ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ ԿՐԹՈՒԹՅԱՆ ԵՎ ԳԻՏՈՒԹՅԱՆ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ
ԵՎՐՈՊԱԿԱՆ ԿՐԹԱԿԱՆ ՏԱՐԱԾԱՇՐՋԱՆԱՅԻՆ ԱԿԱԴԵՄԻԱ

ՀՈՎՀԱՆՆԵՍ ԳԵՎՈՐԳԻ ԿԱՍԱՐՋՅԱՆ

ԱՆԱԼՈԳԱՅԻՆ ԻՆՏԵԳՐԱԼ ՍԽԵՄԱՆԵՐԻ ԱՎՏՈՄԱՏԱՑՎԱԾ ՖԻԶԻԿԱԿԱՆ ՆԱԽԱԳԾՄԱՆ ՄԻՋՈՑՆԵՐԻ ՄՇԱԿՈՒՄԸ ԵՎ ՀԵՏԱԶՈՏՈՒՄԸ

ԱՏԵՆԱԽՈՍՈՒԹՅՈՒՆ

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

Գիտական ղեկավար՝ ՀՀ ԳԱԱ թղթակից անդամ,
ՀՀ գիտության վաստակավոր գործիչ,
տ.գ.դ., պրոֆ. Վ. Շ. Մելիքյան

ԵՐԵՎԱՆ 2016

ԲՈՎԱՆԴԱԿՈՒԹՅՈՒՆ

ՆԵՐԱԾՈՒԹՅՈՒՆ

<u>Թեմայի արդիականությունը:</u> Կանխատեսվում է, որ անալոգային ինտեգրալ սխեմաների (ԻՍ) նախագծման կարիքը շատ և շատ անգամ աճելու է հաջորդ տարիների ընթացքում: Եթե համեմատենք թվային ԻՍ-երի նախագծման հետ, որը հնարավոր է իրագործել նվազագույն ջանքեր գործադրելով, օգտագործելով ավտոմատ գործիքներ, անալոգային նախագծումը շարունակում է մնալ բարդ և ժամանակատար աշխատանք: Առ այսօր անալոգային նախագծերի մեծ մասն իրականացվում է անալոգային նախագծողների կողմից ձեռքով՝ հիմնվելով իրենց գիտելիքների և փորձի վրա: Սկսնակ աշխատողի համար բարձր ճշգրտության անալոգային ԻՍ-ի նախագծումը բավականին դժվար գործ է [6, 90]: Արդյունավետության բարելավման և նախագծման ժամանակի նվազեցման համար անհրաժեշտ է անալոգային ԻՍ-երի ավտոմատացված նախագծման գործիքների մշակում: Ավտոմատացման բարդությունները հետևյալն են [92, 95].

- ի տարբերություն թվային ԻՍ-երի, անալոգային ԻՍ-երը գերզգայուն են աղմուկի նկատմամբ: Աղմուկով պայմանավորված լարման ցանկացած շեղում կարող է սխալ արդյունքների հանգեցնել [4, 41]: Տարբերակվում է երկու տիպի աղմուկի աղբյուր՝ պատահական աղմուկներ և միջավայրով պայմանավորված աղմուկներ: Պատահական աղմուկներն առաջանում են ռեզիստորների և շղթայի ակտիվ տարրերի աշխատանքի արդյունքում: Միջավայրի աղմուկներն առաջանում են, երբ մեծ թվով թվային բլոկներ շրջապատում են անալոգային բլոկը: Այդ թվային բլոկների փոխանջատումներն առաջացնում են մեծ դինամիկ հոսանքներ, որոնք ազդում են զգայուն անալոգային հանգույցների վրա,
- հաջորդ բարդությունը անալոգային և թվային բլոկների ինտեգրումն է մեկ հարթակի վրա: Հիմնական խնդիրն այն է, որ անալոգային հանգույցները գերզգայուն են աղմուկի նկատմամբ, իսկ թվային հանգույցները շատ աղմկոտ են: Այս խնդիրից խուսափելու համար նախագծողները պետք է մեկուսացնեն կամ անալոգային հանգույցները պաշտպանեն թվային հանգույցներից: Ավտոմատացված համակարգը

այս պարագայում կարող է մեծ դեր ունենալ նշված պոտենցիալ բարդությունները ոչ միայն բացահայտելու, այլ նաև արագ լուծում առաջարկելու գործում [78]:

Տեխնոլոգիական գործընթացի փոփոխումը և էլեկտրական ագրեսիվ պահանջները առաջ են բերում անալոգային և ռադիոհաճախական սխեմաների ֆիզիկական նախագծման նոր մոտեցումների անհրաժեշտություն [1]: Բարձր ճշգրտության ֆիզիկական նախագծերում մեծ դեր են խաղում ֆիզիկական նախագծման երևույթները, որոնցից են՝ տարրերի համապատասխանեցումը և համաչափությունը, մակաբույծ տարրերը, միջմիացումներում հոսանքի խտությունը և հարթակի երևույթները: Բարձր ճշգրտության համար անհրաժեշտ է այս երևույթները մոդելավորել ֆիզիկական նախագծման ավտոմատացված միջոցներում, ինչն իրականում շատ բարդ է: Այս բարդությունը հանգեցնում է նրան, որ անալոգային ԻՍ-երի ֆիզիկական նախագծումը շատ դեպքերում մասնագետների կողմից կատարվում է ձեռքով [67, 68]:

Ֆիզիկական նախագծումը կոմպլեմենտար մետաղ-օքսիդ-կիսահաղորդիչ (ԿՄՕԿ) անալոգային ԻՍ-երի նախագծման կարևորագույն փուլերից մեկն է: Բարձր որակի ֆիզիկական նախագիծը պետք է ցածր զգայունություն ունենա աղմուկների և գործընթացի փոփոխությունների նկատմամբ [24, 95]: Ֆիզիկական նախագծի պարամետրերի բարելավման համար անհրաժեշտ են նոր ծրագրային գործիքներ, որոնք արագորեն կգեներացնեն մի քանի տիպի ֆիզիկական նախագծեր և նմանակումների միջոցով կընտրեն դրանցից լավագույնը [69, 79, 96]:

Ատենախոսությունը նվիրված է անալոգային ինտեգրալ սխեմաների ավտոմատացված ֆիզիկական նախագծման միջոցների մշակմանը և հետազոտմանը՝ հաշվի առնելով վերը դիտարկված նախագծման բարդությունները: Մուտքային տվյալները և խնդրի դրվածքը տալու համար մշակված ծրագրային ապահովման գործիքը շատ քիչ ժամանակ է պահանջում: Գործիքը կարճ ժամանակում տալիս է ճշգրիտ և միանգամից կիրառելի արդյունքներ, այն բավականին ճկուն է և կարելի է օգտագործել ամեն տիպի ԻՍ–երի ֆիզիկական նախագծման ժամանակ:

<u>Հետազոտության առարկան</u> անալոգային ԻՍ-երի ֆիզիկական նախագծման երկրորդական երևույթների ազդեցությունն է ԻՍ-երի ելքային պարամետրերի վրա:

Ավտոմատացված նախագծման ընթացքում վերջիններիս առաջացրած խնդիրները և երկրորդական երևույթների նվազեցման եղանակները:

Աշխատանքի նպատակը: Անալոգային ԻՍ-երի ֆիզիկական նախագծման ժամանակ առաջացած նախագծման կանոնների ուսումնասիրումը և ներդրումը ավտոմատացված նախագծման գործիքի մեջ: Անալոգային ԻՍ-երի նախագծման երկրորդական երևույթների նվազեցման մեթոդների ուսումնասիրումը և ներդրումը ավտոմատ նախագծման գործիքի մեջ, որը կհանգեցնի նախագծման արագագործության աճին և նախագծի ճշգրտությանը:

Հետազոտության մեթոդները: Ատենախոսությունում օգտագործվել են էլեկտրական շղթաների և կիսահաղորդչային սարքերի տեսությունները, անալոգային ԻՍ-երի մոդելավորման, ֆիզիկական նախագծման եղանակները, անալոգային ԻՍ-երի և մասնավորապես դրանց ֆիզիկական նախագծերի ավտոմատացված նախագծման մեթոդները:

Գիտական նորույթը

- Առաջարկվել են անալոգային ԻՍ-երի ավտոմատացված ֆիզիկական նախագծման նոր սկզբունքներ, որոնք ապահովում են գործնական պահանջներին բավարարող էներգասպառում, կիսահաղորդչային բյուրեղի վրա զբաղեցրած փոքր մակերես, արդյունքների անհրաժեշտ ճշտություն և նախագծման կարճ ժամանակ:
- Մշակվել է ֆիզիկական նախագծման տարրերի համապատասխանեցման նոր մեթոդ: Այն փորձարկվել է դիֆերենցիալ ուժեղարարի ֆիզիկական նախագծում, և համապատասխանեցման գոյություն ունեցող այլ մեթոդների համեմատ, տալիս է 20…30% սխալանքի կրճատում նույն մակերեսի պայմաններում:
- Առաջարկվել է գրպանիկի մոտիկության երևույթի (ԳՄԵ) նվազեցման տոպոլոգիական իրականացում՝ փոխարինելով նախագծում առկա P տիպի մետաղ-օքսիդ-կիսահաղորդիչ (P-ՄՕԿ) տրանզիստորները N տիպի մետաղ-օքսիդ-կիսահաղորդիչ (N-ՄՕԿ) տրանզիստորներով: Նախագծվել է SAED32/28 *նմ* տեխնոլոգիական գործընթացի համար հոսանքի հայելու երկու ֆիզիկական նախագիծ, առաջինը P-ՄՕԿ տրանզիստորներով, երկրորդը՝ զուտ N-ՄՕԿ տրանզիստորներով:

Վերը նշված մեթոդի կիրառման շնորհիվ գրպանիկի մոտիկության երևույթի ազդեցությունը հոսանքի հայելում նվազել է 10...100 անգամ:

- Ստեղծվել է անալոգային ԻՍ-երի ավտոմատացված ֆիզիկական նախագծման գործիքների փաթեթ, որը կարելի է օգտագործել ցանկացած տիպի անալոգային ԻՍ-ի ավտոմատացված ծրագծման համար: Այն ապահովում է անհրաժեշտ ճշտություն՝ շնորհիվ նախագծողի կողմից սահմանվող սահմանափակումների: Ծրագրային միջոցի փորձարկումը փուլահաճախական ինքնահամալարման համակարգերում (ՓԻՀ) վկայում է դրա բարձր արդյունավետության մասին. ծրագծման ժամանակը նվազել է մոտ 35 անգամ:

<u>Պաշտպանության են ներկայացվում հետևյալ դրույթները.</u>

- Անալոգային ԻՍ-երի ֆիզիկական նախագծման տարրերի համապատասխանեցման նոր լուծում, որն ապահովում է ջերմային գրադիենտի նկատմամբ կայուն ֆիզիկական նախագիծ:
- ԳՄԵ-ի նվազեցման լուծում՝ օգտագործելով զուտ N-ՄՕԿ տրանզիստորներ:
- Անալոգային ԻՍ-երի ավտոմատացված նախագծման սահմանափակումների տրման և պահպանման Constraint Designer ծրագրային ապահովման գործիքը:
- Անալոգային ԻՍ-երի ավտոմատացված ֆիզիկական ծրագծման AAR Designer գործիքը:

<u>Աշխատանքի գործնական արժեքը:</u> Անալոգային ԻՍ-երի ֆիզիկական նախագծման երկրորդական երևույթների ազդեցության թուլացման մշակված դրույթները, մեթոդները և եղանակները կիրառվել են AAR Designer և Constraint Designer ծրագրային գործիքային փաթեթում, որն ունի օգտատերի «հարմարավետ» միջավայր, սկրիպտային կատարման տող, նմանակումներին հարմարեցված կառուցվածք, և միջինը 20...40 անգամ կրճատում է նախագծման գործընթացի տևողությունը՝ ձեռքով նախագծման համեմատ: Նախագծված AAR Designer և Constraint Designer ծրագրային փաթեթի կիրառումը փուլահաճախական ինքնահամալարման համակարգի ավտոմատացված ֆիզիկական նախագծման ժամանակ հաստատում է դրա բարձր արդյունավետությունը, քանի որ ավտոմատ

գործիքով ֆիզիկական նախագիծը ստացվել է 35 անգամ արագ՝ առանց ֆիզիկական նախագծման կանոնների խախտման և բավարարում է նախագծի բոլոր պահանջները:

Գիտական դրույթների հավաստիությունը հաստատվել է գործնական փորձարկումների արդյունքների անհրաժեշտ ճշգրտությամբ և ստացված գիտական արդյունքների մաթեմատիկական հիմնավորմամբ:

Ներդրումը: Անալոգային ԻՍ-երի ավտոմատացված ֆիզիկական նախագծման AAR Designer և Constraint Designer ծրագրային միջոցները ներդրված են «Սինոփսիս Արմենիա» ՓԲԸ-ում: AAR Designer ծրագրային միջոցով նախագծվել և փորձարկվել են մի շարք անալոգային ԻՍ-եր, այդ թվում՝ ՓԻՀ համակարգեր: AAR Designer և Constraint Designer ծրագրային միջոցների փաթեթն ընդգրկված է «Սինոփսիս Արմենիա» ընկերության ծրագրային միջոցների կազմում:

Հրապարակումներ: Ատենախոսության հիմնական դրույթները ներկայացված են 6 գիտական հրապարակումներում:

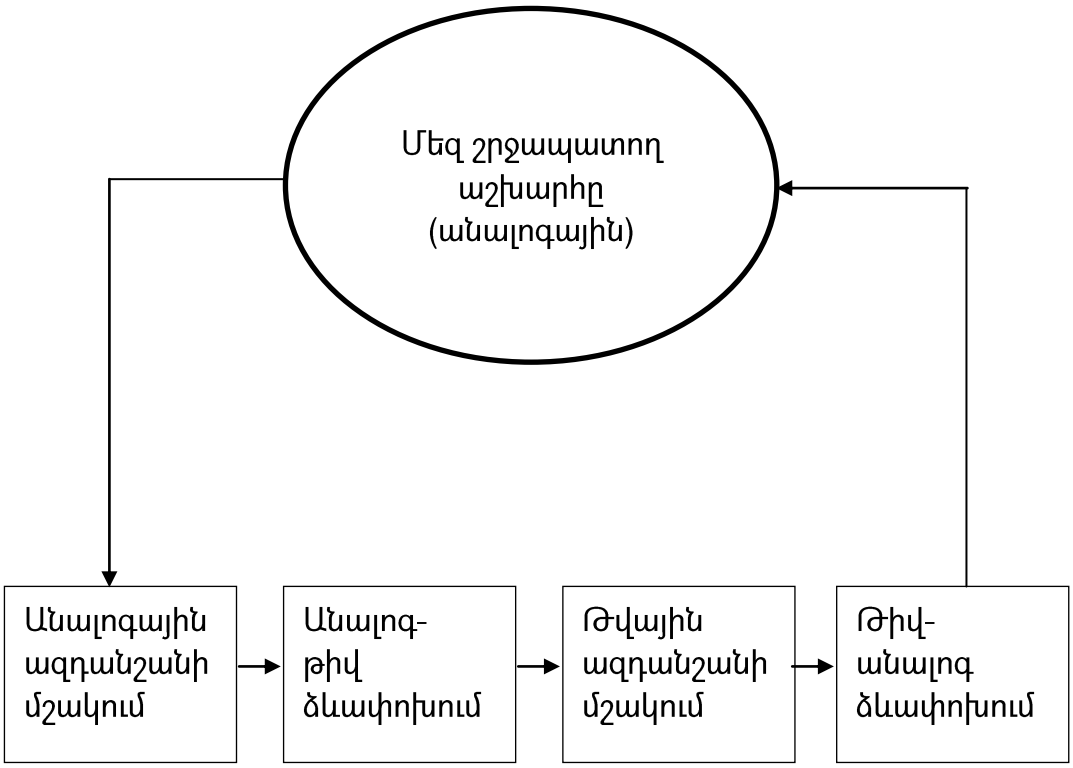
Ատենախոսության կառուցվածքը: Ատենախոսությունը բաղկացած է ներածությունից, 3 գլխից, եզրահանգումից, 112 անուն գրականության ցանկից և 4 հավելվածից (առաջին հավելվածում բերված է ատենախոսության ներդրման ակտը, երկրորդում՝ օգտագործված փաստաթղթերի նկարագրությունները և թեստավորման ֆայլերը, երրորդում՝ մշակված ծրագրային միջոցի տեքստից հատվածներ, չորրորդում՝ ատենախոսությունում օգտագործված նկարների, աղյուսակների և հապավումների ցանկերը): Հիմնական տեքստը կազմում է 132 էջ, որում ընդգրկված են 103 նկար և 8 աղյուսակ: Աշխատանքի ընդհանուր ծավալը, հավելվածների հետ միասին, կազմում է 159 էջ:

ԳԼՈՒԽ 1. ԱՆԱԼՈԳԱՅԻՆ ԻՆՏԵԳՐԱԼ ՍԽԵՄԱՆԵՐԻ ԱՎՏՈՄԱՏԱՑՎԱԾ ՖԻԶԻԿԱԿԱՆ ՆԱԽԱԳԾՄԱՆ ԱՌԱՆՁՆԱՀԱՏԿՈՒԹՅՈՒՆՆԵՐԸ

1.1. Անալոգային ԻՍ-երի նախագծման անհրաժեշտությունը

Անալոգային ԻՍ-երի նախագծումը մեծ կարևորություն ունի, քանի որ մեզ շրջապատող աշխարհը ամբողջովին անալոգային է, և անալոգային ԻՍ-երը լայն կիրառություն են ստանում: Վերջին տասնամյակում ԻՍ-երի զարգացումը հանգեցրեց անալոգային ԻՍ-երի նախագծման բարդացմանը [50, 53]: Անալոգային ԻՍ-երն անհրաժեշտ են [103]՝

- անալոգ-թիվ ձևափոխիչներում, որպեսզի թվայնացվի մուտքային ազդանշանը,
- թիվ-անալոգ ձևափոխիչներում, ազդանշանը ձևափոխություններից հետո անալոգայինի վերափոխելու համար,
- անալոգային ազդանշանի մշակման այնպիսի գործողությունների համար, ինչպիսիք են՝ ուժեղացումը, ֆիլտրումը և այլն (նկ. 1.1):



Նկ. 1.1. Անալոգային ԻՍ-երի կիրառման օրինակը

1.2. Անալոգային ԻՍ-երի ավտոմատացված ֆիզիկական նախագծումը

ԻՍ-երի ինտեգրման աստիճանի մեծացմանը զուգընթաց նախագծողներին առաջադրվում են ավելի կոշտ պահանջներ նախագծման ժամանակի վերաբերյալ: Առաջանում է ԻՍ-երի նախագծման այնպիսի նոր մեթոդների մշակման անհրաժեշտություն, որոնք որակապես կփոխեն նախագծման տևողությունը: Սա հատկապես անհրաժեշտ է անալոգային ԻՍ-երի նախագծման ոլորտում, որտեղ ավտոմատացման մակարդակը շատ անգամ զիջում է թվային ԻՍ - երի նախագծմանը [57, 84, 85]:

Վերջին տարիներին էլեկտրոնային նախագծումը հիմնականում կենտրոնացված է անալոգային սխեմաների ավտոմատացված նախագծման մեթոդների զարգացման վրա [11, 59]: Ի սկզբանե կենտրոնացումը կատարվել էր հետևյալ հինգ հիմնական ուղղությամբ՝ անալոգային նախագծման ստանդարտ լեզուներ, տրամաբանական նախագծում, անալոգային և թվային սխեմաների սիմուլյացիա, անալոգային և թվային բլոկների ինտեգրում, անալոգային ԻՍ-երի ֆիզիկական սինթեզ [31, 62, 70]: Չնայած այն փաստին, որ նշված բոլոր ուղղություններով էլ կատարվել են մեծ աշխատանքներ, դեռևս վաղ է մտածել անալոգային ԻՍ-երի ամբողջովին ավտոմատ նախագծման մասին [2]:

Անալոգային ԻՍ-երի ավտոմատացման համար նախատեսված գործիքին որևէ ձևով պետք է տրվեն ստացվող ֆիզիկական նախագծի մուտքային պարամետրերը՝ մակերես, ծրագծման թույլատրվող շերտերի հաստություն, հեռավորություն, առավելագույն հոսանք և այլն. այս մուտքային տվյալները կոչվում են նախագծման սահմանափակումներ [59, 60]:

Սահմանափակումների ներմուծումը գործել է մինչև ավտոմատ նախագծման առաջացումը, սխեմայում որպես տեքստ գրվել են սահմանափակումները, և ֆիզիկական նախագիծը կատարվել է՝ հաշվի առնելով դրանք [37]: Հետագայում նախագծերի բարդացմանը զուգընթաց ավելացել է սահմանափակումների քանակը, և սխեմատեխնիկական նախագծում այլևս հնարավոր չի եղել այդքան տվյալներ գրի առնել, որի պատճառով սկսել են սահմանափակումները պահել հատուկ տվյալների հենքերում [60]:

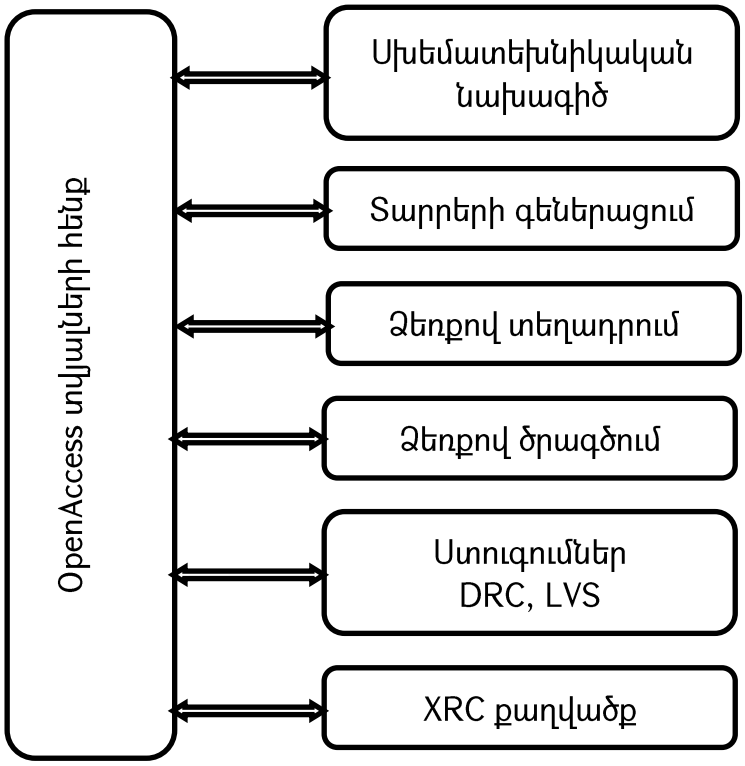
Ֆիզիկական նախագիծը կարող է ֆունկցիոնալ տեսանկյունից ճշգրիտ աշխատել և բավարարել բոլոր պահանջներին, բայց սահմանափակումներին չբավարարելու դեպքում, այն պատվիրատուի կողմից չի ընդունվի: Այսպիսով, ֆիզիկական նախագիծը պետք է մշակվի այնպես, որ բավարարի առաջադրվող բոլոր պահանջներին՝ մակերես, ցրվող հզորություն, առավելագույն հոսանք և այլն:

1.2.1. Ավտոմատացված և ձեռքով նախագծման տարբերությունները

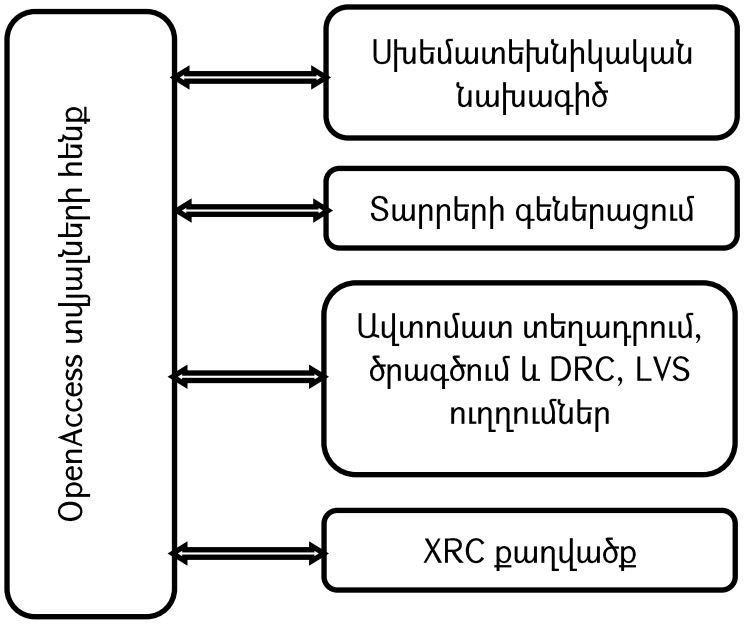
Դիտարկենք ինտեգրալ սխեմաների ֆիզիկական նախագծման ավանդական (ձեռքով) և ավտոմատացված նախագծման առանձնահատկությունները:

