունով և արագագործությամբ: Նշենք, որ ըստ ֆիզիկական չափերի պարամետրացումը «կոշտ» բլոկների դեպքում կատարվում է ինչպես մակերեսով (A), այնպես էլ ուղղանկյան երկարության (w) և բարձրության (h) հարաբերակցությամբ, իսկ «փափուկ» բլոկների դեպքում՝ միայն մակերեսով: Այս բլոկները գնալով ավելի մեծ կիրառություն են գտնում տարաբնույթ համակարգային ԻՍ-երի նախագծերում: «փափուկ» և «կոշտ» բլոկների օրինակներ են բերված նկ. 2.8-ում [51]:



ա



բ

Նկ. 2. 8 «Փափուկ» մակրոսներ (ա) մինչև պարամետրացումը և «կոշտ» մակրոսներ (բ) պարամետրացումից հետո

Նկարներից երևում է, որ առանձին ֆիզիկական դոմենների տիրույթները, պարամետրացումից հետո ստանում են կանոնավոր ձևեր:

ԻՄ-երի ինտեգրման աճին զուգընթաց՝ դրանցում ավելանում է մտավոր սեթիականության (ՄՄ) բլոկներով իրագործվող ֆունկցիոնալությունը, որը ներկայումս կարող է հասնել ամբողջ ֆունկցիոնալության ավելի քան 90%-ին: [52, 53] «Կոշտ» ՄՄ-ն հանդես է գալիս ավարտված ֆիզիկական նախագծի տեսքով, որի դեպքում հայտնի են ինչպես էլեկտրական, այնպես էլ ֆիզիկական պարամետրերը: «Փափուկ» ՄՄ-ն հանդես է գալիս սարքավորումների նկարագրման Verilog կամ VHDL լեզվով գրված սինթեզվող մոդուլի տեսքով: Այն, «կոշտ» բլոկի համեմատ, ավելի հեշտ է ենթարկվում տեխնոլոգիական մասշտաբավորման, սակայն ավելի դժվար է իրագործել դրա բնութագրումը: Ներկայումս սկսել են ավելի հաճախ կիրառել բավականին մեծ «փափուկ» բլոկներ, այդ թվում՝ պրոցեսորներ, քանի որ դրանց նախագծումը տնտեսապես շատ շահավետ է, և փոխհատուցվում են դրանց հետագա ներդրման լրացուցիչ ծախսերը: «Փափուկ» ՄՄ-ների արդյունավետությունը առավել ընդգծվում է ինտերֆեյսային բլոկների պարագայում, քանի որ այս դեպքում հաճախ անհրաժեշտություն է առաջանում դրանց որոշակի ձևափոխության՝ կախված օգտագործվող կապուլդիների տեսակից և ճարտարապետությունից:

Այդ բլոկներն ունեն հետևյալ հատկությունները

«Փափուկ» մակրոսները՝

- ֆիզիկական կառուցվածքի տեսանկյունից ավելի ճկուն են, քան «կոշտ» մակրոսները,
- ավելի անկախ են կոնկրետ արտադրական գործընթացից,
- ունեն թերություն որոշ չափով անկանխատեսելի են՝ կատարողականության, ժամանակային, տարածքային կամ հզորության առումներով,
- ֆիզիկական նախագծման մեջ հնարավորություն է տրվում արտաքին էլուստների տեղաբաշխման մանիպուլացման:

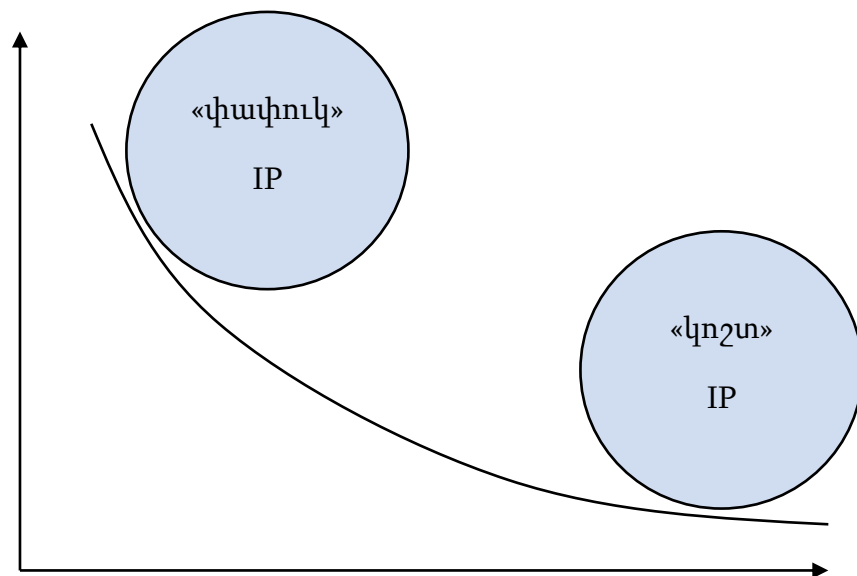
«Կոշտ» մակրոսները՝

- հիմնականում լինում են ապարատային ՄՄ-ների տեսքով,
- բլոկային մակարդակի նախագծեր են,
- նպատակաուղղված են հատուկ ինտեգրալ սխեմաների արտադրությանը,

- օպտիմալացված են կամ ըստ հզորության, կամ ըստ զբաղեցրած տարածքի, կամ ըստ ժամանակի,
- ֆիզիկական նախագծման մեջ հասանելիություն է տրվում «կոշտ» բլոկների արտաքին ելուստներին,
- օգտագործողը կարող է ազատ տեղաշարժել, պտտել, բայց չի կարող որևէ բան փոխել արդեն պատրաստի «կոշտ» մակրոսի մեջ,

«Կոշտ» և «փափուկ» ՄՍ բլոկների որակական համեմատականությունը ըստ տարբեր պարամետրերի բերված է նկ. 2.9-ում:

Վերաօգտագործում, շարժունակություն, ձկունություն



Կանխատեսելիություն, կատարողականություն, արժեք, վաճառողի կողմից ներդրված ջանք

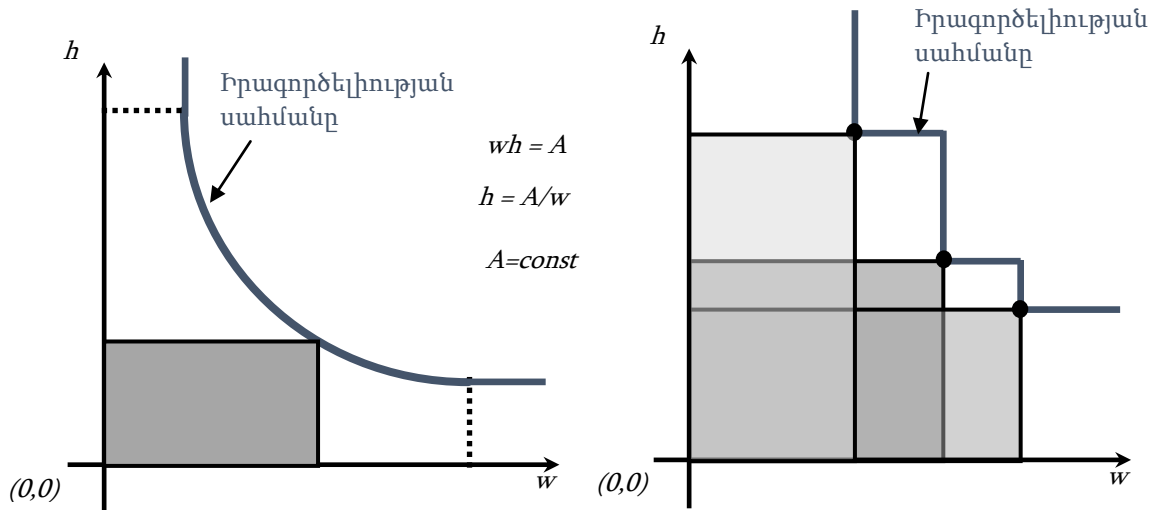
Նկ. 2. 9. «Փափուկ» և «կոշտ» IP-ների համեմատության գրաֆիկը

Գոյություն չունեն պարզ չափանիշներ՝ «կոշտ» մակրոսների նախնական տեղաբաշխումը գնահատելու համար: Այդ տեղաբաշխման արդյունքների որակի գնահատումը կարող է լինել շատ սուբյեկտիվ և հիմնականում կախված է նախագծման գործնական փորձերից: Այնուամենայնիվ, տեղաբաշխման որոշիչ չափանիշ է սխեմայի աշխատանքի արագագործությունը:

«Կոշտ» մակրոսների ավելի արդի օրինակ է հիշողությունը: Այն կարող է լինել ցանկացած տեսքի, որն ապահովում է հատուկ ֆունկցիոնալություն: LEF, GDS2 ֆայլերի ֆորմատները թույլ են տալիս մակրոսների հեշտ օգտագործումը տարբեր ծրագրերում:

«Կոշտ» բլոկը պարամետրացման ճանապարհով կարող է սինթեզվել «փափուկ» բլոկից՝ ավտոմատացված ձևով:

«Փափուկ» բլոկից «կոշտ»-ի անցումը հանգեցնում է բլոկի երկրաչափական ձևի ընտրության: Այս ընտրությունը կատարվում է մակերեսի հաստատունության պայմանով՝ ձևի կորի հիման վրա (նկ. 2.10):



Նկ. 2. 10. «Փափուկ» բլոկից «կոշտ» բլոկն անցման «ձևի» կորը (w) և կողերի տարրեր հարաբերակցություններով ներկայացվող «կոշտ» բլոկների տարրերակները (r)

«Փափուկ» բլոկից «կոշտին» անցման ժամանակ առաջ է գալիս յուրաքանչյուր մակարդակում գազաթների դասավորությունը որոշելու խնդիր [54]: Այդ դասավորությունը պետք է լինի այնպիսին, որ շղթաների հատումների քանակը ձգտի նվազագույնի, ինչն էլ, իր հերթին, նպաստում է ամբողջ սխեմայում միջմիացումների երկարությունների նվազմանը:

Նախ նշենք, որ շղթաների հատումների քանակը կախված է ոչ թե տարրերի վերջնական կոորդինատներից, այլ տվյալ ռանգում տարրերի հարաբերական դիրքերից: Այդ առումով այս քայլի խնդիրը ոչ թե երկրաչափական է, այլ կոմբինատորական: Սակայն այդ խնդիրը NP-բարդության է արդեն իսկ ընդամենը երկու ռանգ ունեցող գրաֆի դեպքում[55]:

Հիմնականում շղթաների հատումների քանակի նվազեցման խնդիրը ոչ թե լուծվում է գլոբալ՝ ամբողջ գրաֆի համար, այլ լոկալ՝ բոլոր հարևան տարրերի զույգերի համար: Փորձերը ցույց են տալիս [55], որ երկու ռանգերի համար շղթաների հատումների քանակի նվազեցման խնդիրը բավականին հեռու է օպտիմալ լինելուց [54, 56]: Ավելի ընդունելի, հատումների գլոբալ քանակը նվազեցնելու համար, դիտարկվում են բոլոր հարևան ռանգերը և հատումների նվազեցումը դրանց միջև: Այդ մոտեցումներն ունեն մի քանի մոդիֆիկացիաներ, որոնք տարբերվում են հետևյալ չափորոշիչներով.

- Ռանգերի ֆիքսման մեթոդի ընտրություն: Երկու հարևան ռանգեր դիտարկելիս մի ռանգի տարրերը կարելի է համարել ֆիքսված և ընտրել տարրերի օպտիմալ դասավորություն մյուս ռանգի համար: Կարելի է նաև R_i ռանգը դիտարկելիս ֆիքսված համարել R_{i+1} և R_{i-1} ռանգերը:
- Հերթականության ընտրություն, ըստ որի դիտարկվելու են տվյալ ռանգի զագաթները: Ռանգերը կարող են դիտարկվել ձախից աջ կամ հակառակը, հնարավոր է նաև դիտարկման ուղղությունների փոփոխություն:
- Դիտարկման ավարտի չափորոշիչի ընտրություն: Դա կարող է լինել դիտարկումների ֆիքսված քանակ, կամ դիտարկումները կարող են կատարվել, քանի դեռ դրանք փոքրացնում կամ զգալիորեն փոքրացնում են հատումների ընդհանուր քանակը:

Դիտարկենք հատումների թվի նվազեցումը երկու ռանգերի համար [54]: Ենթադրենք, մեզ տրված է $G = (R_1, R_2, W)$ գրաֆը, որտեղ R_1 -ը առաջին ռանգի տարրերի բազմությունն է, R_2 -ը՝ երկրորդ, իսկ E -ն՝ շղթաների բազմությունը: Առաջին ռանգի զագաթների դիրքերը համարելով ֆիքսված, կդասավորենք երկրորդ ռանգի տարրերը: Ինչպես արդեն նշվել է, այդ խնդիրն ունի NP-բարդություն, և դրա մոտավոր լուծումները գտնելու համար կոգտագործվեն մոտավոր լուծման մեթոդներ:

Վերոհիշյալ մեթոդներից մեկը հիմնված է տեսակավորման ալգորիթմների վրա [57, 58]: R_2 ռանգի տարրերը տեսակավորվում են այնպես, որ շղթաների հատումները ձգտեն նվազագույնի: Հեշտ է նկատել, որ ընտրված երկու տարրերի համար՝ $u \in R_2$ և $v \in R_2$, դրանց հարևան տարրերի հետ հատումների քանակը կախված է միայն այդ երկու տարրերի հարաբերական դիրքերից: Այսպիսով, կարող ենք դիտարկել հատումների

քանակը՝ փոխելով տրված ռանգում տարրերի հարաբերական դիրքերը: Հենց տարրերի հարաբերական դիրքերը փոխելու վրա էլ հիմնված են տեսակավորման ալգորիթմներն օգտագործող մեթոդները: Այդպիսի մեթոդները պետք է արագ հաշվեն հատումների պոտենցիալ քանակը: Դա կարող է կատարվել ուղակի $O(|W|^2)$ ժամանակում կամ լավարկել մինչև $O(\sum K_{uv})$, որտեղ K_{uv} -ն հատումների քանակն է, որն առաջանում է u և v տարրերի գույգի միջոցով: Հնարավոր հատումների հաշվարկը յուրաքանչյուր գույգի դեպքում պետք է կատարել միայն մեկ անգամ (դա էլ հենց տալիս է տարրերի բազմության դասավորությունը):

Տեսակավորման մեթոդների վրա հիմնված ալգորիթմներից մեկն իր էությանմբ նման է պղպջակների մեթոդին (bubble sort) [57, 59]: Հերթականությամբ դիտարկվում են հարևան գագաթների բոլոր գույգերը, և դրանք տեղերով փոխվում են, եթե այդ դեպքում հատումների թիվը կփոքրանա: Գործընթացն ավարտվում է, երբ հերթական լրիվ դիտարկումից հետո (բոլոր տարրերով անցնելուց հետո) հատումների քանակը չի նվազում (չկարգավորված գույգ չի գտնվում): Այս ալգորիթմի բարդությունը $O(|R_2|^2)$ է, քանի որ վերջնական կարգավորություն ստանալու համար կարող են անհրաժեշտ լինել մինչև անգամ $|R_2|$ դիտարկումներ: Այսպիսի ալգորիթմի ընտրությունը խիստ կախված է նախնական դասավորության ընտրությունից, քանի որ դիտարկվում և տեղափոխվում են միայն իրար հարևան տարրերը:

Հաջորդ մեթոդը հիմնված է արագ տեսակավորման (quick sort) ալգորիթմի գաղափարի վրա [57, 60, 61]: Սկզբում ընտրվում է «միջին» տարր, իսկ մնացած տարրերը բաժանվում են երկու բազմությունների՝ կախված նրանից, թե դրանք ինչ հարաբերությունների մեջ են գտնվում ընտրված տարրի հետ: Հետո նման պրոցեդուրա ռեկուրսիվ կիրառվում է ստացված բազմություններից յուրաքանչյուրի նկատմամբ: Վատագույն դեպքում տվյալ ալգորիթմի բարդությունը, ինչպես արագ տեսակավորման ալգորիթմի դեպքում, համարվում է քառակուսային, սակայն միջինում ալգորիթմը պահանջում է ընդամենը $O(|R_2| \log |R_2|)$ գործողություններ:

Հաջորդ մեթոդի համաձայն՝ ալգորիթմը ամեն քայլում վերցնում է տարրերից մեկը և փորձում այն դնել այնպիսի տեղում, որ ստացվի հատումների փոքրագույն թիվ: Դրա համար ալգորիթմը փորձում է տարրը տեղադրել յուրաքանչյուր հնարավոր տեղում և

հաշվում է ստացված հատումների քանակը: Քանի որ տարրը մի դիրքով տեղափոխելիս շղթաների հատումների քանակի փոփոխությունը կարելի է հաշվել հաստատուն ժամանակում (օգտագործելով լրացուցիչ հավաքված ինֆորմացիա), ապա ալգորիթմն ունի քառակուսային բարդություն և իր էությանմբ նման է ներդրումներով տեսակավորման մեթոդին (insertion sort)[57, 62, 63]:

Ինչպես արդեն նշվել է, ցանկացած դասավորության որոշման մեթոդ, որը հիմնված է տեսակավորման ալգորիթմների վրա և պետք է աշխատի տարրերի զույգերի հետ, չի կարող իրագործվել գծային ժամանակում: Հետևաբար, վերոհիշյալ մեթոդներից ոչ մեկը չի կարող ունենալ գծային բարդություն:

Գործնականում ավելի լայն կիրառություն ունեն բարիկենսորոնի և միջին արժեքի վրա հիմնված մեթոդները[64, 65]: Բարիկենսորոնի մեթոդի դեպքում $v \in R_2$ տարրի կոորդինատը որոշվում է որպես R_1 ռանգին կապած բոլոր տարրերի կոորդինատների միջին թվաբանական.

$$\text{avg}(u) = \frac{1}{\deg(u)} \sum_{v \in N(u)} x_1(v), \text{ որտեղ } N(u) := \{v \mid (u,v) \in W\}: \quad (2.8)$$

Եթե վերադասավորելուց հետո որոշ տարրերի կոորդինատներ համընկնում են, ապա դրանք տեղադրվում են նվազագույն հեռավորության վրա՝ կամայական հերթականությամբ: Այս մեթոդի ամենահայտնի տարբերակը միջին արժեքի մեթոդն է [66], երբ միջին թվաբանականի փոխարեն օգտագործվում է x_1 ռանգի միջին հարևանի կոորդինատը, այսինքն՝ եթե $u_1, u_2, \dots, u_m \in R_1$ v տարրին հարևան տարրերն են, ընդ որում, $x_1(u_1) < x_1(u_2) < \dots < x_1(u_m)$, ապա $x_2(v) = \text{med}(v)$ -ն որոշվում է որպես $x_1(u_{\text{avg}})$: Այն դեպքում, երբ $\text{med}(v) = \text{med}(u)$, և v -ն ունի կենտ աստիճան, իսկ u -ն՝ զույգ, ապա v -ն տեղադրվում է u -ից առաջ: Իսկ եթե աստիճանների զույգությունը համընկնում է, ապա հերթականությունը որոշվում է կամայականորեն: [67] աշխատանքում ցույց է տրված, որ միջինները կարող են գտնվել $O(|N(u)|)$ ժամանակում, ինչը այս մեթոդը դարձնում է նույնքան արագ, որքան բարիկենսորոնի մեթոդը:

Ենթադրենք՝ $\text{opt}(G, x_1)$ -ն տվյալ գրաֆի համար հնարավոր նվազագույն հատումների քանակն է, որտեղ x_1 -ը տարրերի դիրքերն է R_1 -ում, $\text{avg}(G, x_1)$ -ն՝ հատումների քանակը բարիկենսորոնի մեթոդի կիրառումից հետո, իսկ $\text{med}(G, x_1)$ -ը՝ հատումների քանակը

միջին արժեքի մեթոդից հետո: Բերենք մի քանի պնդումներ բարիկենսորոնի և միջին արժեքի մեթոդների վերաբերյալ [54]:

1. Եթե տրված G գրաֆի համար հնարավոր է դասավորություն՝ առանց շղթաների հատումների, ապա և՛ բարիկենսորոնի, և՛ միջին արժեքի մեթոդները կգտնեն այդ լուծումը, այսինքն, եթե x_1 դասավորությունը այնպիսին է, որ $\text{opt}(G, x_1) = 0$, ապա $\text{avg}(G, x_1) = \text{med}(G, x_1) = 0$:
2. Ցանկացած n -ի համար գոյություն ունի երկկողմանի գրաֆ՝ $G = (R_1, R_2, W)$, որտեղ $|R_1| = n$, $|R_2| = 2$, և R_1 -ի տարրերի այնպիսի x_1 դասավորություն, որ $\text{avg}(G, x_1) / \text{opt}(G, x_1)$ համեմատական է $n^{1/2}$:
3. Ցանկացած n -ի համար գոյություն ունի երկկողմանի գրաֆ՝ $G' = (R'_1, R'_2, W')$, որտեղ $|R'_1| = n$, $|R'_2| = 2$, և R'_1 -ի տարրերի այնպիսի x'_1 դասավորություն, որ $\text{med}(G', x'_1) / \text{opt}(G', x'_1) \geq 3 - O(1/n)$
4. $G = (R_1, R_2, W)$ երկկողմանի գրաֆի և R_1 -ի տարրերի ցանկացած x_1 դասավորության համար ճիշտ է $\text{med}(G', x'_1) \leq 2 * \text{opt}(G', x'_1)$ անհավասարությունը, եթե R_2 -ի գագաթները չունեն երեքից ավելի շղթաներ:

Այս մեթոդի ձևափոխված տարբերակը կիրառվել է այս աշխատանքում, որի ավելի մանրամասն նկարագրությունը ներկայացված է 3. 4 գլխում:

Այսպիսով, հետազոտվել է ինտեգրալ սխեմաների՝ «փափուկ» և «կոշտ» բլոկների՝ ֆիզիկական կառուցվածքների տեսքով ներկայացումը: Ներկայացվել են «փափուկ» և «կոշտ» բլոկների հատկությունները: Հետազոտվել են արդի ալգորիթմական լուծումներ՝ գրաֆում և ԻՄ-ում շղթաների հատումների քանակի նվազարկման և արագագործ ավտոմատացման միջոց մշակելու նպատակով:

2.5. Տրամաբանական տարրերի փոխանջատման ակտիվությունների միջոցով դինամիկ հզորությունների գնահատումը՝ որպես սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ ֆիզիկական կառուցվածքի ստացման միջոց

Եթե (1.6) արտահայտությունում տարրերի տեսակարար դինամիկ հզորություններն ու բջիջների թվային ստանդարտ գրադարանի աշխատանքային հաճախականությունը

հայտնի են դառնում գրադարանի ընտրության արդյունքում և կախված չեն սխեմայի տրամաբանական ակտիվությունից, ապա տարրերի փոխանջատման ակտիվությունը ուղղակիորեն կախված է սխեմայի տրամաբանությունից և տվյալ տարրի սխեմայում տեղաբաշխման վայրից: Ուստի ֆիզիկական նախագծման համար կարևոր է մշակել այնպիսի միջոցներ, որոնք թույլ կտան հաշվի առնել տրամաբանական սխեմայում տարրերի փոխանջատման ակտիվությունը՝ դրանց իրական հզորության գնահատման ժամանակ:

Փոխանջատման ակտիվության գնահատականը հիմնված է տրամաբանական տարրի ելքին ազդանշանային հավանականության բնութագրման վրա [68].