- Նախագծման ավանդական մեթոդ: Սկզբում կատարվում է սխեմատեխնիկական նախագիծ, հետո իրականացվում է ֆիզիկական մոդելների գեներացում. գեներացվում են ռեզիստորները, տրանզիստորները, ունակությունները և ֆիզիկական նախագծման անհրաժեշտ այլ տարրեր: Այնուհետև կատարվում է ձեռքով տեղակայում (placement), և ձեռքով ծրագծում (routing): Քանի որ տեղակայումը և ծրագծումը նախագծողների կողմից կատարվում են ձեռքով, ուստի հնարավոր է դրանց ընթացքում տեղի ունենան նախագծման կանոնների խախտումներ [97]. նշված պատճառով ծրագծման ավարտից հետո նախագծողը պետք է կատարի նախագծման կանոնների ստուգում (DRC), ինչպես նաև՝ սխեմատեխնիկական և ֆիզիկական նախագծերի համեմատում (LVS): Եթե նշված ստուգումները սխալներ չեն վերադարձնում, նախագծողն անցնում է նմանակման փուլին: Եթե ստուգումների արդյունքում սխալներ են հայտնաբերվում, նախագծողը հետ է վերադառնում նախորդ փուլերին:
- Նախագծման ավտոմատացված մեթոդ: Այս դեպքում տեղակայումը և ծրագծումը կատարվում են ավտոմատ՝ ծրագրային ապահովման գործիքի միջոցով [34, 35]: Նշված պարագայում բացառվում է սխալների առաջացումը, քանի որ համակարգիչը կատարում է տեղակայում և ծրագծում՝ հաշվի առնելով նախագծման կանոնները [87, 88, 97]: Ֆիզիկական նախագիծը ստանալուց հետո կատարվում են նմանակումներ:

Ձեռքով նախագծման քայլերի հաջորդականությունը բերված է նկ. 1.2 - ում, իսկ ավտոմատացված նախագծմանը՝ նկ. 1.3 - ում:



Նկ. 1.2. Ձեռքով նախագծման մոդելը



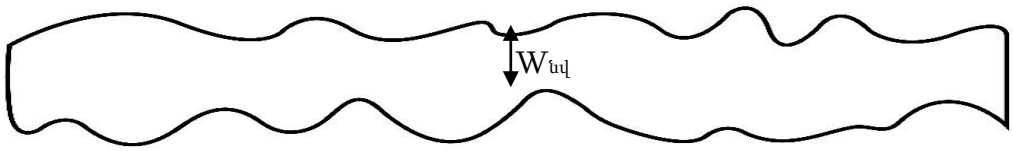
Նկ. 1.3. Ավտոմատացված նախագծման մոդելը

1.2.2. Անալոգային ԻՍ-երի ֆիզիկական նախագծման կանոնների հետազոտումը

Ֆիզիկական նախագծման ժամանակ պետք է նախատեսել, որ նախագիծը բավարարի նախագծման կանոններին: Թվային ԻՍ-երի հետ համեմատած՝ անալոգային ԻՍ-երի ֆիզիկական նախագծման կանոններն ավելի բարդ են և խիստ [42, 89]:

Քննարկենք անալոգային ԻՍ-երում հաճախ հանդիպող ֆիզիկական նախագծման կանոնները:

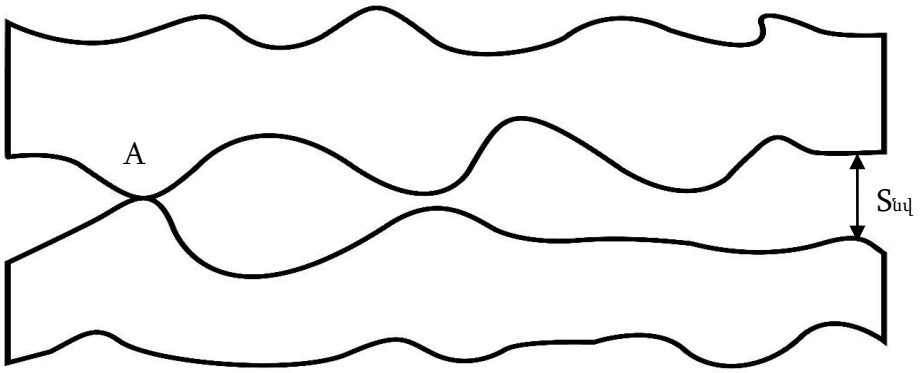
- Լարի նվազագույն լայնություն: Այս կանոնով սահմանվում է լարի նվազագույն լայնությունը: Նշված կանոնի խախտումը կարող է հանգեցնել լարի գերտաքացման և դրա արդյունքում լարը կարող է կտրվել և հանդիսանալ շղթայի բաց տեղամասի առաջացման պատճառ: Նկ. 1.4 - ում պատկերված է արտադրական գործընթացի արդյունքում ստացված լարի մի հատված, որի թույլատրելի նվազագույն լայնությունը $W_{նվ}$ է: Լարը, որը չունի «նվազագույն լայնություն» սահմանափակումը, արտադրության ժամանակ կարող է ունենալ կտրված տեղամասեր: Բացի դրանից, մեծ հոսանքների պատճառով լարերը բարակում են, և եթե նվազագույն լայնության սահմանափակմանը չեն բավարարում, որոշակի ժամանակահատված հետո դրանք կարող են կտրվել: Ներկայում գոյություն ունեն ծրագրային ապահովման գործիքներ, որոնք կատարելով շղթայի հոսանքների անալիզ նախագծողին տրամադրում են այն լարերի ցուցակը, որոնք հնարավոր է ԻՍ-ի աշխատանքի ընթացքում կտրվեն:



Նկ. 1.4. Լարի «նվազագույն լայնություն» կանոնի գրաֆիկական պատկերումը

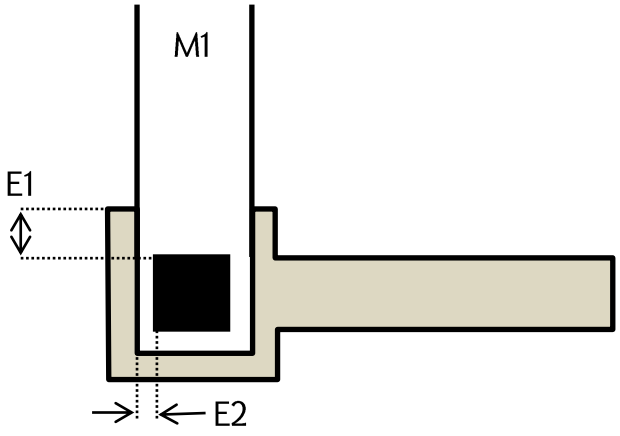
- Նվազագույն հեռավորություն: Այս կանոնով սահմանվում է լարերի միջև եղած նվազագույն հեռավորությունը: Արտադրական գործընթացի ժամանակ կարող է այնպես ստացվել, որ երկու՝ միմյանց մոտ դասավորված լարեր արտադրման գործընթացի անկատարելության պատճառով հպվեն իրար, և առաջանա կարճ միացում, ուստի անհրաժեշտ է պահել երկու լարերի միջև գործող նվազագույն հեռավորությունը:

Նկ. 1.5 - ում պատկերված է լարերի միջև գործող նվազագույն հեռավորության ($S_{նվ}$) խախտման և արտադրման գործընթացով պայմանավորված խորդուբորդությունների պատճառով կարճ միացման առաջացումը A կետում:



Նկ. 1.5. Լարերի միջև «նվազագույն հեռավորություն» կանոնի գրաֆիկական պատկերումը

- Նվազագույն փոխծածկում: Այս կանոնը կիրառվում է երկու տարբեր շերտերի վրա: Նախագծման ընթացքում մի ծրագծման շերտից մյուսին անցնելիս դրանց հատման կետում տեղադրում են միջմիացում, համոզվելու համար, որ այդ միջմիացումը միացված է երկու ծրագծման շերտին էլ անհրաժեշտ է, որ ծրագծման շերտերը որոշակի մակերեսով ծածկեն միջմիացումը: Անհամապատասխան դասավորությունը մի քանի շերտերի միջև կարող է հանգեցնել շղթայի անցանկալի բաց տեղամասերի կամ կարճ միացման: Նկ. 1.6 - ում պատկերված է նվազագույն փոխծածկման կանոնի գրաֆիկական պատկերումը M1 մետաղական շերտի համար Poly բազմասիլիցիումային շերտով, որտեղ E_1 և E_2 –ը փոխծածկման մեծություններն են:



Նկ. 1.6. Լարի «նվազագույն փոխծածկում» կանոնի գրաֆիկական պատկերումը

1.2.3. Գոյություն ունեցող ծրագրային գործիքների և ավտոմատացման մեթոդների համեմատական վերլուծությունը

Անալոգային ԻՍ-երի ավտոմատացված նախագծման մեթոդները հետևյալն են.

- Մուտքային տվյալների վրա հիմնված [51, 52] – ներառում է շղթայի յուրաքանչյուր տարրի չափերը, պարամետրերը: Այս դեպքում ոչ մի երաշխիք չկա, որ կգտնվի լավագույն լուծումը:

Հաջորդ ավտոմատացման երեք մեթոդները հիմնված են նմանակումների վրա: Դրանք օգտագործում են լավարկման շարժիչը, որպեսզի կատարեն նախագծումը: Լավարկումը քայլ առ քայլ գործողություն է, որի ժամանակ նախագծի փոփոխականները փոփոխվում են ամեն քայլում, մինչև իրենց լավագույն արժեքներին հասնելը: Արագագործության ստուգող ծրագիրը ամեն քայլում ստուգում է արագագործության սահմանափակումների խախտվածությունը [21, 22]:

- Հավասարումների վրա հիմնված [2, 5, 32] – օգտագործվում են անալիտիկ հավասարումներ՝ շղթայի աշխատանքային պայմանները հաշվարկելու համար: Այս հավասարումները կարող են լուծվել կամ ձեռքով, կամ հավասարումների լուծման ավտոմատ գործիքների օգնությամբ: Հաջորդ քայլում խնդիրը վերածվում է նմանակման խնդրի և լուծվում՝ օգտագործելով թվային ալգորիթմներ: Տվյալ մեթոդի հիմնական թերությունն այն է, որ ցանկացած նոր տոպոլոգիայի համար նոր հավասարումների համակարգ է առաջանում, որն անհրաժեշտ է լուծել, բացի դրանից, ոչ բոլոր տարրերն են, որ հեշտությամբ կարելի է ներկայացնել թվային հավասարումների տեսքով [86]: Անալիտիկ հավասարումների լուծման ժամանակ կատարվող մոտարկումները հանգեցնում են ցածր ճշգրտության:
- Նմանակումների վրա հիմնված [11, 12, 55] – օգտագործվում են նմանակումներ, շղթայի աշխատանքը գնահատելու համար: Նմանակումների միջոցով կատարվում են շղթայի պարամետրերի բարելավումներ: Այլ մեթոդների համեմատ սա համարվում է ամենաճկուն մեթոդը, քանի որ այն աշխատում է ցանկացած տոպոլոգիայի համար և ապահովում է գերճշգրիտ արդյունքներ [72]: Նույն շղթայի պարամետրերը հնարավոր

է բարելավել մի քանի անգամ՝ տարբեր ելքային պահանջների համար: Այս պատճառով ցանկացած տիպի շղթա հնարավոր է բարելավել շատ կարճ ժամանակում՝ օգտագործելով նմանակումների մեթոդը: Այս մեթոդի թերությունն այն է, որ պահանջում է համակարգչային ահռելի ռեսուրսներ:

- Ուսուցման վրա հիմնված մեթոդ [17, 25, 26]– մոդելավորման ենթակա շղթայի վարքագիծը կառուցվում է ուսուցման մեխանիզմի հիման վրա՝ ըստ տարատեսակների բաշխվածության: Այս մեթոդի դեպքում պահանջվում են վարժեցման օրինակներ, որոնք պետք է հաշվարկված լինեն մինչև շղթայի պարամետրերի բարելավումը: Վարժեցման տարբերակների քանակությունը խստորեն ազդում է մոդելի ճշգրտության վրա [40, 73]:

Հավասարումների վրա հիմնված մեթոդը չի տալիս այնքան ճշգրիտ արդյունքներ, որ հնարավոր լինի ստանալ ավտոմատ նախագծվող և օգտագործման ենթակա անալոգային սխեմաներ [2, 5, 85]: Ուսուցման վրա հիմնված մեթոդով կարելի է ստանալ մեծ ճշգրտություն, բայց այս դեպքում պետք է վարժեցման օրինակներ ունենալ, որի համար ժամանակ է պահանջվում: Նմանակումների վրա հիմնված մեթոդը ամենաճշգրիտն է, որի արագագործությունն էլ ամենաբարձրերից մեկն է:

Ներկայումս անալոգային ԻՍ-երի ավտոմատացված ֆիզիկական նախագծման ոլորտում կատարվում են մեծ աշխատանքներ և մշակվում են հզոր գործիքներ [28, 75]: Մյուս կողմից՝ այդ գործիքները միշտ չէ, որ ամենաօպտիմալ լուծումն են որպես արդյունաբերական լայն կիրառություն ունեցող գործիք [8, 91]: Այդպիսի գործիքը պետք է ունենա.

- խնդրի դրվածքի և մուտքային տվյալների տեղակայման փոքր ժամանակ,
- ճշգրիտ, օգտագործման համար պատրաստի արդյունքներ,
- բավարար ճկունություն, որ աշխատի կամայական շղթայի համար:

Մի շարք աշխատանքներում օգտագործվում է հավասարումների մեթոդը: Այստեղ հասնում են նմանակման բարձր արագությունների՝ օգտագործելով Matlab գործիքը: Պերեիրա-Արրոյի կողմից առաջարկված ալգորիթմում օգտագործվում է ուսուցման մեթոդը. այն օգտագործում է ճշգրիտ մոտարկումներ՝ վարժեցման օրինակներ գեներացնելու համար: Սոմանին օգտագործում է ուսուցման վրա հիմնված մեթոդը

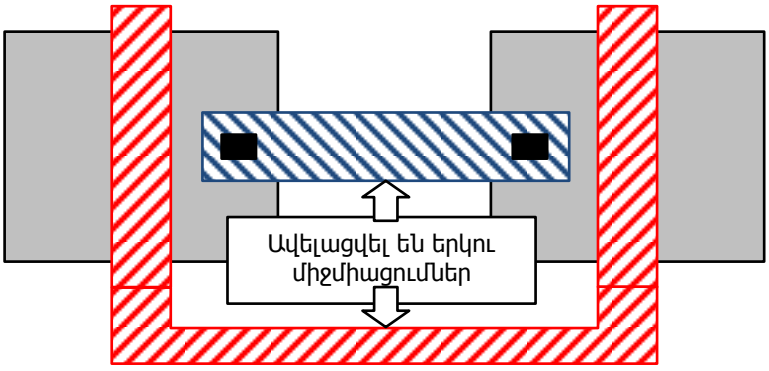
[83]: Այն շատ լավ է աշխատում որոշ տիպի սխեմաների համար, մյուս կողմից՝ այս մոտեցումը ամենաօպտիմալը չէ ընդհանուր տիպի շղթաների համար: Բո-ի կողմից առաջարկված մեթոդը նմանակման հզոր մեթոդ է, որն իրագործում է հիբրիդ մոտեցում: Այն հիմնված է Matlab գործիքի օգնությամբ հավասարումների լուծման և Hspice գործիքով մոտարկումների վրա: Այս մեթոդով կարելի է հասնել մեծ արագագործությունների նույնիսկ բարդ շղթաների համար:

1.3. Անալոգային ԻՍ-երի ֆիզիկական նախագծերի ավտոմատացված ծրագծումը

Անալոգային հանգույցները զբաղեցնում են ամբողջ ԻՍ-ի մակերեսի ընդամենը 10% -ը, սակայն դրանց նախագծումը խլում է ամբողջ նախագծման ժամանակի 90% -ը [35, 41]: Անալոգային ԻՍ-երի նախագծման ավտոմատացումը կարող է խնայել ամբողջ ԻՍ-ի նախագծման ժամանակի զգալի մասը, և քանի որ նախագծման ժամանակի մեծ մասը ծախսվում է ծրագծման վրա, ուստի ավտոմատացման կարևորագույն փուլը ֆիզիկական նախագծի ավտոմատացված ծրագծման փուլն է [76]: Ներկայումս զգալի ջանքեր են գործադրվում անալոգային ԻՍ-երի նախագծման ավտոմատացման համար: Խնդիրը նվազագույն ժամանակում անալոգային սխեմաներ նախագծելն է, որոնք կբավարարեն տվյալ խնդրի պահանջները:

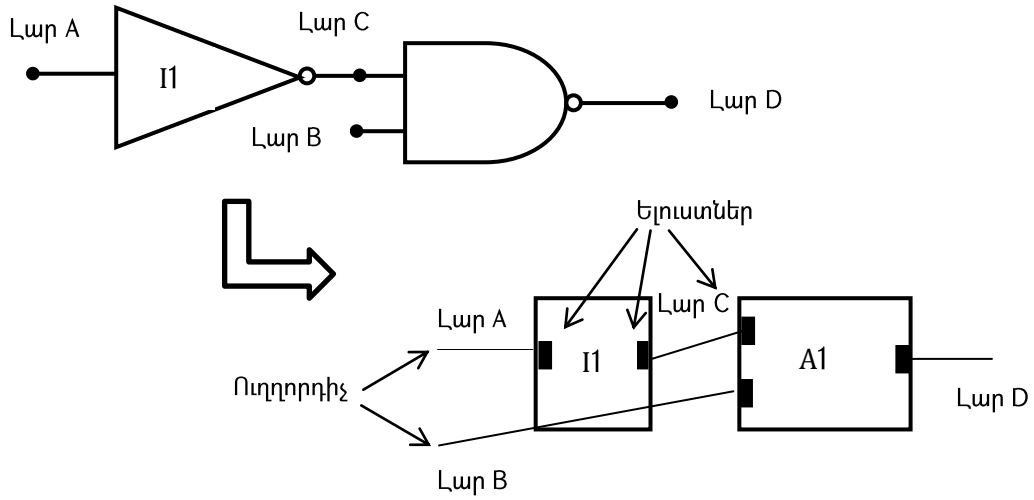
1.3.1. Ավտոմատացված ծրագծման անհրաժեշտությունը

Սկզբնական փուլում, երբ ֆիզիկական նախագծերը պարզ էին և պարունակում էին տասնյակ տարրեր, դրանց ծրագծումը կատարվում էր ամբողջովին ձեռքով (նկ. 1.7):



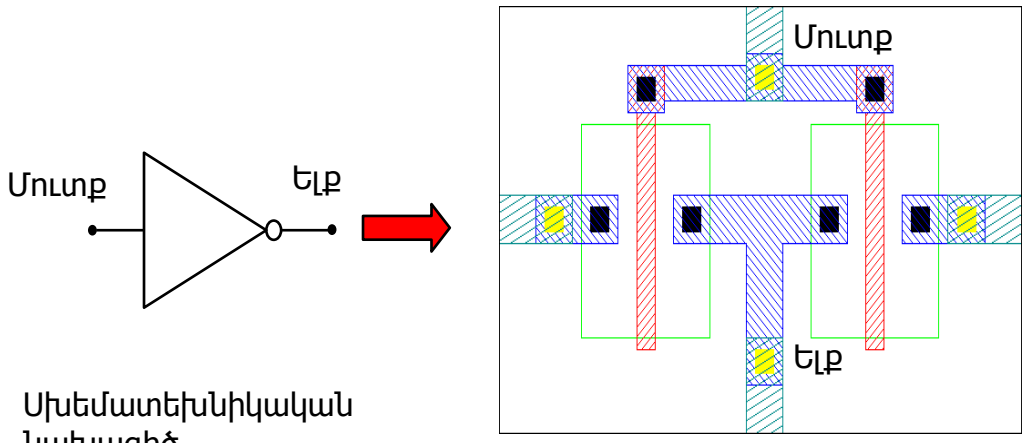
Նկ. 1.7. Միջմիացումների ձեռքով ավելացումը

Հետագայում տարրերի քանակի աճին զուգընթաց առաջանում է ավտոմատացված և լրիվ ավտոմատ նախագծման անհրաժեշտությունը: Սկզբնական փուլում ավտոմատացվեց ֆիզիկական նախագծի տեղադրումը և ուղղորդիչների ընդգծումը
(նկ. 1.8)՝ հիմնվելով սխեմատեխնիկական նախագծի վրա:



Նկ. 1.8. Ֆիզիկական նախագծի և ուղղորդիչների ավտոմատ գեներացումը սխեմատեխնիկական նախագծից

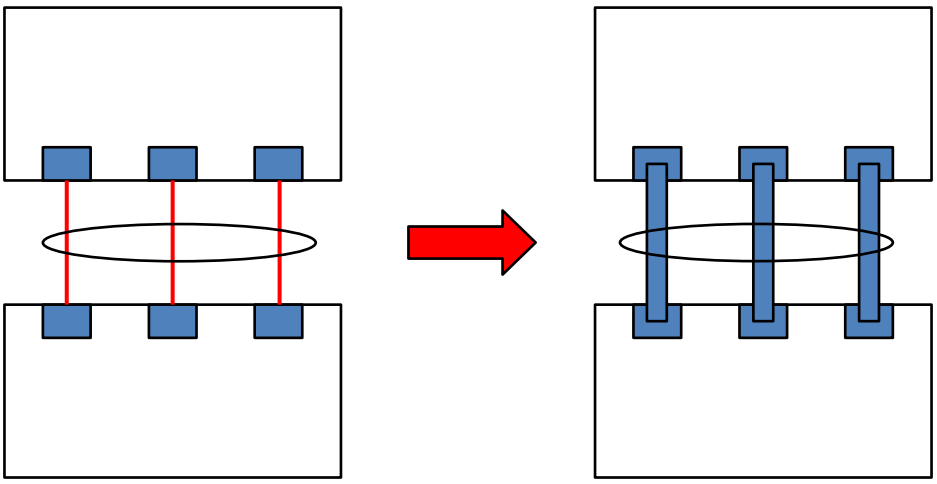
Այս փուլում ֆիզիկական նախագծողին մնում է ուղղորդիչները փոխարինել իրական լարերով և կատարել համապատասխան ելուստների միացում: Այս տիպի ավտոմատացում ապահովելու համար անհրաժեշտ է ունենալ բոլոր տիպի սխեմատեխնիկական տարրերին համապատասխան ֆիզիկական տարրերը. դրանով սկսեցին զբաղվել ստանդարտ գրադարանների նախագծողները: Նկ. 1.9 - ում բերված են ՈՉ տրամաբանական տարրի սխեմատեխնիկական և ֆիզիկական նախագծերը:



Նկ. 1.9. ՈՀ տրամաբանական տարրի սխեմատեխնիկական և ֆիզիկական նախագծերը

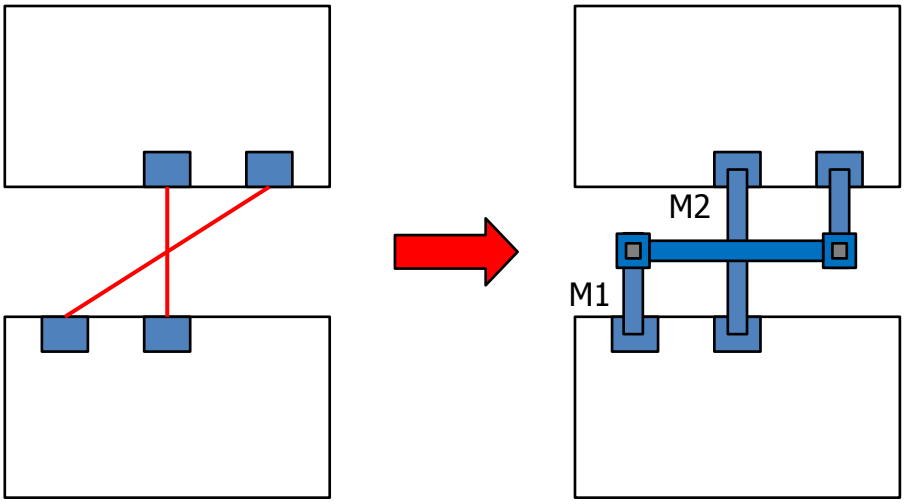
Որոշ դեպքերում ուղղորդիչների ծրագծումը կատարվում է շատ հեշտ, և այն ավտոմատացնելու համար անհրաժեշտ չէ կիրառել բարդ ալգորիթմներ. որպես օրինակ դիտարկենք նկ. 1.10 - ում բերված ելուստների ծրագծումը, այն կարելի է կատարել՝ համապատասխան ելուստները ուղղակի միացնելով զուգահեռ լարերով:

Շատ դեպքերում ծրագծումը դառնում է բարդ գործընթաց, և ավտոմատացնել այն պարզ ալգորիթմներով հնարավոր չէ: Նկ. 1.11 - ում պատկերված է մի դեպք, երբ երկու տարբեր M1 և M2 շերտերի լարերը պետք է անցնեն միմյանց վրայով, բայց այդ դեպքում կստացվի կարճ միացում, որի պատճառով ավտոմատ ծրագծող գործիքը պետք է տեղադրի միջմիացում և անցնի հաջորդ մետաղական շերտ, այդ շերտով անցնի առաջին շերտի վրայից և կրկին տեղադրի միջմիացում, որպեսզի հետ վերադառնա առաջին շերտին և ավարտի ծրագծումը:

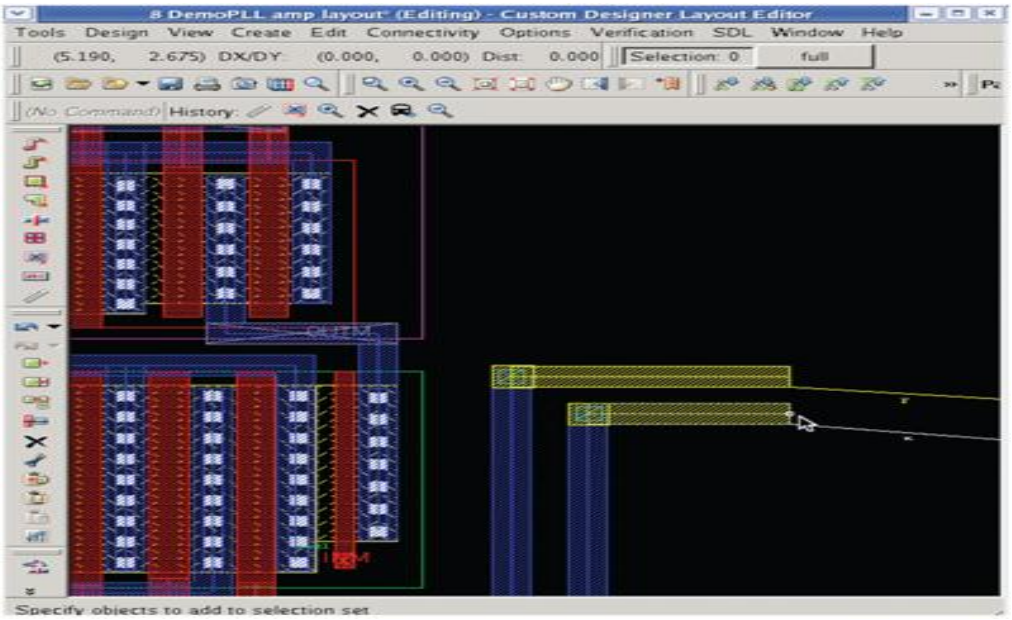


Նկ. 1.10. Ելուստների ծրագծման պարզ ալգորիթմը

Շատ ֆիզիկական նախագծման գործիքներում առկա է ինտեգրված ինտերակտիվ ծրագծող գործիք: Այդ գործիքներում կատարվում է ավտոմատ միջմիացումների տեղադրում, ավտոմատ մետաղական շերտի փոփոխություն, լարերի՝ միմյանցից հեռացում: Այն կատարում է նաև ավտոմատ DRC կանոնների կիրառում, այսինքն՝ նախագծողը չի կարող խախտել DRC կանոնները ձեռքով ծրագծման ժամանակ: Նկ. 1.12 - ում բերված է ինտերակտիվ ծրագծման օրինակ Սինոփսիս ընկերության Custom Designer միջավայրում:

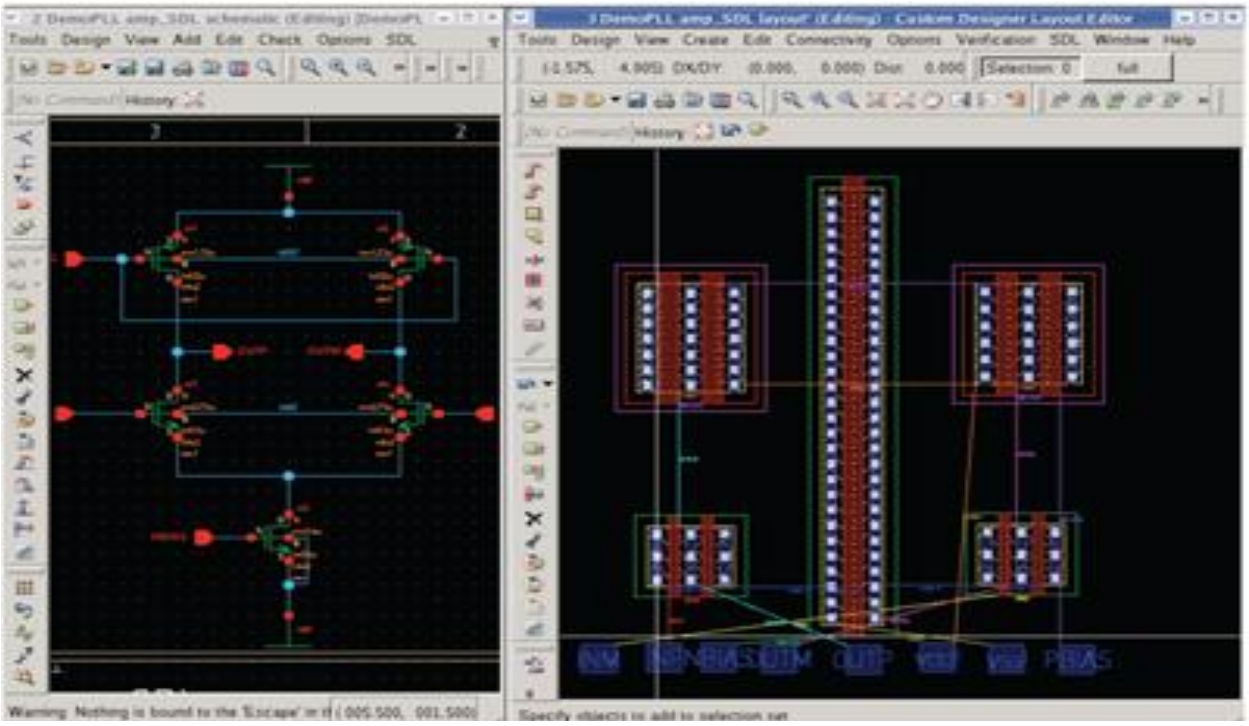


Նկ. 1.11. Երկու տարբեր մետաղական շերտերով ծրագծումը



Նկ. 1.12. Դիֆերենցիալ զույգի ինտերակտիվ ծրագծումը

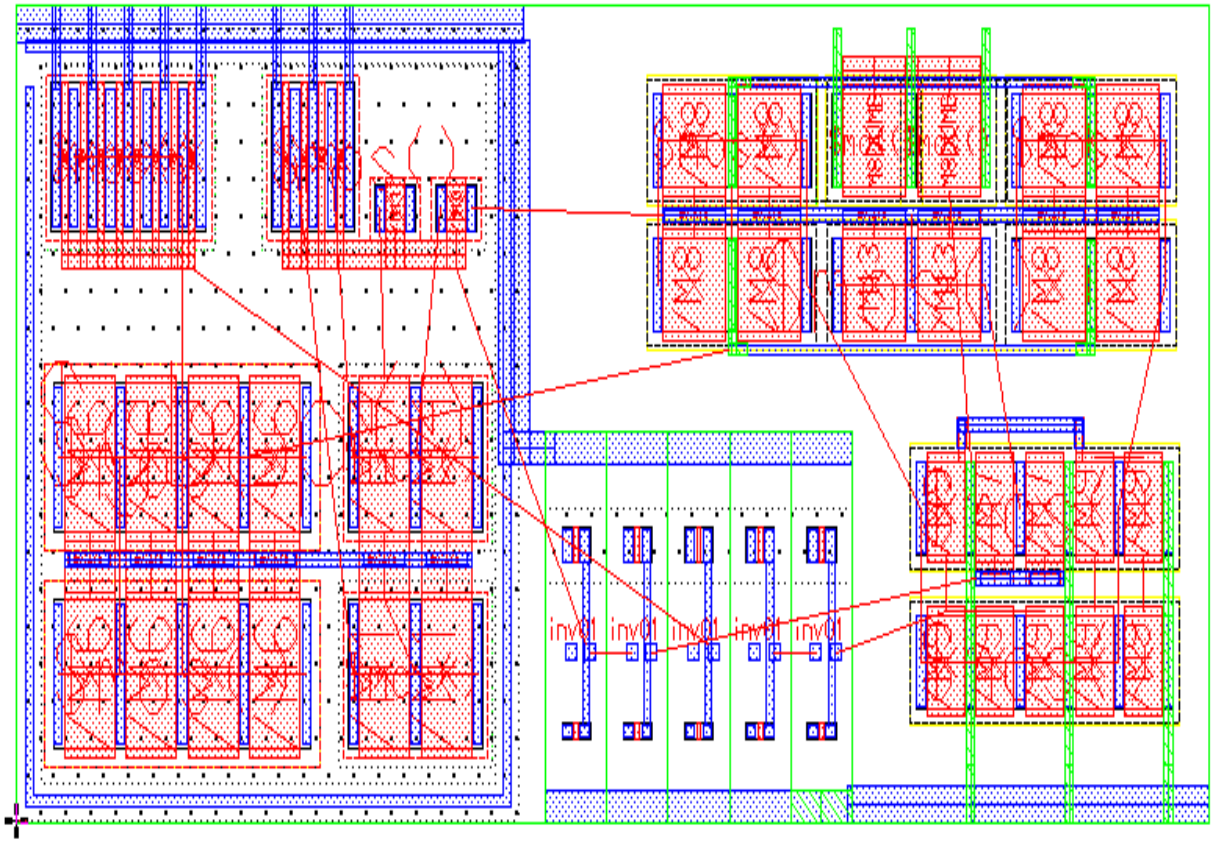
Ավտոմատացված կերպով իրականացվում է ոչ միայն միջմիացումների ինտերակտիվ ծրագծումը, այլ նաև սխեմատեխնիկական նախագծից ֆիզիկական նախագծի գեներացումը: Այս գործընթացը կատարվում է՝ նախօրոք ունենալով սխեմատեխնիկական տարրերին համապատասխան ֆիզիկական նախագծերը: Ֆիզիկական նախագծի տարրերը ավտոմատ գեներացվում են սխեմատեխնիկական նախագծի պարամետրերին համապատասխան, այնուհետև կատարվում է տեղադրում այն դիրքերով, ինչ դիրքերում գտնվում էին դրանք սխեմատեխնիկական նախագծում [29]: Նկ. 1.13 - ում բերված է Custom Designer միջավայրում սխեմատեխնիկական նախագծից ֆիզիկական նախագծի հանգույցների ավտոմատ գեներացում: Այս պարագայում նախագծողին մնում է կատարել համապատասխան ելուստների ծրագծում: Այս մեթոդնով ծրագծումը բավականաչափ պարզեցնում և արագացնում է ֆիզիկական նախագծման գործընթացը, սակայն ներկայումս կան այնպիսի մեծ նախագծեր, որ նույնիսկ այդ գործիքի կիրառմամբ նախագծումը խլում է բավականաչափ մեծ ժամանակ՝ օրեր, շաբաթներ, իսկ որոշ դեպքերում՝ նաև ամիսներ: Այս դեպքում առաջանում է ամբողջովին ավտոմատ գործիքի մշակման անհրաժեշտություն, որը հիմնվելով ֆիզիկական նախագծի սահմանափակումների և ուղղորդիչների վրա կկատարի ավտոմատ ծրագծում [13, 14]:



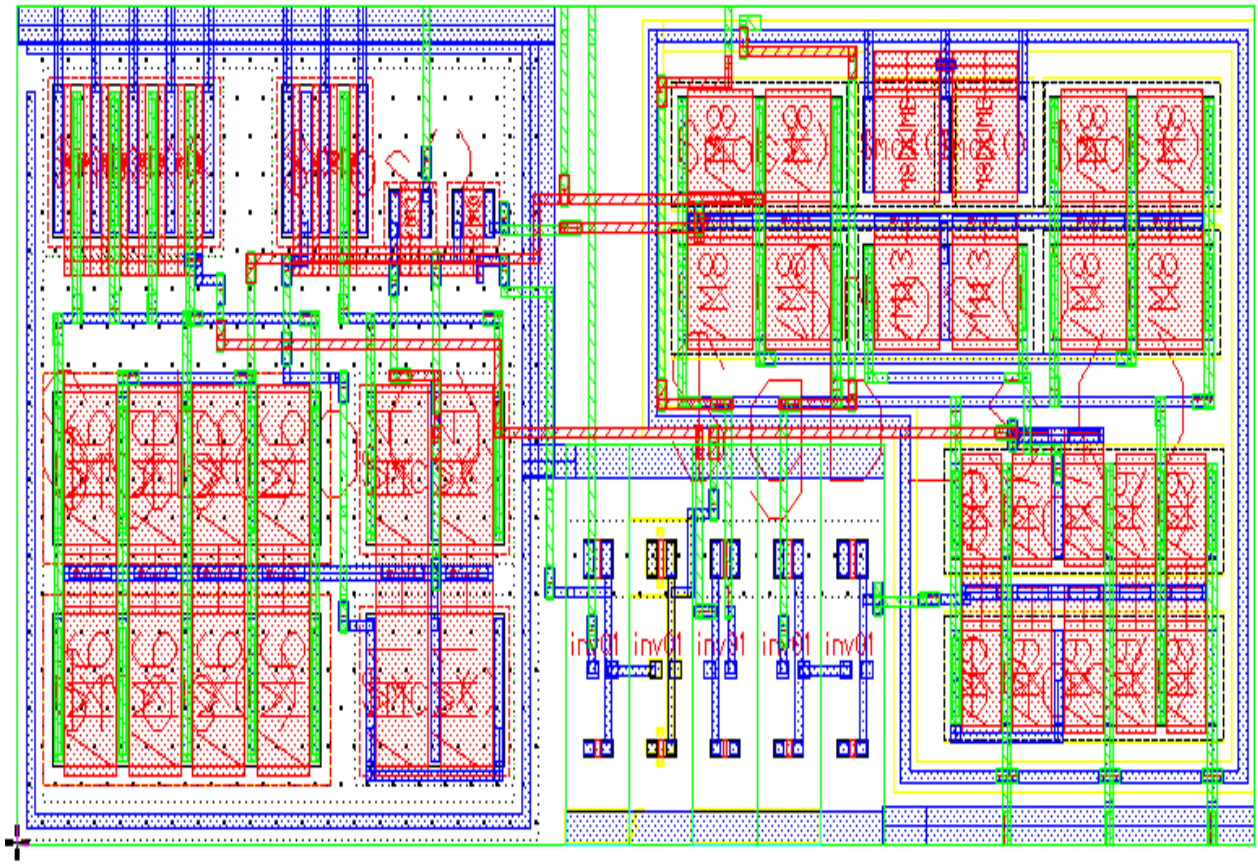
Նկ. 1.13. Ֆիզիկական նախագծի ավտոմատ գեներացումը և տեղադրումը

Նկ. 1.14 և նկ. 1.15 - ում բերված են ֆիզիկական նախագծի ուղղորդիչներով պատկերը և ավտոմատ գործիքի կողմից վերջնական ծրագծված նախագիծը:

Ավտոմատ ծրագծման արդյունավետությունը կախված է բազմաթիվ գործոններից՝ ծրագծման համար թույլատրելի մետաղական շերտերից, լարերի հաստություններից, տարրերի միջև հեռավորություններից, նախագծի վրա դրվող սահմանափակումներից և այլն [19, 20]:



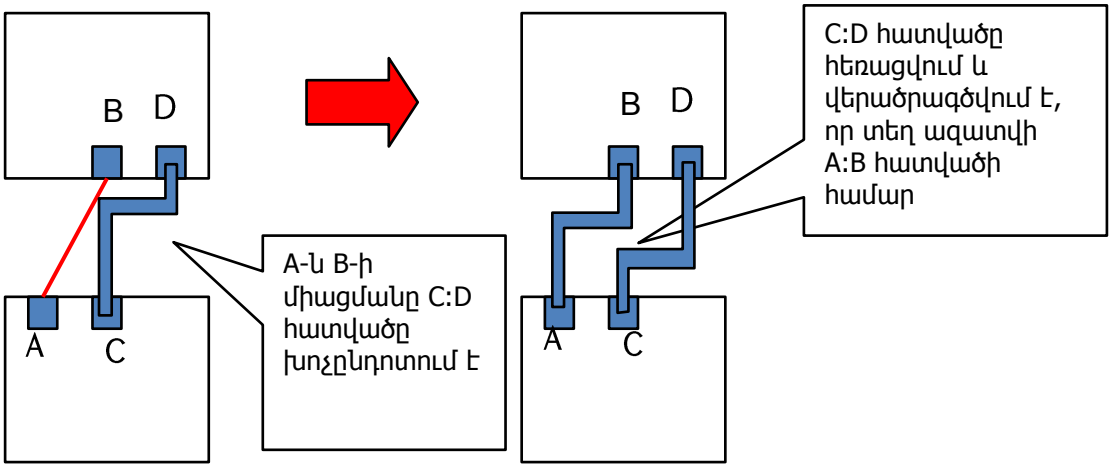
Նկ. 1.14. Ուղղորդիչներով ֆիզիկական նախագիծը



Նկ. 1.15. Ավտոմատ գործիքով ծրագծված նախագիծը

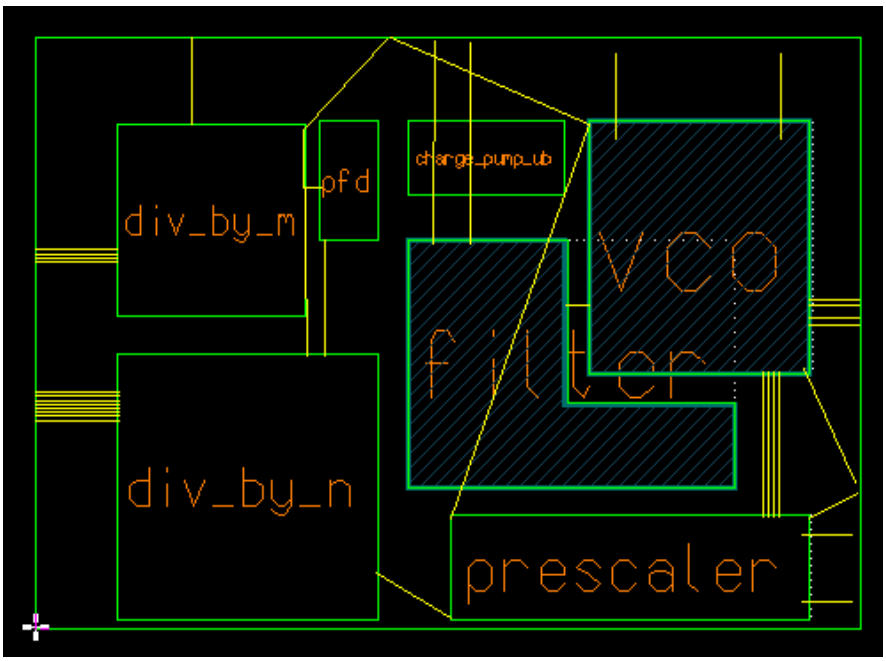
Սկզբնական շրջանի ավտոմատ ծրագծման գործիքները սահմանափակված էին. դրանք կարող էին ծրագծում կատարել միայն նախօրոք սահմանված հատուկ ցանցի եզրերով, հետագայում նախագծվեցին գործիքներ, որոնք հնարավորություն ունեն տեղադրել լարերը այնտեղ, որտեղ դիրքը ամենաարդյունավետն է [15, 39]: Ավտոմատ ծրագծման ալգորիթմում պետք է մշակել նաև վերածրագծման գաղափարը, երբ ավելի արդյունավետ նախագիծ ստանալու նպատակով արդեն ծրագծված հատվածում լարերը ջնջվում են և կատարվում է վերածրագծում: Նկ. 1.16 - ում պատկերված է այդպիսի մի դեպք, երբ A ելուստը B-ին միացնելու համար անհրաժեշտ է վերածրագծել C:D հատվածը:

Երբ սխեմայի բլոկների քանակը և միջմիացումների քանակը շատ է մեծանում, վերածրագծումը անարդյունավետ և շատ դեպքերում անիմաստ և չափազանց ժամանակատար գործողության է վերածվում և կարող է երբևէ չավարտվել և չտալ վերջնական նախագիծ:



Նկ. 1.16. Օպտիմալ սխեմայի ապահովման համար նախագծի վերածրագծումը

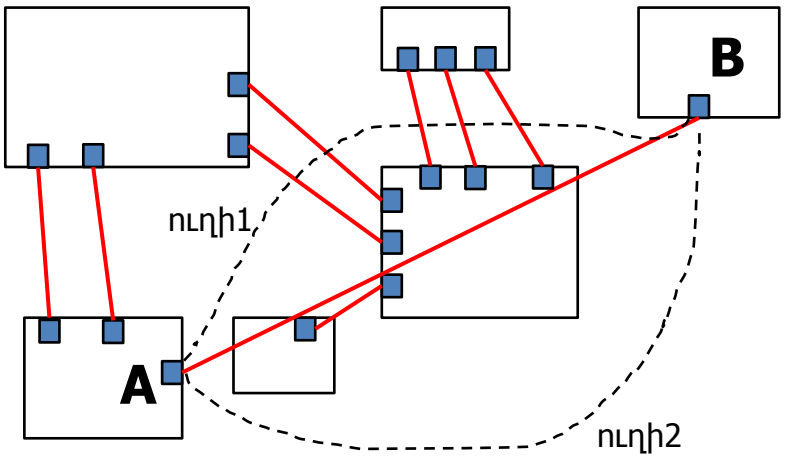
Օրինակ նկ. 1.17 - ում բերված է փուլահաճախական ինքնահամալարման համակարգի ֆիզիկական նախագիծ, որում ծրագծում արված չէ, այն անցել է բջիջների տեղակայում փուլը և սխեմատեխնիկական նախագծի հիման վրա գեներացվել են ուղղորդիչները: Նախագծում այնքան շատ է տարրերի և ուղղորդիչների քանակը, որ այն ծրագծել, կիրառելով վերածրագծման մեթոդը, հնարավոր չէ:



Նկ. 1.17. Փուլահաճախական ինքնահամալարման համակարգի ֆիզիկական նախագիծը

1.3.2. Ավտոմատ ծրագծման ալգորիթմների առանձնահատկությունները

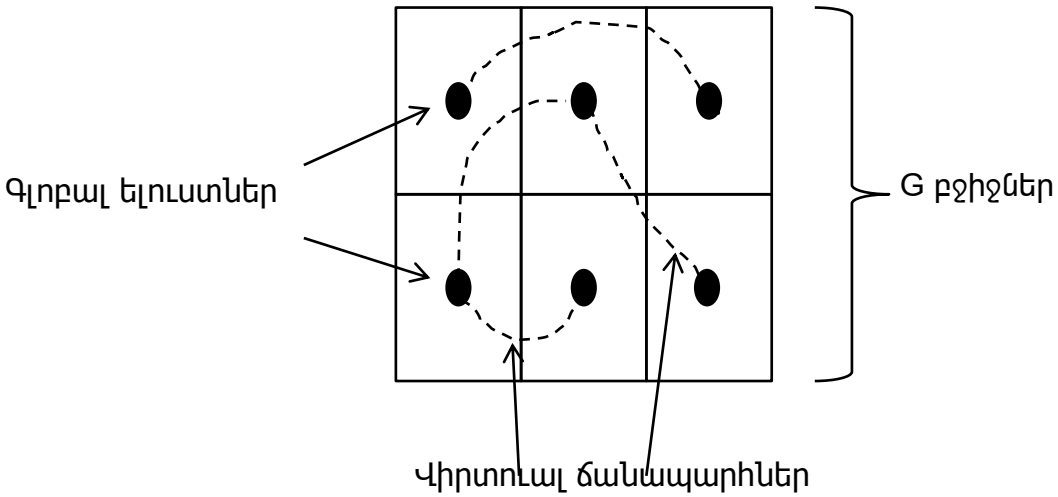
Ավտոմատ ծրագծում կատարող գործիքը պետք է օժտված լինի որոշակի բանականությամբ. այն պետք է կարողանա ընտրել օպտիմալ լուծումներ, որից հետո միայն կատարել ծրագծում, որպեսզի հնարավորին չափով խուսափի վերածրագծման ժամանակատար գործողությունից [15]: Եթե գործիքը նախօրոք իմանա, թե նախագծի որ հատվածներում է լինելու լարերի մեծ խտություն, այն կկարողանա ավելի օպտիմալ որոշումներ կայացնել և շրջանցել այդ հատվածները: Նախագծում ուղղորդիչներն օգնում են նախագծողին հասկանալ խիտ լարերով հատվածները և փորձել շրջանցել դրանք: Նկ. 1.18 - ում բերված է մի օրինակ, որտեղ պետք է միացվեն A և B ելուստները, և նախագծողը պետք է ընտրի դա իրագործելու ուղին: Ուղի1-ի դեպքում այն պետք է անցնի ավելի մեծ քանակությամբ ուղղորդիչների վրայով, քան ուղի2-ի դեպքում, ուստի նախագծողը կընտրի ուղի2-ը, որպեսզի խուսափի ավելորդ միջմիացումներ տեղադրելուց և մետաղական շերտեր փոխելուց:



Նկ. 1.18. Ծրագծման ուղու ընտրությունը, ըստ ուղղորդիչների խտության

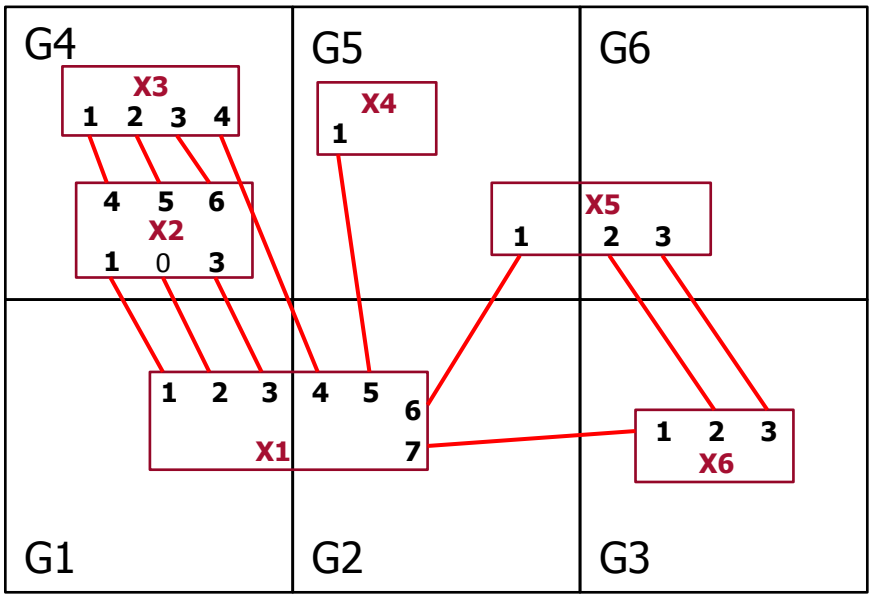
Նմանատիպ մեխանիզմ անհրաժեշտ է ավտոմատ ծրագծման գործիքին ևս՝ ծրագծման ուղիները նախապես գնահատելու համար: Լուծումը հետևյալն է. բաժանել ֆիզիկական նախագիծը փոքր բջիջների զանգվածի՝ այսպես կոչված G բջիջների: Յուրաքանչյուր բջջին պատկանող ելուստները միավորվում են մեկ ելուստի մեջ, որը տեղադրված է բջջի կենտրոնում: Այդ ելուստները կոչվում են գլոբալ ելուստներ: Նկ. 1.19 - ում պատկերված են G բջիջներ իրենց գլոբալ ելուստներով:

Այնուհետև կառուցվում են համապատասխան գլոբալ ելուստները միացնող վիրտուալ ճանապարհները: Դիտարկենք նկ. 1.20 - ում բերված փոքր ֆիզիկական նախագիծը՝ բաղկացած 6 G բջջից և 6 տարրից:



Նկ. 1.19. Ֆիզիկական նախագծի բաժանումը G բջիջների

Ծրագծման գործիքը պարունակում է ինֆորմացիա յուրաքանչյուր G բջջում պարունակվող տարրերի և ելուստների մասին: Այդ ինֆորմացիան պահվում է աղյուսակի տեսքով (աղ. 1.1):



Նկ. 1.20. G բջիջների բաժանած ֆիզիկական նախագիծը

Աղյուսակ 1.1

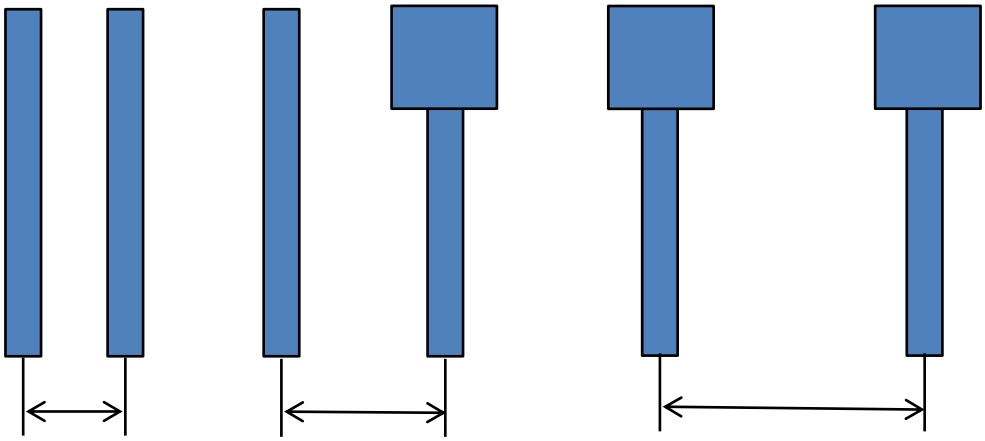
Ֆիզիկական նախագծի G բջիջների և դրանցում պարունակվող տարրերի և ելուստների աղյուսակ

-	G1	G2	G3	G4	G5	G6
Ելուստներ	X1/1,2,3	X1/4,5,6,7	X6/1,2,3	X2/1,2,3,4,5,6 X3/1,2,3,4	X4/1 X5/1	X5/2,3

Իմանալով յուրաքանչյուր բջջում առկա ելուստները՝ գործիքը հնարավորություն ունի որոշել օպտիմալ միջմիացման ճանապարհները:

Կարևոր դեր ունի նաև լարերի միջև նվազագույն հեռավորության ընտրման գործողությունը: Գոյություն ունի երեք տիպի լարերի ծրագծման հեռավորության սահմանափակում: Նկ. 1.21 - ում պատկերված են այդ հեռավորությունները: Առաջինը լար-լար հեռավորությունն է, որն ամենափոքր հեռավորությունն է, այս դեպքում ծրագծումը կատարվում է ամենամեծ խտությամբ, սակայն գործիքին այլևս տարածք չի մնում, որ տեղադրի միջմիացում և անցնի հաջորդ մետաղական մակարդակ: Երկրորդը լար-միջմիացում հեռավորությունն է, որի դեպքում լարերից մեկի վրա

հնարավոր է տեղադրել միջմիացում, և երրորդը միջմիացում-միջմիացում հեռավորությունն է, որն ապահովում է ծրագծման ճկունություն, բայց ունենում ենք նախագծի ամենամեծ մակերեսը:



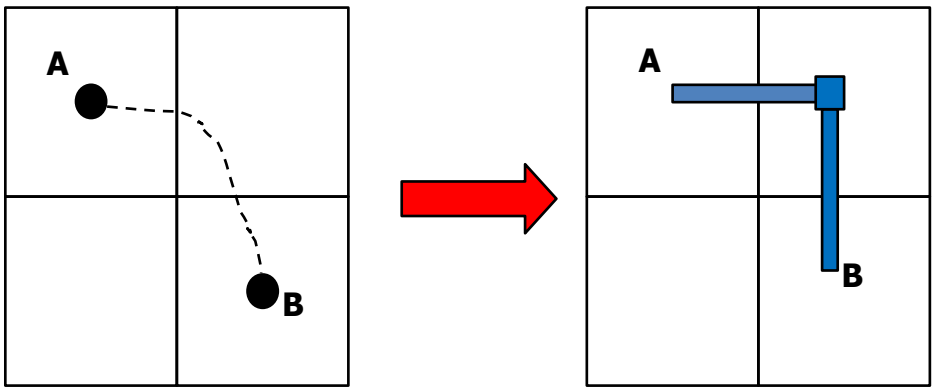
Նկ. 1.21. Ծրագծման նվազագույն հեռավորության ընտրումը. ա) լար-լար, բ) լար-միջմիացում, գ) միջմիացում-միջմիացում

Ցածր մետաղական շերտերի ծրագծման համար, որտեղ մետաղից մետաղ անցման հավանականությունը շատ մեծ է, պետք է օգտագործել լար-միջմիացում կամ միջմիացում–միջմիացում հեռավորությունը, իսկ, սնուցման լարերի ծրագծման համար, որտեղ միջմիացումների տեղադրումը անհրաժեշտ չէ, պետք է ծրագծումը կատարել լար-լար հեռավորությամբ:

Յուրաքանչյուր մետաղական շերտի համար ծրագծման գործիքին պետք է տալ նախընտրելի ծրագծման ուղղությունը: Օրինակ M1, M3, M5 և M7 մետաղական շերտերը ծրագծվում են հորիզոնական, իսկ M2, M4, M6 և M8 մետաղներով ծրագծում կատարվում է ուղղահայաց ուղղություններով:

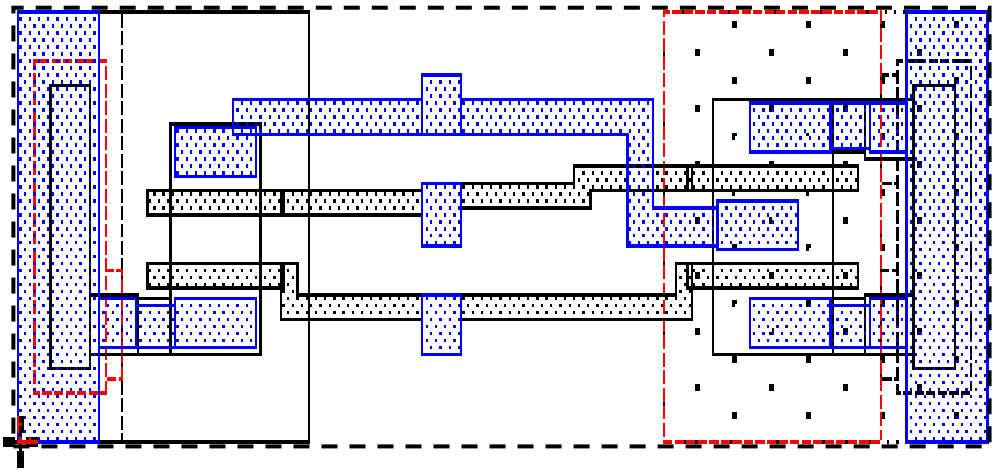
Այն բանից հետո, երբ վիրտուալ ճանապարհները ծրագծվել են, գործիքն անցնում է իրական ծրագծմանը մետաղական շերտերով: Նկ. 1.22 - ում պատկերված է վիրտուալ ծրագծումից իրական ծրագծման անցումը:

Օպտիմալ ծրագծման համար կարևոր նշանակություն ունի սահմանափակումների ճիշտ սահմանումը, օրինակ, ծրագծման լարերի հաջորդականությունը. սկզբում ծրագծվում են ավելի բարձր կարևորություն ունեցող լարերը [37, 38]: Լարերի առավելագույն երկարությունը ևս կարող է ծառայել որպես սահմանափակում:



Նկ. 1.22. Վիրտուալ ճանապարհների կիրառմամբ ծրագծումը

Հաջորդ կարևորագույն սահմանափակումներից մեկը լարերի լայնությունների վերաբերյալ սահմանափակումն է: Օրինակ, սնուցման լարերը պետք է ծրագծվեն ավելի լայն, քան մնացած լարերը, ամենակարևոր սահմանափակումներից մեկը նախագծի եզրերի սահմանումն է, սրանով տրվում են ելքային ելուստների դիրքերը և ֆիզիկական նախագծի մակերեսը [18]: Նկ. 1.23 - ում կետագծերով պատկերված է նախագծի եզրային սահմանը: Այն ելուստները, որոնք գտնվում են սահմանից դուրս, անտեսվում են գործիքի կողմից:



Նկ. 1.23. Ֆիզիկական նախագծի սահմանի նշանակումը

Ծրագծում իրականացնելու համար գործիքին անհրաժեշտ է երկու հիմնական տվյալ: Առաջինը տեխնոլոգիական փաստաթուղթն է. այն հաճախ տրվում է LEF ընդլայնմամբ, հաջորդը նախագծի մասին տեղեկությունների փաստաթուղթն է՝ DEF ընդլայնմամբ: Տեխնոլոգիական փաստաթղթում տրվում են տեխնոլոգիական գործընթացում առկա բոլոր տվյալները. ֆիզիկական լարերի միջև հեռավորությունը,

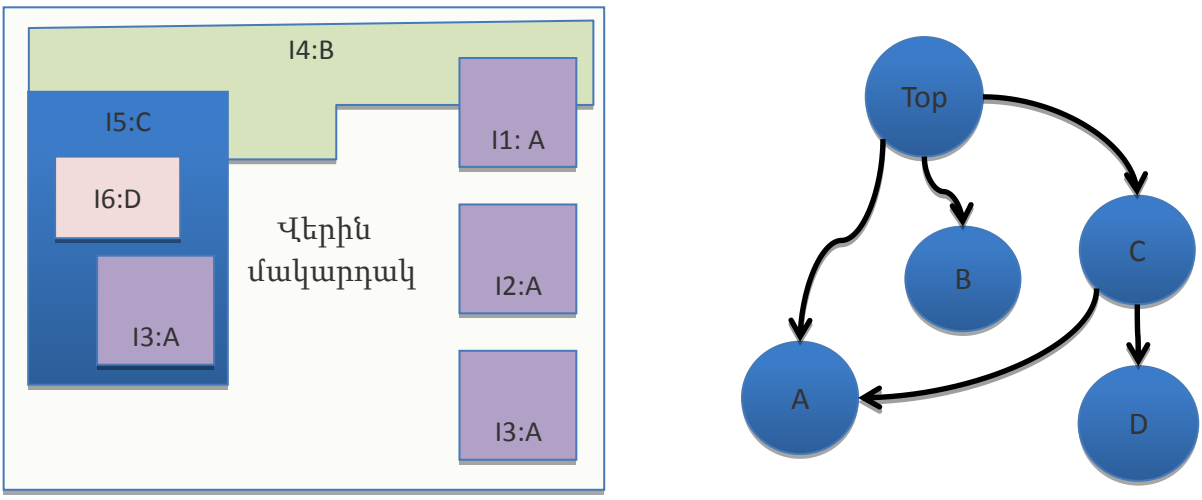
դրանց լայնությունը, ելուստների տիպերը, չափերը և այլն: DEF փաստաթղթում պարունակվում են ֆիզիկական նախագծի բջիջների անունները, տեղաբաշխվածությունը և համապատասխան ելուստների միջմիացումները:

1.3.3. Հիերարխիկ ծրագծում

Մեծ ֆիզիկական նախագծերն ունենում են մի քանիսից մինչև տասնյակ մակարդակներ: Ծրագծման գործիքը պետք է կարողանա ծրագծում կատարել այդ բոլոր հիերարխիական մակարդակներում [29, 30]: Նկ. 1.24 - ում պատկերված է ֆիզիկական նախագիծ, որի բջիջները գտնվում են 6 հիերարխիական մակարդակներում:

Բազմամակարդակ ծրագծման համար գործիքը պետք է բավարարի հետևյալ պահանջներին.

- ծրագծման գործիքը միաժամանակ ծրագծում կատարում է միայն մեկ մակարդակում, այսինքն՝ ծրագծում կատարվում է հաջորդաբար անցնելով մի մակարդակից մյուսին,
- գործիքը պետք է տեսնի ծրագծման բոլոր մակարդակներում գտնվող բջիջները, և երբ ծրագծում է կատարվում ինչ-որ մակարդակում, ապա անմիջապես նրա ներքևի մակարդակի բջիջները արգելքի բջիջներ են, այնպես որ դրանց վրայով ծրագծում չի կատարվում,
- ծրագծում կատարվում է ներքևից վերև: Սկզբում ծրագծվում են նախագծի ներքևի մակարդակները, բարձրանալով վերև, ծրագծվում է ամենավերևի մակարդակը:

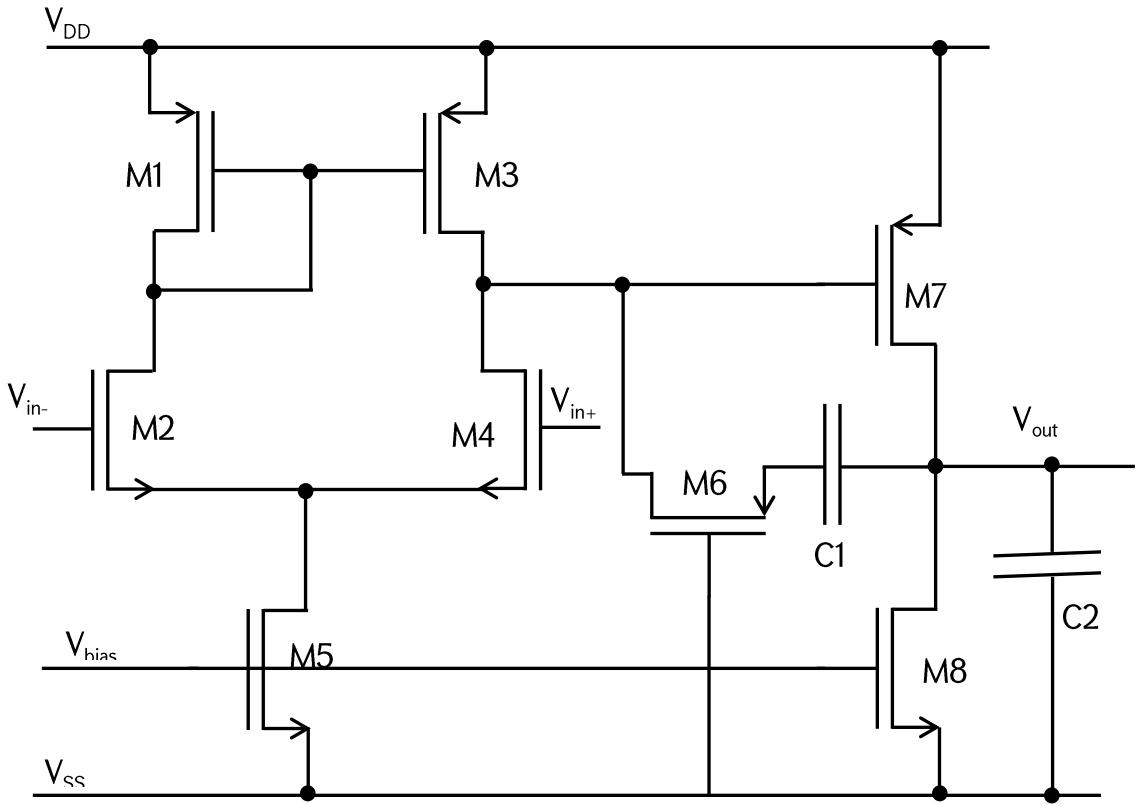


Նկ. 1.24. Բազմամակարդակ ֆիզիկական նախագիծը

1.3.4. Ավտոմատացված ծրագծման ժամանակ սահմանափակումների ազդեցությունը ֆիզիկական նախագծի պարամետրերի վրա

Անալոգային ԻՍ-երի ֆիզիկական նախագծման ժամանակ մեծ նշանակություն ունեն ելքային նախագծի վրա դրվող պահանջները՝ հզորություն, մակերես, առավելագույն հոսանք և այլն, որոնք նախագծման գործիքին տրվում են սահմանափակումների միջոցով [38, 39]:

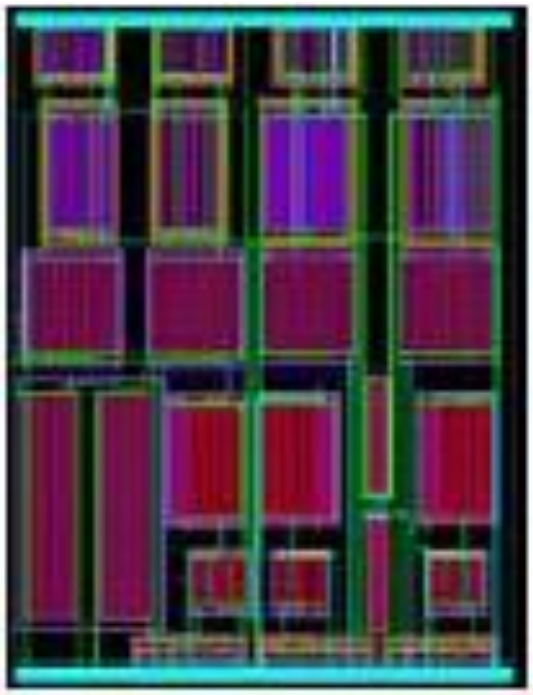
Հետազոտենք նկ. 1.25 - ում պատկերված ուժեղարարը [36, 89]:



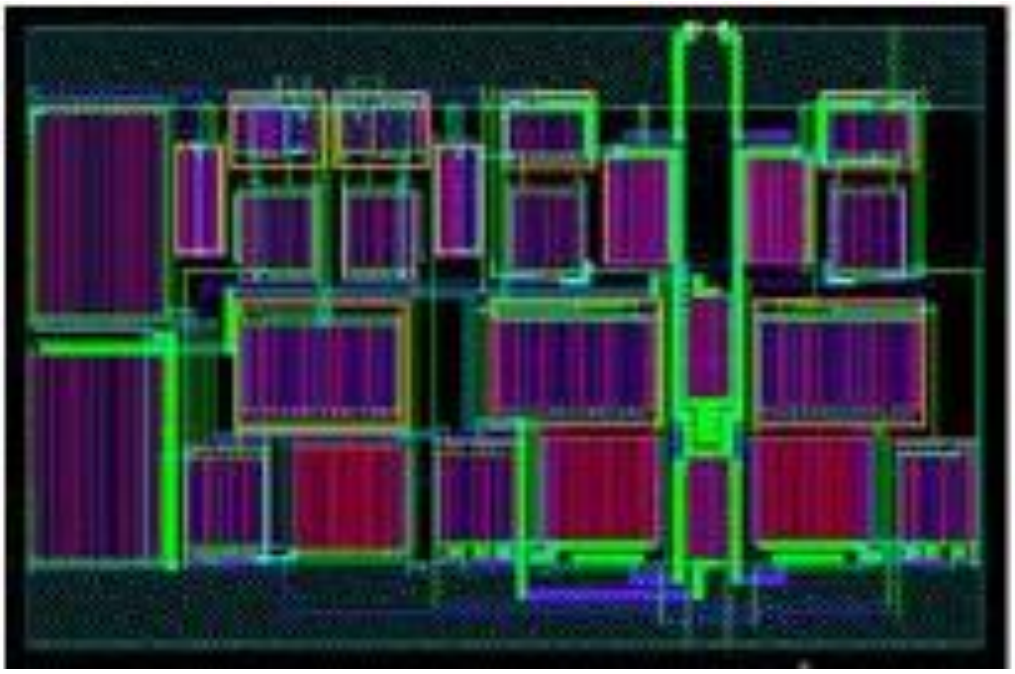
Նկ. 1.25. ՄՕԿ ուժեղարարի սխեմատեխնիկական նախագիծը

Այս նույն սխեմայի համար կատարվել է երկու ֆիզիկական նախագիծ, առաջինը պետք է ապահովի 3 *մՎտ* (նկ. 1.26), երկրորդը՝ 9 *մՎտ* (նկ. 1.27) հզորություն:

Կատարված ֆիզիկական նախագծերից ստացվում է, որ ըստ աշխատանքային հզորության պահանջների, ֆիզիկական նախագծի մակերեսը կարող է մի քանի անգամ տարբեր ստացվել: Այսպես, առաջին դեպքում նախագիծը զբաղեցնում է 0,004 $մմ^2$ իսկ երկրորդ դեպքում՝ 0,009 $մմ^2$ մակերես:



Նկ. 1.26. 3 մՎտ ելքային հզորությամբ ուժեղարարի ֆիզիկական նախագիծը



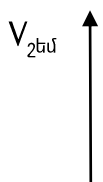
Նկ. 1.27. 9 մՎտ ելքային հզորությամբ ուժեղարարի ֆիզիկական նախագիծը

1.4. Անալոգային ԻՍ-երի ֆիզիկական նախագծման ժամանակ երկրորդական երևույթների հետազոտումը

Անալոգային ԻՍ-երի ֆիզիկական նախագծման ժամանակ առաջ են գալիս մի շարք անցանկալի երկրորդական երևույթներ, որոնց ազդեցությունը նախագծի ելքային պարամետրերի վրա տեխնոլոգիական գործընթացի նվազմանը զուգընթաց մեծանում է [1, 41, 98]:

ԻՍ-երի ֆիզիկական նախագծման տեխնոլոգիական գործընթացի փոքրացմանը զուգընթաց առաջանում են նոր տեսակի երևույթներ, որոնք կարճ կոչվում են ֆիզիկական նախագծի երկրաչափական չափերից կախվածության երևույթներ [16, 23]:

Այս երևույթներից մեկը հարթակի եզրերին տարրերի մոտիկության երևույթն է [43, 16]: Տարրի հեռավորությունը հարթակի եզրից ազդում է տարրի շեմային լարման վրա [27]: Պատճառը իմպլանտի իոններն են, որոնք հարթակի եզրերից ցրվում են և մեծացնում շեմային լարումը մի քանի միլիվոլտից մինչև մի քանի տասնյակ միլիվոլտ: Նկ. 1.28 - ում պատկերված են հարթակից տրանզիստորի հեռավորությունը և շեմային լարման կախվածությունը այդ հեռավորությունից:



ա) *բ)*

Նկ. 1.28. Ա) տարրի հեռավորությունը հարթակի եզրից, բ) շեմային լարման կախվածությունը հարթակի եզրից ունեցած հեռավորությունից

Շեմային լարման փոփոխությունը հանգեցնում է ոչ միայն անհամապատասխանության առաջացման, այլ նաև նախագծի արագագործության զգալի փոփոխության [64, 65]:

Այս երևույթներից խուսափելու համար անհրաժեշտ է.