Ազդանշանային հավանականությունը տրամաբանական տարրի ելքին «1» ազդանշանի առաջացման հավանականությունն է, որը կարելի է հաշվարկել տվյալ տարրի իսկության աղյուսակով:

Հիմնական տրամաբանական տարրերի ելքին ազդանշանային հավանականությունները՝ փոխակերպիչի (ինվերտորի) տիպի, և(AND), կամ(OR), որոշվում են հետևյալ կերպ՝

$$P_{inv} = 1 - P_1; \quad P_{AND} = \prod_{i=1}^m P_i; \quad P_{OR} = 1 - \prod_{i=1}^m (1 - P_i), \quad (2.9)$$

որտեղ P_i –ն տրամաբանական տարրի i -րդ մուտքի ազդանշանային հավանականությունն է, m –ը՝ տրամաբանական մուտքերի քանակը:

Եթե հայտնի է տարրի ելքի ազդանշանային հավանականությունը (P), ապա փոխանջատումների ակտիվությունը որոշվում է հետևյալ բանաձևով [68].

$$K = 2P(1 - P): \quad (2.10)$$

Որոշակի տրամաբանական սխեմայի տարրերի փոխանջատման ակտիվության գնահատման ժամանակ անհրաժեշտ է ունենալ սխեմայի առաջնային մուտքերի ազդանշանային հավանականությունները: Սխեմայի առաջնային մուտքերում ազդանշանային հավանականությունը բնութագրելու համար առավել ընդհանրացված մոտեցում է համարվում հավանական մոտեցումը, երբ սխեմայի բոլոր առաջնային մուտքերի համար ազդանշանային հավանականության արժեքը հավասար է 0,5:

Այսպիսով, որոշակի ստանդարտ թվային բջիջների գրադարանի հիման վրա իրականացվող տրամաբանական սխեմայի համար, հիմնվելով (2. 9) և (2.10) –ով բոլոր տարրերի փոխանջատումների ակտիվության արժեքների հաշվարկի վրա, կարելի է (1.6)-ի միջոցով գնահատել տրամաբանական տարրերի աշխատանքային դինամիկ հզորությունները:

Այսպիսով, կարևորվել է ԻՄ-ի տրամաբանական նախագծումից ֆիզիկականին անցնելիս տրամաբանական տարրերի աշխատանքային հզորությունների արժեքների հաշվի առնելը ԻՄ-ի տակդիրի վրա դրանց տեղադրման ժամանակ:

Ֆիզիկական նախագծման ընթացքում տրամաբանական սխեմայի աշխատանքի առանձնահատկությունների տեսանկյունից հրատապ է դառնում, տրամաբանական տարրերի աշխատանքային հզորություններից ելնելով, դրանց տեղաբաշխման մեթոդների մշակումը: Եթե ընդունենք, որ հզորության ստատիկ բաղադրիչը գործնականում է կախված չէ սխեմայի աշխատանքի տրամաբանությունից, ապա տարրերի տեղաբաշխման փուլում հզորությունների գնահատման համար կարող է օգտագործվել միայն հզորությունների դինամիկ բաղադրիչը:

Եզրակացություններ

1. Կարևորվել է ԻՄ-երում տարրերի ժամանակային ուղղորդմամբ տեղաբաշխումը՝ որպես սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ ֆիզիկական կառուցվածքի ստացման միջոց: Ներկայումս գոյություն ունեցող ժամանակային ուղղորդմամբ տեղաբաշխման մեթոդներն ունեն որոշակի թերություններ, և կարիք կա դրանց հետագա զարգացման:
2. Առաջարկվում է նոր մոտեցում՝ ԻՄ-ի ֆիզիկական նախագծման ժամանակ սխեմայի ֆունկցիոնալ-տրամաբանական առանձնահատկությունների հաշվառման համար:
3. Մշակվել է ԱԳՀ-ի օգտագործման մեխանիզմը, և կարևորվել է սխեմայի Verilog նկարագրից ԱԳՀ-ի ստացման այնպիսի մոտեցման մշակումը, որով հնարավոր կլինի խուսափել անցանկալի մեծ հիշողության օգտագործումից և ստանալ ավտոմատացված միջոցի ցանկալի մեծ արագագործություն:

4. Նկարագրվել և կարևորվել է սխեմայի տարրերի ռանգավորումը՝ որպես սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ ֆիզիկական կառուցվածքի ստացման միջոց:
5. Հետազոտվել է ինտեգրալ սխեմաների՝ «փափուկ» և «կոշտ» բլոկների ֆիզիկական կառուցվածքների տեսքով ներկայացումը: Ուսումնասիրվել են «փափուկ» և «կոշտ» բլոկների հատկությունները:
6. Հետազոտվել են արդի ալգորիթմական լուծումներ՝ գրաֆում և ԻՄ-ում շղթաների հատումների քանակի նվազարկման և արագագործ ավտոմատացման միջոց մշակելու նպատակով:
7. Կարևորվել են ԻՄ-ի տրամաբանական նախագծումից ֆիզիկականին անցնելիս, տրամաբանական տարրերի տեղաբաշխման ժամանակ, դրանց աշխատանքային հզորությունների արժեքների գնահատումը և հաշվառումը:
8. Ֆիզիկական նախագծման ընթացքում տրամաբանական սխեմայի աշխատանքի առանձնահատկությունների տեսանկյունից՝ հրատապ է դառնում, տրամաբանական տարրերի աշխատանքային հզորություններից ելնելով, դրանց տեղաբաշխման մեթոդների մշակումը: Եթե ընդունենք, որ հզորության ստատիկ բաղադրիչը գործնականում կախված չէ սխեմայի աշխատանքի տրամաբանությունից, ապա տարրերի տեղակայման փուլում հզորությունների գնահատման համար կարող է օգտագործվել միայն հզորությունների դինամիկական բաղադրիչը:
9. Առաջարկվում է տրամաբանական տարրերի դինամիկ հզորությունների գնահատման մեթոդ՝ կախված տարրի տիպից, սխեմայի տրամաբանությունից և դրանում տարրի դիրքից:

**ԳԼՈՒԽ 3. ՄԽԵՄԱՏԵԽՆԻԿԱԿԱՆ ԱՌԱՆՁՆԱՀԱՏԱՏՎՈՒԹՅՈՒՆՆԵՐԻ
ՀԱՇՎԱՌՄԱՍԲ ԻՆՏԵԳՐԱԼ ՄԽԵՄԱՆԵՐԻ ՖԻԶԻԿԱԿԱՆ ՆԱԽԱԳԾՄԱՆ
ԱՎՏՈՄԱՏԱՑՄԱՆ ԵՆԹԱՀԱՄԱԿԱՐԳԻ ԱԼԳՈՐԻԹՄԱԿԱՆ ԵՎ
ԾՐԱԳՐԱՅԻՆ ԱՊԱՀՈՎՈՒՄԸ**

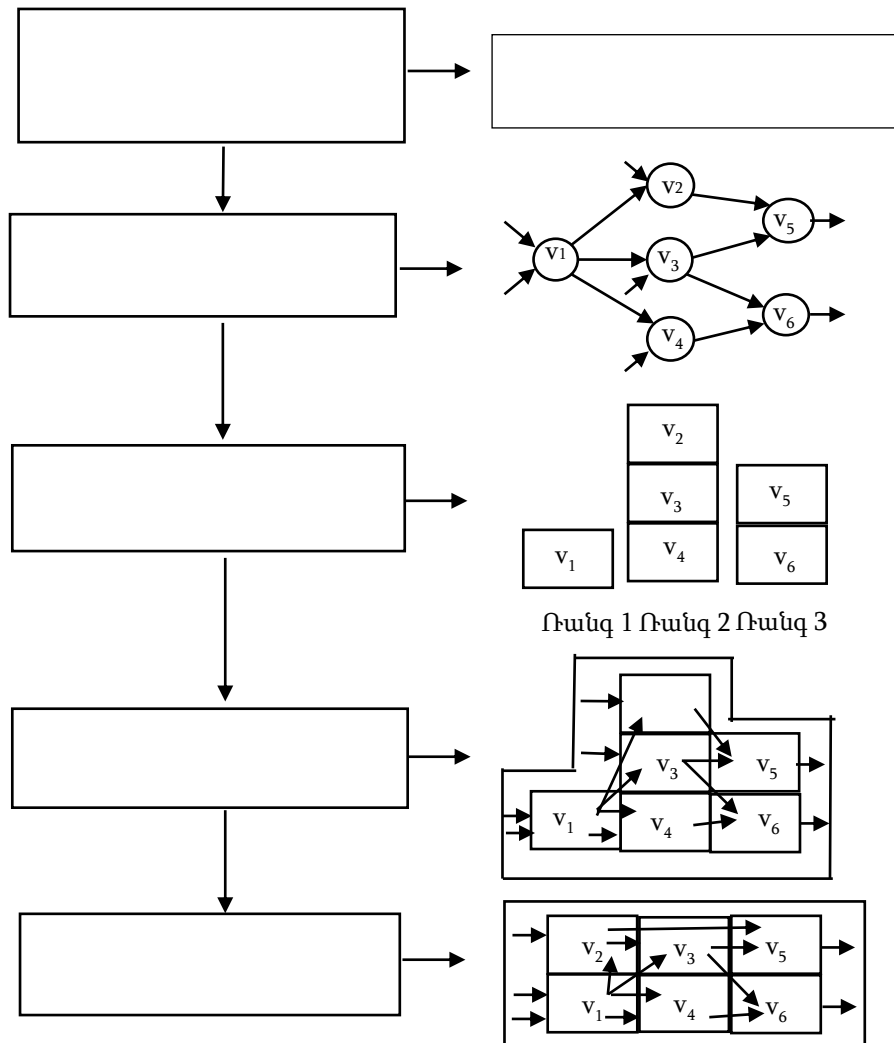
**3.1. Սխեմայի աշխատանքի տրամաբանության հաշվառմամբ ԻՄ-ի
ավտոմատացված ֆիզիկական նախագծման հիմնական փուլերը**

Սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ ԻՄ-ի ֆիզիկական նախագծման համար առաջարկվող և իրականացված մոտեցումը հիմնված է հետևյալ քայլերի հաջորդական իրագործման վրա (նկ. 3. 1).

- 1) սխեմայի Verilog նկարագրի փոխարկումը ազդանշանների հոսքի գրաֆի,
- 2) ազդանշանների հոսքի գրաֆով սխեմայի տարրերի՝ ըստ աշխատանքի հաջորդականության ռանգավորման իրականացում,
- 3) ռանգավորված սխեմայի ֆիզիկական «փափուկ» կառուցվածքի բլոկի ավտոմատացված սինթեզը,
- 4) «փափուկ» բլոկից ֆիզիկական «կոշտ» կառուցվածքի բլոկի ավտոմատացված սինթեզը՝ կառուցվածքային սահմանափակումների հաշվառմամբ:

Այսպիսի մոտեցումը պետք է նպաստի միջմիացումների երկարությունների և հատումների կրճատմանը: Այս մոտեցմամբ մշակված են որոշակի կանոններ, որոնք թույլ են տալիս տրամաբանական սխեմայի տարրերի ռանգավորումից «փափուկ» և «կոշտ» բլոկների սինթեզման ժամանակ, ըստ հնարավորության, ԻՄ-ի տակդիրի վրա տարրերը տեղաբաշխել այնպես, որ դրանց ֆիզիկական հաջորդականությունը համապատասխանի ազդանշանների հոսքի ուղղությանը: Բացի այդ, հատուկ ուշադրություն է դարձված ամենաերկար կապերի երկարությունների կրճատման անհրաժեշտությանը, որոնք որոշվում են որոշիչ ուղիների վրա ազդանշանների տարածման հապաղմամբ: Այս նպատակով ներմուծված է տարրերի փոխկապվածության կարևորության գործակից, որը հաշվի է առնում համապատասխան շղթայի կարևորությունը սխեմայի արագագործության տեսանկյունից:

Նկ. 3.1.-Ում դիտարկվել է նախորդ գլխում ներկայացված տրամաբանական սխեմայի Verilog նկարագրից մինչև «կոշտ» բլոկի ստացման գործընթացի ալգորիթմական ապահովումը:



Նկ. 3. 1. Սխեմատիկական առանձնահատկությունների հաշվառմամբ ԻՄ-ի ֆիզիկական նախագծման քայլերի հաջորդականությունը

Ներկայացված ալգորիթմական ապահովման համար որպես մուտքային տվյալներ են ծառայում 1.4 ենթագլխում ներկայացված խնդրի դրվածքին համապատասխանող մուտքային պարամետրերը:

3.2. Տրամաբանական սխեմայի verilog նկարագրի փոխարկումը ազդանշանների հոսքի գրաֆի

ԻՄ-երի ֆիզիկական նախագծման ժամանակ սխեմայի ֆունկցիոնալ-տրամաբանական առանձնահատկությունների հաշվառման առաջարկվող մոտեցումը ենթադրում է տվյալների մշակման սխեմայի փականային մակարդակի Verilog նկարագրության նախնական փոխակերպումը ազդանշանների հոսքի գրաֆի (ԱՀԳ)՝ $G(W,C)$ [69-72], որտեղ $C=\{c_1, c_2, \dots, c_m\}$ –ը տարրերի բազմությունն է, $W=\{w_1, w_2, \dots, w_n\}$ –ը՝ շղթաների բազմությունը: Հետագայում, ստացված ԱՀԳ-ի հիման վրա, կատարվում է սխեմայի տրամաբանական բջիջների դասակարգում, որի արդյունքներն օգտագործվում են դրանց տեղաբաշխման ժամանակ:

Այսպիսով, եթե տրված են ինտեգրալ սխեմայի փականային մակարդակի Verilog նկարագիրը, սահմանափակումները (չափերը՝ $\{a_i, h_i\}$, $i = \overline{1, m}$), որտեղ m -ը տարրերի քանակն է) պարունակող մուտքային զանգվածը, ապա ԱՀԳ-ի մոդելի ստեղծման հիմքում ընկած է հետևյալ հիմնական քայլերից բաղկացած ալգորիթմը.

- 1) կատարել ԻՄ-ի փականային մակարդակի verilog ֆայլի ընդհանուր վերլուծություն և ստուգել՝ արդյո՞ք այն վավեր է, թե ոչ: Կատարել համեմատություն նախնապես տրված վավեր ֆայլի հետ,
- 2) դուրս բերել մուտքային շղթաների բազմությունը, W_I , $I = \overline{1, k}$; k -ն մուտքային շղթաների քանակն է,
- 3) դուրս բերել ելքային շղթաների բազմությունը, W_O , $O = \overline{k+1, q}$; q -ն ելքային և մուտքային շղթաների գումարային քանակն է,
- 4) դուրս բերել միջանկյալ շղթաների բազմությունը, W_i , $m = \overline{q+1, n}$; n -ը բոլոր շղթաների քանակն է,
- 5) դուրս բերել տարրերի բազմությունը, C_i , $i = \overline{1, m}$; m -ը տարրերի քանակն է,
- 6) շղթաներն ու տարրերը դուրս բերելիս հաստատել բոլոր կապերը միմյանց հետ,
- 7) բոլոր տարրերում ներմուծել լայնություն և բարձրություն՝ ըստ տրված թվային ստանդարտ բջիջների գրադարանի՝ $\{a_i, h_i\}$, $i = \overline{1, m}$, որտեղ a_i և h_i համապատասխանաբար C_i տարրի երկարությունը և բարձրությունն են,

- 8) բոլոր շղթաներում ներմուծել ժամանակի պահուստները՝ $Res_i, i = \overline{1, n}$, որոնց արժեքները վերցվում են սխեմայի ստատիկ ժամանակային վերլուծության արդյունքից:

Այս աշխատանքում ներկայացված ԱՀԳ-ի ստացման մոտեցումը, որը ավելի մանրամասն նկարագրված է նկ. 3.2. -ում, նպատակաուղղված է՝ արագացնելու ԱՀԳ-ի վրա կիրառվող հետագա ռանգավորման և ֆիզիկական կառուցվածքի ստացման ալգորիթմների աշխատանքը: Ալգորիթմի արագագործությունը հիմնականում ապահովվում է շղթաների և տարրերի անունների փոխարեն՝ նրանց id (identity number) ինքնության համարների կիրառմամբ, որտեղ id-ն բնական թիվ է և ամեն տարրի կամ շղթայի համար եզակի է: Բացի id-երի ներմուծումից, արագագործության ապահովմանը ծառայում են նաև զանգվածների և բինար որոնման բալանսավորված ծառի տեսքով տվյալների (տարրեր, շղթաներ. . .) նկարագրությունները:

Դիտարկենք ալգորիթմի աշխատանքը Iscas85 գրադարանի [46] թեստային c17.v վերիլոգ ֆայլի օրինակով (նկ. 3.2):

```
1 // c17 Verilog file
2
3 module c17(G1,G16,G17,G2,G3,G4,G5);
4 input G1,G2,G3,G4,G5;
5 output G16,G17;
6
7 wire G8,G9,G12,G15;
8
9 nand NAND2_0(G8,G1,G3);
10 nand NAND2_1(G9,G3,G4);
11 nand NAND2_2(G12,G2,G9);
12 nand NAND2_3(G15,G9,G5);
13 nand NAND2_4(G16,G8,G12);
14 nand NAND2_5(G17,G12,G15);
15
16 endmodule
```

Նկ. 3. 2. Սխեմայի փականային մակարդակի Iscas85 գրադարանի թեստային c17.v Verilog նկարագիրը

Ալգորիթմի նկարագրության համար կօգտագործենք օբյեկտակողմնորոշված ծրագրավորման (ՕԿՕ) որոշակի տերմինաբանություն, որը կնպաստի նկարագրության հակիրճ և ձևայնացած ներկայացմանը: Ինչպես հայտնի է, ՕԿՕ –ն ծրագրավորման մոտեցում է, որի գաղափարական հիմք են հանդիսանում օբյեկտ և դաս հասկացությունները [73, 74]:

Այս ծրագրի աշխատանքը կատարվում է մի քանի հիմնական դասերի միջոցով՝ Graph (որը հենց մեր ԱՀԳ-ն է), Cell, Wire (5-րդ կետը): Ծրագիրը կարդում է փականային մակարդակի վերիլոգ ֆայլը և ստանում ԱՀԳ:

Ֆայլից տվյալների կարդալը կատարվում է հետևյալ կերպ՝

- 1) ֆայլից կարդալը կատարվում է տող առ տող,
- 2) եթե կան մեկնաբանություններ (որոնք սկսվում են հետևյալ `//` սիմվոլով) կամ դատարկ տողեր, ապա ծրագիրը դրանք անտեսում է, օրինակ՝ 1, 2, 6, 8 և 15-րդ տողերը,
- 3) Առաջին հերթին ծրագիրը կարդում է, թե ինչ մոդուլ պետք է գրանցի ԱՀԳ-ում (տող 3),
- 4) ամեն մոդուլ ունի իր՝ input, output և wire: Ծրագրը հերթով կարդում և գրանցում է ամեն տարրի ինֆորմացիան. դա կատարվում է մի քանի քայլերով (տող 4, 5, 7)՝
 - Հերթական էլեմենտը կարդալուց հետո, նրան տրվում է id: id-ն ներկայացնում է բնական թիվ, այսինքն՝ ամեն էլեմենտ կունենա իր եզակի թիվը: Օրինակ 4-րդ տողում 1-ին տարրը կլինի G1-ը, իսկ նրա id-ն կլինի 0, 2-րդը, կլինի G2-ը, իսկ նրա id-ն՝ 1 և այդպես շարունակ: Նույնը կատարվում է output և wire -ների (տող 5 և 6) դեպքում:
 - ԱՀԳ-ում ստեղծվում է map[75, 76](տարրի անունից դեպի իր id) կամ արդեն ստեղծված map-ի մեջ ներմուծվում է հերթական տարրը՝ իրեն տրված id-ով: map-ը բինար որոնման բալանսավորված ծառ է: Նրա ամեն տարրը ներկայացնում է {key, value} զույգ: Որոնումը և բալանսավորումը կատարվում է key-ի միջոցով, որոնք ծառի մեջ չեն կարող կրկնվել, իսկ value-ն ներկայացնում է ցանկացած տվյալ: Մեր օրինակում key-ը Id-ն է, իսկ value-ն՝ G1 կամ G2 ... (նկ. 3. 5)

- ԱՀԳ-ում առանձին կոնտեյներում (C++ լեզվի ստանդարտ գրադարանից են, ներկայացնում են՝ զանգված[77,78]) պահվում են input, output, wires՝ իրենց համապատասխան տարրերի id-երով: Այսպիսով՝ փականների անունների փոխարեն ստանում ենք id- եր՝ թվի տեսքով, ինչն էլ հետագայում կազդի ծրագրի արագագործության վրա:

5) ԱՀԳ-ի ճիշտ աշխատանքի համար հարկավոր են Cell և Wire դասերը[79]: Առաջինը պահում է տրամաբանական տարրի ինֆորմացիան: Երկրորդը՝ այդ տարրերը իրար կապող շղթաների ինֆորմացիան (աղ. 3. 1, աղ. 3. 2):

Օրինակ՝ նկ.2.4 -ում պատկերված գրաֆում NAND2_0, NAND2_1, NAND2_2, NAND2_3, NAND2_4, NAND2_5 անուններով տարրերը գրաֆի գագաթներն են, որոնք էլ այս ծրագրում հանդես են գալիս որպես Cell-օբյեկտներ, իսկ այդ գագաթները իրար կապող շղթաները՝ որպես Wire-ներ՝ G1, G2, G3, G4, G5, G8, G9, G12, G15, G16, G17 :

6) 4)-րդ կետը կատարելուց հետո ծրագիրը հասնում է տրամաբանական տարրերը կարդալուն(տող 9-14): Դիտարկենք ծրագրի աշխատանքը 9-րդ տողում: Այն ներկայացնում է՝

- ստանդարտ տարրի անուն՝ nand,
- տարրի անուն՝ NAND2_0,
- output փականի անուն՝ G8,
- input փականների անունները՝ G1,G3:

Արդեն ունենալով ամեն տարրի id-ին՝ ԱՀԳ-ում կարողանում ենք ստեղծել Cell-եր և Wire-ներ հերթական տողը կարդալիս, որոնք էլ ներմուծում ենք ԱՀԳ-ում պահվող g_cells և g_wires զանգվածներում: Ներմուծումն իրականացվում է այնպես, որ զանգվածում տարրի համարը համապատասխանի տարրի id-ին, ինչն էլ կարագացնի հետագա ալգորիթմների աշխատանքը (զանգվածում տրված համարով տարրին դիմելը հաստատուն ժամանակում է կատարվում՝ $O(1)$ է):

7) Հասնելով “endmodule”-ին, հերթական մոդուլի կարդալն ավարտվում է, և սկսվում է հաջորդ մոդուլի կարդալը, եթե այն առկա է:

Այսպիսով, ԱՀԳ-ում ստացվում է՝

- “vector<Wire> g_wires”՝ բոլոր շղթաների զանգվածը (աղ. 3. 1),
- “vector<Cell > g_cells”՝ բոլոր տարրերի զանգվածը (աղ. 3. 2),
- “set<int> g_input”՝ input փականների id-երի set (նկ. 3.3),
- “set<int> g_output”՝ output փականների id-երի set (նկ. 3.4),
- “map<string, int> g_gate_name_to_id”՝ map՝ տարրի անունից դեպի իր id (նկ. 3.5):

Թվարկված կոնտեյներները C++ ստանդարտ գրադարանից են[80]:

vector-ը զանգված է, set-ը և map-ը բինար որոնման բալանսավորված ծառեր են:

ԱՀԳ-ի տրամաբանական տարրը կապող շղթաների բազմությունը ներկայացնում է զանգված աղ. 3. 1-ը:

Աղյուսակ 3.1

ԱՀԳ-ի շղթաների «Wire» զանգվածի կառուցվածքը

Զանգվածի տարրի համարը	0	1	2	3	4	5	6	7	8	9	10
Շղթայի անունը	G1	G16	G17	G2	G3	G4	G5	G8	G9	G12	G15
Շղթայի id	0	1	2	3	4	5	6	7	8	9	10
Շղթայի սկիզբ	-	5	5	-	-	-	-	0	1	2	3
Շղթայի վերջ	0	-	-	2	3 2	1	3	4	3 2	5 4	5
Ժամանակի պահուստը	-	-	-	-	-	-	-	65	0	0	7

Զանգվածի ամեն տարրում պահվող տվյալները հետևյալն են՝

- 1.գրաֆի շղթայի (Wire-ի) անունը,
- 2.գրաֆի շղթայի id-ն, որը համընկնում է զանգվածում տարրի համարի հետ,

3.որտեղից է սկիզբ առնում տվյալ շղթան: Այն ներկայացնում է բինար որոնման բալանսավորված ծառ: Ծառի ամեն տարրը պահում է տրամաբանական տարրի (Cell-ի) id-ն,

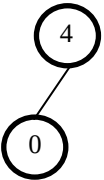
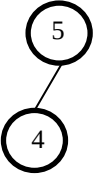
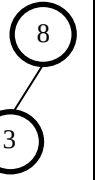
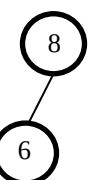
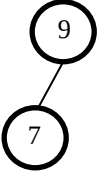
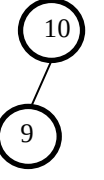
4.որտեղ է ավարտվում տվյալ շղթան: Այն ներկայացնում է բինար որոնման բալանսավորված ծառ: Ծառի ամեն տարրը պահում է տրամաբանական տարրի (Cell-ի) id-ն,

5.շղթայի ժամանակի պահուստը. արժեքները ալգորիթմի համար մուտքային տվյալներ են, վերցվում են սխեմայի ստատիկ ժամանակային վերլուծության արդյունքից:

Օրինակ՝ 8 id ունեցող շղթան(G9) սկսվում է 1 id ունեցող տրամաբանական տարրից, որը NAND2_1-ն է, և ավարտվում է 2 և 3 id ունեցող տրամաբանական տարրերին միանալով, որոնք էլ NAND2_2 և NAND2_3 տարրերն են: Ավելի մանրամասն տարրի id-ն ներկայացված է աղ. 3.2 տարրի զանգվածի նկարագրությունում:

Աղյուսակ 3.2

ԱՀԳ-ի տարրերի «Cell» զանգվածի կառուցվածքը

Զանգվածի տարրի համարը	0	1	2	3	4	5
Տարրի անունը	NAND2_0	NAND2_1	NAND2_2	NAND2_3	NAND2_4	NAND2_5
Տարրի id	0	1	2	3	4	5
Տարրի մուտքային շղթայի id						
Տարրի ելքային շղթայի id	7	8	9	10	4	5
Տարրի ռանգը	-	-	-	-	-	-

Դիտարկենք տարրերի զանգվածի 0-րդ տարրը՝

1. անուն՝ NAND2_0,

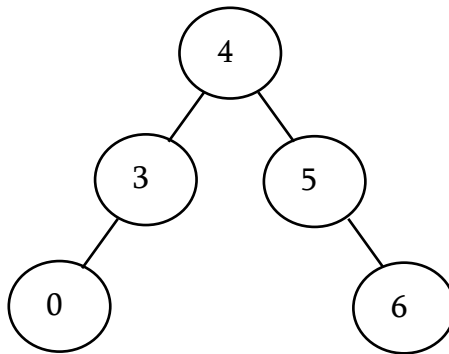
2. տարրի մուտքային կողերի id-երն են՝ 0 և 4: Նայելով շղթաների զանգվածի 0 և 4 տարրերի տվյալները, կտեսնենք, որ դրանք G1 և G3 շղթաներն են,
3. տարրի ելքային կողի id-ն է՝ 7: Այս դեպքում ևս նայելով կողերի զանգվածում, կտեսնենք, որ այն ներկայացնում է՝ G8,
4. տարրի ռանգը արժեքավորված չէ. այն կարժեքավորվի ռանգավորման ալգորիթի աշխատանքից հետո:

Այսպիսով, ստացանք NAND2_0(G8,G1,G3), ինչն էլ համապատասխանում է թեստային verilog ֆայլի 9-րդ տողին(նկ. 3.2):

Ալգորիթի 2-րդ և 3-րդ կետերի աշխատանքի արդյունքում՝ վերոնշյալ տեստային վերլուգ ֆայլի 2-րդ և 3-րդ տողերում գրված ինֆորմացիան պահվում է հետևյալ կերպ:

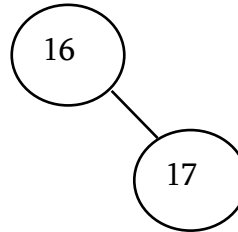
Մուտքային շղթաներ: Ալգորիթի աշխատանքի արդյունքում պահվում է բինար որոնման բալանսավորված ծառ [76, 80]: Ծառի ամեն տարր իր մեջ պահում է մուտքային շղթայի id: Verilog ֆայլի (նկ. 3. 2) 4-րդ տողում գրված է՝ G1,G2,G3,G4,G5; այսինքն՝ կպահվի նրանց id-երը, որոնք 0, 3, 4, 5, 6 -ն են (աղ.3. 1):

Այսպիսով, ստացվեց նկ.3.3-ում ներկայացված պատկերը:



Նկ. 3. 3. Մուտքային շղթաների id-երը բինար որոնման բալանսավորված ծառում

Ելքային շղթաներ : Աշխատում է նույն ձևով, ինչպես մուտքայինի դեպքում, սակայն այստեղ պահվում են 5-րդ տողում գրված շղթաների G16,G17-ի id-երը (նկ. 3. 4):

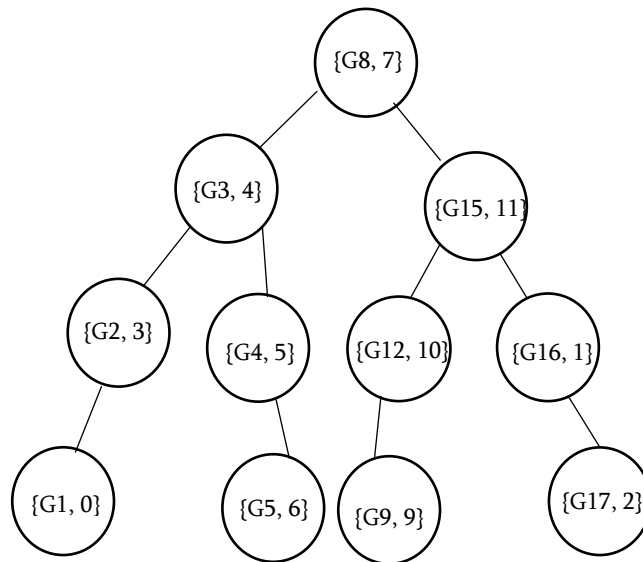


Նկ. 3. 4. Ելքային շղթաների id-երը բինար որոնման բալանսավորված ծառում

Մուտքային և ելքային շղթաների այսպիսի ծառերի[76] տեսքով պահվող ինֆորմացիան թույլ է տալիս հեշտությամբ $O(\log n)$ (լոգարիթմական) ժամանակում կատարել որոնում՝ մուտքային կամ ելքային շղթաների բազմությունում, ինչն էլ, իր հերթին, նպաստում է ԱՀԳ-ի վրա հետազայում կիրառելի ալգորիթմների արագագործությանը:

Նման գրաֆով աշխատանքի համար անհրաժեշտություն է առաջանում շղթայի անունով գտնել շղթայի id-ն: Բացի այն բանից, որ ունենալով ցանկացած շղթայի id-ն, կարելի է կողերի զանգվածում գտնել նրա անունը (աղ. 3. 1), նաև կարելի է անել հակառակը, ունենալով անունը՝ գտնել id-ն նկ. 3.4 -ի և աղ 2-ի միջոցով: Այս հնարավորության համար անհրաժեշտ է ունենալ map անունից դեպի id. map-ը բինար որոնման բալանսավորված ծառ է: Նրա ամեն տարրը ներկայացնում է {key, value} զույգ: Որոնումը և բալանսավորումը կատարվում է key-ի միջոցով, որոնք ծառի մեջ չեն կարող կրկնվել, իսկ value - ն ներկայացնում է ցանկացած տվյալ: Մեր օրինակու՝ key-ը (բանալի) շղթայի անունն է (օրինակ՝ G1 կամ G2 . . .), իսկ value-ն՝ համապատասխան կողի id-ն (աղ.3. 1):

Վերոնշյալ verilog ֆայլի օրինակում կունենանք հետևյալ map-ը (նկ. 3.5):



Նկ. 3.5. ԱՀԳ-ի շղթաների բազմության անունից դեպի իրենց id կառուցված map

Այսպիսով, սխեմայի ֆունկցիոնալ-տրամաբանական առանձնահատկությունների հաշվառման, տվյալների մշակման և ԻՍ ֆիզիկական նախագծման համար նախագծվել է տվյալների պահման այնպիսի մոտեցում, որը հիմնված է բինար որոնման բալանսավորված ծառի[76], տարրերի և շղթաների զանգվածների վրա[78, 79], ինչն էլ, իր հերթին, նշանակում է սխեմայի մուտքային փականային մակարդակի Verilog նկարագրության նախնական փոխակերպումը ազդանշանների հոսքի գրաֆի (ԱՀԳ-ի):

3.3. Տրամաբանական սխեմայի ռանգավորումը

Սխեմայի տրամաբանական տարրերի ռանգավորումը կատարվում է ԱՀԳ-ի՝ $G(W, C)$ բազմամակարդակ գրաֆի փոխակերպմամբ, որի համար հարկավոր է C բազմությունը բաժանել այնպիսի m մակարդակների (կամ ռանգերի)՝ $C_1, C_2, \dots, C_m$, որ ցանկացած c_i -ից դեպի c_j ուղղորդված $w(c_i, c_j)$ շղթայի համար, որտեղ $w \in W$, տեղի ունենա (3.1)՝

$$p < q, \text{ եթե } c_i \in C_p \text{ և } c_j \in C_q: \quad (3.1)$$

Սխեմայի տրամաբանական բջիջների ռանգավորումը կատարվում է այն ենթադրության հիման վրա, որ ցանկացած տարրի ռանգ մեկով ավելի է, քան այդ տարրի մուտքերին միացված այլ տարրերից առավելագույնի ռանգը: Որոշակի i տարրի R_i ռանգը որոշվում է հետևյալ կերպ՝

$$R_i = \begin{cases} 1, & \text{if } \forall I_i \in W_i, \\ \max\{R_k: k \in C_i\} + 1, & \text{if } \exists I_i \notin W_i, \text{ for } i = 1, 2, \dots, N, \end{cases} \quad (3.2)$$

որտեղ I_i - ն i -րդ տարրի մուտքային շղթաների բազմությունն է, W_i -ը՝ սխեմայի առաջնային մուտքերի բազմությունը, R_k -ն k -րդ տարրի ռանգը, C_i -ն՝ i -րդ տարրի մուտքերին միացված տարրերի բազմությունը, N -ը սխեմայի տարրերի քանակը:

Տրամաբանական սխեմայի Verilog նկարագրից ազդանշանների հոսքի գրաֆի ստացման և սխեմայի տարրերի ռանգավորման նպատակով օգտագործված է տվյալների ներկայացման և հետագա փնտրման բինար որոնման բալանսավորված ծառի սկզբունքը: Սկզբնական վիճակում բոլոր տարրերն էլ ռանգ չունեն, և դրա համար նրանց բոլորին տրվում է 0 ռանգ: Ռանգավորման ալգորիթը բաղկացած է 4 հիմնական քայլերից.

1. Բոլոր առաջնային մուտքային շղթաներին (Verilog ֆայլում input դաշտի դիմաց գրված շղթաները) վերագրել 1-ին ռանգ: Այն բոլոր տարրերին, որոնց բոլոր մուտքային շղթաները սխեմայի առաջնային մուտքային շղթաներ են, տրվում է 1-ին ռանգը:

2. Բոլոր այն տարրերը, որոնց բոլոր մուտքային շղթաներն ունեն 0-ից տարբեր ռանգ, ստանում են առավելագույն ռանգ ունեցող շղթայի ռանգը, իսկ նրանց ելքային շղթային տրվում է 1-ով ավել ռանգ:

Եթե c_i տարրի ռանգը նշանակենք $r(c_i)$, իսկ w_i շղթայի ռանգը՝ $r(w_i)$, ապա սկզբնական վիճակում կլինի՝

$$\begin{aligned} r(c_i) &= 0, \\ r(w_i) &= 0, \\ r(c_i) &= r(w_{imax}), \\ r(c_{out}) &= r(c_i) + 1, \end{aligned} \quad (3.3)$$

որտեղ w_{imax} -ը i -րդ տարրի մուտքային շղթաներից առավելագույն ռանգ ունեցողն է, իսկ c_{out} -ը i -րդ տարրի ելքային շղթայի ռանգն է:

3. Եթե 2-րդ քայլից հետո դեռ կան այնպիսի տարրեր, որոնք չունեն ռանգ (այսինքն՝ $r(c_i) = 0$), ապա դա նշանակում է, որ գրաֆում գոյություն ունի տարրերի այնպիսի հաջորդականություն, որը կազմում է փակ շրջան՝ ցիկլ: Անհրաժեշտ է գտնել չռանգավորված տարրերի այդ ցիկլը:

Այս քայլում պետք է սկսենք կամայական տարրից՝ c_i , որը չունի ռանգ՝ $r(c_i) = 0$: Եթե տարրը ռանգ չունի, դա նշանակում է, որ նրա գոնե մեկ մուտքային շղթա՝ $r(w_i) = 0$ ռանգ չունի, այլապես 2-րդ քայլում c_i , տարրը կռանգավորվեր: Գրանցում ենք c_i , -ն և շարունակում հետևյալ կերպ:

Հաշվի առնելով, որ ունենք ցանկացած շղթայի սկզբին և վերջին միացված բոլոր տարրերը, հետևաբար՝ կգտնենք w_i շղթայի սկզբին միացված տարրը՝ c_p -ն, որը չունի ռանգ, հակառակ դեպքում w_i շղթան կունենար ռանգ: Գրանցում ենք նաև c_p տարրը և կունենանք մեր որոնելի ցիկլի արդեն 2 տարրերի հաջորդականություն՝ c_i, c_p : Այսպես շարունակ շարժվում ենք այնքան ժամանակ, մինչև հասնենք արդեն իսկ մեզ մոտ գրանցված տարրը նորից գրանցելուն: Այստեղ արդեն գտանք մեր որոնելի ցիկլերից մեկը:

4. Նախորդ քայլում հայտնաբերված տարրերի ցիկլից՝ $c_i, c_p \dots$ հանվում է 1 շղթա, որը նշված տարրերի ցիկլի 2 տարր իրար էր կապում: Արդյունքում՝ ցիկլը փոխարկվում է ուղղակի տարրերի հաջորդականության:

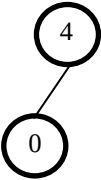
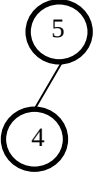
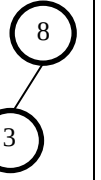
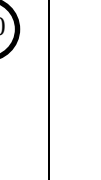
Այս քայլից հետո վերադառնում ենք 2-րդ քայլին: Այս ամենը կատարվում է այնքան ժամանակ, մինչև 2-րդ քայլից հետո բոլոր տարրերը ռանգավորվեն:

Գլուխ 3.2-ում ներկայացված օրինակի համար աղ. 3.2-ի «տարրի ռանգ» տողը ռանգավորումից հետո կունենա հետևյալ տեսքը (աղ. 3.3):

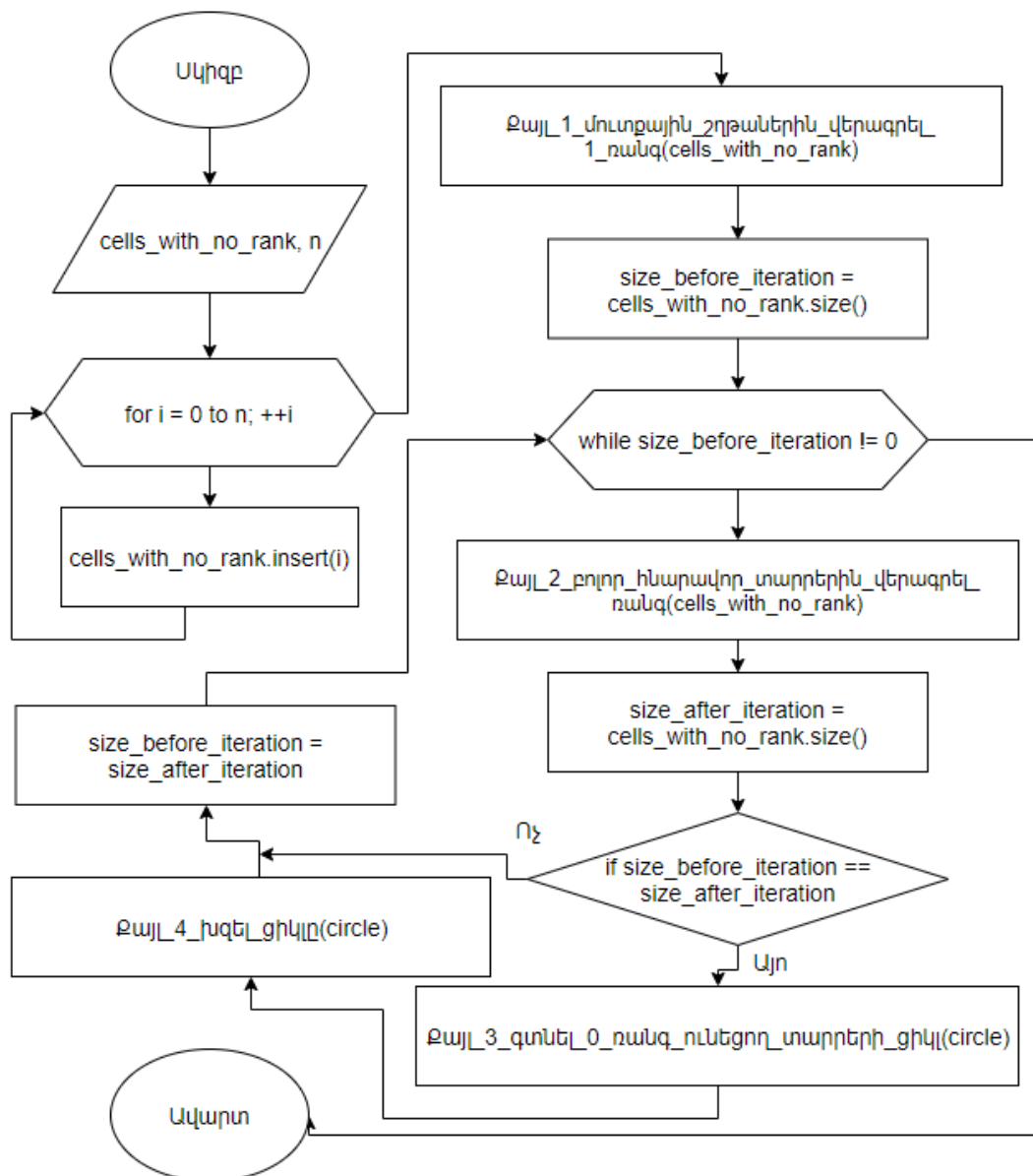
Աղյուսակ 3.3

ԱՀԳ-ի տարրերի զանգվածի կառուցվածքը ռանգավորումից հետո

Զանգվածի տարրի համարը	0	1	2	3	4	5
Տարրի անունը	NAND2_0	NAND2_1	NAND2_2	NAND2_3	NAND2_4	NAND2_5
Տարրի id	0	1	2	3	4	5

Տարրի մուտքային շղթայի id						
Տարրի ելքային շղթայի id	7	8	9	10	4	5
Տարրի ռանգը	1	1	2	2	3	3

Տարրերի ռանգավորման նպատակով մշակված ռանգավորման ալգորիթմի հիմնական քայլերը ներկայացված են նկ. 3.6-ում:



Նկ. 3. 6. Ռանգավորման ալգորիթմի հիմնական քայլերը

Այսպիսով, այս փուլը գազաթների բաժանումն է մակարդակների՝ համաձայն (3.2)-ի: Յուրաքանչյուր գազաթի վերագրվում է ռանգ, որը ցույց է տալիս տվյալ գազաթի մակարդակը այնպես, որ բոլոր կողերն ուղղված լինեն փոքր մակարդակից մեծը, ընդ որում, նույն մակարդակի գազաթները պետք է իրար հետ կապված չլինեն: Նույն մակարդակի գազաթները տեղադրվելու են նույն ուղղահայացի վրա:

3.4. Թվային ինտեգրալ սխեմաների տրամաբանական նկարագրությունից

«փափուկ» ֆիզիկական կառուցվածքի բլոկների սինթեզը

Մխեմայի համար ունենալով $G=(W,C)$ գրաֆը, կարող ենք տարրերի ռանգերին համապատասխանության մեջ դնել տեղաբաշխման X և Y կոորդինատները: Որևէ i -րդ տարրի X կոորդինատը կլինի՝

$$\forall c_i \in C, c_i(x) = \beta r(c_i), \quad (3.4)$$

որտեղ $r(c_i)$ -ն c_i տարրի ռանգն է, իսկ β -ն՝ ամենամեծ երկարությունն ունեցող տարրի երկարությունը:

Տարրերի Y կոորդինատների արժեքները որոշվում են ապագա միջմիացումների երկարությունների մինիմումի պայմանից: Եթե որպես ապագա միջմիացումների երկարությունների գնահատման մոդել վերցնենք լայնորեն օգտագործվող կիսապարագծային մոդելը [39, 7], ապա ակնհայտ է, որ նշված պայմանի ապահովման նպատակով անհրաժեշտ է, որ տարբեր ռանգերում գտնվող միևնույն շղթային պատկանող տարրերի Y կոորդինատների արժեքները լինեն հնարավորինս իրար մոտ՝

$$\forall w_k = \{c_i, c_j\}, |c_i(y) - c_j(y)| \rightarrow \min: \quad (3.5)$$

Այս խնդիրը լուծելու համար առաջարկվում է օգտագործել գրաֆային ալգորիթմներից հայտնի ըստ լայնության փնտրման ալգորիթմը [81]: Այն կիրառվում է ամենաբարձր ռանգ ունեցող տարրերից սկսած, դեպի ձախ, համաձայն (3.4)-ի: Արդյունքում, համաձայն (3.5)-ի, ամեն տարրի (բացառությամբ 1-ին ռանգի) ձախ կողմում կտեղադրվի նրա հետ կապված 1-ով փոքր ռանգ ունեցող տարր, ինչն էլ կնպաստի միջմիացումների երկարությունների կրճատմանը:

Տարրերի X և Y կոորդինատների նախնական արժեքները ստանալուց հետո կատարվում է տարրերի տեղաբաշխման լեգալացում, այսինքն՝ տարրերի տողերի և

այսինքն միջև եղած հեռավորությունների փոքրացում՝ մինչև նվազագույն թույլատրելի չափը, որի արդյունքում ստացվում է «փափուկ» բլոկի վերջնական տեսքը:

Ռանգավորման արդյունքներով «փափուկ» բլոկի ավտոմատացված սինթեզի ալգորիթմի կեղծ կոդը բերված է նկ. 3.7-ում:

For ամեն առավելագույն ռանգ ունեցող cell-ի համար

If cell -ը դեռ չի դիտարկվել

Կիրառել ըստ լայնության փնտրման ալգորիթմը cell-ի համար

Ամեն նոր դիտարկվող child_cell-ին վերագրել parent_cell.Y-ին մոտ Y

child_cell-ը նշել որպես դիտարկված

For ռանգ i to max_rank բոլոր cells-ի համար

Լեզալացնել տարրերը

If cell_a -ն հասվում է cell_b -ի հետ

Shift i-րդ ռանգի տարրերը cell_b-ի բարձրության չափով դեպի վար

Նկ. 3. 7. «Փափուկ» բլոկի ավտոմատացված սինթեզի ալգորիթմի կեղծ կոդը

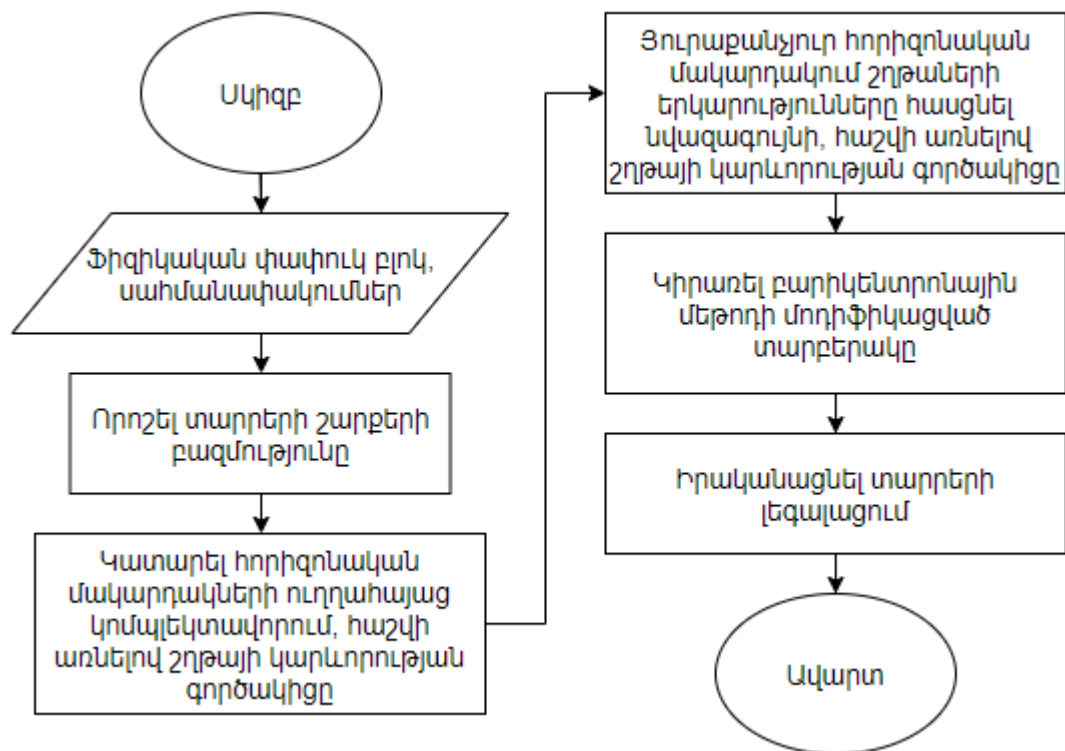
Այս փուլում տարբեր ռանգերի տարրերի քանակները հիմնականում տարբեր կլինեն: Հետևաբար, տարրերի նույն բարձրության և դրանց խիտ կոմպլեկտավորման պայմաններում տարբեր ռանգերի բարձրությունները կտարբերվեն: Նման տեղաբաշխման արդյունքում ձեռք բերված բլոկը կունենա ուղղանկյունից տարբերվող ձև և հաշվի չի առնի բլոկի ձևի վրա դրվող նախագծային սահմանափակումները:

Այսպիսով, մշակվել է «փափուկ» բլոկի ավտոմատացման միջոցի ստացման մեթոդ՝ տրված սխեմայի Verilog նկարագրությունից, և ալգորիթմական լուծման ծրագրային միջոց՝ հիմնված ԱՀԳ-ի ստացման, ռանգավորման և ըստ լայնության փնտրման ալգորիթմների վրա: Այս մեթոդը նպաստում է ինչպես ավտոմատացված միջոցի արագագործության մեծացմանը, այնպես էլ, միջմիացումների երկարությունների կրճատման հաշվին նախագծվող ԻՍ-ի արագագործության աճին:

3.5. «Փափուկ» ֆիզիկական կառուցվածքի բլոկների փոխակերպումը «կոշտ»

բլոկների

«Փափուկ» ֆիզիկական կառուցվածքի բլոկներից «կոշտ» բլոկների ավտոմատացված սինթեզի ալգորիթմի բլոկ-սխեման բերված է նկ. 3.8-ում:



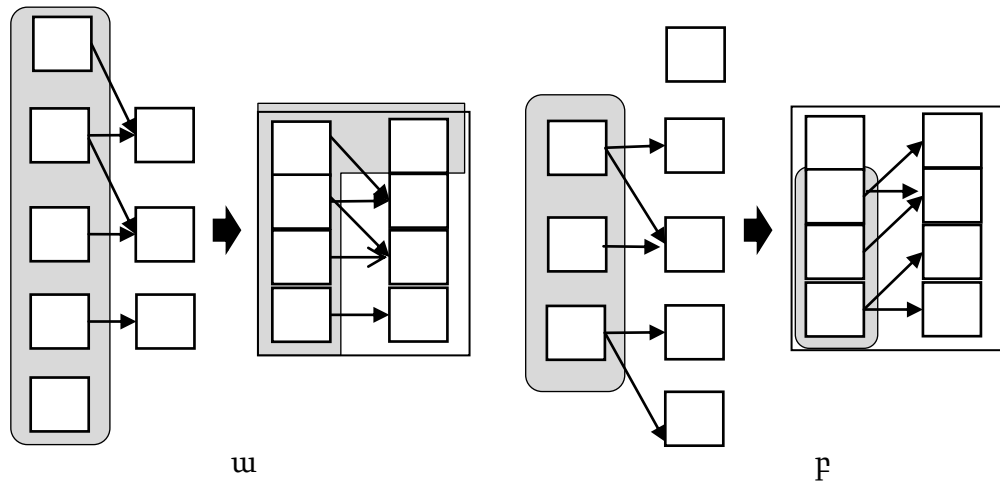
Նկ. 3. 8. Ֆիզիկական «փափուկ» բլոկից «կոշտ» բլոկի ստացման ալգորիթմի հիմնական քայլերը

«Կոշտ» բլոկների փոխարկման ժամանակ կարող են փոխվել ինչպես տարրերի հորիզոնական մակարդակների քանակը, այնպես էլ տարրերի յուրաքանչյուր ուղղահայաց մակարդակների կազմը: Ավելին, եթե բլոկի ձևի վրա դրված են կողմերի հետևյալ հարաբերակցությամբ սահմանափակումներ՝ $l_{min} \leq y/x \leq l_{max}$, ապա n տարրերի շարքերի բազմությունը կարող է ընտրվել՝ ելնելով հետևյալ պայմանից.

$$\left\lfloor \frac{\sqrt{Sl_{min}}}{h} \right\rfloor \leq n \leq \left\lfloor \frac{\sqrt{Sl_{max}}}{h} \right\rfloor, \quad (3.6)$$

որտեղ S -ը տեղաբաշխվող տրամաբանական տարրերի գումարային տոպոլոգիական մակերեսն է, h -ը՝ տրված գրադարանի տարրերի բարձրությունը:

Հորիզոնական մակարդակների ուղղահայաց կոմպլեկտավորումը կատարվում է՝ սկսած առաջին R_1 ռանգից մինչև վերջինը: Այս դեպքում հերթական i -րդ R_i ռանգի կոմպլեկտավորման ժամանակ կարող է լինել երկու դեպք. կամ R_i - ն ամբողջական չէ, և անհրաժեշտ է տարրերը ավելացնել R_j ռանգից. $j < i$, կամ հակառակը (նկ. 3.9):



Նկ. 3. 9. Հորիզոնական մակարդակների ուղղահայաց կոմպլեկտավորումը, որտեղ λ ռանգի տարրերը ա) գերազանցում են, բ) քիչ են աջ ռանգի տարրերից

Որևէ i -րդ տարրը մեկ մակարդակից մյուսը տեղափոխելու կարգը հիմնված է տեղափոխվող տարրի կոմպլեկտավորվող մակարդակի տարրերի ենթաբազմության հետ առավելագույն և դեռևս չկոմպլեկտավորված մակարդակի տարրերի ենթաբազմության հետ նվազագույն կապերի վրա, ինչը համապատասխանում է հետևյալ պայմանին.

$$\beta_i = \min_{i \in R_n} (\sum_{j \in R_n} \lambda_{ij} - \sum_{j \in R_k} \lambda_{ij}), \quad (3.7)$$

որտեղ R_k և R_n համապատասխանաբար կոմպլեկտավորված և չկոմպլեկտավորված տարրերի բազմություններն են, λ_{ij} - ն՝ i - րդ և j -րդ տարրերը կապող շղթայի կարևորության գործակիցը: Կարևորության գործակիցը հաշվի է առնում շղթայի ժամանակային պարամետրերը և որոշվում է՝ ելնելով իր՝ շղթայի ժամանակի պահուստի նկատմամբ հակառակ համեմատականության նկատառումներից [3,82]: Ժամանակային ահմանափականների ստուգման նպատակով օգտագործվում է շղթաների երկարությունը գնահատելու կիսապարագծային մեթոդը, որը որոշվում է

նվազագույն ուղղանկյան տրված շղթայի կոնտակտներն ընդգրկող նվազագույն մակերեսով ուղղանկյան կիսապարագծով [3]: Ռանգավորման հիման վրա տեղաբաշխման ժամանակ կիսապարագծային մեթոդը կասոցացվի տվյալ շղթայի հետ կապված տարրերի առավելագույն և նվազագույն ռանգերի և ուղղահայաց մակարդակների միջև եղած տարբերությունների գումարի հետ:

Ժամանակային սահմանափակումները հաշվի առնելու համար նպատակահարմար է վերանայել տարրերի գծային կարգը՝ յուրաքանչյուր հորիզոնական մակարդակում շղթաների երկարությունները նվազագույնի հասցնելու համար՝ հաշվի առնելով դրանց կարևորությունը: Ընդհանուր առմամբ, այս խնդիրը NP բարդության է, և դրա ճշգրիտ լուծումը ենթադրում է բոլոր տարբերակների ամբողջական հաշվարկ, ինչը անընդունելի է: Հայտնի են այս խնդրի արագ լուծման մի շարք մեթոդներ, որոնցից բարիկենտրոնային մեթոդն առանձնանում է իր արագությամբ և օգտագործման հարմարավետությամբ [83]: Այս աշխատանքում առաջարկվում է բարիկենտրոնային մեթոդի մոդիֆիկացված տարբերակ, որը հաշվի է առնում ոչ միայն հարևան մակարդակների տրամաբանական տարրերը կապող շղթաները, այլև շղթաների կարևորության գործակիցները:

Որևէ k -րդ հորիզոնական մակարդակում i -րդ տարրի՝ $c_{ki} \in C$ ուղղահայաց կոորդինատը որոշվում է հետևյալ կերպ՝

$$c_{ki}(y) = \frac{1}{card(\Omega_k)} \sum_{j \in \Omega_k} \lambda_{kij} c_j(y), \quad (3.8)$$

որտեղ Ω_k -ն արդեն փոխատեղված տարրերի բազմություն է, իսկ $card(\Omega_k)$ -ն՝ այդ բազմության հզորությունը:

Տարրերի ուղղահայաց տեղադրման հերթականությունը՝ V_k , որոշակի k -րդ հորիզոնական մակարդակում, որոշվում է այս տարրերի ուղղահայաց կոորդինատների աճման կարգով՝ $v_{ki}(y_1) \leq v_{ki}(y_2) \leq \dots \leq v_{ki}(y_k)$: Որոշ տարրերի ուղղահայաց կոորդինատների համընկնելու դեպքում ընտրվում է դրանց տեղադրման կամայական կարգը:

Հորիզոնական մակարդակներում տարրերի գծային կարգը վերակազմակերպելու առաջարկվող ալգորիթմի ռազմավարությունը հետևյալն է.