- օգտագործել նույն չափի դիֆուզիաներ,
- հարթակի եզրից տարրերը տեղադրել մեծ հեռավորությունների վրա,
- կատարել տարրերի համապատասխանեցում,
- կատարել կեղծ տարրերի ներդնում:

1.5. Անալոգային ԻՍ - երում Ֆիզիկական տարրերի նախագծումը

1.5.1. Ռեզիստորների և կոնդենսատորների նախագծումը

Անալոգային ԻՍ-երի ֆիզիկական նախագծման ժամանակ կարևոր նշանակություն ունի նաև ռեզիստորների և ունակությունների ճշգրիտ նախագծումը [53]: Ռեզիստորներն անալոգային ԻՍ-երում նախագծվում են հիմնականում երեք տիպի շերտերից՝ բազմասիլիցիումից, դիֆուզիայից և հարթակից: Քանի որ մետաղի դիմադրությունը շատ փոքր է, ուստի ռեզիստորների նախագծման մեջ այն գրեթե չի օգտագործվում [105]: Աղ. 1.2 - ում բերված են վերը թվարկված նյութերի միավոր հաստության տեսակարար դիմադրությունները, որը կոչվում է նաև քառակուսային դիմադրություն: Կախված երկրաչափական չափերից՝ մարմնի դիմադրությունը որոշվում է հետևյալ բանաձևով [41, 99].

$$R = \rho \frac{L}{S} = \frac{\rho}{t} \frac{L}{W} = R_{քառ} \frac{L}{W}\,, \qquad (1.1)$$

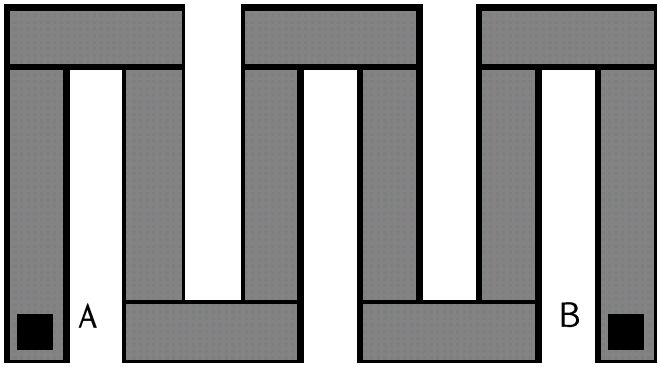
որտեղ ρ -ն նյութի տեսակարար դիմադրությունն է, L-ը՝ ռեզիստորի երկարությունը, S-ը՝ ռեզիստորի մակերեսը, W-ն՝ ռեզիստորի լայնությունը, t-ն՝ հաստությունը, իսկ $R_{քառ}$-ը՝ քառակուսային դիմադրությունը:

Աղյուսակ 1.2

Նյութերի միավոր հաստության տեսակարար դիմադրությունները

Հաղորդիչ	Քառակուսային դիմադրություն	Ճշգրտությունը
Բազմասիլիցիում	20-50 *Օհմ/□*	+/-10%
Դիֆուզիա n+ Դիֆուզիա n-	15-30 *Օհմ/□* 40-200 *Օհմ/□*	+/-5% +/-5%
Հարթակ	1-5 *կՕհմ/□*	+/-30%

Այսպիսով, մեծ դիմադրություններ նախագծելու համար պետք է որպես հաղորդիչ օգտագործել հարթակը, սակայն այն ունի նախագծման ամենամեծ սխալանքը, այդ իսկ պատճառով օգտագործում են բազմասիլիցում կամ դիֆուզիա [41, 93]: Այս դեպքերում *կՕհմ*- երի հասնող ռեզիստորի նախագծման համար կպահանջվի շատ մեծ մակերես: Ռեզիստորի մակերեսի նվազեցման համար օգտագործում են *գալարածև* կառուցվածքներ [4, 41]: Այդպիսի կառուցվածք պատկերված է նկ. 1.29 - ում:



Նկ. 1.29. Ռեզիստորի գալարածև կառուցվածքը

Այս դեպքում պետք է հաշվել A և B կետերի միջև ընդհանուր դիմադրությունը: Ընդհանուր դիմադրությունը հաշվելու համար ռեզիստորը բաժանվում է միավոր երկարությամբ քառակուսիների: Այս պարագայում պետք է հաշվի առնել, որ

անկյուններում միավոր երկարության դիմադրությունն ավելի փոքր է ստացվում: Յուրաքանչյուր անկյուն ավելացնում է ընդհանուր դիմադրությանը 0,5 քառակուսի, իսկ յուրաքանչյուր ելուստ 0,14 քառակուսի չափով:

Բոլոր անալոգային ԻՍ-երում էլ առկա են կոնդենսատորներ [57]: ԻՍ-երում օգտագործվող կոնդենսատորներն իրականացվում են հիմնականում հետևյալ հինգ եղանակով. բազմասիլիցիում - դիֆուզիա, ՄՕԿ, բազմասիլիցիում – բազմասիլիցիում, մետաղ – բազմասիլիցիում, մետաղ – մետաղ: Բազմասիլիցիում – դիֆուզիա տիպի կոնդենսատորները հիմնականում կիրառվում են թվային ԻՍ – երի նախագծերում: Այս տիպի կոնդենստատորներն ապահովում են 5% ճշգրտություն [4, 41]: ՄՕԿ տիպի կոնդենսատորները ՄՕԿ տրանզիստորների մակաբույծ երևույթների արդյունք են, դրանք հիմնականում ունեն փոքր ունակություն և նունյպես օգտագործվում են թվային ԻՍ-երում: Անալոգային ԻՍ-երի նախագծման մեջ հիմնականում օգտագործում են մետաղ – բազմասիլիցիում և մետաղ – մետաղ տիպի կոնդենսատորներ, որոնք պահանջում են մեծ մակերես:

1.5.2. Ռեզիստորների և կոնդենսատորների անհամապատասխանությունները

ԻՍ-երի բոլոր հանգույցներն ենթարկվում են մակրոսկոպիկ շեղումների [7, 54]: Այդ շեղումները լինում են [3, 66].

- եզրային, որոնք տարածվում են նախագծի եզրերում,
- մակերեսային, որոնք տարածվում են նախագծի ամբողջ մակերեսով:

Մակերեսային շեղումները որոշվում են հետևյալ արտահայտությամբ [41].

$$s = m\sqrt{\frac{k}{2A}}\ , \tag{1.2}$$

որտեղ m - ը և s - ը A ակտիվ մակերես ունեցող հանգույցի հետազոտվող պարամետրերի միջին և ստանդարտ շեղումներն են, k – ն անհամապատասխանության գործակիցն է: Համեմատականության k գործակցի արժեքը կախված է անհամապատասխանության աղբյուրների տիպից:

ԻՍ – ի ֆիզիկական նախագծի երկու հանգույցների միջև առկա անհամապատասխանության ստանդարտ շեղումը որոշվում է հետևյալ արտահայտությամբ [4, 41]

$$s_\delta = \sqrt{\left(\frac{s_1}{m_1}\right)^2 + \left(\frac{s_2}{m_2}\right)^2}\ , \qquad (1.3)$$

որտեղ m1 և m2-ը մեզ հետաքրքրող պարամետրի միջին արժեքներն են՝ յուրաքանչյուր սարքի համար, s1 և s2-ը տվյալ պարամետրի ստանդարտ շեղումներն են:

Ռեզիստորների անհամապատասխանության գործակիցը որոշելու համար հետազոտենք L երկարությամբ և W լայնությամբ ուղղանկյունաձև ռեզիստոր

$$A=L*W\ , \qquad (1.4)$$

որտեղ A – ն ռեզիստորի մակերեսն է: Ռեզիստորի R դիմադրությունը որոշվում է հետևյալ արտահայտությամբ [4, 41].

$$R = \frac{L}{W} Rs\ , \qquad (1.5)$$

որտեղ Rs – ը ռեզիստորի քառակուսային դիմադրությունն է:

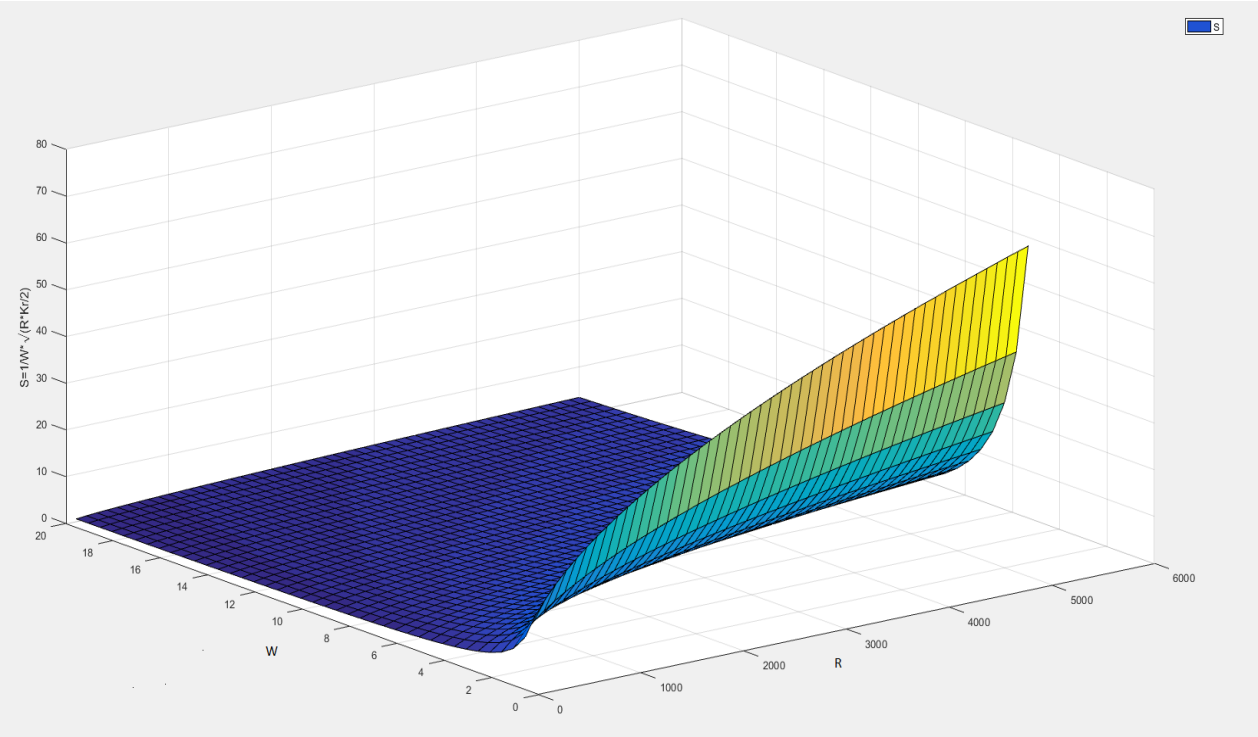
(1.4) -ից և (1.5) -ից կարելի է գրել

$$A = \frac{R}{R_s} W^2\ , \qquad (1.6)$$

օգտվելով (1.6) և (1.2) բանաձևերից, ռեզիստորի ստանդարտ շեղման համար կարող ենք գրել

$$s = \frac{1}{W}\sqrt{\frac{R*k_R}{2}}\ , \qquad (1.7)$$

որտեղ k_R- ը դիմադրության անհամապատասխանության գործակիցն է: Matlab գործիքի օգնությամբ ստացվել է անհամապատասխանության ստանդարտ շեղման 3D մոդելը (նկ. 1.30): Մոդելից երևում է, որ W լայնության նվազումը բերում է անհամապատասխանության զգալի աճի:



Նկ. 1.30. Անհամապատասխանության սրանդարտ շեղման 3D մոդելը

Հավասար անվանականներով ռեզիստորների դեպքում անհամապատասխանության ստանդարտ շեղումը որոշվում է հետևյալ արտահայտությամբ [41].

$$s_\delta = \frac{1}{W}\sqrt{\frac{k_R}{R}} \ , \tag{1.8}$$

բանաձևից պարզ է դառնում, որ անհամապատասխանության ստանդարտ շեղումը հակադարձ համեմատական է ռեզիստորի լայնությանը և դիմադրության քառակուսի արմատին:

Տարբեր անվանականներով ռեզիստորների դեպքում կունենանք [41].

$$s_\delta = \frac{1}{W}\sqrt{\frac{k_R(R_1+R_2)}{2R_1R_2}} : \tag{1.9}$$

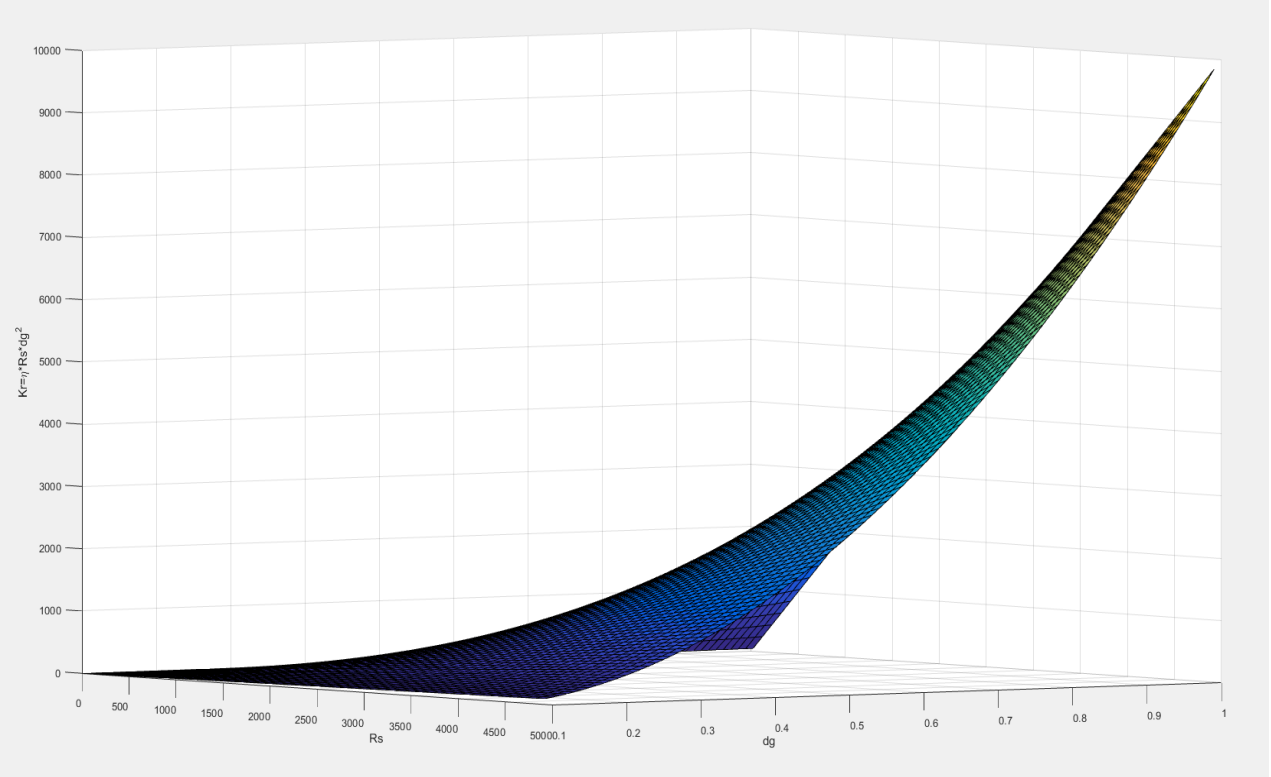
(1.9)– ից պարզ է դառնում, որ տարբեր անվանականներով ռեզիստորների անհամապատասխանությունը կախված է փոքր անվանական ունեցող ռեզիստորի դիմադրության արժեքից:

Կիսահաղորդչային ռեզիստորի k_R անհամապատասխանության գործակցի կախվածությունը ռեզիստորի նյութից որոշվում է հետևյալ արտահայտությամբ [41, 102].

$$k_R = \eta R_s d_g^2 \ , \tag{1.10}$$

որտեղ η-ն հաստատուն է, Rs-ը քառակուսային դիմադրությունն է, d_g-ն կառուցվածքի միջին տրամագիծն է: (1.10) արտահայտությունը գործում է, երբ կառուցվածքի d_g միջին տրամագիծը փոքր է ռեզիստորի լայնությունից: Կայուն համապատասխանեցման համար ռեզիստորի լայնությունը պետք է ընտրել 1 *մկմ* - ից ավելի:

Matlab գործիքի օգնությամբ ստացվել է ռեզիստորի անհամապատասխանության k_R գործակցի 3D մոդելը (նկ. 1.31): Մոդելից երևում է, որ կառուցվածքի միջին տրամագծի աճը բերում է ռեզիստորի անհամապատասխանության գործակցի թռիչքային աճի:



Նկ. 1.31. Ռեզիստրի անհամապատասխանության գործակցի 3D մոդելը

1.6. ԻՍ – երի հանգույցների պարբերական և համակարգված շեղումները

Արտադրության ընթացքում ԻՍ – ի հանգույցների պարամետրերը շեղվում են ֆիզիկական նախագծում տրված պարամետրերից: Այդ տիպի շեղումներին անվանում են տեխնոլոգիական գործընթացով պայմանավորված:

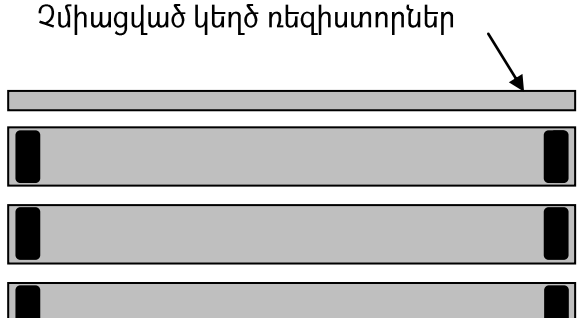
Գործընթացային շեղումները կախված են տարրի երկրաչափական ձևից: Կոնդենսատորների մակերես - պարագիծ հարաբերակցությունը հավասարեցնելով

կարելի է ազատվել արտադրական գործընթացով պայմանավորված շեղումներից: Նույն ունակությունն ունեցող կոնդենսատորների համար ընտրելով միևնույն երկրաչափական պարամետրերը՝ կստանանք արտադրական գործընթացով պայմանավորված նվազագույն շեղում:

Պարբերական այլ անհամապատասխանությունների պատճառ կարող են լինել ֆիզիկական նախագծում ծրագծման համար օգտագործվող լարերը: Ծրագծման լարերի մակաբույծ դիմադրությունները և ունակությունները պետք է լինեն այնքան փոքր, որ դրանցով պայմանավորված շեղումները չազդեն նախագծի պարամետրերի վրա, սակայն ֆիզիկական նախագծի մակաբույծ քաղվածքից երևում է, որ որոշ դեպքերում դրանք ունենում են մեծ արժեքներ և զգալիորեն ազդում են նախագծի պարամետրերի վրա: Այդ ազդեցությունը նվազեցնելու համար կատարվում է նախագծում առկա հանգույցների համապատասխանեցում [41, 94]:

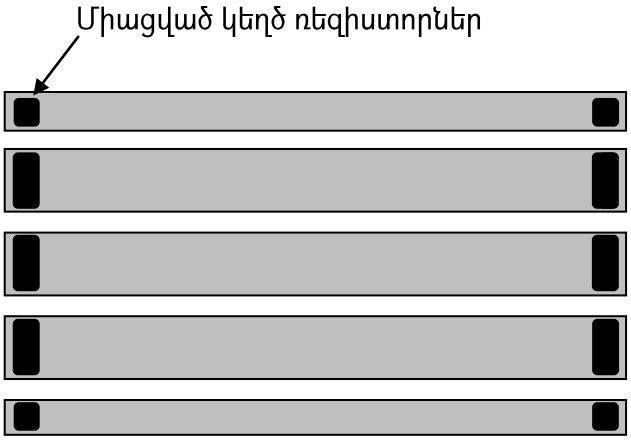
Ֆիզիկական նախագծում ռեզիստորների լավագույն համապատասխանեցման համար, պետք է հաշվի առնել նաև ռեզիստորային մատրիցի սեգմենտների ծրագծման լարերի դիմադրությունները: Դրանց ազդեցությունը հնարավոր է նվազարկել մեծացնելով համապատասխանեցված ռեզիստորի սեգմենտների դիմադրությունը: Լարերի մակաբույծ դիմադրության նվազարկման համար կարելի է նաև կարճացնել ծրագծման լարերի երկարությունները կամ ավելացնել դրանց վրա միջմիացումների քանակը:

ԻՍ-ի արտադրության ժամանակ, արտադրական գործընթացի անկատարելության հետևանքով հանգույցներում առաջանում են խածատման մակարդակի տարբերություններ, շատ դեպքերում դրանք կարող են պարբերական անհամապատասխանությունների պատճառ հանդիսանալ և ազդել նախագծի ելքային պարամետրերի վրա [3, 56]: Խածատման մակարդակի տարբերությունները չեզոքացնելու համար հանգույցի եզրերում ավելացվում են կեղծ ռեզիստորներ: Կեղծ ռեզիստորները կարելի է օգտագործել երկու եղանակով՝ չմիացված (նկ. 1.32) և միացված (նկ. 1.33):