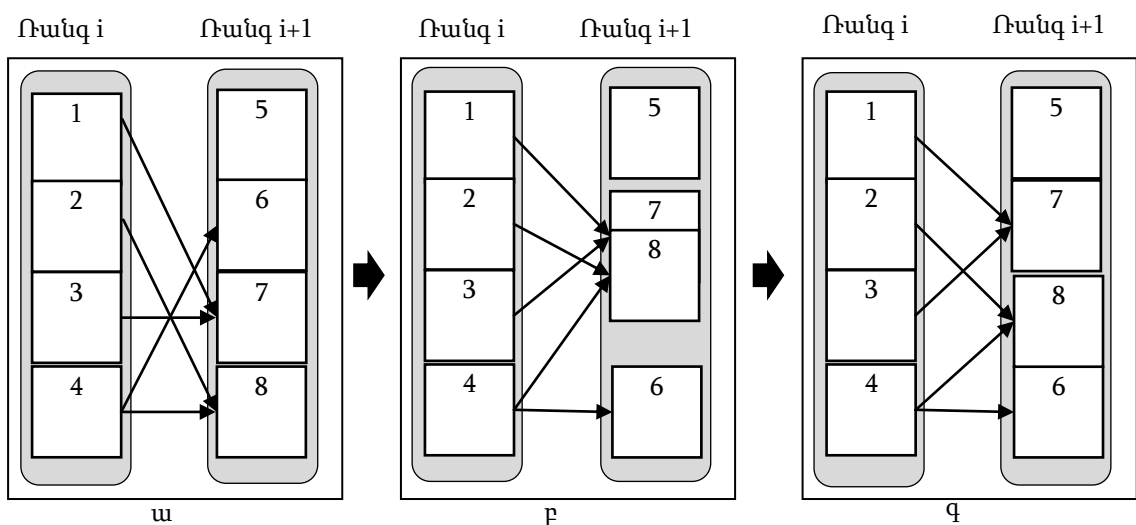
1. Առաջին մակարդակի տարրերի ըստ կապվածության տեղաբաշխումը՝ օգտագործելով հաջորդական տեղաբաշխման ամենապարզ ալգորիթմը [3]:

Միևնույն ժամանակ, հերթական ամենամոտ դիրքում տեղաբաշխվում է տարր, որն ունի նախադրյալի ֆունկցիայի առավելագույն արժեքը: Վերջինս սահմանվում է որպես տվյալ տարրի՝ արդեն տեղաբաշխված և դեռևս չտեղաբաշխված տարրերը միացնող կապերի կարևորության գումարային գործակիցների միջև տարբերություն:

2. Հաջորդ մակարդակների տարրերի հաջորդական տեղաբաշխում՝ ըստ դրանց աճման ուղղության: Այս դեպքում որոշ ռանգերում տեղաբաշխումը կատարվում է՝ հաշվի առնելով տարրերի կապը միայն նախորդ հիերարխիաների տարրերի հետ, այսինքն՝ արդեն տեղաբաշխված տարրերի հետ (նկ. 3.10 ա, բ):
3. Երկրորդ քայլի պարբերական կրկնությունը՝ հաջորդաբար ռանգերի կարգերի ավելացման և նվազեցման ուղղություններով, մինչև ժամանակի սահմանափակումների բավարարումը:

Յուրաքանչյուր մակարդակի ներսում տարրերի տեղաբաշխման համար, առաջին հերթին, սահմանվում են բոլոր տարրերի ուղղահայաց կոորդինատները, և հետագայում տարրերի տեղաբաշխումը կատարվում է ըստ կոորդինատների ավելացման: Հավասար կոորդինատների դեպքում տարրերը տեղաբաշխվում են կամայականորեն:

Վերջնական տեղադրումը ենթադրում է տեղաբաշխման արդյունքների լեզալացում՝ հաշվի առնելով նախագծի սահմանափակումները (նկ. 3.10 բ, գ):



Նկ. 3. 10. Հորիզոնական մակարդակներում ուղղահայաց ուղղությամբ տարրերի կոմպլեկտավորումը՝ ա) նախնական վիճակ, բ) մոդիֆիկացված բարիկենտրոնային ալգորիթմի աշխատանքից հետո, գ) արդյունքների լեզալացում

Եզրակացություն: Առաջարկվել է տրամաբանական բլոկների ֆիզիկական կառուցվածքի սինթեզի նոր մոտեցում, որը հիմնված է տրամաբանական սխեմայի ռանգավորման ճանապարհով ստացված ֆիզիկական «փափուկ» բլոկից «կոշտ» բլոկի ձևափոխման վրա, որն իրագործվել և փորձակվել է թեստային սխեմաների համար: Փորձարկումները ցույց են տվել մեթոդի բարձր արդյունավետությունը՝ արագագործության և միջմիացումների երկարությունների կրճատման և, հետևաբար, դրանցում հապաղումների նվազեցման տեսանկյունից:

Այսպիսով, մշակվել է ֆիզիկական «կոշտ» բլոկի ավտոմատացված սինթեզի նոր մոտեցում, որը հիմնված է տրամաբանական սխեմայի ռանգավորման ճանապարհով ստացված ֆիզիկական «փափուկ» բլոկի ձևափոխման և մոդիֆիկացված բարիկենտրոնային ալգորիթմի վրա, որը նպաստում է բարձր արդյունավետությանը՝ արագագործության և միջմիացումների երկարությունների կրճատման և, հետևաբար, դրանցում հապաղումների նվազեցման տեսանկյունից:

3. 6. Թվային ինտեգրալ սխեմաների տարրերի տեղաբաշխումը՝ դինամիկ հզորությունների հաշվառմամբ

Ինչպես արդեն նշվել է, ԻՄ-ի տակդիրի տեսակարար հզորության ավելացումը հանգեցնում է նրա հուսալիության զգալի նվազմանը: Ֆիզիկական նախագծման փուլում, որպես ջերմային հուսալիության բարելավման միջոց, կարող է հանդիսանալ տարրերի այնպիսի տեղաբաշխումը, որը կնպաստի ԻՄ-ի տակդիրի ջերմային դաշտի հարթմանը: Ընդ որում, որպես տեղաբաշխման մեթոդ կարող է ծառայել տարրերի հաջորդական տեղաբաշխման ալգորիթմը, հաշվի առնելով դրանց հզորությունները: Որպես ջերմային դաշտի տոպոլոգիական հարթմանը նպաստող տեղադրման չափանիշ կարող է ծառայել [3].

$$F = \underbrace{\min}_{\Pi} \left(\sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n |\Delta P_{ij}| d_{ij} \right), \quad (3. 9)$$

որտեղ ΔP_{ij} -ն i -րդ և j -րդ տարրերի հզորությունների տարբերության բացարձակ արժեքն է, d_{ij} -ն՝ i -րդ և j -րդ տարրերի դիրքերի միջև հեռավորությունը, Π -ն՝ տարրերի տեղաշարժերի բազմությունը, n -ն՝ տարրերի քանակը:

Ֆիզիկական նախագծման տեսանկյունից (3. 9) պայմանը նշանակում է, որ հզորությունների մեծ տարբերություն ունեցող տարրերը պետք է ավելի մոտ տեղաբաշխվեն իրար նկատմամբ, և հակառակը, ինչը կհագեցնի ԻՄ-ի տակդիրի ջերմային դաշտը հավասարակշռելուն, և կնվազեցնի առավել վտանգավոր հատվածների ջերմաստիճանը:

Հաջորդական տեղաբաշխման ավգորիթմի էությունը տարրերի հերթական ընտրությունն է՝ հիմնվելով β հաշվարկին, և այդ ֆունկցիայի նվազագույն արժեք ունեցող տարրի տեղաբաշխումը արդեն տեղաբաշխված տարրերի մոտակա ազատ դիրքում: Հաշվի առնելով (3. 9), որպես հավակնորդության ֆունկցիա կարող է ծառայել հերթական տեղաբաշխվող տարրի հզորությունների տարբերությունը արդեն տեղաբաշխվածների և դեռ չտեղաբաշխված տարրերի հետ, ինչը որոշ i -րդ տարրի համար կունենա հետևյալ տեսքը.

$$\beta_i = \min_{i \in E_{\text{in}}} \left(\sum_{j \in E_{\text{in}}} |\Delta P_{ij}| - \sum_{j \in E_{\text{z}}} |\Delta P_{ij}| \right), \quad (3. 10)$$

որտեղ E_{in} և E_{z} համապատասխանաբար տեղադրված և չտեղադրված տարրերի բազմություններն են:

Հզորության տեղաբաշխման հավասարաչափության և տեղային առավելագույն հզորության նվազեցման տեսանկյունից՝ տարրերի տեղաբաշխման արդյունավետության առումով, կարելի է օգտագործել կանխորոշված տարրերի քանակի տեղային տեղաբաշխման շրջանում գումարային առավելագույն միջին հզորության արտահայտությունը:

Եթե որպես տեղային գոտի դիտարկում ենք երկու հարևան տարրերի շրջանը, ապա այս արտահայտությունը կունենա հետևյալ տեսքը.

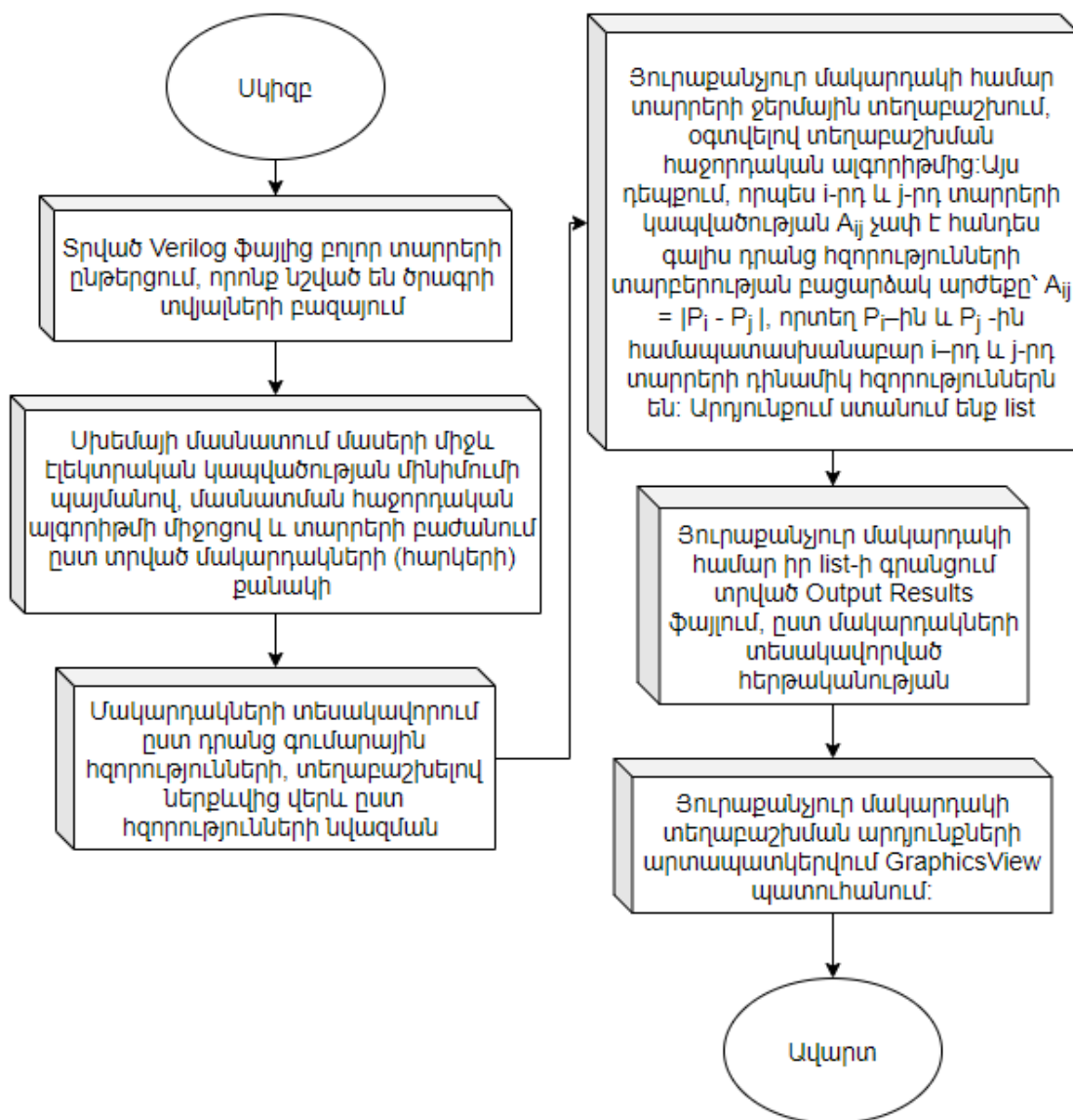
$$W = \max_{i=1,2,\dots,n-1} \frac{P_i + P_{i+1}}{2}; \quad (3. 11)$$

Մինևույն ժամանակ, համեմատելով տեղաբաշխման մի քանի տարբերակներ, առավելագույնը արդյունավետ կհամարվի ըստ (3. 11)-ի նվազագույն արժեքով տարբերակը:

Տեղաբաշխման ընդհանուր որակը հզորությունների համահարթեցման տեսանկյունից կարող է գնահատվել (3.11) արտահայտության հարաբերական արժեքով՝

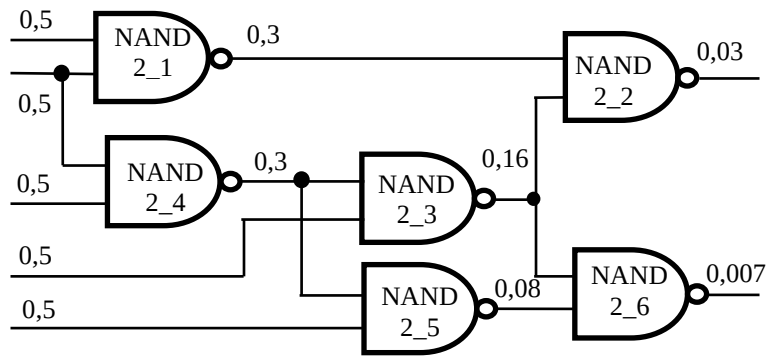
$$W = \left(\frac{\max W - \min W}{\max W} \right) 100\% . \quad (3. 12)$$

Տարրերի՝ ըստ դինամիկ հզորությունների տեղաբաշխման ալգորիթմի քայլերը ներկայացված են նկ. 3.11-ում:



Նկ. 3. 11. Ըստ հզորությունների ֆիզիկական տեղաբաշխման ալգորիթմի հիմնական քայլերը

Տրամաբանական տարրերի տեղաբաշխման ժամանակ դինամիկ հզորությունների հաշվառման վերոհիշյալ մոտեցման արդյունավետությունը ցույց է տրված ISCAS 85 շարքի C16 փորձնական սխեմայի տարրերի գծային տեղաբաշխման ամենատարածված օրինակով, որը բերված է նկ. 3.12-ում [46]:



Նկարում նաև ցուցադրված են տարրերի փոխանջատման ակտիվության արժեքները, որոնց հաշվարկը տրված է աղ. 3.4-ում:

Աղյուսակ 3.4

C16 փորձնական սխեմայի տարրերի պարամետրերի հաշվարկային արժեքները

Տրամաբանական տարրեր	NAND 2_1	NAND 2_2	NAND 2_3	NAND 2_4	NAND 2_5	NAND 2_6
Հատուկ դինամիկ հզորություն [նՎտ/ՄՀց]	0,013	0,013	0,016	0,016	0,013	0,013
Ազդանշանային հավանականություն	0,188	0,016	0,085	0,188	0,041	0,0035
Փոխանջատման ակտիվություն	0,3	0,03	0,16	0,3	0,08	0,007
Դինամիկ հզորություն[նՎտ]	1,95	0,19	1,28	2,4	0,52	0,05

Որպես տարրերի բազա օգտագործված է մշակված «Մինոփսիս-Արմենիա» ՓԲԸ-ի կրթական դեպարտամենտում մշակված[12] թվային ստանդարտ բջիջների գրադարանը՝ SAED_EDK 32/28, որի անվանական հաճախականությունը 500 ՄՀց է:

Տրամաբանական տարրերի նշված գրադարանից ընտրված գրադարանային տեսակարար դինամիկ հզորությունների ազդանշանային հավանականության և փոխանջատման ակտիվության արժեքները՝ հաշվարկված ըստ (2. 7) և (2. 8)-ի, և

աշխատանքային դինամիկ հզորությունների արժեքները՝ գնահատված ըստ (1. 6)-ի, ներկայացված են աղ. 3.4-ում: Զույգ տարրերի հզորությունների բացարձակ տարբերությունների արժեքների մատրիցը և բջիջների հաջորդական գծային տեղաբաշխման գործընթացը, որը հաշվարկված է համաձայն (3. 10)-ի, բերված է նկ.

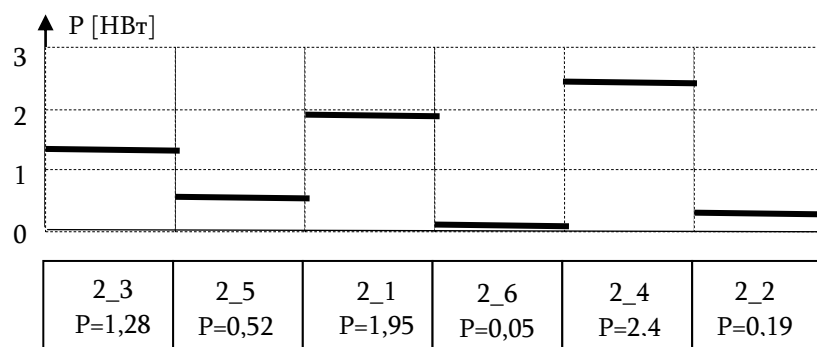
3. 13-ում:

	NAND						β_0	β_1	β_2	β_3	β_4	β_5
	2_1	2_2	2_3	2_4	2_5	2_6						
2_1	0	1,8	0,7	0,5	1,4	1,9	6,3	4,9	2,1	x	x	x
2_1	1,8	0	1,1	2,2	0,3	0,2	5,6	3,4	2,8	-0,8	-1,2	-5,6
2_1	0,7	1,1	0	0,1	0,8	1,2	3,9	x	x	x	x	x
2_1	0,5	2,2	0,1	0	1,9	2,4	7,1	6,9	3,1	2,1	-2,7	x
2_1	1,4	0,3	0,8	1,5	0	0,5	4,5	2,9	x	x	x	x
2_1	1,9	0,2	1,2	2,4	0,5	0	6,2	3,8	2,8	-1	x	x

Նկ. 3.13. ISCAS- 85 շարքի C16 փորձնական տրամաբանական սխեմայի գծային տեղաբաշխման հաշվարկի և հզորությունների տարբերության մատրից

Արդյունքում՝ բջիջները տեղաբաշխվում էին հետևյալ հաջորդականությամբ. NAND 2_3 → NAND 2_5 → NAND 2_1 → NAND 2_6 → NAND 2_4 → NAND 2_2: Նկ. 3.14-ը ցույց է տալիս բջիջների սխեմատիկ տեղաբաշխումը՝ նշելով դրանց դինամիկ հզորությունները և համապատասխան դիագրամը:

Հզորության տեղաբաշխման հավասարաչափությունը վերլուծելու համար նշված



Նկ. 3.14. Փորձարկվող սխեմաների տարրերի գծային տեղաբաշխումը և դրանց դինամիկ հզորությունները

տեղաբաշխման արդյունքը համեմատվում է նույն սխեմայի տարրերի տեղաբաշխման արդյունքի հետ, առանց հաշվի առնելու համար հզորությունը [4]: (3.11) Յուցանիշի համեմատությունը բարելավվել է տեղաբաշխման որակը 56% -ով:

Այսպիսով, ԻՄ-ի տրամաբանական նախագծումից ֆիզիկականին անցնելիս, կարևոր է հաշվի առնել տրամաբանական տարրերի աշխատանքային հզորությունների արժեքները ԻՄ-ի տակդիրի վրա դրանց տեղաբաշխման ժամանակ: Մշակվել է տրամաբանական տարրերի դինամիկ հզորությունների գնահատման համար ավտոմատացված միջոցի ստացման ալգորիթմական լուծում՝ կախված սխեմայի տրամաբանությունից և տարրերի տեղակայման ժամանակ ստացված արդյունքների կիրառումից:

Եզրակացություններ

1. ԻՄ-ի ֆունկցիոնալ-տրամաբանական առանձնահատկությունների հաշվառման, տվյալների մշակման և ԻՄ ֆիզիկական նախագծման համար մշակվել է տվյալների պահման այնպիսի մոտեցում, որը հիմնված է բինար որոնման բալանսավորված ծառի, տարրերի և շղթաների զանգվածների վրա, ինչն էլ, իր հերթին, թույլ է տվել ավտոմատացնել սխեմայի մուտքային փականային մակարդակի Verilog նկարագրության նախնական փոխակերպումը ԱՀԳ-ի:
2. Մշակվել է ԱՀԳ-ի գազաթների ռանգավորման ալգորիթմ, որը բինար բալանսավորված ծառի օգտագործման շնորհիվ՝ պարզ փնտրման համեմատ ապահովել է արագագործության մեծացում $O(n*m)$ -ից $O(n*\log m)$:
3. Մշակվել է «փափուկ» բլոկի ավտոմատացման միջոցի ավտոմատացված ստացման մեթոդ՝ տրված սխեմայի Verilog նկարագրությունից, որը հիմնված է ԱՀԳ-ի ստացման, ռանգավորման և ըստ լայնության փնտրման ալգորիթմների վրա: Մշակվել է մեթոդի ալգորիթմական և ծրագրային ապահովումը: Այս մեթոդը նպաստում է ինչպես ավտոմատացված միջոցի արագագործության մեծացմանը, այնպես էլ, միջմիացումների երկարությունների կրճատման հաշվին նախագծվող ԻՄ-ի արագագործության աճին:
4. Մշակվել է ֆիզիկական «կոշտ» բլոկի ավտոմատացված սինթեզի նոր մոտեցում, որը հիմնված է տրամաբանական սխեմայի ռանգավորման ճանապարհով

ստացված ֆիզիկական «փափուկ» բլոկի ձևափոխման և մոդիֆիկացված բարիկենտրոնային ալգորիթմի վրա, որը նպաստում է բարձր արդյունավետությանը՝ արագագործության և միջմիացումների երկարությունների կրճատման և, հետևաբար, դրանցում հապաղումների նվազեցման տեսանկյունից:

5. Մշակվել է ալգորիթմ՝ ԻՄ-ի տակդիրի վրա տեղաբաշխման դեպքում տրամաբանական տարրերի դինամիկ հզորությունների արժեքները գնահատելու համար:
6. Մշակվել է ալգորիթմ՝ ավտոմատացված միջոցի ստացման համար, կախված սխեմայի տրամաբանությունից և տարրերի տեղակայման ժամանակ 5-րդ կետում ստացված արդյունքների կիրառումից:

ԳԼՈՒԽ 4. ՄՇԱԿՎԱԾ ԱՎՏՈՄԱՏԱՑՎԱԾ ԵՆԹԱՀԱՄԱԿԱՐԳԻ ԾՐԱԳՐԱՅԻՆ ԻՐԱԳՈՐԾՈՒՄԸ ԵՎ ՀԵՏԱԶՈՏՈՒԹՅԱՆ ԱՐԴՅՈՒՆՔՆԵՐԸ

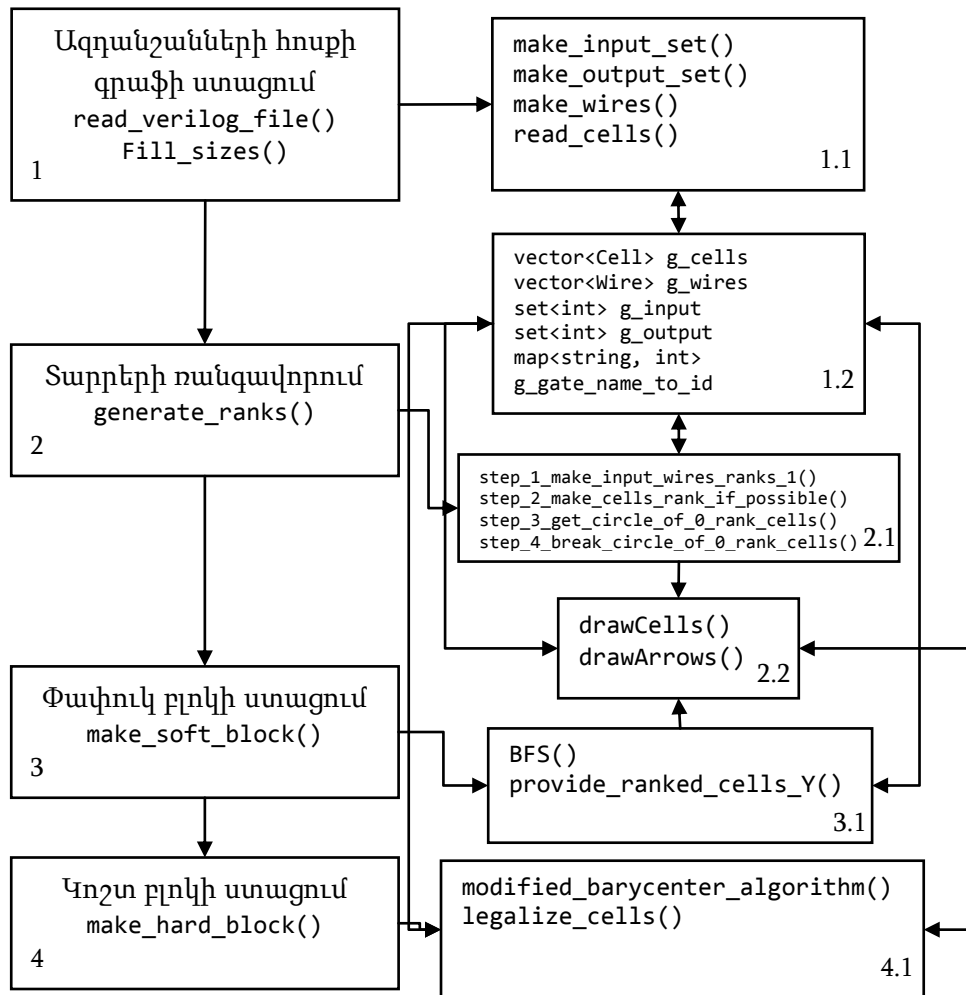
4.1. Մխեմատելիսնիկական առանձնահատկությունների հաշվառմամբ ինտեգրալ սխեմաների ֆիզիկական կառուցվածքի ավտոմատացված նախագծման ծրագրային միջոցի նկարագրությունը և փորձնական հետազոտության արդյունքները

Ինչպես նեկայացվեց նախորդ բաժիններում, ԻՍ-երի սխեմատելիսնիկական առանձնահատկությունների հաշվառմամբ ֆիզիկական նախագծման համար ստեղծվել է ԻՍ-երի ավտոմատացված նախագծման ենթահամակարգի ծրագրային ապահովում:

Տրամաբանական սխեմայի ֆիզիկական նախագծման և «կոշտ» բլոկի կառուցման ծրագրային միջոցի հիմնական ֆունկցիաները ներկայացված են նկ. 4.1-ում[84]:

Ստորև ներկայացված են ֆունկցիաների նշանակությունները.

1. ազդանշանների հոսքի գրաֆի ստացում (1 բլոկ)[85]`
 - `read_verilog_file()` – կարդում է տրված Verilog ֆայլը և կառուցում ԱՀԳ` (1.2 բլոկ)
 - `make_input_set()` – մուտքային շղթաների ծառի կառուցում` `g_input` -ի մեջ (1.3 բլոկ),
 - `make_output_set()` – ելքային շղթաների ծառի կառուցում` `g_output` -ի մեջ (1.3 բլոկ),
 - `make_wires()` – բոլոր շղթաների տվյալների կառուցում` `g_wires` զանգվածում (1.3 բլոկ),
 - `read_cells()` – տարրերի տվյալների կառուցում` `g_cells` զանգվածում (1.3 բլոկ),
`g_gate_name_to_id` բինար որոնման բալանսավորված ծառում գրվում են բոլոր շղթաները և դրանց `id` -երը (նույնականության համարը), որպեսզի ծրագիրը հետագայում հեշտությամբ աշխատի միայն `id`-ով:
 - `fill_sizes()` – բոլոր տարրերին վերագրվում են բարձրություն և երկարություն. `g_cells` զանգվածում կատարվում է տվյալների թարմացում:



Նկ. 4.1. Մշակված ծրագրային միջոցի աշխատանքը բնութագրող քայլերի հաջորդականությունը և կիրառված

2. Տարրերի ռանգավորում (2 բլոկ)[85]: Իրականացնում է տարրերի ռանգավորում որոշակի ալգորիթմով, որն էլ ներկայացվում է 3 ֆունկցիաների աշխատանքով (2.1 բլոկ)՝

- step_1_make_input_wires_ranks_1() - բոլոր մուտքային շղթաներին տրվում է 1 ռանգ,
- step_2_make_cells_rank_if_possible() - տրված որոշակի պայմանը բավարարելու դեպքում համապատասխան տարրերին վերագրել համապատասխան ռանգ,
- step_3_break_circle_of_0_rank_cells() – եթե բոլոր տարրերը դեռ ռանգավորված չեն, հետևաբար՝ գոյություն ունի տարրերի աշխատանքի ցիկլ, կտրել հետադարձ կապը:

3. Փափուկ բլոկի ստացում (3 բլոկ)[86, 87]՝

- BFS() – իրականացնում է հայտնի փնտրում՝ ըստ խորության[57] ալգորիթմի մոդիֆիկացված տարբերակը, միջմիացումների հատումների քանակի և

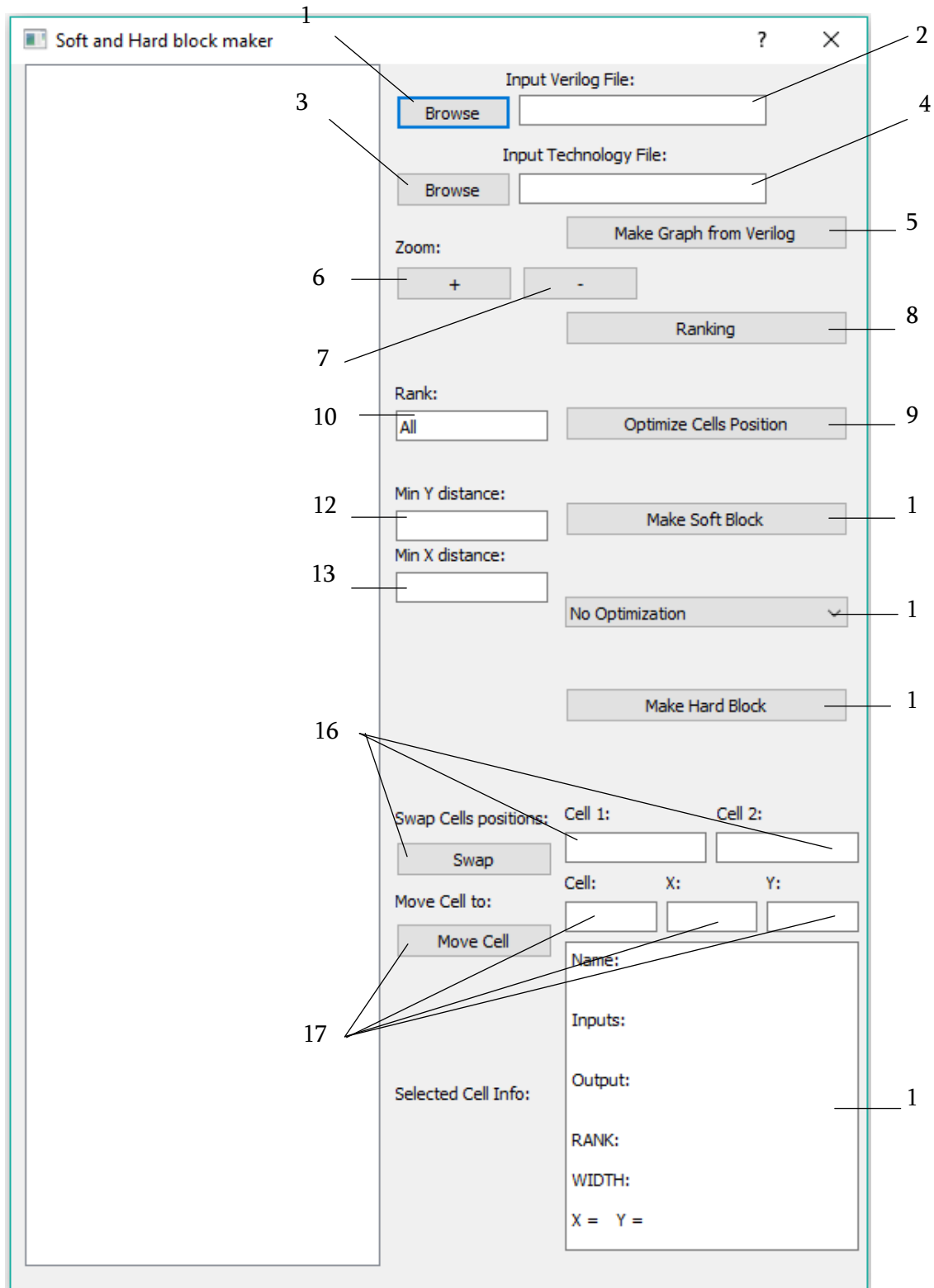
երկարությունների կրճատման նպատակով, որից հետո `provide_ranked_cells_Y()` ֆունկցիայով տարրերին տրվում է որոշակի Y կոորդինատ (3.1 բլոկ):

4. Կոշտ բլոկի կառուցում (4 բլոկ)[84, 87]՝

- `modified_barycenter_algorithm()` – իրականացնում է այս աշխատանքում ներկայացված բարիկենտրոնային մոդիֆիկացված ալգորիթմը, որի արդյունքում տարրերը ստանում են որոշակի X և Y կոորդինատներ(4.1 բլոկ),
- `legalize_cells()` – այս ֆունկցիան բացառում է տարրերի հատումների քանակը և ստանում է վերջնական տեսքի կոշտ բլոկ (4.1 բլոկ):

Նշված բոլոր քայլերում ծրագիրը ինտենսիվ հարցումներ է անում 1. 2 բլոկին և ստանում տվյալների վերաբերյալ ցանկացած ինֆորմացիա: Բոլոր բլոկներում ներկայացված ֆունկցիաները հնարավորություն ունեն ցանկացած պահի գրաֆիկական ինտերֆեյսում արտապատկերելու տվյալ պահի տարրերի և տարրերը իրար կապող շղթաների ընդհանուր գրաֆային տեսքը(2.2 բլոկ):

Ծրագրային ապահովումն իրագործված է «Soft and Hard block maker» (SHBM) անունով ծրագրային գործիքի տեսքով, որի մուտքային աշխատանքային պատուհանը բերված է նկ. 4.2-ում[87]: Ծրագրային միջոցի աջ մասում տեղադրված են նախագծողի համար նախատեսված ծրագրի աշխատանքի բոլոր գործիքները:



Նկ. 4.2. SHBM ծրագրային գործիքի սկզբնական պատուհանի տեսքը
SHBM-ի գործիքներն են՝

1. -ը «Browse» կոճակ է, որի միջոցով ներմուծվում է մուտքային՝ փականային մակարդակի Verilog ֆայլը (նկ. 4.3),

```

module s28(G1,G2,G3,G5,G6,G7,G18);
input G1,G2,G3,G5,G6,G7;
output G18;
wire G4,G8,G9,G10,G11,G12,G14,G15,G16,G17;

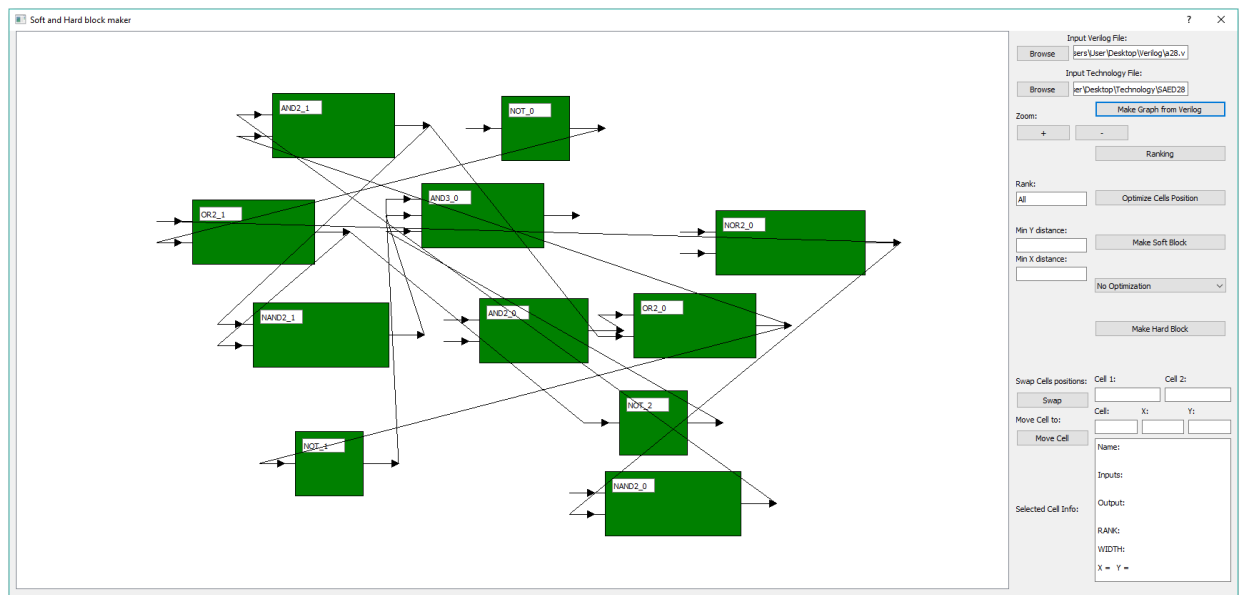
and AND2_0(G8,G1,G2);
nand NAND2_0(G9,G3,G4);
nor NOR2_0(G4,G5,G6);
not NOT_0(G11,G7);
or OR2_0(G12,G8,G10);
and AND2_1(G10,G12,G9);
or OR2_1(G14,G4,G11);
not NOT_1(G15,G12);
nand NAND2_1(G16,G10,G14);
not NOT_2(G17,G14);
and AND3_0(G18,G15,G16,G17);

endmodule

```

Նկ. 4.3. a28 փականային մակարդակի Verilog նկարագիր

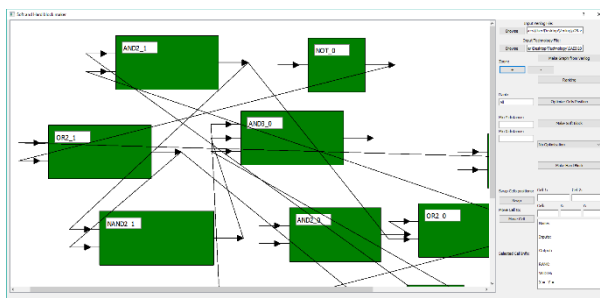
2. -ը 1-ինի միջոցով ներմուծված ֆայլի հասցեն է ցույց տալիս,
3. -ը «Browse» կոճակ է, որի միջոցով ներմուծվում է մուտքային՝ տեխնոլոգիայի ֆայլը,
4. -ը 3-րդի միջոցով ներմուծված ֆայլի հասցեն է ցույց տալիս,
5. -ը «Make Graph from Verilog» կոճակ է, որը ստեղծում է ազդանշանների հոսքի գրաֆը՝ ըստ 3.2 գլխում նշված ալգորիթմի, և արտապատկերում է գրաֆիկական



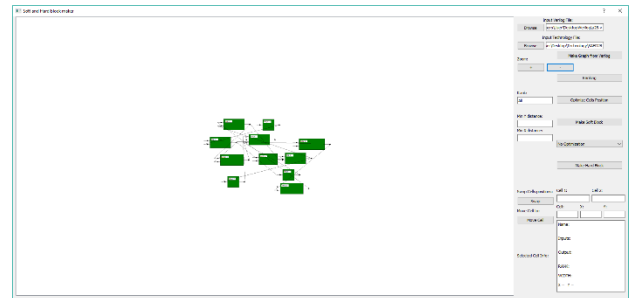
Նկ. 4.4. a28 թեստային Verilog ֆայլի՝ ազդանշանների հոսքի գրաֆի գրաֆիկական տեսքը

պատուհանում(նկ. 4.4), տարրերին վերագրելով կամայական X և Y կոորդինատներ տրված միջակայքում: ԱՀԳ-ն պատկերվում է տարրերի և այդ տարրերը իրար միացնող շղթաների տեսքով: Ամեն տարր պատկերվում է ուղղանկյան տեսքով, ունի գույն և վրան գրված է իր անունը: ԱՀԳ-ի ամեն տարր հնարավորություն ունի տեղաշարժվելու, եթե օգտատերը այն մկնիկով բռնի և տեղաշարժի: Տարրերի հետ միասին կտեղաշարժվեն նրանց շղթաները:

6. -ը « + » կոճակն է, որի միջոցով կատարվում է սխեմայի խոշորացում(Zoom in)(նկ. 4.5 ա),
7. -ը « - » կոճակն է, որի միջոցով կատարվում է սխեմայի մանրացում(Zoom out)(նկ. 4.5 բ),



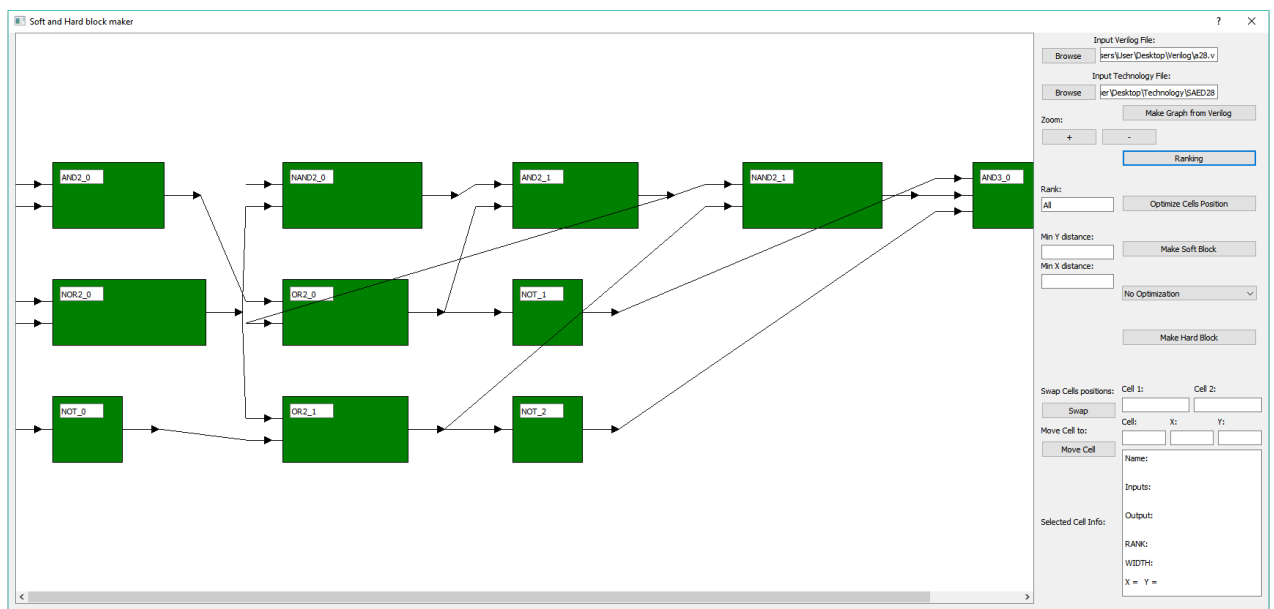
ա



բ

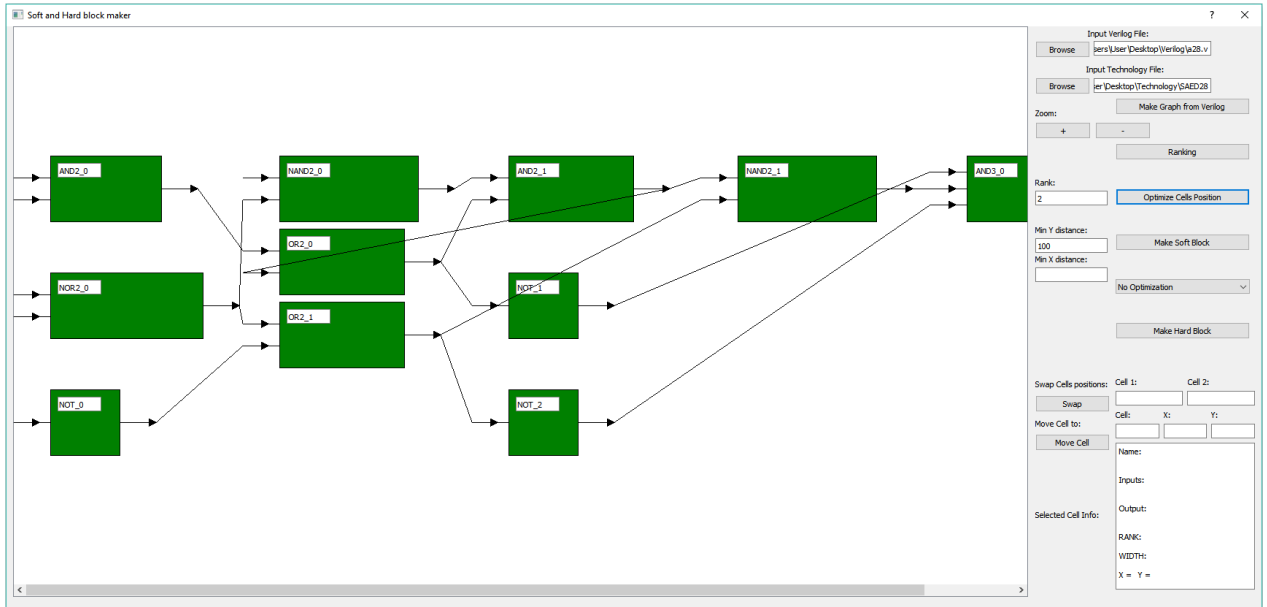
Նկ. 4.5. Սխեմայի 'ա) խոշորացումը բ) մանրացումը

8. -ը «Ranking» կոճակն է, որի միջոցով կատարվում է տարրերի ռանգավորում՝ ըստ 3.3 գլխում նշված ալգորիթմի (նկ. 4.6)[85],



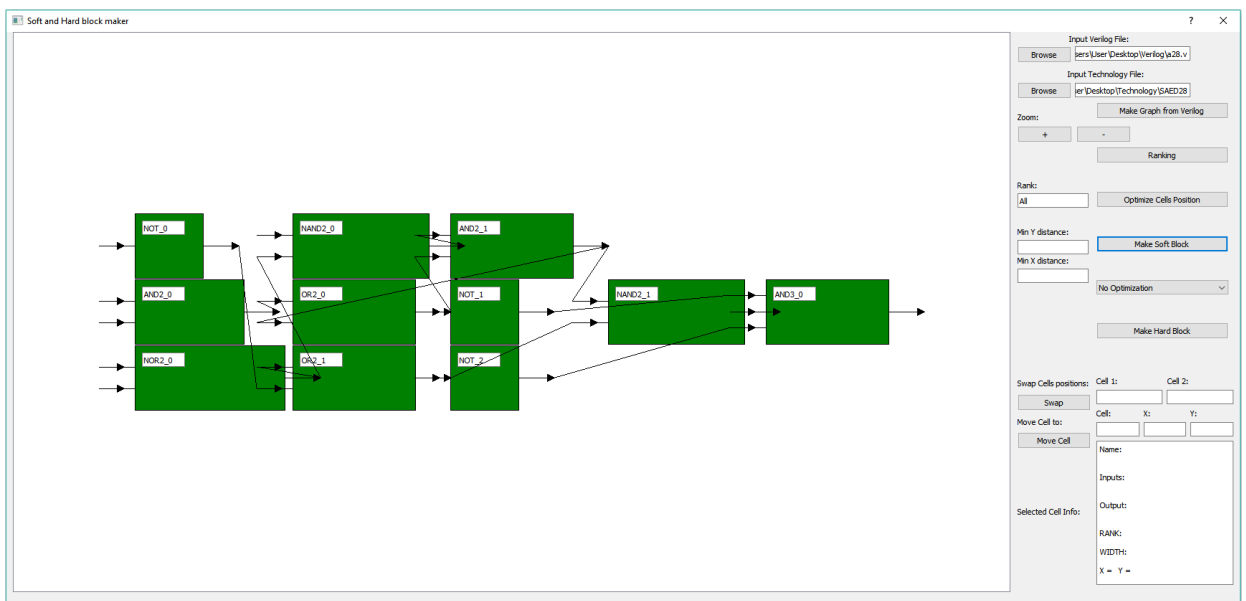
Նկ. 4.6. a28 թեստային Verilog ֆայլի՝ ռանգավորված FU-ի գրաֆիկական տեսքը

9. -ը «Optimize Cells Position» կոճակն է, որի միջոցով կատարվում է 10-ում տրված ռանգ ունեցող տարրերի միջև հեռավորությունների օպտիմացում՝ 12, 13-ում տրված նվազագույն X և Y առանցքներով հեռավորության համաձայն: Օրինակ՝ ռանգ 2, Y նվազագույն հեռավորությունը՝ 100-ի դեպքում, գրաֆիկական տեսքը՝ նկ. 4.7-ում պատկերված տեսքը կունենա՝



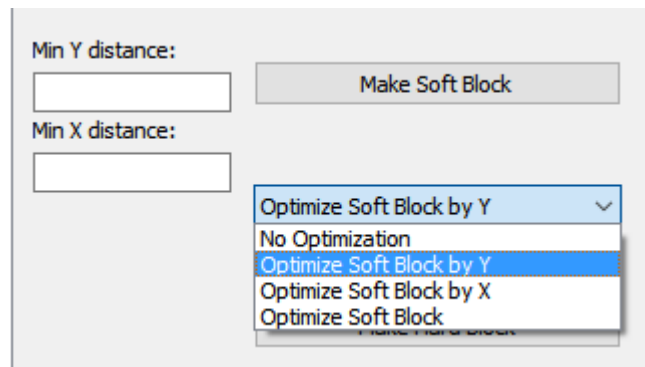
Նկ. 4.7. 2-րդ ռանգի տարրերի միջև հեռավորության օպտիմալացումը

10. -ում նշվում է այն ռանգի համարը, կամ եթե բոլորն է պետք «All» կամ ոչինչ, որը պետք է օպտիմացվի «Optimize Cells Position» կաճակով,



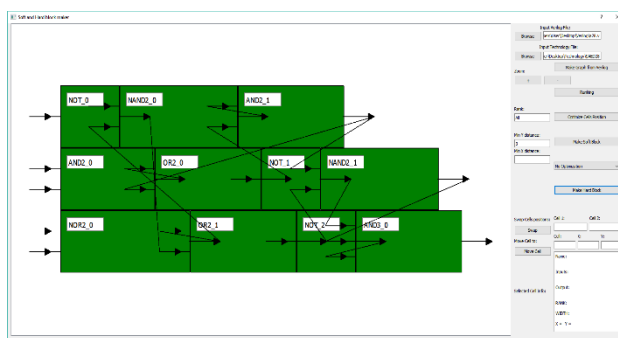
Նկ. 4.8. Ֆիզիկական «փափուկ» բլոկի գրաֆիկական տեսքը