Նկ. 1.32. Չմիացված կեղծ ռեզիստորները

Նկ. 1.32 - ի հիմնական թերությունն այն է, որ չմիացված կեղծ ռեզիստորները կարող են կուտակել լիցքեր, որոնց ստեղծած էլեկտրական դաշտն էլ կազդի մոտ գտնվող հանգույցների պարամետրերի վրա, դրանից խուսափելու համար պետք է կատարել կեղծ ռեզիստորի հողանցում (նկ. 1.33): Անալոգային ԻՍ-երում նախագծվող կոնդենսատորները նույնպես զգայուն են խաճատման մակարդակի տարբերություններով պայմանավորված շեղումների նկատմամբ, հետևաբար կոնդենսատորների դեպքում նույնպես պետք է ավելացնել կեղծ կոնդենսատորներ և իրականացնել համապատասխան միջմիացումներ [41, 106]:



Նկ. 1.33. Միացված կեղծ ռեզիստորները

1.7. Անալոգային ԻՍ-երում ՄՕԿ տրանզիստորների համապատասխանեցման վերլուծությունը

Ֆիզիկական նախագծի արդյունքների բարելավման համար իրականացվում են ՄՕԿ տրանզիստորների համապատասխանեցումներ: Հետազոտենք միևնույն արտաբերի հոսանքով աշխատող, համապատասխանության մեջ դրված երկու ՄՕԿ

տրանզիստորի բնութագրերը: Իդեալական տրանզիստորների դեպքում դրանց փական - ակունք լարումները կլինեն միմյանց հավասար: Անհամապատասխանության առկայության պատճառով պայմանավորված լարումների տարբերությունը որոշվում է հետևյալ բանաձևով [4].

$$\Delta V_{ФԱ} = V_{ФԱ1} - V_{ФԱ2} \ , \qquad (1.11)$$

որտեղ $\Delta V_{ФԱ}$-ն երկու տրանզիստորների միջև գոյություն ունեցող փական - ակունք լարումների տարբերությունն է, իսկ $V_{ФԱ1}$ և $V_{ФԱ2}$ – ը՝ համապատասխանաբար առաջին և երկրորդ տրանզիստորների փական - ակունք լարումները:

Ընդունելով, որ տրանզիստորները գտնվում են հագեցման ռեժիմում, $\Delta V_{□□}$ – ի համար կստացվի

$$\Delta V_{ФԱ} = \Delta V_2 - V_{ФԱ1}\left(\frac{\Delta k}{2k_2}\right) , \qquad (1.12)$$

որտեղ ΔV_2- ն շեմային լարումների միջև գոյություն ունեցող տարբերությունն է, k_2- ը երկրորդ տրանզիստորի հաղորդականությունն է, իսկ Δk - ն՝ տրանզիստորների հաղորդականությունների տարբերությունը: $\Delta V_{ФԱ}$-ն կարելի է նվազարկել՝ կրճատելով $V_{ФԱ1}$ լարումը:

Դիտարկենք ՄՕԿ ԻՍ-երի վարքը, որտեղ կիրառվել է տրանզիստորների հոսանքների համապատասխանեցում: Այս դեպքում արտաբերների հոսանքների անհամապատասխանության համար կստանանք [4].

$$\frac{I_{D2}}{I_{D1}} = \frac{k_2}{k_1}\left(1 + \frac{\Delta V_2}{V_{ФԱ1}}\right) , \qquad (1.13)$$

որտեղ $\frac{I_{D2}}{I_{D1}}$ –ը երկրորդ և առաջին տրանզիստորների արտաբերների հոսանքների հարաբերությունն է: (1.13) - ից պարզ է դառնում, որ $V_{ФԱ1}$-ի նվազումը բերում է անհամապատասխանության աճի, հետևաբար անհամապատասխանության ազդեցությունը նվազարկելու համար տրանզիստորների փականների լարումները պետք է լինեն բավականին մեծ:

ՄՕԿ տրանզիստորների երկրաչափական պարամետրերը ազդում են դրանց համապատասխանեցման վրա: Այդ ազդեցությունը բնութագրելու համար հետազոտենք հոսքուղու երկարության մոդուլյացիայի ազդեցությունը տրանզիստորի պարամետրերի վրա: Հոսքուղու երկարության մոդուլյացիայով պայմանավորված ակունք – արտաբեր հոսանքը որոշվում է [4, 74]

$$I_D = I_{D□□□} * [1 + \lambda(V_{DS} - V_{DS□□□}) , \quad (1.14)$$

որտեղ I_D-ն արտաբերի հոսանքն է, $I_{D□□□}$-ը հագեցման հոսանքն է, V_{DS}-ը արտաբերակունք լարումն է, $V_{DS□□□}$-ը հագեցման լարումն է, իսկ λ-ն՝ հոսքուղու երկարության մոդուլյացիայի գործակիցը, որը որոշվում է հետևյալ արտահայտությամբ [4]

$$\lambda = \frac{1}{□}\frac{dX_{dl}}{dV_{DS}} , \quad (1.15)$$

որտեղ L-ը հոսքուղու երկարությունն է: (1.15) հավասարումից երևում է, որ տրանզիստորի հոսքուղու մոդուլյացիայի երևույթի գործակիցը հակադարձ համեմատական է հոսքուղու երկարությանը, հետևաբար կարճ հոսքուղով տրանզիստորներն ավելի վատ են համապատասխանեցվում, քան երկար հոսքուղիներովը:

ՄՕԿ տրանզիստորների հոսքուղու հաղորդականության անհամապատասխանությունը որոշվում է հետևյալ բանաձևով [4, 41].

$$\frac{S_k}{k} = \frac{C_k}{\sqrt{W_{էֆ}L_{էֆ}}} , \quad (1.16)$$

որտեղ S_k-ն հաղորդականության անհամապատասխանությունն է, k-ն հոսքուղու հաղորդականությունն է, $W_{էֆ}L_{էֆ}$-ը հոսքուղու լայնության և երկարության էֆեկտիվ արժեքների արտադրյալն է, C_k-ն հաստատուն գործակից է:

Տրանզիստորի համապատասխանեցման աստիճանը կախված է տրանզիստորի օքսիդի շերտի հաստությունից. բարակօքսիդները ավելի լավ են համապատասխանեցվում, քան հաստօքսիդ տրանզիստորները [41, 71, 74]:

Ճշգրիտ համապատասխանեցման համար տրանզիստորները պետք է նախագծվեն՝ հաշվի առնելով հետևյալ պայմանները [41]:

- Կողմնորոշում և համընկնում - համապատասխանեցման մեջ դրված հանգույցների սեգմենտների քանակը բոլոր ուղղություններով պետք է լինի նույնը և դրանք պետք է ունենան ընդհանուր կենտրոն:
- Սիմետրիկություն և կոմպակտություն – համապատասխանեցման մեջ դրված հանգույցները պետք է լինեն սիմետրիկ X-երի և Y-ների առանցքների նկատմամբ և ցանկալի է, որ հանգույցն ունենա քառակուսու տեսք:

Անալոգային ԻՍ-երում հանգույցների անհամապատասխանություններ առաջանում են նաև արտաքին տարրերի իրար մոտ դասավորության պատճառով.

- փականին մոտ գտնվող հանգույցները կարող են տրանզիստորի էֆեկտիվ երկարության և լայնության փոփոխություններ առաջացնել,
- հոսքուղու հարևանությամբ տեղակայված դիֆուզիոն շերտերն ազդում են հարթակի լեգիրացման աստիճանի վրա, որը հանգեցնում է տրանզիստորի հաղորդականության և շեմային լարման փոփոխության:

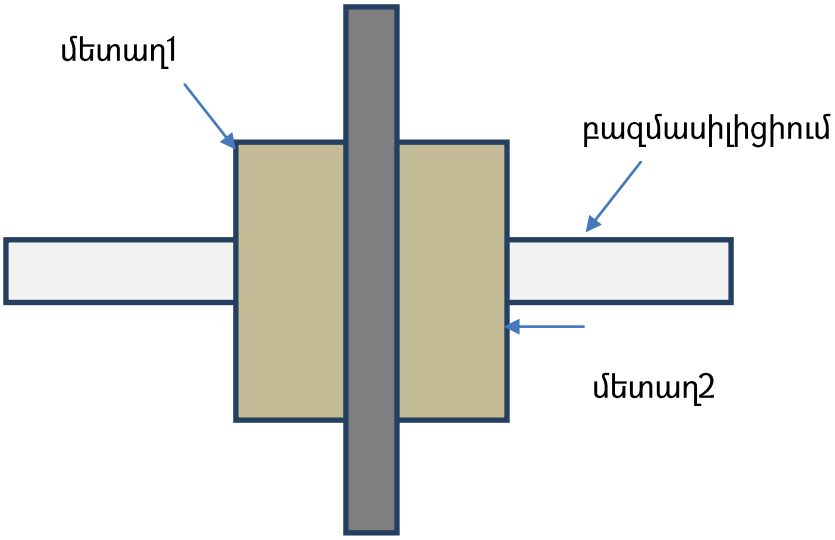
Տեխնոլոգիական գործընթացից կախված ԻՍ-ում տրանզիստորների համապատասխանեցման համար օգտագործվում են կեղծ փականներ, որոնք միացվում են տրանզիստորի ակունքին կամ հարթակին, որը նվազեցնում է էլեկտրական ազդեցությունները:

Հետազոտությունները ցույց են տվել, որ N-ՄՕԿ տրանզիստորները ցուցաբերում են ավելի լավ համապատասխանեցում, քան P-ՄՕԿ տրանզիստորները, այս պատճառով համապատասխանեցման նկատմամբ զգայուն նախագծերում ձգտում են հնարավորինս օգտագործել N-ՄՕԿ տրանզիստորներ:

1.8. Աղմուկի նկատմամբ զգայուն ազդանշանների էլեկտրական պաշտպանություն

ԻՍ-երի ֆիզիկական նախագծերում յուրաքանչյուր հանգույց մնացած հանգույցների համար աղմուկի աղբյուր է, դրանց ազդեցությունը որոշ դեպքերում հնարավոր է անտեսել, սակայն երբ գործ ունենք աղմուկի նկատմամբ զգայուն հանգույցների հետ, ապա անտեսել աղմուկների ազդեցությունը հնարավոր չէ: Հայտնի է, որ աղմուկների նկատմամբ հատկապես զգայուն են անալոգային ԻՍ-երը, և երբ կարիք է առաջանում նույն ԻՍ-ում ինտեգրել թվային և անալոգային հանգույցներ, պետք նկատի ունենալ, որ թվային հանգույցների առաջացրած աղմուկները անխուսափելիորեն կազդեն անալոգային հանգույցների վրա: Այդպիսի ԻՍ-երի նախագծման սկզբնական փուլում պետք է պարզել սխեմայում աղմկոտ և աղմուկի նկատմամբ զգայուն լարերը և դրանք միմյանցից հնարավորին չափով հեռացնել, չպետք է թույլ, տալ որ այդ լարերը ծրագծվեն մեկը մյուսի վրայով: Այն դեպքում, երբ

այդպիսի ծրագծումը անխուսափելի է, պետք է իրականացվի էլեկտրական պաշտպանություն (նկ. 1. 34) [47]:



Նկ. 1. 34. Ֆիզիկական նախագծում էլեկտրական պաշտպանության իրականացումը

Բազմասիլիցիում և մետաղ2 շերտերի միջև էլեկտրական պաշտպանություն իրականացնելու համար դրանց հատման տեղամասում տարվել է մետաղ1 հողանցված շերտը (նկ. 1.34): Մագինիսական դաշտի ազդեցությունից պաշտպանվելու համար պաշտպանիչ շերտը պետք է լարերի հատման մակերեսից ավելի մեծ լինի:

Եզրակացություններ

1. Տեխնոլոգիական գործընթացի փոփոխման և էլեկտրական ագրեսիվ պահանջների պատճառով առաջանում է անալոգային սխեմաների ֆիզիկական նախագծման նոր մոտեցումների անհրաժեշտություն: Ցույց է տրվել, որ բարձր ճշգրտության ֆիզիկական նախագծերում մեծ դեր են խաղում ֆիզիկական նախագծման երևույթները որոնցից են՝ տարրերի համապատասխանեցումը և համաչափությունը, մակաբույծ տարրերը, միջմիացումներում հոսանքի խտությունը և հարթակի երևույթները:

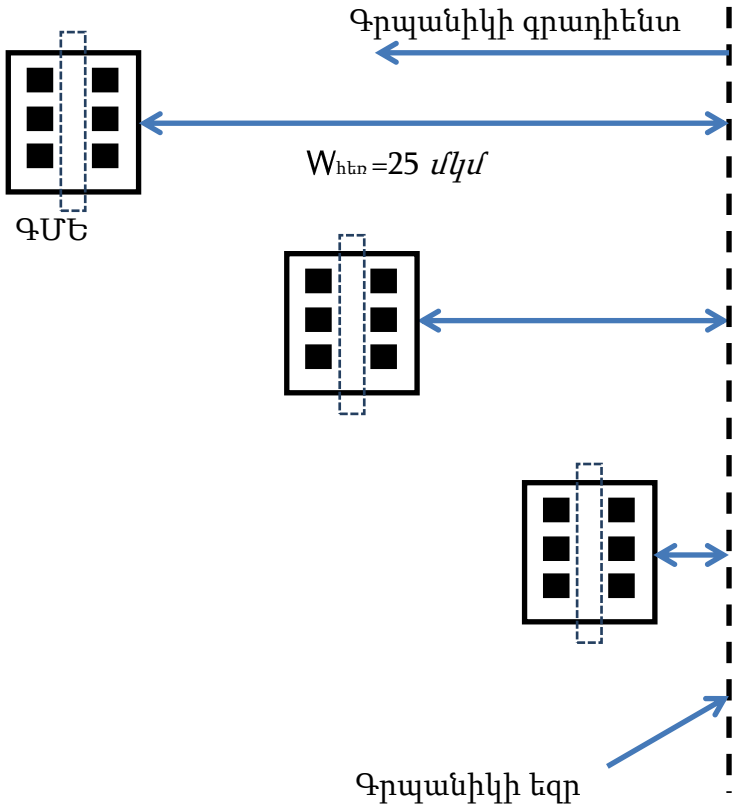
2. Հետազոտվել են ավտոմատ ծրագծման մեթոդները: Ուսումնասիրվել է սահմնափակումների ազդեցությունը անալոգային ԻՍ-երի ավտոմատացված նախագծման վրա:

3. Հետազոտվել են անալոգային ԻՍ-երի ֆիզիկական նախագծման երկրորդական երևույթների ազդեցությունը և դրանց նվազեցման եղանակները: Կատարվել է ջերմային գրադիենտի ազդեցության վերլուծություն ԻՍ-երի ելքային պարամետրերի վրա: Ուսումնասիրվել է պաշտպանիչ օղակների, էկրանավորման և համապատասխանեցման կիրառման ազդեցությունը ֆիզիկական նախագծի ելքային պարամետրերի վրա:

ԳԼՈՒԽ 2. ԱՆԱԼՈԳԱՅԻՆ ԻՆՏԵԳՐԱԼ ՍԽԵՄԱՆԵՐԻ ՖԻԶԻԿԱԿԱՆ ՆԱԽԱԳԾՄԱՆ ԵՐԿՐՈՐԴԱԿԱՆ ԵՐԵՎՈՒՅԹՆԵՐԻ ԱԶԴԵՑՈՒԹՅԱՆ ԹՈՒԼԱՑՄԱՆ ԱՌԱՋԱՐԿՎՈՂ ՄԻՋՈՑՆԵՐԸ

2.1. Գրպանիկի մոտիկության երևույթի ազդեցության քանակական վերլուծությունը

ԳՄԵ-ի ազդեցությունը գնահատելու համար p և n տիպի տրանզիստորներից կազմված նախագիծը տեղադրվել է այնպես, որ տրանզիստորները գրպանիկի եզրից գտնվեն տարբեր հեռավորությունների վրա (նկ. 2.1) [82]:

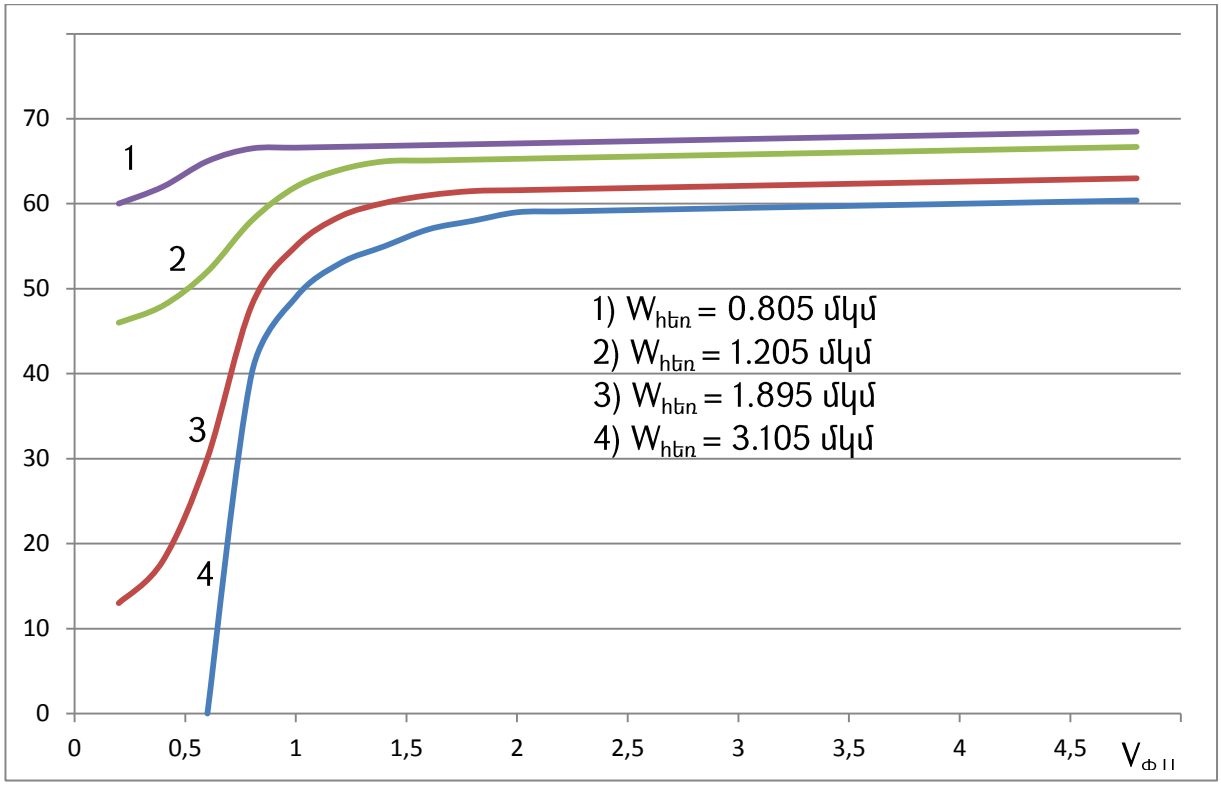


Նկ. 2.1. ԳՄԵ-ի գնահատումը

Միայն մեկ տրանզիստորի վրա ԳՄԵ-ի ազդեցությունը դիտարկելու համար նախագիծն իրականացված է այնպես, որ մյուս տարրերը տրանզիստորից հեռացված են 25 *մկմ*-ով: Փորձնական նախագծում առկա է նաև մեկ այլ՝ գրպանիկի եզրից 25 *մկմ*-ով հեռացված տրանզիստոր՝ որպեսզի համոզվենք, որ այդ տրանզիստորի վրա

ԳՄԵ-ն չի ազդում: Մյուս տրանզիստորների պարամետրերը համեմատվում են այդ տրանզիստորի պարամետրերի հետ:

Հետազոտությունները ցույց են տվել, որ ԳՄԵ - ի ազդեցությունը դառնում է էական գրպանիկի եզրից 2.5 *մկմ* – ից փոքր հեռավորությունների վրա: Տրանզիստորի շեմային լարումը կախված է նաև ակունք/արտաբեր կողմնորոշումից, որն էլ առաջացնում է 10 *մՎ* շեղում: Նմանակումների արդյունքում ստացվել է տրանզիստորի հոսանքի շեղման կախվածությունը փական – ակունք լարումից տարբեր $W_{հեռ}$ հեռավորությունների դեպքում (նկ. 2.2):

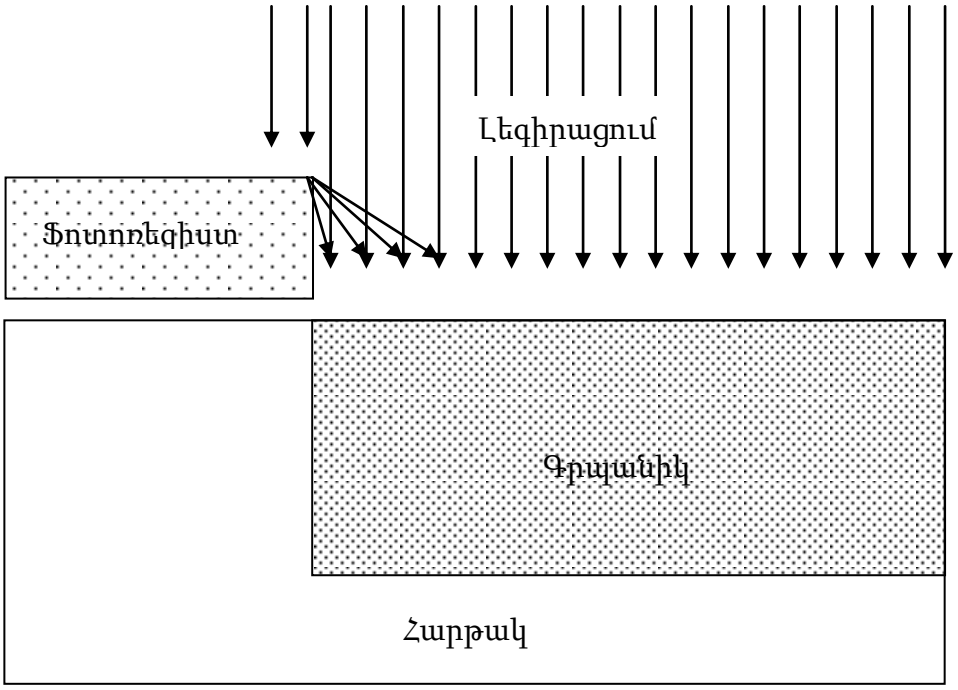


Նկ. 2.2. Տրանզիստորի հոսանքի շեղման կախվածությունը փական – ակունք լարումից տարբեր $W_{հեռ}$ հեռավորությունների դեպքում

2.1.1. Գրպանիկի մոտիկության երևույթի մոդելավորումը

ԻՍ-երի արտադրական գործընթացում գրպանիկներ ստանալու համար օգտագործում են N և P տիպի իոնային իմպլանտացիա, որի ընթացքում իոնները անդրադառնում են ֆոտոռեզիստի եզրերից և փոխում են գրպանիկի կոնցենտրացիան (նկ. 2.3) [43]: Խորը գրպանիկներ ստանալու տեխնոլոգիական գործընթացում

հարկավոր է օգտագործել մեծ էներգիայով իոններ, որոնք ավելի շատ են ցրվում ֆոտոռեզիստի եզրերից և դրանով մեծացնում ԳՄԵ-ի ազդեցությունը [82]:



Նկ. 2.3. Գրպանիկի կոնցենտրացիայի փոփոխությունը լեգիրացման ընթացքում

Հոսքուղու մեծ երկարություն և լայնություն ունեցող ԿՄՕԿ տրանզիստորների շեմային լարումը որոշվում է հետևյալ բանաձևով [4]՝

$$V_{th} = V_{FB} + \Phi_s + \gamma\sqrt{\Phi_s - V_{bs}} = V_{TH0} + \gamma\left(\sqrt{\Phi_s - V_{bs}} - \sqrt{\Phi_s}\right), \quad (2.1)$$

որտեղ V_{FB}–ն հարթակի (flat band) լարումն է, γ-ն՝ հարթակի երևույթի գործակիցը, Φ_s-ը՝ մակերևութային պոտենցիալը, V_{TH0}-ն՝ տարրի շեմային լարումը: ԳՄԵ-ն կարելի է մոդելավորել HSPICE միջավայրում՝ մեծացնելով հարթակի կոնցենտրացիայի SCA, SCB, SCC պարամետրերը: Տրանզիստորի շեմային լարման փոփոխությունը որոշվում է հետևյալ արտահայտությամբ [4, 82]՝

$$\mathrm{d}V_{th} = K_{VTH0WE} * (SCA + W_{EB} * SCB + W_{EC} * SCC), \quad (2.2)$$