11. -ը «Make Soft Block» կոճակն է, որի միջոցով իրականացվում է ֆիզիկական «փափուկ» բլոկի ստացումը՝ ըստ 3.4 գլխում նշված ալգորիթմի (նկ. 4.8)[86],
12. -ում տրվում «փափուկ» բլոկ կառուցելիս ըստ Y առանցքի տարրերի միջև հեռավորության նվազագույն չափը,
13. -ում տրվում «փափուկ» բլոկ կառուցելիս ըստ X առանցքի տարրերի միջև հեռավորության նվազագույն չափը,
14. -ը հնարավորություն է տալիս «փափուկ» բլոկ կառուցելիս ընտրելու տարրերի միջև հեռավորության օպտիմալացման առանցքը(նկ. 4.9),

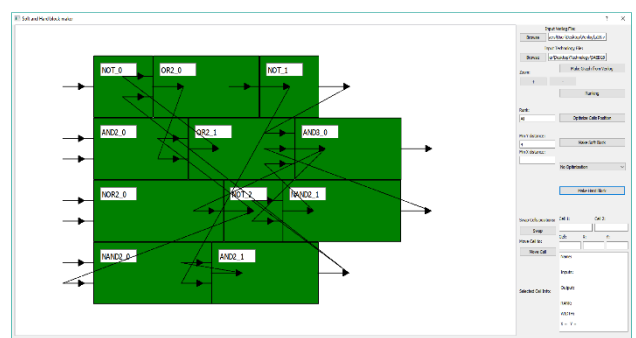


Նկ. 4.9. Ֆիզիկական «փափուկ» բլոկ կառուցելիս տարրերի միջև հեռավորության օպտիմալացման առանցքի ընտրության տարբերակները

15. -ը «Make Hard Block» կոճակն է, որի միջոցով իրականացվում է ֆիզիկական «կոշտ» բլոկի ստացումը՝ ըստ 3.5 գլխում նկարագրված ալգորիթմի(նկ. 4.10)[84, 87], նաև հնարավորություն է տալիս նախագծողին՝ 12-ի միջոցով դեկավարելու «կոշտ» բլոկի կամայական սյան մեջ եղած տարրերի քանակը, օրինակ 3 կամ 4 (նկ. 4.10 ա, բ),



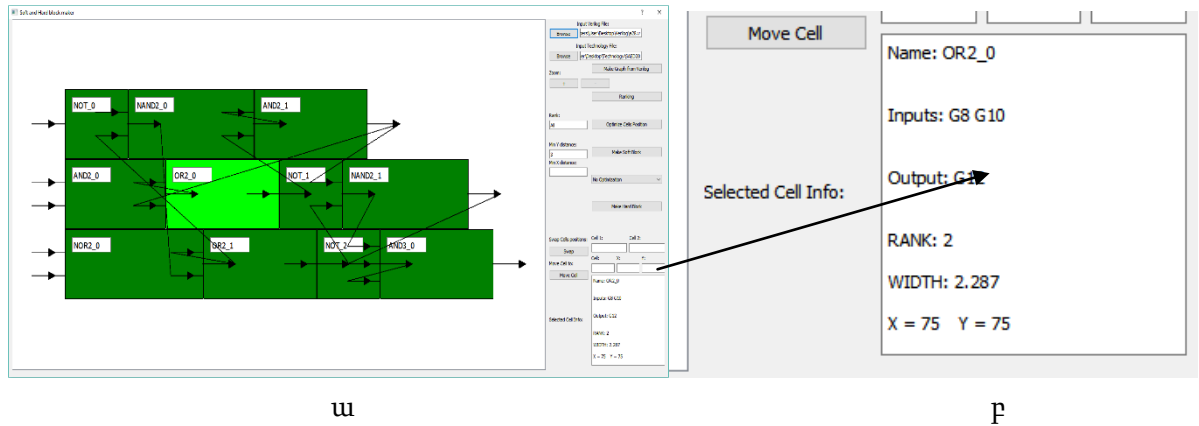
ա



բ

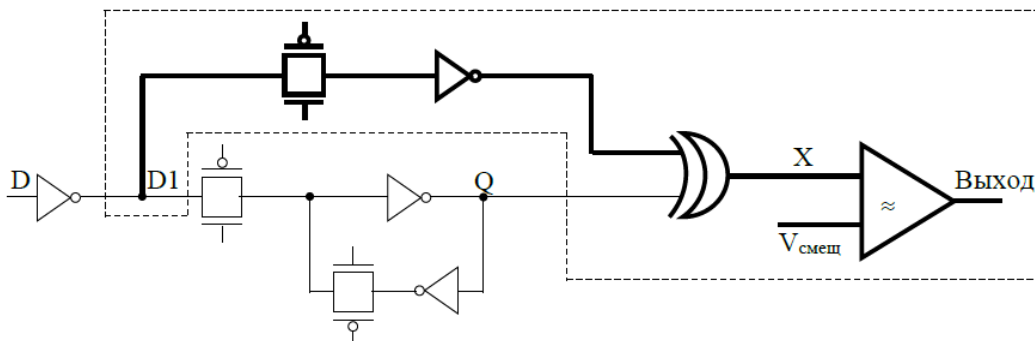
Նկ. 4.10. Ֆիզիկական «կոշտ» բլոկի գրաֆիկական տեսքը՝ ա) 3, բ) 4 տարր ամեն ռանգում

16. -ը հնարավորություն է տալիս նախագծողին՝ «Cell1» և «Cell2» դաշտերում գրված տարրերը «Swap» կոճակի միջոցով տեղերով փոխելու,
17. -ը հնարավորություն է տալիս նախագծողին՝ «Cell» դաշտում գրված տարրը «Move» կոճակի միջոցով տեղափոխելու «X» և «Y» դաշտերում տրված կոորդինատը,
18. -ը տալիս է այն տարրի մանրամասն նկարագրությունը, որը մկնիկով նշել(ընտրել) է նախագծողը (նկ. 4. 11):



Նկ. 4.11. Ֆիզիկական «կոշտ» բլոկում՝ ա) ընտրված OR2_0 տարրը, բ) ընտրված տարրի տվյալները

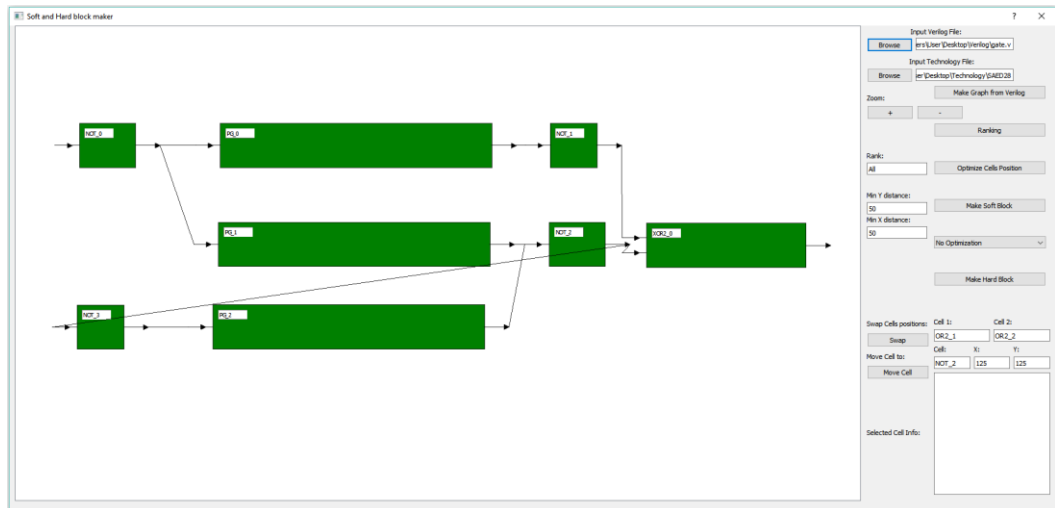
Մշակված ծրագրային գործիքի արդյունավետության գնահատման համար կատարվել է ISCAS85 շարքի մի շարք թեստային սխեմաների [46], ինչպես նաև «Սինոփսիս-Արմենիայում» մշակված՝ փականային մակարդակի PG9 սխեմայի (նկ.4.12) տարրերի ավտոմատացված ռանգավորումը [88]:



Նկ. 4.12. Փականային մակարդակի PG9 մշակված սխեմա

Մասնավորապես PG9 սխեմայի ռանգավորման արդյունքը, կատարված SHBM ծրագրային գործիքի օգնությամբ, բերված է նկ.4.13-ում: Որպես տարրային հենք օգտագործվել է «Սինոփսիս-Արմենիայում» մշակված SAED32/28 թվային ստանդարտ տարրերի գրադարանը [89]:

Շղթաների երկարությունների հաշվարկի համեմատական արդյունքները բերված են աղ. 4.1-ում[87]:



Նկ. 4.13. Փականային մակարդակի PG9 մշակված սխեմայի ավտոմատացված ռանգավորումը

Աղյուսակ 4.1

Թեստային սխեմաների տարրերի տեղաբաշխման շղթաների երկարությունների հաշվարկի արդյունքները

պարամետրեր			Թեստային սխեմա					
			C17	PG9	a28	C432	C1908	C5315
Տարրերի քանակ			6	8	11	161	879	2307
Շղթաների քանակ			11	9	17	272	1028	3008
Կողերի հարաբերություն			2/3	4/2	3/4	10/16	28/32	45/52
Մեկ շղթայի միջին երկարություն (մկմ)	Առավելագույն պահուստ	ա	2.1	2.3	2.9	9.2	34.4	82.8
		բ	3.5	3.6	3.8	15.4	57.3	138.5
		գ	3.5	3.5	3.6	14.1	52.3	125.4
	Միջին պահուստ	ա	-	4.0	4.5	6.9	17.5	34.6
		բ	-	4.0	4.5	7.1	19.5	36.2
		գ	-	4.0	4.5	6.8	17.8	35.6

	Որոշիչ պատաս	ա	1.8	1.9	2.5	3.4	11.4	13.8
		բ	1.8	2.0	2.8	3.7	12.6	15.2
		գ	1.8	1.9	2.5	3.6	12.1	14.4

Աղ. 4.1-ում ա, բ և գ տողերը համապատասխանում են՝ *ա-ռանգավորման հիման վրա տեղաբաշխում, բ- ժամանակային տեղաբաշխում՝ հիմնված հաջորդական տեղաբաշխման ալգորիթմի վրա, գ-Սինոփսիս IC Compiler նախնական տեղաբաշխում, կողերի հարաբերություն- (ուղղաձիգ մակարդակների քանակ) / (հորիզոնական մակարդակների քանակ)*

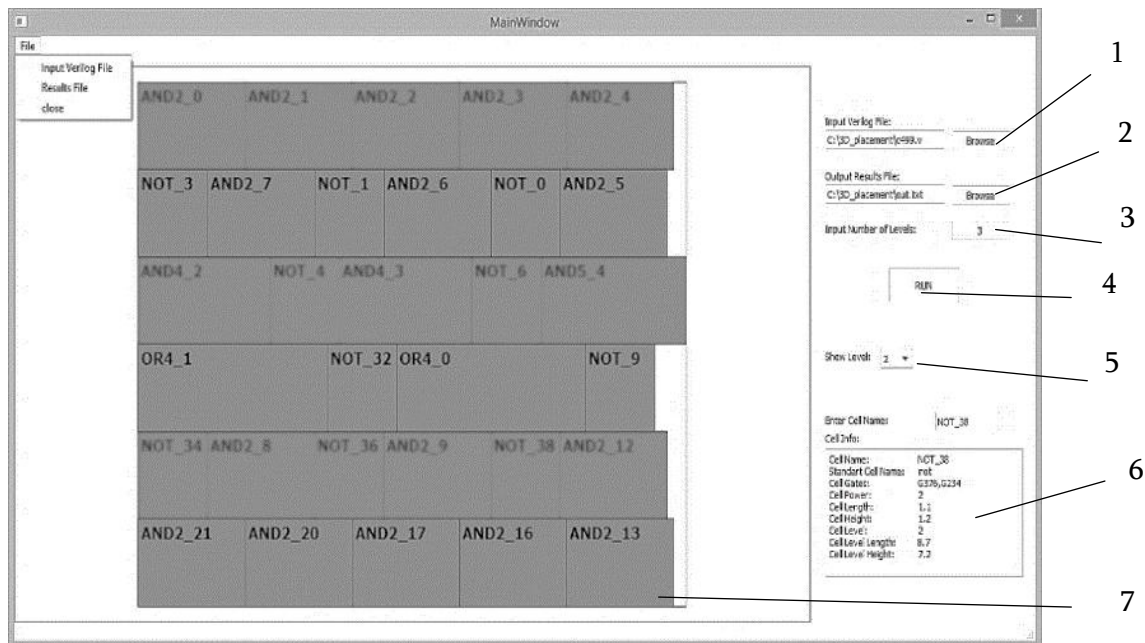
Ինչպես երևում է աղ. 4.1-ից, առաջարկված տեղաբաշխման մեթոդի արդյունքում որոշիչ շղթաների համար, առավելագույն ժամանակի պահուստով շղթաների համեմատ, ապահովվել է 14 ... 80% կարճացում: Տեղաբաշխման արդյունքները համեմատվել են նույն ալգորիթմական բարդությունն ունեցող նախնական տեղաբաշխման հաջորդական ալգորիթմի և «Սինոփսիս-Արմենիա» ընկերության IC Compiler ծրագրային գործիքի նախնական տեղաբաշխման արդյունքների հետ [20, 82]:

Առաջարկվող ալգորիթմի դեպքում որոշիչ շղթաների համար ապահովվում է միջինում 7%, իսկ առավելագույն ժամանակի պահուստով շղթաների համար՝ 35% կարճացում: Միաժամանակ, նկատվել է միջմիացումների գումարային երկարության ավելի քան 10% կարճացում:

4.2. Դինամիկ հզորությունների հաշվառմամբ տարրերի տեղաբաշխման ծրագրային գործիքի իրականացումը

Այս ենթագլխում ներկայացված է 3. 5 ենթագլխում նկարագրված ալգորիթմի հիման վրա իրագործված եռաչափ ԻՍ-երի տարրերի նախնական տեղաբաշխման ծրագրային միջոցը, որը փորձարկված է Iscas 85 սերիայի մի շարք թեստային սխեմաների դեպքում [25]:

Ծրագրի աշխատանքային պատուհանը բերված է նկ. 4. 14-ում:



1. Մուտքային Verilog ֆայլը, 2. Ելքային ֆայլի հասցեն, 3. Մակարդակների թիվը, 4. Ծրագրի աշխատանքի մեկնարկի կոճակը 5. Արտապատկերվող մակարդակի համարը, 6. Տեղաբաշխվող բջջի անվանումը և տվյալները, 7. Տրված մակարդակի տեղաբաշխման արտապատկերումը

Նկ. 4.14. Ջերմային տեղաբաշխման ծրագրի աշխատանքային պատուհանը

Այն ունի նախագծողին հարմարեցված ինտերֆեյս, որը հնարավորություն է տալիս ստանալ և արտապատկերել եռաչափ ԻՄ-երի յուրաքանչյուր մակարդակի բյուրեղի վրա տարրերի տեղաբաշխումը[90]: Ծրագրի համար որպես մուտք են ծառայում սխեմայի Verilog նկարագիրը և որպես տարրային հենք օգտագործվող ստանդարտ գրադարանային բջիջների չափսերի և հզորությունների արժեքները: ԻՄ-ի մակարդակների թիվը ներմուծվում է նախագծողի կողմից: Ընդհանուր դեպքում, ծրագրային միջոցն իրագործում է տեղաբաշխման երեք փուլ [25]՝

- սխեմայի մասնատումը առանձին բյուրեղների վրա իրականացվող ենթասխեմաների,
- առանձին բյուրեղների հզորությունների որոշում և եռաչափ տեղաբաշխում՝ ըստ մակարդակների,
- յուրաքանչյուր մակարդակի համար երկչափ ջերմային տեղաբաշխում:

Այս աշխատանքի շրջանակում իրականացվել է տվյալ ծրագրային միջոցով տարրերի ներմակարդակային տեղաբաշխումը՝ ըստ ջերմային համահարթեցման պայմանի:

Աղ. 4.2-ում ներկայացված է նկ. 4. 14-ի աշխատանքի նկարագրությունը[91]:

Աղյուսակ 4.2

Ջերմային տեղաբաշխման ծրագրի աշխատանքային պատուհանի աշխատանքի նկարագրությունը

	Անվանում	Տիպ	Նկարագրություն
1	[Input Verilog File]	QPushButton QLabel QLineEdit	Տրվում է մուտքային վերիլոգ ֆայլը՝ մուտքային տվյալները վերցնելու համար:
2	[Output Results File]	QPushButton QLabel QLineEdit	Տրվում է ցանկացած դատարկ ֆայլ՝ ելքային ինֆորմացիան գրելու համար: Ֆայլում գրանցվում են մակարդակները իրենց տարրերով և ըստ տեղաբաշխման ճիշտ հերթականության:
3	[Input Number of Levels]	QLabel, QLineEdit	Այս դաշտի միջոցով տրվում է, թե քանի մակարդակի պետք է բաժանվի տրված ինտեգրալ սխեման: Թույլատրվում է ներմուծել միայն ամբողջ թվեր:
4	[RUN]	QPushButton	[RUN]-ի միջոցով աշխատում է ծրագրը, որի քայլերի հաջորդականությունը նկարագրված է 3. 5 գլխում
5	[Show level]	QComboBox	[Show level]-թույլ է տալիս ընտրել ինտեգրալ սխեմայի այն հարկը, որը պետք է պատկերվի:

6	[Enter Cell Name]	QLabel, QLineEdit	Մուտքագրում ենք տրված վերիլոգ ֆայլից ցանկացած cell և ստանում նրա մասին ամբողջ ինֆորմացիան Cell Info դաշտում:
7	[GraphicsView]	QGraphicsView	Պատկերում է նշված մակարդակի ինտեգրալ սխեմայի տարրերի տեղաբաշխումը:

Առաջարկվող մեթոդը փորձարկվել է ISCAS 85 մի շարք մինչև 2300 տարրեր պարունակող փորձական սխեմաների տարրերի տեղաբաշխման համար, և տվել է (3.11) ցուցանիշի արդյունավետություն՝ միջինը 50%-ով:

Տեղաբաշխման ընդհանուր որակը հզորությունների համահարթեցման տեսանկյունից գնահատվել է տեղային ջերմային գոտում՝ միջին հզորության արտահայտության հարաբերական արժեքի գնահատման (3.12) արտահայտությամբ:

Որպես տեղային ջերմային գոտի վերցվել է 4 հարևան բջիջներից կազմված տեղամասը:

Տարբեր թեստային սխեմաների համար տեղաբաշխման որակի համեմատական արդյունքները ներկայացված են աղ. 4.3-ում [90]:

Աղյուսակ 4.3

Ջերմային տեղաբաշխման որակը տարբեր թեստային սխեմաների դեպքում

Սխեմայի նշանակումը	C499	C3540	C5315
Բջիջների քանակը	202	1669	2307
Տեղաբաշխման ընդհանրացված որակը՝ ըստ հզորությունների, W [%]	60	45	40

Այսպիսով, 3-րդ գլխում ներկայացված ալգորիթմների հիման վրա իրագործվել է երկու ծրագրային միջոց, որոնցից առաջինը նախագծողին հնարավորություն է տալիս որպես վերջնական արդյունք ստանալ ֆիզիկական «կոշտ» բլոկ, որը հաշվի է առնում սխեմատեխնիկական առանձնահատկությունները՝ մասնավորապես հիմնված է ԻՄ-ում միջմիացումների կրճատմանը, իսկ երկրորդը նախագծողին հնարավորություն է տալիս որպես վերջնական արդյունք կատարել ֆիզիկական նախագծում, մասնավորապես, հիմնված է ԻՄ-ում միջմիացումների կրճատման և ջերմային հուսալիության

բարձրացման վրա: Ներկայացվել են ծրագրային միջոցների աշխատանքի նկարագրությունը և փորձնական հետազոտության արդյունքները:

Եզրակացություններ

1. Իրագործվել է երկու ավտոմատացված ծրագրային միջոց՝ 3-րդ գլխում ներկայացված ալգորիթմների հիման վրա :
2. Առաջին ավտոմատացված ծրագրային միջոցը նախագծողին հնարավորություն է տալիս որպես վերջնական արդյունք ստանալ ֆիզիկական «կոշտ» բլոկ, որը հաշվի է առնում սխեմատիկական առանձնահատկությունները՝ մասնավորապես հիմնված է ԻՍ-ում միջմիացումների կրճատումը:
3. Երկրորդը ավտոմատացված ծրագրային միջոցը նախագծողին հնարավորություն է տալիս որպես վերջնական արդյունք կատարել ֆիզիկական նախագծում, մասնավորապես, հիմնված է ԻՍ-ում միջմիացումների կրճատման և ջերմային հուսալիության բարձրացման վրա:
4. Ներկայացվել են ավտոմատացված ծրագրային միջոցների աշխատանքի նկարագրությունը և փորձնական հետազոտության արդյունքները:

ԵԶՐԱՀԱՆԳՈՒՄ

1. Ժամանակակից նանոմետրական ԻՄ-երի ֆիզիկական նախագծման մարտահրավերների և առկա իրողությունների վերլուծության արդյունքում փաստագրվել է, որ ԻՄ-երի որոշ ֆիզիկական պարամետրերի ազդեցությունը նրանց աշխատանքի ֆունկցիոնալ-տրամաբանական որակների վրա դարձել է որոշիչ: Գրականության վերլուծությունը ցույց է տվել, որ նշված խնդրի ներկայիս լուծումները բավարար արդյունքների չեն հանգեցնում և կարիք ունեն զարգացման: Այսպիսով, միկրոէլեկտրոնային տեխնոլոգիաների մասշտաբավորմանը զուգընթաց՝ անհրաժեշտություն է առաջանում ԻՄ-երի ավտոմատացված ֆիզիկական նախագծման մեթոդների անընդհատ կատարելագործման և այնպիսի նոր մեթոդների մշակման, որոնք հնարավորինս հաշվի կառնեն դրանց սխեմատեխնիկական առանձնահատկությունները:

2. Հիմնավորված է, որ որպես ԻՄ-երի ավտոմատացված ֆիզիկական նախագծման փուլում սխեմատեխնիկական առանձնահատկությունների հաշվառման հիմնական միջոց կարող է ծառայել տարրերի տեղաբաշխման խնդրի լուծման այնպիսի մեթոդների մշակումը, որոնք հաշվի կառնեն սխեմայի աշխատանքի վրա ազդող առավել կարևոր պարամետրերը: Այդ պարամետրերից, իրենց կարևորությամբ, առանձնանում են միջմիացումներով պայմանավորված հապաղումները և կիսահաղորդչային բյուրեղի վրա տեսակարար հզորության աճով պայմանավորված հուսալիությունը:

3. Առաջարկվել է և ԻՄ-երի տարրերի տեղաբաշխման փուլում միջմիացումներով պայմանավորված հապաղումները հաշվի առնող նոր մեթոդ և դրա ավտոմատացման ծրագրային միջոց, որը, ի տարբերություն ժամանակով ուղղորդված տեղաբաշխման առկա մեթոդների, հաշվի է առնում սխեմայում տեղեկության մշակման տրամաբանական հաջորդականությունը: Այն հիմնված է տրամաբանական սխեմայի ռանգավորման և ֆիզիկական «փափուկ» ու «կոշտ» բլոկների ավտոմատացված սինթեզի վրա և միտված է միջմիացումների երկարության և հետևաբար,՝ դրանցում հապաղումների նվազեցմանը:

4. Առաջարկվում է տրամաբանական տարրերի աշխատանքային դինամիկ հզորությունների գնահատման մեթոդ՝ կախված տարրի տիպից, սխեմայի

տրամաբանությունից և սխեմայում տարրի դիրքից: Սա թույլ է տվել, ի տարբերություն տարրերի ջերմային տեղաբաշխման գոյություն ունեցող մեթոդների, հաշվի առնել սխեմայի աշխատանքի տրամաբանությունը, որը, իր հերթին, հանգեցնում է սխեմայի ջերմային հուսալիության բարձրացմանը:

5. Մշակվել է թվային սխեմաների սխեմատիկական առանձնահատկությունների հաշվառմամբ ԻՄ-երի ֆիզիկական նախագծման ավտոմատացման ենթահամակարգ, որը ներառում է երկու ծրագրային միջոցներ՝

- Verilog նկարագրից ֆիզիկական կառուցվածքի «փափուկ» ու «կոշտ» բլոկների ավտոմատացված սինթեզի SHBM ծրագրային միջոցը,
- տրամաբանական տարրերի աշխատանքային դինամիկ հզորությունների գնահատման և դրանց հիման վրա ջերմային տեղաբաշխման ծրագրային միջոցը:

6. Մշակված ծրագրային միջոցները փորձարկվել են մի շարք թեստային սխեմաների վրա և կիրառման արդյունավետության տեսանկյունից ցուցաբերել են հետևյալ արդյունքները.

- SHBM ծրագրային միջոցն ապահովում է որոշիչ շղթաների երկարությունների հարաբերական նվազում՝ առավելագույն պաշարով շղթաների համեմատ 14 ... 80% -ով: Միննույն մեքենայական ռեսուրսի պայմաններում, համեմատած հաջորդական տեղաբաշխման ալգորիթմի և «Մինոփսիս-Արմենիայի» IC Compiler ծրագրային գործիքի, ապահովվել է որոշիչ շղթաների երկարությունների կրճատում՝ համապատասխանաբար 10% և 5% -ով, իսկ առավելագույն պաշարով շղթաներինը՝ 40% և 33% -ով, ինչը հանգեցնում է միջմիացումների գումարային երկարության 16% և 12%-ով կրճատմանը:
- Ջերմային տեղաբաշխման ծրագրային միջոցը համեմատվել է նույն սխեմաների տարրերի տեղաբաշխման հետ՝ առանց աշխատանքային դինամիկ հզորությունների հաշվառման, և ապահովվել է ջերմային դաշտի՝ միջինը 50% -ով ավելի համահարթեցում:

ՕԳՏԱԳՈՐԾՎԱԾ ԳՐԱԿԱՆՈՒԹՅԱՆ ՑԱՆԿ

1. The International Technology Roadmap for Semiconductors (ITRS).- 2013, <http://www.itrs.net/Links/2013ITRS/2013Chapters/2013SysDrivers.pdf>.
2. The International Technology Roadmap for Semiconductors (ITRS),- More Moore, 2015, <https://www.semiconductors.org/clientuploads/Research Technology/ITRS/2015/5 2015%20ITRS%202.0 More%20Moore.pdf>.
3. Մելիքյան Վ.Շ., Հարությունյան Ա.Գ., Գևորգյան Ա.Ա. Միկրոէլեկտրոնային սխեմաների ֆիզիկական նախագծման մեթոդներ. Մենագրություն/ ՀՊՃՀ. -Եր.: Ճարտարագետ, 2015.-240 էջ:
4. Tong Gao. The physical design challenges of 20 nm processes. - Synopsys, 2012, <http://www.techdesignforums.com/practice/technique/20nm-physical-design-tong-gao-blog/>.
5. Khang A., Carballo J. ITRS 2.0, System Integration Focus Team. - Stanford, 2015.- 39 p.
6. Chang Yao-Wen. Physical Design for Nanometer ICs Graduate Institute of Electronics Engineering; Department of Electrical Engineering; National Taiwan University. - Spring, 2017, <http://cc.ee.ntu.edu.tw/~ywchang>. -28p.
7. A. Sherwani Naveed. Algorithms for VLSI Physical Design Automation. - Intel Corporation. -Kluwer Academic Publishers, 2007. - 572 p.
8. IC Compiler User Guide: Implementation Version B-2008.09.- Synopsys, Inc, September 2008. <http://solvnet.synopsys.com>.- 786p.
9. Talus Automated Chip Creation Methodology Magma Design Automatin 1650 Technology, Inc. www.magma-da.com.-San Jose, 2006.- 12 p.
10. Friedman G.On-Chip Interconnect: The Past, Present, and Future.- 2006.-68p. <https://docplayer.net/6522114-On-chip-interconnect-the-past-present-and-future.html>
11. International Technology Roadmap for Semiconductors, Semiconductor Industry Association, - 2015. <http://public.itrs.net/.2011>.
12. Тепловой режим процессоров Pentium4-athlonx”.

- <http://www.ixbt.com/cpu/Pentium4-athlonxp-thermal-managment.shtml>. - 15c.
13. De Orio R.L., Ceric H., Selberherr S. Physically based models of electromigration: From Black's equation to modern TCAD models Institute for Microelectronics, TU Wien Gußhausstraße 27–29/E360, A-1040.- Microelectronics Reliability 50. Wien, Austria, 2010. –P. 775-789. www.elsevier.com/locate/microrel
 14. Kahng Andrew B., Kang Sung-Mo, Li Wei, Liu Bao. Analytical Thermal Placement for VLSI Lifetime Improvement and Minimum Performance Variation CSE Dept.- UC San Diego, CA, 2007. 7 IEEE. .P. 71-77.
 15. Tsai J.-L., Chen C., Chen G., Goplen B., Qian H., Zhan Y., Kang S.-M., Wong M. D. F., and Sapatnekar S.S. TemperatureAware Placement for SOCs, IEEE Special Issue on On-Chip Thermal Engineering. - Aug. 2006.- P. 1502-1518.
 16. Nassif Sani R., Nowka Kevin J.. IBM Research. Physical design challenges beyond the 22nm node.ISPD '10 Proceedings of the 19th International Symposium on Physical Design.- ACM New York, NY, USA, 2010.- P 13-14.
 17. Physical Design Challenges. - Cadence Design Systems, Inc. All Rights Reserved. - 2012, http://www.cadence.com/Training/Pages/physical_design.aspx.2012.
 18. Jean-Marie Brunet. Challenges Of Physical Design Closure.- Mentor Graphics, 2012, <http://semiengineering.com/challenges-physical-design-closure/>
 19. Угрюмов Е.П. Цифровая схемотехника: Учеб. Пособие для вузов.- 2-е изд., перераб. и доп.- СПб.: БХВ-Петербург, 2004. -544с.
 20. Ren H., Pan D., Kung D. Sensitivity guided net weighting for placement driven synthesis // IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems. - May, 2005 (ISPD'04).- P. 711–721.
 21. Recent Research&Emerging Challenges in Physical Design for manufacturability/Reliability, Chunh-Wei Line, et al., - Retrieved on 01 Apr. 2012.
 22. Michael Keating, David Flynn, Robert Aitken, et al.- Low Power MethodologyManualFor System-on-Chip Design.Springer, 2007.- 299p.
 23. Մելիքյան Վ.Շ., Հարությունյան Ա.Գ., Մաթևոսյան Ա.Վ., Պետրոսյան Հ.Պ. Ցածր էներգասպառմամբ ինտեգրալ սխեմաների նախագծում. Մենագրություն/ ՀՊՃՀ. - Եր.: Ճարտարագետ, 2012.-195 էջ:

24. Reddy V. Introduction to semiconductor reliability// 2003 International Reliability Physics Symposium. - Dallas, Texas, 2003.- 111. -P. 1-74.
25. Հարությունյան Ա.Գ., Մարտիրոսյան Ա.Ռ., Հարությունյան Հ.Ժ., Գասպարյան Տ.Ա. Եռաչափ ինտեգրալ սխեմաների տարրերի ջերմային տեղաբաշխումը // ՀԱՊՀ Լրաբեր. Գիտական հոդվածների ժողովածու. - Երևան, 2017.- Մաս 1. – էջ 283-288:
26. Geert Vanwijnsberghe, VLSI-Physical Design- Tool Terminology. - 2013.- 105p.
<https://www.slideshare.net/muralikancham/vlsi-floor-planning>.
27. Alpert Charles J., Shrirang Karandikar, Zhuo Li, Gi-Joon Nam, Techniques for Fast Physical Synthesis, Proceedings of the IEEE. - March 2007.- Vol. 95, No.3.- P. 573-599.
28. Alpert Charles J., Mehta Dinesh P., and Sapatnekar Sachin S. Handbook of Algorithms for Physical Design Automation. - Auerbach Publications, Boston, MA, UA, 2008.
29. <http://www.vlsi-expert.com/2011/03/static-timing-analysis-sta-basic-timing.html>
30. <https://www.coursera.org/lecture/vlsi-cad-layout/interconnect-timing-electrical-models-of-wire-delay-zqafA>
31. http://www.ee.bgu.ac.il/~digivlsi/slides/STA_9_1.pdf
32. <http://scale.engin.brown.edu/classes/EN0291S40F06/lecture04.pdf>
33. http://www.ee.hacettepe.edu.tr/~alkar/ELE432/ELE432_2.pdf
34. <https://www.synopsys.com/>
35. <https://www.cadence.com/>
36. <https://www.mentor.com/>
37. Lockhart D. Place and Route using Synopsys IC Compiler. ECE5745 Tutorial 3 (Version 606ee8a). - January 30, 2016,
<https://web.csl.cornell.edu/courses/ece5745/handouts/ece5745-tut3-icc.pdf>
38. PrimeTime Golden Timing Signoff Solution and Environment.
<https://www.synopsys.com/content/dam/synopsys/implementation&signoff/datasheets/primetime-ds.pdf>
39. Kahng Andrew B., Lienig Jens, Markov Igor L., Hu Jin. VLSI Physical Design: From Graph Partitioning to Timing Closures. - Springer Science+Business Media B.V, 2011.-310p.
40. How accurately can we model timing in a placement engine/ A. Chowdhary, K. Rajagopal, S. Venkatesan, et al // Proc. Design Automation Conf.- 2005.- P. 801–806.

41. Джурунтаев Д.З. О сокращении длины соединений по критическому пути при размещении элементов СБИС Математическое моделирование научно-технологических и экологических проблем в нефтегазодобывающей промышленности: vii Казахстанско-Российская Междун. науч.-практ. конференция. - 2010. - С. 87-92.
42. Marilyn Wolf. High-Performance Embedded Computing. - Second Edition. - 2014. - 484p.
43. Surin Kittitornkun, Yu-Hen Hu. Electrical Engineering Handbook. - 2005. - P. 911-948.
44. https://en.wikipedia.org/wiki/Signal-flow_graph#Linear_signal-flow_graph_examples
45. Клиначёв Н.В., Введение в технологию моделирования на основе направленных графов.- 2013,
http://model.exponenta.ru/lectures/sml_02.htm
46. Benchmark circuits. ISCAS '85,
<http://www.pld.ttu.ee/~maksim/benchmarks/iscas85/verilog/?C=D;O=A>
47. Digital Standard Cell Library. - SAED_EDK32/28 SYNOPSIS ARMENIA Educational Department.- Yerevan, 2011. -152 p.
48. <https://docplayer.ru/30365041-3-logicheskoe-modelirovanie.html>
49. Ильин В.Н., Фролкин В.Т., Бутко А.И., Камнева Н.Ю., и др. Автоматизация схемотехнического проектирования. -М.: Радио и связь, 1987. -368с.
50. <http://www.myshared.ru/slide/654712/>
51. Melikyan V., Harutyunyan A., Matevosyan A., Chobanyan S., et al. Physikal Design Implementation. - Yerevan: Chartaraget, 2014.-360p.
52. <http://asic-soc.blogspot.com/2007/11>
53. Մելիքյան Վ.Շ., Հարությունյան Ա.Գ., Պողոսյան Ա.Մ., Համակարգային ինտեգրալ սխեմաների նախագծում: Մենագրություն/ ՀՊՃՀ, “Մինոփսիս Արմենիա” ՓԲԸ; ընդ. խմբ. պրոֆ. Մելիքյանի Վ.Շ.. -Եր.: Ճարտարագետ, 2013.-195 էջ:
54. Бабурин Д.Е. Иерархический подход для автоматического размещения ациклических графов // Современные проблемы конструирования программ.- Новосибирск: ИСИ СО РАН, 2002.- С. 7-37.

55. Junger M., Mutzel P. 2-layer straightline crossing minimization: performance of exact and heuristic algorithms // Journal of Graph Algorithms and Applications. - 1997.- vol. 1, no. 1, -1997.- pp. 1-25.
<http://www.cs.brown.edu/publications/jgaa/>
56. Di Battista G., Tamassia R. Algorithms for plane representations of acyclic digraphs // Theor. Comput. Sci. - 1988. - Vol. 61. -P. 175–198.
57. Cormen Thomas H., Stein Clifford, Rivest Ronald L., Leiserson Charles E. "Preface". Introduction to Algorithms (3 ed.). Cambridge, Massachusetts: MIT Press. - 2009.- pp. 147–228.
58. https://en.wikipedia.org/wiki/Sorting_algorithm
59. https://en.wikipedia.org/wiki/Bubble_sort
60. <https://en.wikipedia.org/wiki/Quicksort>
61. <https://www.geeksforgeeks.org/quick-sort/>
62. https://en.wikipedia.org/wiki/Insertion_sort
63. <https://www.geeksforgeeks.org/insertion-sort/>
64. Sugiyama K., Tagawa S., Toda M. Methods for visual understanding of hierarchical systems // IEEE Trans. Syst., Man, and Cybern. - 1981. - Vol. 11, N. 2. -P. 109–125.
65. Makinen E. Experiments on drawing 2-level hierarchical graphs // Intern. J. of Comput. and Mathematics. — 1990. — Vol. 36. — P. 175–181.
66. Shieh F.S., McCreary C.L. Directed graphs drawing by clan-based decomposition // Lect. Notes Comput. Sci. - 1995. - Vol. 1027. - P. 472–482.
67. Eades P., Wormald N. Edge crossing in drawings of bipartite graphs // Algorithmica. - 1994. - Vol. 11. - P. 379–403.
68. Черемисинова Л.Д. Логический синтез комбинационных КМОП схем с учетом рассеивания мощности// Вестник Томского государственного университета. Управление, вычислительная техника и информатика. – Томск, 2014.- № 3 (28). -С. 89-98.
69. Di Battista G., Eades P., Tamassia T., Tollis I.G. Algorithms for drawing graphs: Annotated bibliography Comput. Geom. Theory Appl. — 1994. — P. 235–282.

70. Di Battista G., Eades P., Tamassia T., Tollis I.G. Algorithms for the visualization of graphs. — Prentice–Hall, 1999. — 398 p.
71. Wikipedia the free encyclopedia. - Graph Drawing, - 2019, https://en.wikipedia.org/wiki/Graph_drawing
72. Di Battista G., Garg A., Liotta G., Parise A. et al Drawing directed acyclic graphs: an experimental study Lect. Notes Comput. Sci. — 1997. — Vol. 1190. — P.76–91.
73. Wikipedia the free encyclopedia: Object-oriented programming.- 2014, https://en.wikipedia.org/wiki/Object-oriented_programming
74. Vankayala Karunakar. Basic Concepts of Object-Oriented Programming using C++. -2016, <https://www.geeksforgeeks.org/basic-concepts-of-object-oriented-programming-using-c/>
75. Map In C++ Standart Template Library (STL).- 2016, <https://www.geeksforgeeks.org/map-associative-containers-the-c-standard-template-library-stl/>
76. Wikipedia the free encyclopedia: Self-balancing binary search tree, last modified.- 2019, https://en.wikipedia.org/wiki/Self-balancing_binary_search_tree
77. Wikipedia the free encyclopedia: Array programming, last edited. - 2019, https://en.wikipedia.org/wiki/Array_programming
78. std::vector, last edited.- 2018, <https://en.cppreference.com/w/cpp/container/vector>
79. Wikipedia the free encyclopedia: last edited.- 2019, [https://en.wikipedia.org/wiki/Class_\(computer_programming\)](https://en.wikipedia.org/wiki/Class_(computer_programming))
80. Wikipedia the free encyclopedia, C++ Standard Library, last modified.- 2019, https://en.wikipedia.org/wiki/C%2B%2B_Standard_Library
81. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. - 2-е изд-е. - М.: Вильямс, 2005. - С. 622-632.
82. Арутюнян А.Г. Начальное размещение логических ячеек интегральных схем с учетом важности цепей // Проблемы разработки перспективных микро- и нанoeлектронных систем – 2014: Сборник трудов / Под общ. ред. академика РАН А.Л. Стемпковского. - М.: ИППМ РАН, 2014. - Часть 1. - С.143-146.

83. Roberto Tamassia. Handbook Graph Drawing and Visualization. - CRC Press 2013Taylor&Francis Group. Northampton, Massachysetts, USA. -2014.- 852p.
84. Մարտիրոսյան Ա.Ռ., ISSN 0002-306X. ՀԳԱԱ և ՀԱՊՀ Տեղ. Տեխն. գիտ. սերիա. 2019. Հ. LXXII, N2. – էջ 225-234.
85. Мартиросян А.Р. Размещение элементов логических схем на основе графа потока сигналов Вестник Инженерной академии Армении (БИАА). - 2018. - Т.150, № 4. - С. 620-626.
86. Մարտիրոսյան Ա.Ռ., «Փափուկ» ֆիզիկական կառուցվածքով բլոկների սինթեզը թվային ինտեգրալ սխեմաների տրամաբանական նկարագրությունից // ISSN 0002-306X. ՀԳԱԱ և ՀԱՊՀ Տեղ. Տեխն. գիտ. սերիա. 2018. Հ. LXXI, N4. – էջ 475-484.
87. Melikyan V.Sh., Harutyunyan A.G., Martirosyan A.R. Ranking-based Placement of IC Logic Cells // IEEE 39th International Conference on Electronics and Nanotechnology (ELNANO). –2019. –P. 46-49.
88. Петросян О.А., Аветисян З.М., Мартиросян А.Р. Метод обнаружения метастабильности в синхронизаторах // Вестник НПУА: Информационные технологии, электроника, радиотехника. –2018. –№ 2. –С. 56-66.
89. Digital Standard Cell Library. - SAED_EDK32/28 - SYNOPSYS ARMENIA Educational Department. - Yerevan, 2011. -152 P.
90. Melikyan V.Sh., Harutyunyan A.G. 3D Integrated Circuits Multifactor Placement // Proceedings of IEEE East-West Design & Test Symposium (EWDTS'17). - Novi Sad, Serbia, 2017.-P.45-48.
91. Qt Documentation, - 2019
<https://doc.qt.io/qt-5/classes.html>

ՀԱՎԵԼՎԱԾ 1
ՆԵՐԴՐՄԱՆ ԱԿՏ

SYNOPSIS®

№ 434/19

"21" 05 2019



Հաստատում եմ՝

«ՄԻՆՈՓՄԻՍ ԱՐՄԵՆԻԱ» ՓԲԸ

Գլխավոր տնօրեն՝

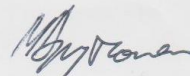
Հ. Մուսայելյան

Արման Ռաֆիկի Մարտիրոսյանի «Ինտեգրալ սխեմաների ավտոմատացված ֆիզիկական նախագծումը սխեմատեխնիկական առանձնահատկությունների հաշվառմամբ» թեմայով թեկնածուական ատենախոսության արդյունքների

ՆԵՐԴՐՄԱՆ ԱԿՏ

ՀԱՊՀ «Ավտոմատացման համակարգեր» մասնագիտությամբ 3-րդ կուրսի ասպիրանտ Ա.Ռ. Մարտիրոսյանի կողմից մշակված «Soft and Hard block maker» ծրագրային միջոցը ներդրված է «ՄԻՆՈՓՄԻՍ ԱՐՄԵՆԻԱ» ՓԲԸ-ում: Այն նախատեսված է ինտեգրալ սխեմաների ֆիզիկական «փափուկ» և «կոշտ» բլոկների ստացման և արագագործության բարձրացման համար: Մշակված գործիքի կիրառումը զգալիորեն կրճատում է միջմիացումներով պայմանավորված սխեմայի հապաղումները և բարձրացնում է նախագծվող սխեմաների արագագործությունը: Մշակված ծրագրային միջոցի կարևորությունը ինտեգրալ սխեմաների ֆիզիկական նախագծման գործընթացում և համապատասխանում է ժամանակակից էլեկտրոնային նախագծման բնագավառում կիրառվող ծրագրային գործիքային միջոցին ներկայացվող բոլոր պահանջներին:

Լուծումների խմբի հետազոտման և
մշակման բաժնի կառավարիչ՝

 Մ. Հարությունյան

"ՄԻՆՈՓՄԻՍ ԱՐՄԵՆԻԱ" ՓԲԸ
0026, ԳՂ, ԵՐԵՎԱՆ, ԱՐՇԱԿՈՒՆՅԱՆ 41
Հեռ.՝ (+374 10) 49 21 00, ֆաքս՝ (+374 10) 49 26 96
ԳՎԳԴ 02236362

"SYNOPSIS ARMENIA" CJSC
41 ARSHAKUNYATS AVE., YEREVAN, ARMENIA, 0026
TEL.: (+374 10) 49 21 00, FAX: (+374 10) 49 26 96
TAX PAYER'S ID 02236362

ՀԱՎԵԼՎԱԾ 2

«SHBM» ծրագրային որոշ հասկանքներ:

// Graph.hpp

```
#pragma once
#include <vector>
#include <set>
#include <map>
using namespace std;

class Graph
{
private:
    typedef std::set<int> int_set;
    typedef int_set::iterator int_set_iterator;
    typedef std::map<std::string, int> string_int_map;
    typedef string_int_map::iterator string_int_map_iterator;
private:
    enum gate_is_from
    {
        module = 0,
        input = 1,
        output = 2,
        wire = 3
    };

    inline const char* gate_to_string(gate_is_from g)
    {
        switch (g)
        {
            case module: return "module";
            case input: return "input";
            case output: return "output";
            case wire: return "wire";
            default: return "module";
        }
    }

    class Wire
    {
    public:
        Wire(const std::string& name, int id, const int rank, bool
is_only_input = true) :
                                                                    w_name(name),
                                                                    w_id(id),
                                                                    w_rank(rank),

w_keep_max_rank(rank),

w_connected_cells_which_got_rank(0)
                                                                    {}

    public:
        std::string w_name;
        int w_id;
        int w_rank;
        int w_keep_max_rank;
```

```

        int w_connected_cells_which_got_rank;
        std::set<int> w_front_connected_with;
        std::set<int> w_end_connected_with;
    };

    class Cell
    {
    public:
        string n_type;
        string n_name;
        int n_rank;
        set<int> n_in_edges_id;
        int n_out_edge_id;
        bool visited = false;
    };

public:
    Graph();
    ~Graph();
    void read_verilog_file(const std::string& file_name);
    void write_to_output(const std::string& file_name);
    void write_ranks_to_output(const std::string& file_name);
    void write_types_to_output(const std::string & file_name);
    void write_wire_to_output(const std::string& file_name);
    void draw_graph();
    void ranking();
    void test();

private:
    int get_max_wires_count(const std::string& file_name);
    bool make_cells_from_line(const std::string& line, std::string&
type, Cell& c);
    void make_gate_helper(const std::string& file_name, gate_is_from&
gate_is, gate_is_from& next_gate_is);
    void make_input_set(const std::string& file_name);
    void make_output_set(const std::string& file_name);
    void make_wires(const std::string& file_name);
    //void make_wires(int count);
    void read_cells(const std::string& file_name);

private:
    void step_1_make_input_wires_ranks_1(int_set& cells_with_no_rank);
    void step_2_make_cells_rank_if_possible(int_set&
cells_with_no_rank);
    void step_3_get_circle_of_0_rank_cells(int_set& cells_with_no_rank,
std::vector<int>& c_circle);

private:
    void draw_cell(const Cell& c);

private:
    vector<Cell> g_cells;
    vector<Wire> g_wires;
    std::set<int> g_input;
    std::set<int> g_output;
    std::map<std::string, int> g_gate_name_to_id;

```

```
};
```

```
// customrect.hpp
```

```
#include "stdafx.h"
#include "Graph.h"
#include <ctime>
#include <iostream>

int main()
{
    clock_t start = clock();
    Graph G;
    G.read_verilog_file("a1.v");
    G.ranking();
    clock_t end = clock();
    std::cout << "Total duration file_read: " << (float(end - start)) /
CLOCKS_PER_SEC << " seconds" << std::endl;
    G.draw_graph();
    G.test();
    return 0;
}

#ifndef CUSTOMRECT_H
#define CUSTOMRECT_H

#include <QGraphicsRectItem>
#include <QBrush>

class CustomRect : public QGraphicsRectItem
{
public:
    CustomRect(const QRectF &rect);
    void addOutG(QGraphicsLineItem *out_line, QGraphicsPolygonItem
*out_triangle);
    void addInG(QGraphicsLineItem *in_line, int id, QGraphicsPolygonItem
*in_triangle);
    void addText(QGraphicsTextItem *text);
    void addLine(uint wire_id, QGraphicsLineItem *line, bool isPoint1);
    QVariant itemChange(GraphicsItemChange change, const QVariant
&value);
    void moveLineToCenter(QPointF newPos);
private:
    typedef std::map<int, QGraphicsLineItem*> int_line;
    typedef int_line::iterator int_line_iterator;
    typedef std::vector<std::pair<uint, QGraphicsLineItem*> >
uint_line_vec;
private:
    uint_line_vec line_vec;
    std::vector<bool> isP1_vec;
    QGraphicsTextItem* text;

    std::map<int, QGraphicsLineItem*> in_g;
    QGraphicsLineItem* out_g;
```

```

        std::map<int, QGraphicsPolygonItem*> in_g_triangle;
        QGraphicsPolygonItem* out_g_triangle;
};
#endif // CUSTOMRECT_H

// customrect.cpp

#include "customrect.h"

CustomRect::CustomRect (const QRectF& rect) : QGraphicsRectItem(rect)
{
    setFlag(QGraphicsItem::ItemIsMovable);
    setFlag(QGraphicsItem::ItemSendsScenePositionChanges);
    text = nullptr;
    out_g = nullptr;
    out_g_triangle = nullptr;
}

void CustomRect::addOutG(QGraphicsLineItem* out_line,
    QGraphicsPolygonItem* out_triangle)
{
    out_g = out_line;
    QBrush blackBrush(Qt::black);
    out_g_triangle = out_triangle;
    out_g_triangle->setBrush(blackBrush);
}

void CustomRect::addInG(QGraphicsLineItem* in_line, int id,
    QGraphicsPolygonItem* in_triangle)
{
    in_g[id] = in_line;
    QBrush blackBrush(Qt::black);
    in_triangle->setBrush(blackBrush);
    in_g_triangle[id] = in_triangle;
}
/* void addLine(QGraphicsLineItem *line, bool isPoint1, uint index) {

    if(index < line_vec.size()){
        line_vec[index] = line;
        isPl_vec[index] = isPoint1;
    }
}
*/
void CustomRect::addText(QGraphicsTextItem* text)
{
    this->text = text;
}

void CustomRect::addLine(uint wire_id, QGraphicsLineItem *line, bool
isPoint1) {
    line_vec.push_back(std::make_pair(wire_id,line));
    isPl_vec.push_back(isPoint1);
}
QVariant CustomRect::itemChange(GraphicsItemChange change, const QVariant
&value)

```

```

{
    if (change == ItemPositionChange && scene()) {
        // value is the new position.
        QPointF newPos = value.toPointF();

        moveLineToCenter(newPos);
        if(text){
            text->setPos(newPos+QPointF(40,40));
        }
    }
    return QGraphicsItem::itemChange(change, value);
}

void CustomRect::moveLineToCenter(QPointF newPos) {
    // Converts the ellipse position (top-left)
    // to its center position
    int xOffset = static_cast<int>(rect().x());
    int yOffset = static_cast<int>(rect().y() + rect().height()/4);

    // QPointF newCenterPos = QPointF(newPos.x() + xOffset, newPos.y() +
    yOffset);

    if(out_g)
    {
        int outG_xoffset = static_cast<int>(rect().x() + rect().width());
        int outG_yoffset = static_cast<int>(rect().y() +
rect().height()/2);
        QPointF newOutGPos = QPointF(newPos.x() + outG_xoffset,
newPos.y() + outG_yoffset);
        QPointF out_p1 = newOutGPos;
        QPointF out_p2 = QPointF(out_p1.x()+50, out_p1.y());
        out_g->setLine(QLineF(out_p1, out_p2));
        if (out_g_triangle) {
            QPolygonF Triangle;
            Triangle.append(QPointF(out_p2.x()-10, out_p2.y()+5));
            Triangle.append(QPointF(out_p2.x()-10, out_p2.y()-5));
            Triangle.append(QPointF(out_p2.x(), out_p2.y()));
            Triangle.append(QPointF(out_p2.x()-10, out_p2.y()+5));
            out_g_triangle->setPolygon(Triangle);
            QBrush blackBrush(Qt::black);
            out_g_triangle->setBrush(blackBrush);
        }
    }

    if(!in_g.empty())
    {
        uint i = 1;
        for(int_line_iterator it = in_g.begin(); it != in_g.end(); ++it)
        {
            int inG_xoffset = static_cast<int>(rect().x());
            int inG_yoffset = static_cast<int>(rect().y() +
i*((rect().height()/(in_g.size()+1))));
            QPointF newInGPos = QPointF(newPos.x() + inG_xoffset,
newPos.y() + inG_yoffset);

```

```

        QPointF in_p1 = newInGPos;
        QPointF in_p2 = QPointF(in_p1.x()-50, in_p1.y());
        it->second->setLine(QLineF(in_p1,in_p2));

        QPolygonF Triangle;
        Triangle.append(QPointF((in_p2.x()+in_p1.x())/2,
in_p2.y()+5));
        Triangle.append(QPointF((in_p2.x()+in_p1.x())/2,
in_p2.y()-5));
        Triangle.append(QPointF((in_p2.x()+in_p1.x())/2+10,
in_p2.y()));
        Triangle.append(QPointF((in_p2.x()+in_p1.x())/2,
in_p2.y()+5));
        in_g_triangle[it->first]->setPolygon(Triangle);
        QBrush blackBrush(Qt::black);
        in_g_triangle[it->first]->setBrush(blackBrush);
        ++i;
    }
}

// Move the required point of the line to the center of the ellipse
for(uint i=0; i < line_vec.size(); ++i){
    // QPointF p1 = isP1_vec[i] ? newCenterPos : line_vec[i]-
>line().p1();
    // QPointF p2 = isP1_vec[i] ? line_vec[i]->line().p2() :
newCenterPos;
    QPointF p1 = isP1_vec[i] ? out_g->line().p2() :
line_vec[i].second->line().p1();
    QPointF p2 = isP1_vec[i] ? line_vec[i].second->line().p2() :
in_g[line_vec[i].first]->line().p2();

    line_vec[i].second->setLine(QLineF(p1, p2));
}
}

```

// zooming.hpp

```

#ifndef ZOOMING_H
#define ZOOMING_H

#include <QObject>
#include <QGraphicsView>

class Graphics_view_zoom : public QObject {
    Q_OBJECT
public:
    Graphics_view_zoom(QGraphicsView* view);
    void gentle_zoom(double factor);
    void set_modifiers(Qt::KeyboardModifiers modifiers);
    void set_zoom_factor_base(double value);

private:
    QGraphicsView* _view;
    Qt::KeyboardModifiers _modifiers;
    double _zoom_factor_base;
}

```

```

        QPointF target_scene_pos, target_viewport_pos;
        bool eventFilter(QObject* object, QEvent* event);

signals:
    void zoomed();
};
#endif // ZOOMING_H

// main.cpp

#include <QApplication>
#include <iostream>
#include <ctime>
#include "graph.h"
#include "customrect.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    Graph w;
    w.show();

    return a.exec();
}

#ifndef GRAPH_H
#define GRAPH_H

#include <QDialog>
#include <QtCore>
#include <QtGui>
#include <QGraphicsScene>
#include <vector>
#include <set>
class CustomRect;

namespace Ui {
class Graph;
}

class Graph : public QDialog
{
    Q_OBJECT
private:
    enum Cell_Type{
        t_not = 0,
        t_and = 1,
        t_nand = 2,
        t_or = 3,
        t_xor = 4,
        t_nor = 5,
        t_xnor = 6,
        t_dff = 7
    };
public:

```

```

explicit Graph(QWidget *parent = 0);
~Graph();

public:
    void BFS(int s, int y);
    void read_file(const std::string& filename);
    void read_rank_file(const std::string& filename);
    void read_type_file(const std::string& filename);
    void read_wire_file(std::string& filename);
    void drawArrows(std::vector<CustomRect *> &rect_vec);
    void drawCells(std::vector<CustomRect *> &rect_vec);
    void moveCellsToCurrentPlace(std::vector<CustomRect *> &rect_vec);
    void read_VS_generated_files();
    void fill_width_map_datas();
    void get_cells_layout_positions(std::vector<CustomRect *> &rect_vec,
std::vector< std::vector<int> >& layout_of_cells);
private:
    Ui::Graph *ui;
public:
    QGraphicsScene *scene;

public:
    struct Wire{
        std::string w_name;
        std::set<int> w_front_connected_with;
        std::set<int> w_end_connected_with;
    };

    struct Cell
    {
    public:
        std::string n_type;
        std::string n_name;
        int n_rank;
        std::set<int> n_in_edges_id;
        int n_out_edge_id;
        int n_y = 0;
        int n_number_cells_connected_with_out_edge = 0;
        double n_width = 0;
    };

public slots:
    void onBrowse1();
    void onBrowse2();
    void callGentleZoomPlus();
    void callGentleZoomMinus();
    void onRankingButtonClicked();
    void onSoftBlockButtonClicked();
    void onHardBlockButtonClicked();
    void onReset();
    void onSendInfo();
    void onMoveCell();

private:
    typedef std::set<int> int_set;
    typedef std::set<int>::iterator int_set_iterator;

```

```

        typedef std::map<std::string, std::map<int, double> >
string_p_int_double;
        typedef string_p_int_double::iterator string_p_int_double_iter;
        typedef std::map<int, double>::iterator int_double_iter;
        int getCellsMaxRank();
        void provideRankedCellsY();
        void provideRankedCellsMatrix(std::vector< std::vector<int> >&
matrix);
        void fill_inv_width();
        void fill_and_width();
        void fill_nand_width();
        void fill_or_width();
        void fill_nor_width();
        void fill_xor_width();
        void fill_xnor_width();
        void fill_dff_width();
        void count_cells_out_edge_power();
        void make_cells_width();
        void make_soft_block(std::vector<CustomRect*>& rect_vec,
std::vector<std::vector<int> > &layout_of_cells);
public:
        std::vector<Cell> g_cells;
        std::vector<Wire> g_wires;
        std::vector<bool> visited;
        std::vector<std::set<int>> g_x_y;
        int g_num_of_visiteds = 0;
        string_p_int_double g_cells_width;
        void make_hard_block(std::vector<CustomRect*>& rect_vec,
std::vector<std::vector<int> > &layout_of_cells, const uint
number_of_h_elements);
};

#endif // GRAPH_H

#include "graph.h"
#include "ui_graph.h"
#include "zooming.h"
#include <QGraphicsItem>
#include <fstream>
#include <iostream>
#include "customrect.h"
#include <QGraphicsProxyWidget>

Graph::Graph(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Graph)
{
    ui->setupUi(this);

    scene = new QGraphicsScene(this);
    Graphics_view_zoom* z = new Graphics_view_zoom(ui->graphicsView);
    z->set_modifiers(Qt::NoModifier);
    ui->graphicsView->setScene(scene);
/*
    QBrush redBrush(Qt::red);

```

```

    QBrush blueBrush(Qt::blue);
    QPen blackPen(Qt::black);

    blackPen.setWidth(2);

    QGraphicsRectItem *rect = scene->addRect(-100,-
100,100,100,blackPen,blueBrush);
    rect->setFlag(QGraphicsItem::ItemIsMovable, true);*/
}
Graph::~~Graph()
{
    delete ui;
}

void Graph::make_hard_block(std::vector<CustomRect*>& rect_vec,
std::vector< std::vector<int> >& layout_of_cells, const uint
number_of_h_elements)
{
    uint new_cell_row = 0;
    bool one_time_flag = true;
    std::vector<double> X_pos;
    double start_X_pos;
    int start_Y_pos;
    bool can_push = true;

    for(uint i = 1; i < layout_of_cells.size(); ++i){
        for(uint j = 0; j < layout_of_cells[i].size() - 1; ++j){
            if(new_cell_row == number_of_h_elements){
                new_cell_row = 0;
                can_push = false;
            }

            int cell_id = layout_of_cells[i][j];
            if(one_time_flag){
                one_time_flag = false;
                X_pos.push_back(rect_vec[cell_id]->x() +
g_cells[cell_id].n_width*75 + 1);
                start_X_pos = rect_vec[cell_id]->x();
                start_Y_pos = g_cells[cell_id].n_y;
                ++new_cell_row;
                continue;
            }

            // if(i == 1)

            double x = 0;
            int y = 0;
            if(new_cell_row < X_pos.size()){
                if(new_cell_row + 1 < X_pos.size() &&
X_pos[new_cell_row+1] < X_pos[new_cell_row]){
                    ++new_cell_row;
                }
                x = X_pos[new_cell_row];
                X_pos[new_cell_row] = x + g_cells[cell_id].n_width*75
+ 1;

                y = start_Y_pos + new_cell_row*91;

```

```

        rect_vec[cell_id]->setX(static_cast<qreal>(x));
        rect_vec[cell_id]->setY(static_cast<qreal>(y));
    }
    else{
        x = start_X_pos;
        if(can_push)
            X_pos.push_back(x + g_cells[cell_id].n_width*75 +
1);
            else
                X_pos[new_cell_row] = x +
g_cells[cell_id].n_width*75 + 1;
                y = start_Y_pos + new_cell_row*91;

    }
    rect_vec[cell_id]->setX(static_cast<qreal>(x));
    rect_vec[cell_id]->setY(static_cast<qreal>(y));
    ++new_cell_row;
    //rect_vec[cell_id]->boundingRect().getCoords(x1, y1, x2,
y2);
    // std::cout << g_cells[cell_id].n_y << std::endl;
    //std::cout << rect_vec[cell_id]->x() << " " <<
rect_vec[cell_id]->y() << std::endl;

    }
}
}

void Graph::get_cells_layout_positions(std::vector<CustomRect*>&
rect_vec, std::vector< std::vector<int> >& layout_of_cells)
{
    if(rect_vec.size() != g_cells.size()){
        std::cout << "ASSSEERT rect_vec != g_cells size" << std::endl;
    }
    std::vector< std::map<int, int> > rank_vec_y_to_id_map;
    typedef std::map<int, int>::iterator map_int_int_iterator;
    for(uint i = 0; i < rect_vec.size(); ++i){
        if(rank_vec_y_to_id_map.size() < g_cells[i].n_rank + 1)
            rank_vec_y_to_id_map.resize(g_cells[i].n_rank + 1);

        rank_vec_y_to_id_map[g_cells[i].n_rank].insert(std::make_pair(g_cells[i].
n_y, i));
    }

    layout_of_cells.resize(rank_vec_y_to_id_map.size());
    for(uint i = 0; i < rank_vec_y_to_id_map.size(); ++i){
        map_int_int_iterator map_it = rank_vec_y_to_id_map[i].begin();
        layout_of_cells[i].reserve(rank_vec_y_to_id_map[i].size());
        int max_width = 0;
        for( ; map_it != rank_vec_y_to_id_map[i].end(); ++map_it){
            layout_of_cells[i].push_back(map_it->second);
            if(rect_vec[map_it->second]->boundingRect().width() >
max_width)
                max_width = rect_vec[map_it->second]-
>boundingRect().width();
        }
        layout_of_cells[i].push_back(max_width);
    }
}

```

```

    }
    /*      uint j = 0;
    bool not_finished = true;
    while(not_finished){
        not_finished = false;
        for(uint i = 1; i < layout_of_cells.size(); ++i){
            if(j < layout_of_cells[i].size()){
                std::cout << layout_of_cells[i][j] << '\t';
                not_finished = true;
            }
        }
        ++j;
        std::cout << std::endl;
    }*/
}

void Graph::make_soft_block(std::vector<CustomRect*>& rect_vec,
std::vector< std::vector<int> >& layout_of_cells)
{
    uint j = 0;
    bool not_finished = true;
    while(not_finished){
        not_finished = false;
        for(uint i = 1; i < layout_of_cells.size(); ++i){
            if(j < layout_of_cells[i].size()-1){
                std::cout << layout_of_cells[i][j] << '\t';
                not_finished = true;
            }
        }
        ++j;
        std::cout << std::endl;
    }
}

void Graph::fill_width_map_datas()
{
    fill_inv_width();
    fill_and_width();
    fill_nand_width();
    fill_or_width();
    fill_nor_width();
    fill_xor_width();
    fill_xnor_width();
    fill_dff_width();
    count_cells_out_edge_power();
    make_cells_width();
}

void Graph::count_cells_out_edge_power()
{
    for(int i = 0; i < g_wires.size(); ++i)
    {
        int_set_iterator c_it =
g_wires[i].w_front_connected_with.begin();
        if(c_it != g_wires[i].w_front_connected_with.end())

```

```

        g_cells[*c_it].n_number_cells_connected_with_out_edge =
g_wires[i].w_end_connected_with.size();
    }
}

void Graph::make_cells_width()
{
    for(int i = 0; i < g_cells.size(); ++i)
    {
        if(g_cells[i].n_number_cells_connected_with_out_edge == 0)
            ++g_cells[i].n_number_cells_connected_with_out_edge;
        int power_id = 10*g_cells[i].n_in_edges_id.size() +
g_cells[i].n_number_cells_connected_with_out_edge;
        string_p_int_double_iter it =
g_cells_width.find(g_cells[i].n_type);
        if(it != g_cells_width.end()){
            int_double_iter it1 = it->second.find(power_id);
            if(it1 == it->second.end()){
                it1 = it->second.find(--power_id);
                assert(it1 != it->second.end());
                ++it1;
            }
            g_cells[i].n_width = it1->second;
        }
    }
}

void Graph::moveCellsToCurrectPlace(std::vector<CustomRect*>& rect_vec)
{
    provideRankedCellsY();
    int n = 140;
    std::vector<int> counting_rank_visited_elements;
    counting_rank_visited_elements.resize(getCellsMaxRank()+1);
    for(int i =0; i < rect_vec.size(); ++i)
    {
        n = g_cells[i].n_rank*157;
        rect_vec[i]->moveBy(n, g_cells[i].n_y*2 +
counting_rank_visited_elements[g_cells[i].n_rank]*100);
    }
}

void Graph::provideRankedCellsY()
{
    std::vector< std::vector<int> > matrix;
    provideRankedCellsMatrix(matrix);
    g_x_y.resize(getCellsMaxRank()+1);
    // Mark all the vertices as not visited
    if(visited.size() == 0){
        for(int i = 0; i < g_cells.size(); ++i)
            visited.push_back(false);
    }
    int i1 = matrix.size()-1;
    int j1 = -1;
    int y = 0;

```

```

        while(i1 > 0 && (g_num_of_visiteds == 0 || g_num_of_visiteds !=
visited.size())) {
            if(i1 < matrix.size() && matrix[i1].size() != 0)
                ++j1;
            if(j1 < (matrix[i1].size())){
                BFS(matrix[i1][j1], y);
            }
            else{
                --i1;
                j1 = -1;
            }
        }
    }
}

void Graph::provideRankedCellsMatrix(std::vector< std::vector<int> >&
matrix)
{
    int max_rank = getCellsMaxRank();
    matrix.resize(max_rank+1);
    for(int i = 0; i < g_cells.size(); ++i)
    {
        matrix[g_cells[i].n_rank].push_back(i);
    }
}

int Graph::getCellsMaxRank()
{
    int max_rank = g_cells[0].n_rank;
    for(int i =1; i < g_cells.size(); ++i)
    {
        if(g_cells[i].n_rank > max_rank)
        {
            max_rank = g_cells[i].n_rank;
        }
    }
    return max_rank;
}

void Graph::drawCells(std::vector<CustomRect*>& rect_vec)
{
    rect_vec.reserve(g_cells.size());
    int h = 75;
    int w = 75;
    for(uint i = 0; i < g_cells.size(); ++i ){
        rect_vec.push_back(new CustomRect(QRectF(h,h,g_cells[i].n_width*w,
1.2*h)));
    }
    for(uint i = 0; i < rect_vec.size(); ++i){
        scene->addItem(rect_vec[i]);
    }
}

void Graph::drawArrows(std::vector<CustomRect*>& rect_vec)
{
    QBrush blackBrush(Qt::black);

```

```

QBrush greenBrush(Qt::green);
QBrush dgreenBrush (Qt::darkGreen);
for(uint i = 0; i < g_cells.size(); ++i )
{
    QGraphicsLineItem *outLine = scene->addLine(QLineF(40, 40, 80,
80));
    QPolygonF Triangle;
    Triangle.append(QPointF(70, 85));
    Triangle.append(QPointF(70, 75));
    Triangle.append(QPointF(80, 80));
    Triangle.append(QPointF(70, 85));
    QGraphicsPolygonItem* pTriangleItem = scene-
>addPolygon(Triangle);
    rect_vec[i]->addOutG(outLine, pTriangleItem);
    for(int_set_iterator it = g_cells[i].n_in_edges_id.begin(); it !=
g_cells[i].n_in_edges_id.end(); ++it){
        QGraphicsLineItem *inLine = scene->addLine(QLineF(40, 40, 80,
80));
        QPolygonF T1;
        T1.append(QPointF(10, 10));
        T1.append(QPointF(10, 15));
        T1.append(QPointF(15, 10));
        T1.append(QPointF(10, 10));
        QGraphicsPolygonItem* pti = scene->addPolygon(T1);
        rect_vec[i]->addInG(inLine, *it, pti);
        //std::cout << "Gcell_wire_id = " << *it << std::endl;
    }
    rect_vec[i]->setBrush(dgreenBrush);
    //FOR SELECTION OF NAND3_2
    if(g_cells[i].n_name == "OR2_0"){
        rect_vec[i]->setBrush(greenBrush);
        for(int_set_iterator it = g_cells[i].n_in_edges_id.begin();
it != g_cells[i].n_in_edges_id.end(); ++it)
            std::cout << "IN EDGES = " << g_wires[*it].w_name <<
std::endl;
        std::cout << "RANK = " << g_cells[i].n_rank << std::endl;
        std::cout << "OUT WIRE = " <<
g_wires[g_cells[i].n_out_edge_id].w_name << std::endl;
        std::cout << "WIDTH = " << g_cells[i].n_width << std::endl;
        std::cout << "X = " << rect_vec[i]->rect().x() << "Y = " <<
rect_vec[i]->rect().y() << std::endl;
    }
    //QGraphicsTextItem* text = w.scene-
>addText(w.g_cells[i].n_name.c_str());
    QLineEdit* pLineEdit = new QLineEdit(g_cells[i].n_name.c_str());
    pLineEdit->setFixedWidth(60);
    QGraphicsProxyWidget* pMyProxy = new
QGraphicsProxyWidget(rect_vec[i]);
    pMyProxy->setWidget(pLineEdit);
    pMyProxy->setPos(rect_vec[i]->rect().x()+10, rect_vec[i]-
>rect().y()+10);
    //rect_vec[i]->addText(text);
}
for(uint i = 0; i < g_wires.size(); ++i){

```

```

        for(int_set_iterator it =
g_wires[i].w_front_connected_with.begin(); it !=
g_wires[i].w_front_connected_with.end(); ++it){
            for(int_set_iterator it_end =
g_wires[i].w_end_connected_with.begin(); it_end !=
g_wires[i].w_end_connected_with.end(); ++it_end){
                QGraphicsLineItem *line = scene->addLine(QLineF(40, 40,
80, 80));
                rect_vec[*it]->addLine(i, line, true);
                rect_vec[*it_end]->addLine(i, line, false);
                // std::cout << "front = " << *it << " end = " << *it_end
<< std::endl;
            }
        }
    }
}

void Graph::read_file(const std::string& filename)
{
    std::fstream fin(filename);
    //read nodes
    uint cells_cnt = 0;
    fin >> cells_cnt;
    for(uint i = 0; i < cells_cnt; ++i){
        Cell c;
        uint in_edges_cnt;
        fin >> c.n_name >> c.n_rank >> c.n_out_edge_id >>
in_edges_cnt;
        for(uint i = 0; i < in_edges_cnt; ++i){
            int id;
            fin >> id;
            c.n_in_edges_id.insert(id);
        }
        g_cells.push_back(c);
    }
    fin.close();
}

```

ՀԱՎԵԼՎԱԾ 3 ՆԿԱՐՆԵՐԻ ՑԱՆԿ

1.	Նկ. 1.1. ԻՍ-ի տրամաբանական փականներում և միջմիացումներում ազդանշանի հապաղումների էվոլյուցիան	15
2.	Նկ. 1.2. ԻՍ-երի ֆիզիկական նախագծման խնդիրների վաղաժամ դիտարկման երթուղին.....	20
3.	Նկ. 1.3. ԻՍ-երի ֆիզիկական նախագծման խնդիրների վաղաժամ դիտարկման երթուղին.....	22
4.	Նկ. 1.4. ա-տրամաբանական հիերարխիա, որտեղ A,B,c,d տարրերը ֆիզիկական նախագծման ժամանակ պետք է պահվեն միասին, բ-ֆիզիկական հիերարխիա, որտեղ A,B,c,d տարրերը ներկայացված են մեկ ֆիզիկական բլոկի տեսքով.....	29
5.	Նկ. 1.5. Տրամաբանական և ֆիզիկական հիերարխիաների կապը լավարկման հետ.....	30
6.	Նկ. 2.1. ԱՀԳ-ի պարզագույն օրինակ	477
7.	Նկ. 2.2. ԱՀԳ-ում կոդերի դասակարգումը՝ ա)մուտքային ,բ)միջանկյալ, գ)ելքային	477
8.	Նկ. 2.3. Ազդանշանի հոսքի գրաֆի հապաղման հանգույցով օրինակ	488
9.	Նկ. 2.4. ISCAS85 գրադարանի c17 verilog տրամաբանական նկարագրության ազդանշանների հոսքի գրաֆը	49
10.	Նկ. 2.5. Տրամաբանական սխեմայի օրինակ.....	511
11.	Նկ. 2. 6. Տրամաբանական մոդելավորման մեթոդների դասակարգումը	522
12.	Նկ. 2.7. Տրամաբանական սխեմայի ռանգավորված օրինակ	544

13.	Նկ. 2. 8 «Փափուկ» մակրոսներ (ա) մինչև պարամետրացումը և «կոշտ» մակրոսներ (բ) պարամետրացումից հետո.....	555
14.	Նկ. 2. 9. «Փափուկ» և «կոշտ»IP -ների համեմատության գրաֆիկը.....	57
15.	Նկ. 2. 10. «Փափուկ» բլոկից «կոշտ» բլոկն անցման «ձևի» կորը (ա) և կողերի տարբեր հարաբերակցություններով ներկայացվող «կոշտ» բլոկների տարբերակները (բ)	58
16.	Նկ. 3. 1. Մխեմատեխնիկական առանձնահատկությունների հաշվառմամբ ԻՍ-ի ֆիզիկական նախագծման քայլերի հաջորդականությունը	677
17.	Նկ. 3. 2. Մխեմայի փականային մակարդակի Iscas85 գրադարանի թեստային c17.v Verilog նկարագիրը.....	69
18.	Նկ. 3. 3. Մուտքային շղթաների id-երը բինար որոնման բալանսավորված ծառում.....	744
19.	Նկ. 3. 4. Ելքային շղթաների id-երը բինար որոնման բալանսավորված ծառում	755
20.	Նկ. 3.5. ԱՀԳ-ի շղթաների բազմության անունից դեպի իրենց id կառուցված map	766
21.	Նկ. 3. 6. Ռանգավորման ալգորիթմի հիմնական քայլերը.....	79
22.	Նկ. 3. 7. «Փափուկ» բլոկի ավտոմատացված սինթեզի ալգորիթմի կեղծ կողը .	811
23.	Նկ. 3. 8. Ֆիզիկական «փափուկ» բլոկից «կոշտ» բլոկի ստացման ալգորիթմի հիմնական քայլերը.....	822

24.	Նկ. 3. 9. Հորիզոնական մակարդակների ուղղահայաց կոմպլեկտավորումը, որտեղ ձախ ռանգի տարրերը ա) գերազանցում են, բ) քիչ են աջ ռանգի տարրերից	833
25.	Նկ. 3. 10. Հորիզոնական մակարդակներում ուղղահայաց ուղղությամբ տարրերի կոմպլեկտավորումը՝ ա) նախնական վիճակ, բ) մոդիֆիկացված բարիկենտրոնային ալգորիթմի աշխատանքից հետո, գ) արդյունքների լեգալացում.....	855
26.	Նկ. 3. 11. Ըստ հզորությունների ֆիզիկական տեղաբաշխման ալգորիթմի հիմնական քայլերը.....	88
27.	Նկ. 3. 12. ISCAS 85 շարքի C16 փորձական տրամաբանական սխեմա.....	889
28.	Նկ. 3.13. ISCAS- 85 շարքի C16 փորձական տրամաբանական սխեմայի գծային տեղաբաշխման հաշվարկի և հզորությունների տարբերության մատրից	900
29.	Նկ. 3.14. Փորձարկվող սխեմաների տարրերի գծային տեղաբաշխումը և դրանց դինամիկ հզորությունները	90
30.	Նկ. 4.1. Մշակված ծրագրային միջոցի աշխատանքը բնութագրող քայլերի հաջորդականությունը և կիրառված ֆունկցիաները	944
31.	Նկ. 4.2. SHBM ծրագրային գործիքի սկզբնական պատուհանի տեսքը.....	966
32.	Նկ. 4.3. a28 փականային մակարդակի Verilog նկարագիր	97
33.	Նկ. 4.4. a28 թեստային Verilog ֆայլի՝ ազդանշանների հոսքի գրաֆի՝ գրաֆիկական տեսքը.....	97
34.	Նկ. 4.5. Սխեմայի՝ ա) խոշորացումը բ) մանրացումը.....	98

35.	Նկ. 4.6. a28 թեստային Verilog ֆայլի՝ ռանգավորված ԻՄ-ի գրաֆիկական տեսքը	98
36.	Նկ. 4.7. 2-րդ ռանգի տարրերի միջև հեռավորության օպտիմալացումը.....	99
37.	Նկ. 4.8. Ֆիզիկական «փափուկ» բլոկի գրաֆիկական տեսքը.....	99
38.	Նկ. 4.9. Ֆիզիկական «փափուկ» բլոկ կառուցելիս տարրերի միջև հեռավորության օպտիմալացման առանցքի ընտրության տարբերակները	1000
39.	Նկ. 4.10. Ֆիզիկական «կոշտ» բլոկի գրաֆիկական տեսքը՝ ա) 3, բ) 4 տարր ամեն ռանգում	1000
40.	Նկ. 4.11. Ֆիզիկական «կոշտ» բլոկում՝ ա) ընտրված OR2_0 տարրը, բ) ընտրված տարրի տվյալները	1011
41.	Նկ. 4.12. Փականային մակարդակի PG9 մշակված սխեմա	1011
42.	Նկ. 4.13. Փականային մակարդակի PG9 մշակված սխեմայի ավտոմատացված ռանգավորումը	1022
43.	Նկ. 4.14. Ջերմային տեղաբաշխման ծրագրի աշխատանքային պատուհանը	1044

ՀԱՎԵԼՎԱԾ 4

ԱՂՅՈՒՍԱԿՆԵՐԻ ՑԱՆԿ

1. Աղյուսակ 1.1 ԻՍ-երի ֆիզիկական նախագծմանն առնչվող հիմնական պարամետրերի զարգացումը 14
2. Աղյուսակ 3.1 ԱՀԳ-ի շղթաների «Wire» զանգվածի կառուցվածքը..... 722
3. Աղյուսակ 3.2 ԱՀԳ-ի տարրերի «Cell» զանգվածի կառուցվածքը..... 733
4. Աղյուսակ 3.3 ԱՀԳ-ի տարրերի զանգվածի կառուցվածքը ռանգավորումից հետո 788
5. Աղյուսակ 3.4 C16 փորձնական սխեմայի տարրերի պարամետրերի հաշվարկային արժեքները 889
6. Աղյուսակ 4.1 Թեստային սխեմաների տարրերի տեղաբաշխման շղթաների երկարությունների հաշվարկի արդյունքները 1022
7. Աղյուսակ 4.2 Ջերմային տեղաբաշխման ծրագրի աշխատանքային պատուհանի աշխատանքի նկարագրությունը 1055
8. Աղյուսակ 4.3 Ջերմային տեղաբաշխման որակը տարբեր թեստային սխեմաների համար..... 1066

ՀԱՎԵԼՎԱԾ 5
ՀԱՊԱՎՈՒՄՆԵՐԻ ՑԱՆԿ

1. ԻՍ – ինտեգրալ սխեմա
2. ՏՀԳ – տվյալների հոսքի գրաֆ
3. ԱՀԳ – ազդանշանների հոսքի գրաֆ
4. ՄՄ – մտավոր սեփականություն