որտեղ SCA, SCB, SCC պարամետրերը կախված են ՄՕԿ տրանզիստորի երկրաչափական չափերից և տոպոլագիական իրականացումից: SCA, SCB, SCC պարամետրերը որոշվում են հետևյալ արտահայտություններով [4, 82]՝

$$SCA = \frac{1}{L} SC_{ref}^2 D\left(\frac{1}{SC_L} - \frac{1}{SC_L + L}\right) + \frac{1}{W} SC_{ref}^2 \left(\frac{1}{SC_W} - \frac{1}{SC_W + W}\right), \quad (2.3)$$

$$SCB = \frac{1}{L}\left(\frac{SC_L}{10} + \frac{SC_{ref}}{100}\right) exp\left(-10\frac{SC_L}{SC_{ref}}\right) - \frac{1}{L}\left(\frac{SC_L+L}{10} + \frac{SC_{ref}}{100}\right) exp\left(-10\frac{SC_L+L}{SC_{ref}}\right) +$$

$$\frac{1}{W}\left(\frac{SC_W}{10} + \frac{SC_{ref}}{100}\right) exp\left(-10\frac{SC_W}{SC_{ref}}\right) - \frac{1}{W}\left(\frac{SC_W+W}{10} + \frac{SC_{ref}}{100}\right) exp\left(-10\frac{SC_W+W}{SC_{ref}}\right) , \quad (2.4)$$

$$SCC = \frac{1}{L}\left(\frac{SC_L}{20} + \frac{SC_{ref}}{400}\right) exp\left(-20\frac{SC_L}{SC_{ref}}\right) - \frac{1}{L}\left(\frac{SC_L+L}{20} + \frac{SC_{ref}}{400}\right) exp\left(-20\frac{SC_L+L}{SC_{ref}}\right) + \frac{1}{W}\left(\frac{SC_W}{20} +$$

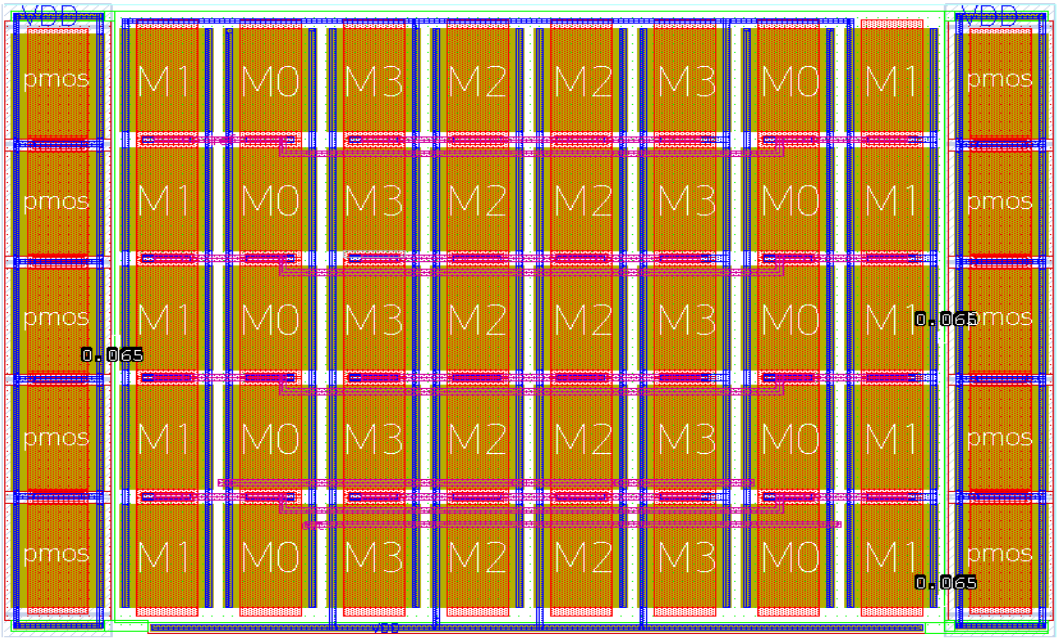
$$\frac{SC_{ref}}{400}\right) exp\left(-20\frac{SC_W}{SC_{ref}}\right) - \frac{1}{W}\left(\frac{SC_W+W}{20} + \frac{SC_{ref}}{400}\right) exp\left(-20\frac{SC_W+W}{SC_{ref}}\right) , \quad (2.5)$$

որտեղ L և W-ն համապատասխանաբար ՄՕԿ տրանզիստորի հոսքուղու երկարությունը և լայնությունն են, SC_W և SC_L-ն հոսքուղու հեռավորություններն են գրպանիկի եզրից: Ստացված պարամետրերը տեղադրելով dVth-ի բանաձևում՝ ստանում ենք շեմային լարման շեղման արժեքը:

2.1.2. ԳՄԵ-ի ազդեցության վերլուծությունը հոսանքի հայելու ֆիզիկական նախագծում

ԳՄԵ-ի ուսումնասիրության համար ընտրվել է հոսանքի հայելու սխեման, որը լայնորեն կիրառվում է անալոգային ԻՍ-երում: Հոսանքի հայելու տրանզիստորները պետք է գտնվեն միևնույն պայմաններում, որպեսզի նրա երկու ճյուղերով անցնի նույն մեծության հոսանք: Հետևաբար, հոսանքի հայելու ֆիզիկական նախագիծը գերզգայուն է տոպոլոգիական իրականացման նկատմամբ: SAED32/28 *նմ* տեխնոլոգիական գործընթացի համար հետազոտվել է ԳՄԵ-ի ազդեցությունը P-ՄՕԿ տրանզիստորի պարամետրերի վրա: SAED32/28 *նմ* տեխնոլոգիական գործընթացի համար իրականացվել է հոսանքի հայելու ֆիզիկական նախագիծը (նկ. 2.4) [33]: Նախ M1 տրանզիստորի դիֆուզիոն շերտը հեռացվել է գրպանիկի եզրից 25 *մկմ*-ով (ԳՄԵ չկա), և նմանակումների արդյունքում ստացվել են տրանզիստորի ելքային հոսանքի և շեմային լարման արժեքները, ապա $W_{հեռ}$ -ը փոքրացվել է մինչև տեխնոլոգիական գործընթացով սահմանված նվազագույն հեռավորությունը (0,065 *մկմ)*: Մեծացնելով հեռավորությունը 0,05 *մկմ* քայլով՝ կատարվել են նմանակումներ և ստացվել՝ յուրաքանչյուր դեպքում M0 տրանզիստորի ելքային հոսանքի և շեմային լարման արժեքները:

Նմանակումների արդյունքում ցույց է տրվել, որ տրանզիստորի շեմային լարման և հոսանքի փոփոխությունը էական է դառնում 1 *մկմ*, 0,7 *մկմ*, 0,4 *մկմ*, 0,28 *մկմ*, 0,17 *մկմ* և 0,065 *մկմ* հեռավորությունների դեպքում [107]:



Նկ. 2.4. SAED32/28 նմ տեխնոլոգիայում P-ՄՕԿ տրանզիստորներով իրականացված հոսանքի հայելու տոպոլոգիան

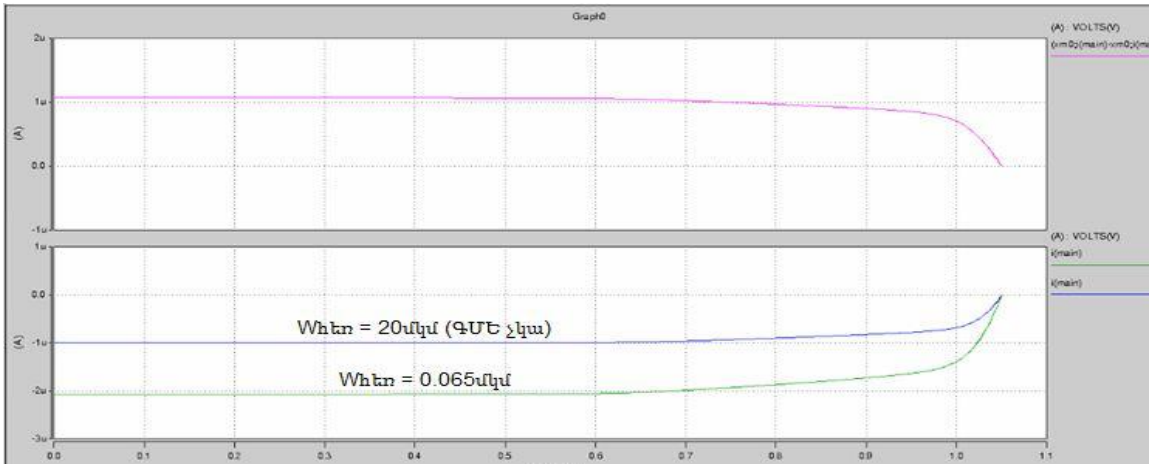
Աղ. 2.1.-ում բերված են *P-ՄՕԿ* տրանզիստորի շեմային լարման և ելքային հոսանքի շեղումները տարբեր $W_{հեռ}$ հեռավորությունների դեպքում [107]:

Աղյուսակ 2.1

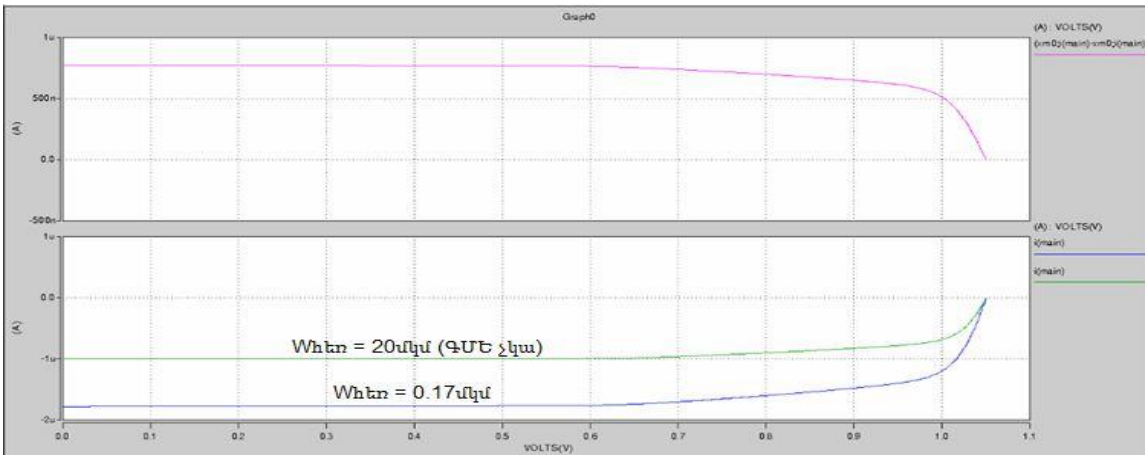
P-ՄՕԿ տրանզիստորի շեմային լարման և ելքային հոսանքի շեղումները

$W_{հեռ}$ (*մկմ*)	Շեմային լարման շեղում (%)	Ելքային հոսանքի շեղում (%)
0,065	10	52,3
0,17	8	44,4
0,28	3	17,3
0,4	1,5	11,5
0,7	0,01	0,06
1	0,01	0,04

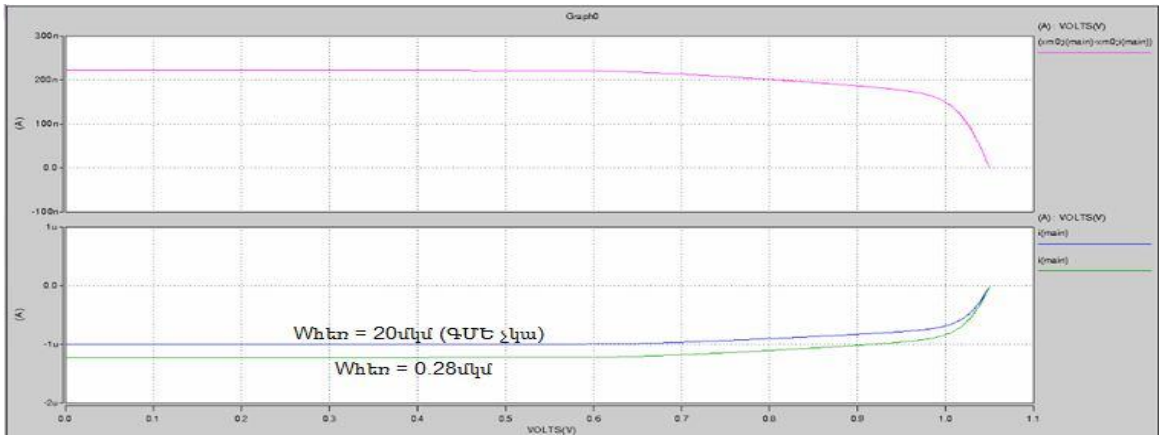
Նմանակումների հիման վրա ստացվել են M0 տրանզիստորի հոսանքի շեղումները $W_{հեռ}$ – ի տարբեր արժեքների դեպքում (նկ. 2.5 – 2.8):



Նկ. 2.5. SAED32/28 նմ տեխնոլոգիայում M0 տրանզիստորի հոսանքի շեղումը $W_{հեռ}$ = 0,065 մկմ հեռավորության դեպքում



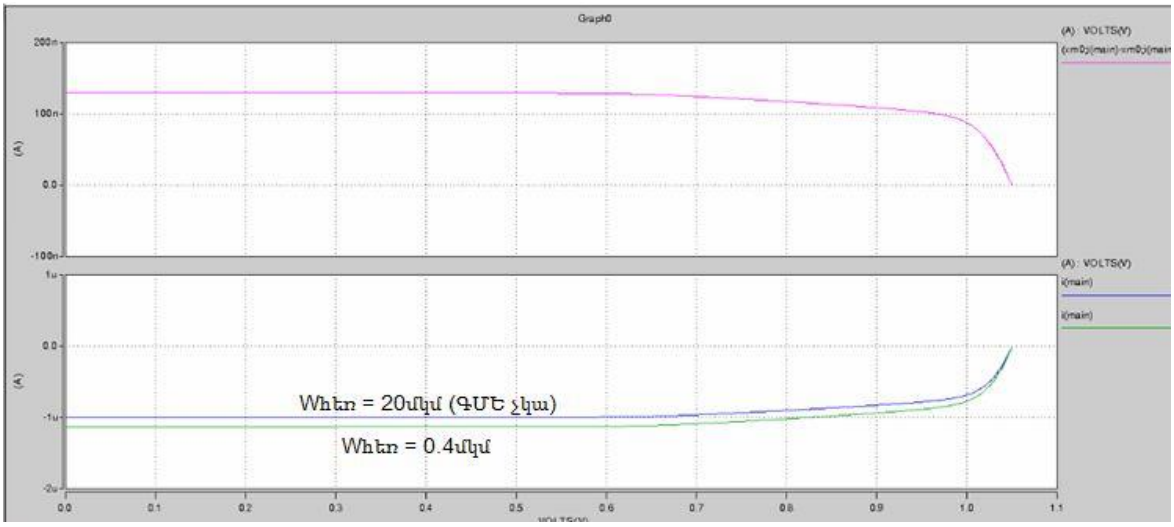
Նկ. 2.6. SAED32/28 նմ տեխնոլոգիայում M0 տրանզիստորի հոսանքի շեղումը $W_{հեռ}$ = 0,17 մկմ հեռավորության դեպքում



Նկ. 2.7. SAED32/28 նմ տեխնոլոգիայում M0 տրանզիստորի հոսանքի շեղումը

$W_{հեռ}$ = 0,28 մկմ հեռավորության դեպքում

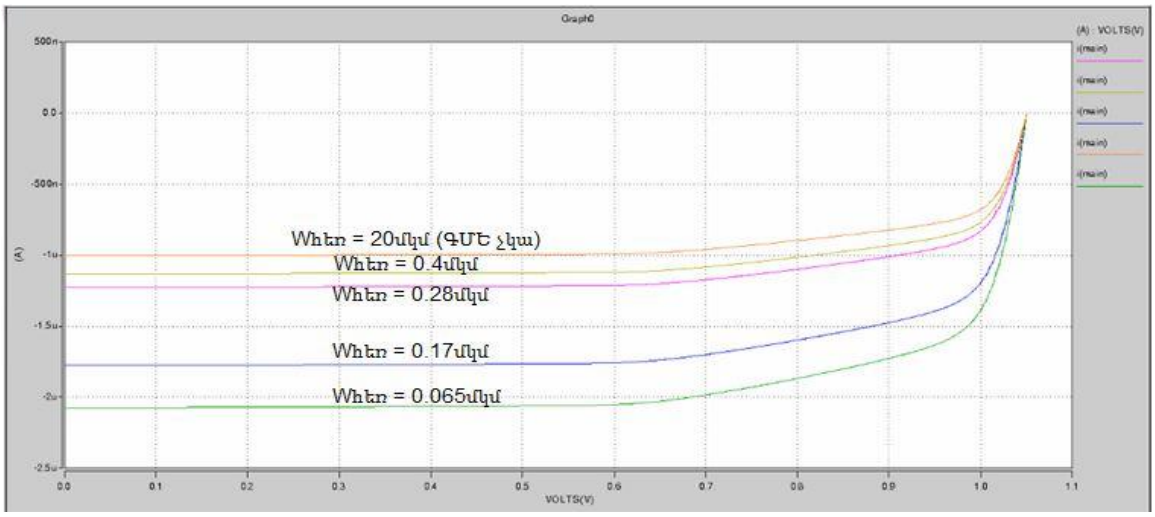
Ստացված արդյունքներից երևում է, որ SAED32/28 *նմ* տեխնոլոգիական գործընթացի համար գրպանիկի մոտիկության երևույթը կարող է փոփոխել շեմային լարումը առավելագույնը 10%-ով, իսկ տրանզիստորի հոսանքը՝ մոտավորապես 52%-ով:



Նկ. 2.8. SAED32/28 նմ տեխնոլոգիայում M0 տրանզիստորի հոսանքի շեղումը

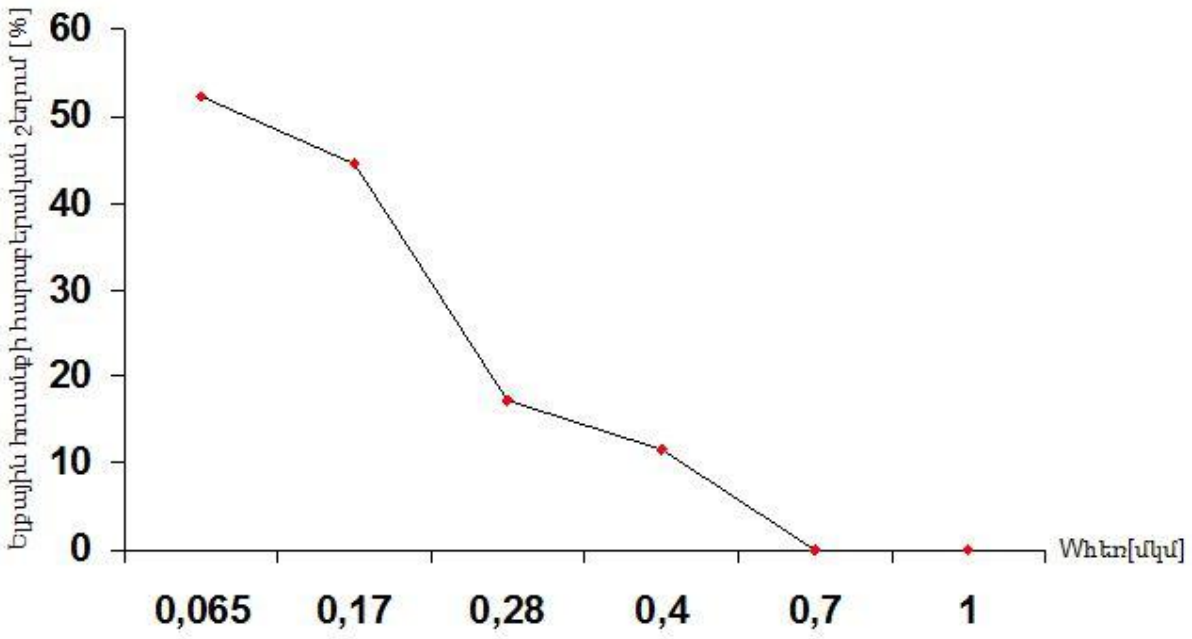
$W_{հեռ}$ = 0,4 մկմ հեռավորության դեպքում

Նկ. 2.9 – ում բերված են M0 տրանզիստորի հոսանքի շեղումները $W_{հեռ}$ հեռավորության տարբեր արժեքների դեպքում:

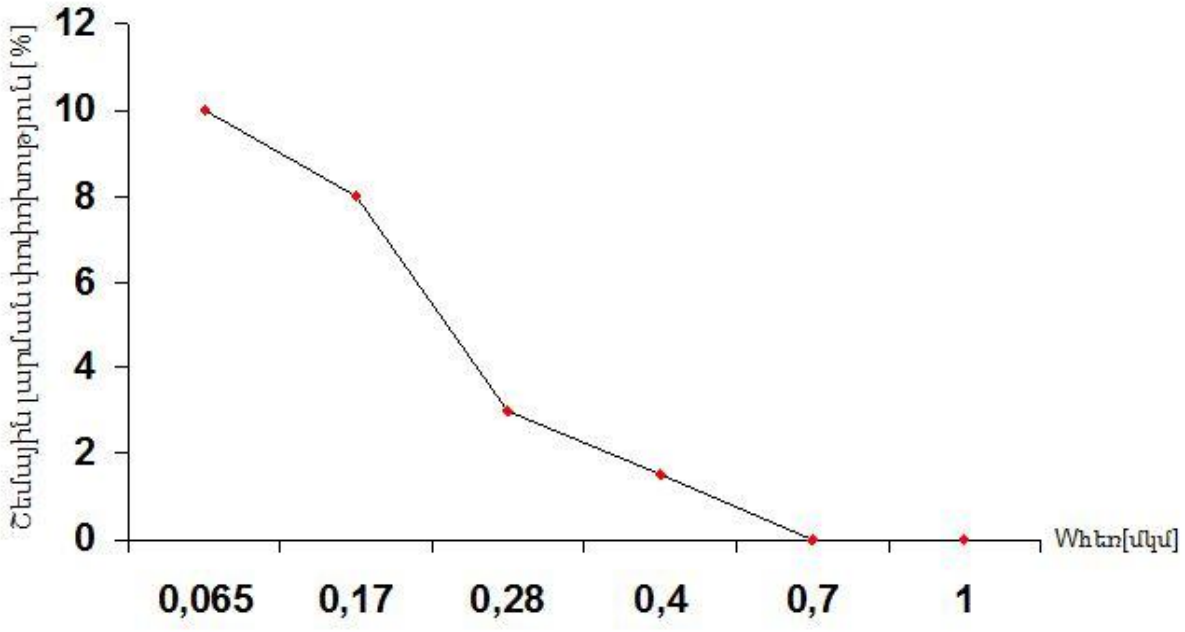


Նկ. 2.9. SAED32/28 նմ տեխնոլոգիայում M0 տրանզիստորի հոսանքի շեղումը $W_{հեռ}$ = 0,065 մկմ, 0,17 մկմ, 0,28 մկմ, 0,4 մկմ հեռավորությունների դեպքում

Հետազոտվել է գրպանիկի մոտիկության երևույթի ազդեցության հետևանքով հոսանքի հարաբերական շեղման (նկ. 2.10) և շեմային լարման փոփոխության (նկ. 2.11) կախվածությունը գրպանիկ – դիֆուզիա հեռավորությունից՝ ներկայացված գրաֆիկի տեսքով:



Նկ. 2.10. SAED32/28 նմ տեխնոլոգիայում հոսանքի հայելու M0 տրանզիստորի ելքային հոսանքի հարաբերական շեղման կախվածությունը գրպանիկ – դիֆուզիա հեռավորությունից



Նկ. 2.11. SAED32/28 նմ տեխնոլոգիայում M0 տրանզիստորի շեմային լարման փոփոխության կախվածությունը գրպանիկ – դիֆուզիա հեռավորությունից

Նմանակումների արդյունքում ցույց է տրվել, որ $W_{հեռ}$ = 0,7 *մկմ* հեռավորության դեպքում M0 տրանզիստորի շեմային լարման և ելքային հոսանքի փոփոխությունը չնչին է: $W_{հեռ}$ – ի հետագա մեծացման դեպքում (0,7 *մկմ* – ից մեծ) շեմային լարման և ելքային հոսանքի փոփոխությունը, համեմատած 0,7 *մկմ* հեռավորության դեպքի հետ, գրեթե ձգտում է զրոյի:

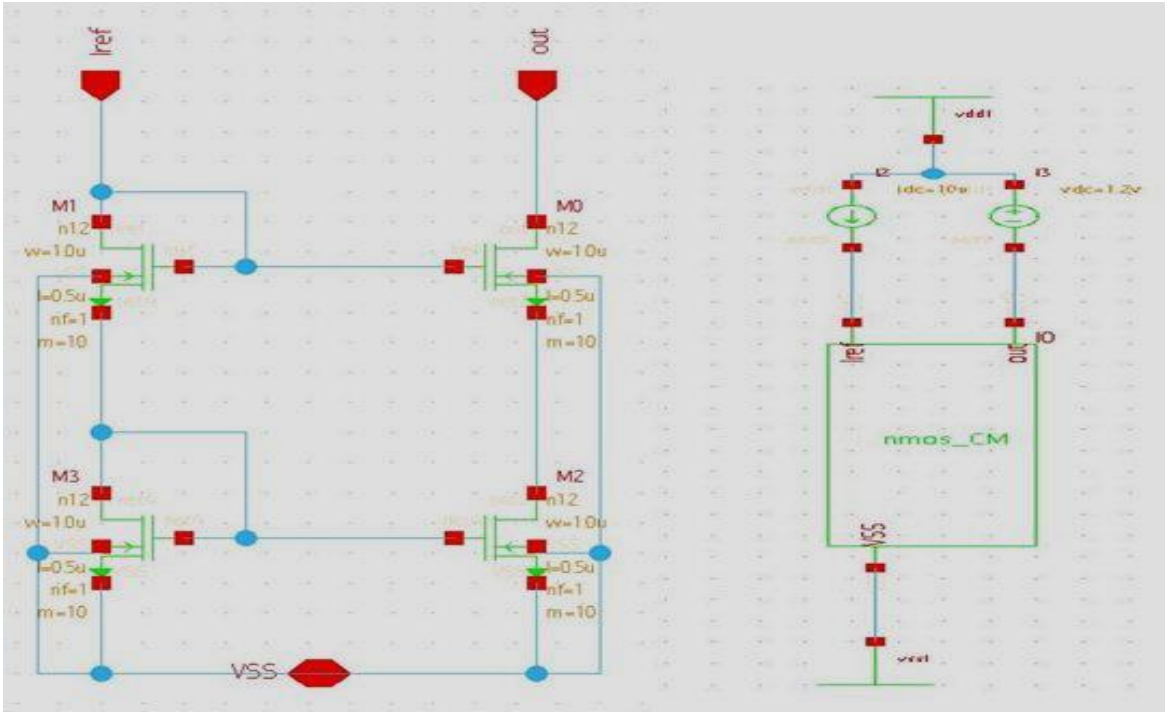
Ստացված արդյունքները վկայում են, որ SAED32/28 *նմ* տեխնոլոգիական գործընթացի համար ԳՄԵ – ի ազդեցությունից խուսափելու համար գրպանիկ – դիֆուզիա հեռավորությունը պետք է լինի 0,7 *մկմ* և ավել, քանի որ դրանից մեծ հեռավորությունների դեպքում տրանզիստորի պարամետրերի փոփոխություն գրեթե չի նկատվում, իսկ ավելի փոքր հեռավորությունների դեպքում նկատվում է տրանզիստորի պարամետրերի էական փոփոխություն:

2.1.3. ԳՄԵ-ի նվազեցման առաջարկված եղանակի հետազոտումը

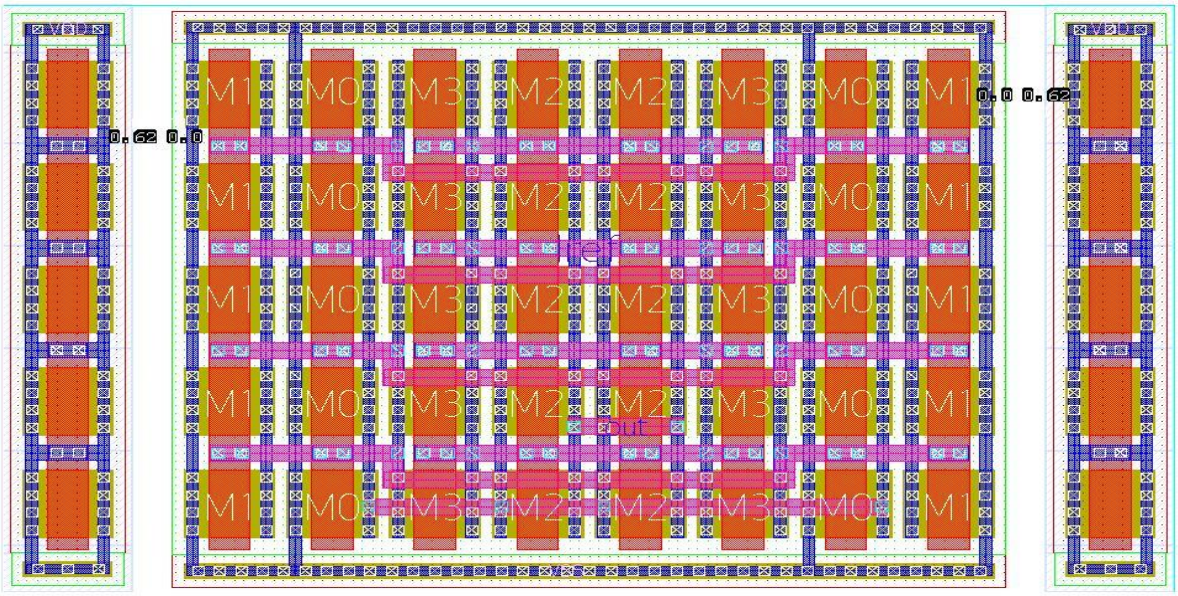
P-ՄՕԿ տրանզիստորներով իրականացված հոսանքի հայելու սխեմայի ելքային արդյունքներից պարզ է դառնում, որ գրպանիկի մոտիկության երևույթը մեծ ազդեցություն ունի ելքային հոսանքի և տրանզիստորի շեմային լարման վրա: Այդ

ազդեցությունը նվազեցնելու նպատակով առաջարկվում է հոսանքի հայելին նախագծել զուտ N-ՄՕԿ տրանզիստորներով [108]:

N–ՄՕԿ տրանզիստորների հիման վրա կառուցված հոսանքի հայելու սխեման բերված է նկ. 2.12 - ում իսկ տոպոլոգիական իրականացումը՝ նկ. 2.13 - ում: Սկզբում ֆիզիկական նախագծում M1 տրանզիստորի դիֆուզիան գրպանիկի եզրերից հեռացվել է 25 *մկմ*-ով (ԳՄԵ չկա), և չափվել են տրանզիստորի շեմային լարման և ելքային հոսանքի արժեքները: Այնուհետև $W_{հեռ}$ - ը ընդունել է 0,065 *մկմ* արժեքը (տեխնոլոգիական նվազագույն թույլատրելի հեռավորություն), մեծացվել 0,01 *մկմ* քայլով, և յուրաքանչյուր հեռավորության համար դիտարկվել է ԳՄԵ-ի ազդեցությունը M0 տրանզիստորի պարամետրերի վրա: Փորձը ցույց է տալիս, որ $W_{հեռ}$ = 0,2 *մկմ* հեռավորության դեպքում M0 տրանզիստորի շեմային լարման և ելքային հոսանքի փոփոխությունը չնչին է: $W_{հեռ}$ – ի հետագա մեծացման դեպքում (0,2 *մկմ* – ից մեծ) շեմային լարման և ելքային հոսանքի փոփոխությունը՝ 0,2 *մկմ* հեռավորության դեպքի համեմատ, գրեթե ձգտում է զրոյի:



Նկ. 2.12. SAED32/28 նմ տեխնոլոգիայում N-ՄՕԿ տրանզիստորներով իրականացված հոսանքի հայելու սխեման և թեստավորող բլոկը



Նկ. 2.13. SAED32/28 նմ տեխնոլոգիական գործընթացի համար հոսանքի հայելու տոպոլոգիական իրականացումը

Նմանակումների արդյունքների վերլուծությունից ստացվում է, որ SAED32/28 *նմ* տեխնոլոգիական գործընթացում ԳՄԵ – ի ազդեցությունից խուսափելու համար գրպանիկ – դիֆուզիա հեռավորությունը պետք է լինի առնվազն 0,2 *մկմ*. ավելի մեծ հեռավորությունների դեպքում տրանզիստորի պարամետրերի փոփոխություն գրեթե չի նկատվում [108]:

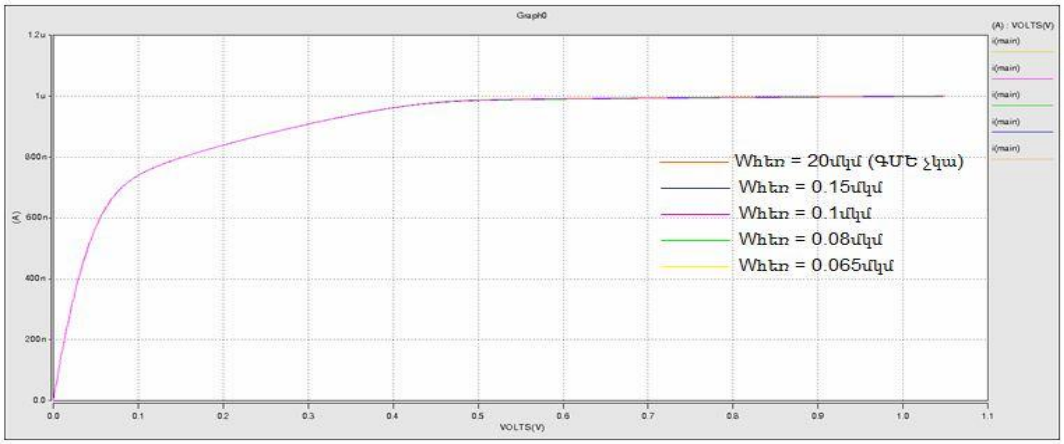
Աղ. 2.2 - ում բերված է N-ՄՕԿ տեխնոլոգիաներով իրականացված հոսանքի հայելու շեմային լարման և ելքային հոսանքի շեղումները տարբեր $W_{հեռ}$ հեռավորությունների դեպքում:

Աղյուսակ 2.2

M0 տրանզիստորի շեմային լարման և ելքային հոսանքի շեղումների կախվածությունը $W_{հեռ}$ հեռավորությունից N-ՄՕԿ տեխնոլոգիայի դեպքում

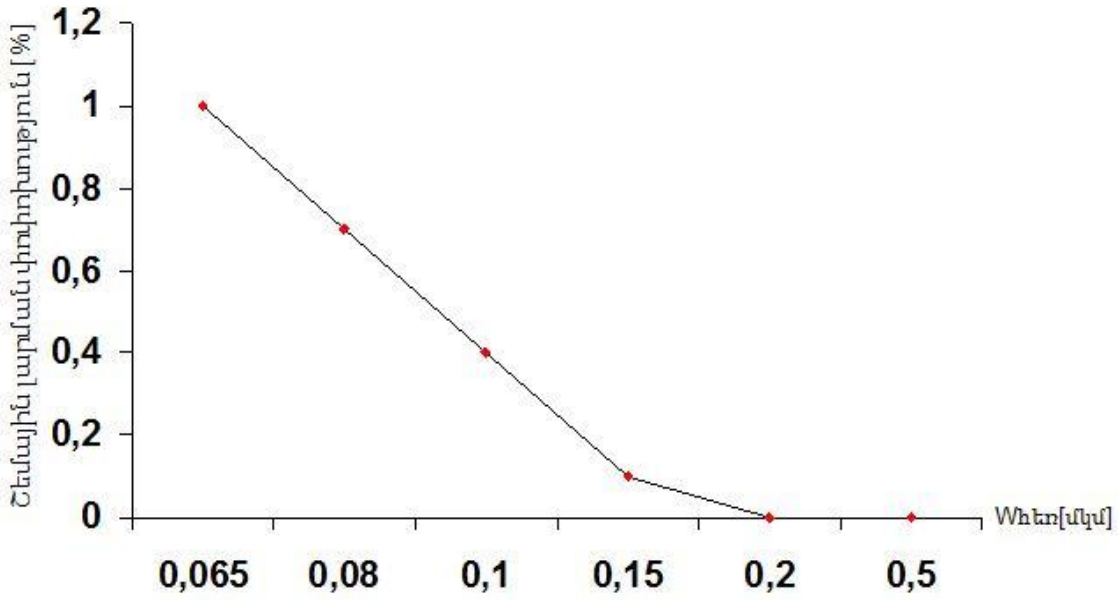
$W_{հեռ}$ (*մկմ*)	Շեմային լարման շեղում (%)	Ելքային հոսանքի շեղում (%)
0,065	1	0,23
0,17	0,1	0,03
0,28	0,01	0,0005
0,4	0,01	0,0004

Նմանակումների արդյունքում ցույց է տրվել, որ SAED32/28 *նմ* տեխնոլոգիական գործընթացի դեպքում գրպանիկի մոտիկության երևույթը կարող է փոփոխել շեմային լարումը առավելագույնը 1%-ով, իսկ տրանզիստորի հոսանքը՝ մոտավորապես 0,23%-ով (աղ. 2.2): Նկ. 2.14-ում բերված են M0 տրանզիստորի հոսանքի շեղումները $W_{հեռ}$ հեռավորության տարբեր արժեքների դեպքում:

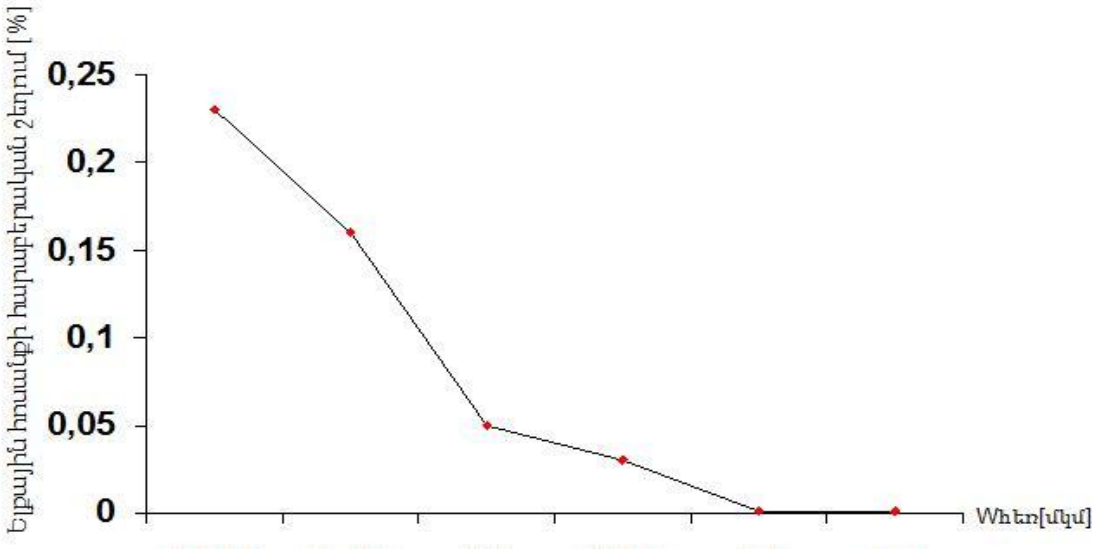


Նկ. 2.14. SAED32/28 նմ տեխնոլոգիայում M0 տրանզիստորի հոսանքի շեղումը $W_{հեռ}$ = 0,065 մկմ, 0,08 մկմ, 0,1 մկմ, 0,15 մկմ հեռավորությունների դեպքում

Նկ. 2.15 և 2.16-ում ներկայացված են գրպանիկի մոտիկության երևույթի ազդեցության հետևանքով շեմային լարման փոփոխության և հոսանքի հարաբերական շեղման կախվածությունները գրպանիկ – դիֆուզիա հեռավորությունից:



Նկ. 2.15. SAED32/28 նմ տեխնոլոգիայում MO տրանզիստորի շեմային լարման փոփոխության կախվածությունը գրպանիկ – դիֆուզիա հեռավորությունից



Նկ. 2.16. SAED32/28 նմ տեխնոլոգիայում հոսանքի հայելու MO տրանզիստորի ելքային հոսանքի հարաբերական շեղման կախվածությունը գրպանիկ – դիֆուզիա հեռավորությունից

Հոսանքի հայելու իրականացված N-ՄՕԿ և P-ՄՕԿ ֆիզիկական նախագծերի հետազոտության արդյունքում հաստատվել է, որ ֆիզիկական նախագծման ժամանակ գրպանիկի մոտիկության երևույթի ազդեցությունը կարելի է նվազեցնել 10…100 անգամ՝ կիրառելով N-ՄՕԿ տեխնոլոգիա:

2.2. Ֆիզիկական նախագծի տարրերի անհամապատասխանությունների նվազեցման մշակված եղանակը

Անալոգային ԻՍ-երի հանգույցների էլեկտրական հատկությունները խիստ կախված են ջերմաստիճանից [63]:

Հետազոտենք ջերմային գրադիենտի ազդեցությունը ԻՍ-երի հանգույցների վրա: Ջերմային երևույթով պայմանավորված s_δ անհամապատասխանությունը որոշվում է [41, 101]

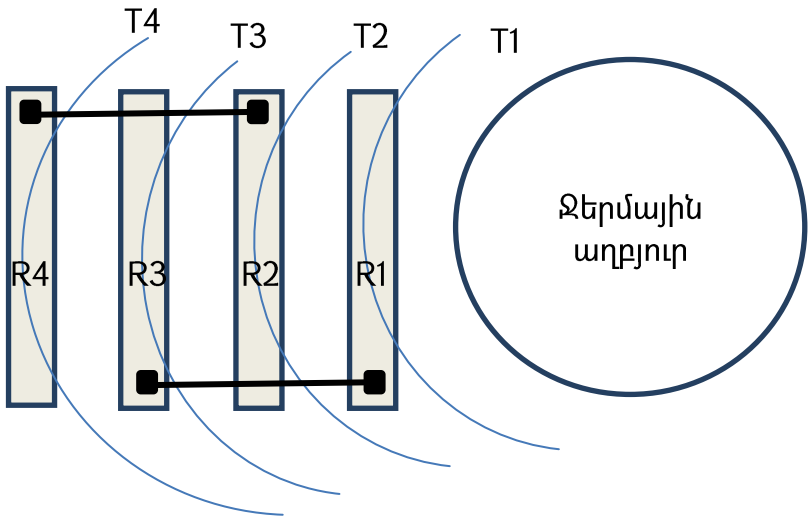
$$s_\delta = TC_1 d_{cc} \nabla T_{cc} \ , \qquad (2.6)$$

որտեղ TC_1-ը տվյալ նյութի դիմադրության գծային ջերմաստիճանային գործակիցն է, d_{cc}-ն՝ ռեզիստորի հեռավորությունը կենտրոնից, ∇T_{cc}- ն՝ կենտրոնը ռեզիստորին միացնող գծի երկայնքով ջերմային գրադիենտը:

Ռեզիստորները ենթարկում են երկու միմյանցից տարբեր ջերմային երևույթների ազդեցության: Ստորև հետազոտվել են այդ երկու ջերմային երևույթները և առաջարկվել դրանց լուծման եղանակներ: Առաջին երևույթը պայմանավորված է նյութի դիմադրության ջերմաստիճանային գործակցով, որի ազդեցությունը հնարավոր է նվազարկել՝ կատարելով ճշգրիտ համապատասխանեցում: Հետազոտենք ջերմային աղբյուրի ազդեցությունը համապատասխանեցման մեջ դրված ռեզիստորների պարամետրերի վրա (նկ 2.17):

Նկ. 2.17 - ում բերված է ջերմային աղբյուր և միմյանցից միևնույն d_{cc} հեռավորությունների վրա գտնվող իզոթերմեր: Քանի որ իզոթերմերի ջերմաստիճանային կախվածությունը աղբյուրից ունեցած հեռավորությունից գծային է, ուստի նկարում բերված իզոթերմերի միջև ΔT ջերմաստիճանային տարբերությունները նույնպես կլինեն իրար հավասար ՝

$$T4 - T3 = T3 - T2 = T2 - T1 = \Delta T: \qquad (2.7)$$



Նկ. 2.17. Դիմադրությունների XYXY համապատասխանեցմամբ սեկցիաների բաշխումը իզոթերմերի վրա

Ընդունենք՝ ջերմային աղբյուրի շրջակայքում ջերմաստիճանը $T_{□□□}$ - է, ուրեմն՝ (2.7)-ից

$$T1 = T_{□□□} - \Delta T,\ T2 = T_{□□□} - 2\Delta T,\ T3 = T_{□□□} - 3\Delta T,\ T4 = T_{□□□} - 4\Delta T: \quad (2.8)$$

Համապատասխանեցված ռեզիստորը բաժանված է սեկցիաների այնպես, որ RA դիմադրության երկու սեկցիան գտնվեն T4 և T2 իզոթերմերի վրա, իսկ RB դիմադրության սեկցիաները՝ T3 և T1 իզոթերմերի վրա (նկ. 2.17): Մետաղական ռեզիստորի դիմադրության կախումը ջերմաստիճանից ունի հետևյալ տեսքը [99, 101]:

$$R = R_0 * (1 + \alpha(T - T_0)) , \quad (2.9)$$

որտեղ R_0-ն ռեզիստորի դիմադրությունն է T_0 ջերմաստիճանում, α-ն մետաղի ջերմաստիճանային գործակիցն է:

(2.9)-ից կստանանք՝

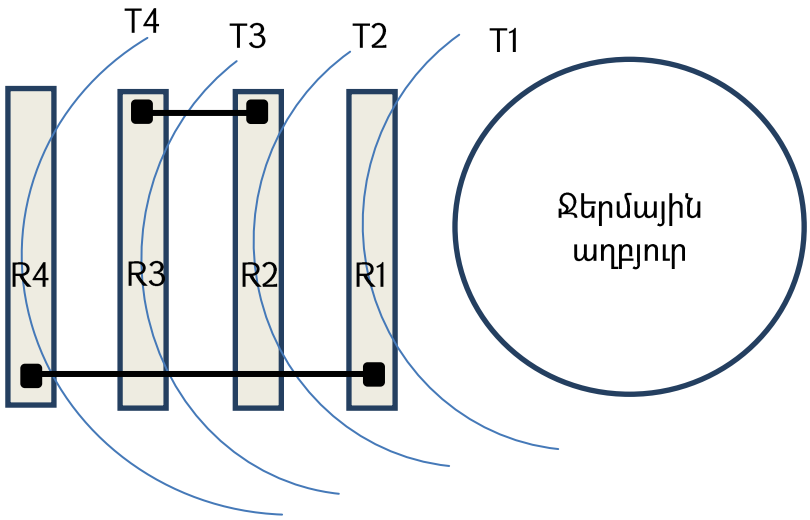
$$RA = R2 + R4 = R_0 * \left(1 + \alpha(T2 - T_0)\right) + R_0 * \left(1 + \alpha(T4 - T_0)\right) , \quad (2.10)$$

$$RB = R1 + R3 = R_0 * \left(1 + \alpha(T1 - T_0)\right) + R_0 * \left(1 + \alpha(T3 - T_0)\right) : \quad (2.11)$$

(2.7) – (2.11) - ից համապատասխանեցման մեջ գտնվող RB և RA դիմադրությունների տարբերության համար ստացվում է՝

$$RB - RA = 2R_0\alpha\Delta T : \quad (2.12)$$

(2.12) - ից երևում է, որ համապատասխանեցման մեջ դրված ռեզիստորների միջև կա դիմադրության տարբերություն, որը անալոգային ԻՍ-երում կարող է բերել անցանկալի արդյունքների: Ջերմաստիճանային գրադիենտով պայմանավորված համապատասխանեցման մեջ դրված ռեզիստորների դիմադրությունների տարբերությունը զրոյացնելու համար մեր կողմից առաջարկվել է ԻՍ-երում տարրերի համապատասխանեցման նոր եղանակ (նկ. 2.18):



Նկ. 2.18. Դիմադրությունների XYYX համապատասխանեցմամբ սեկցիաների բաշխումը իզոթերմերի վրա

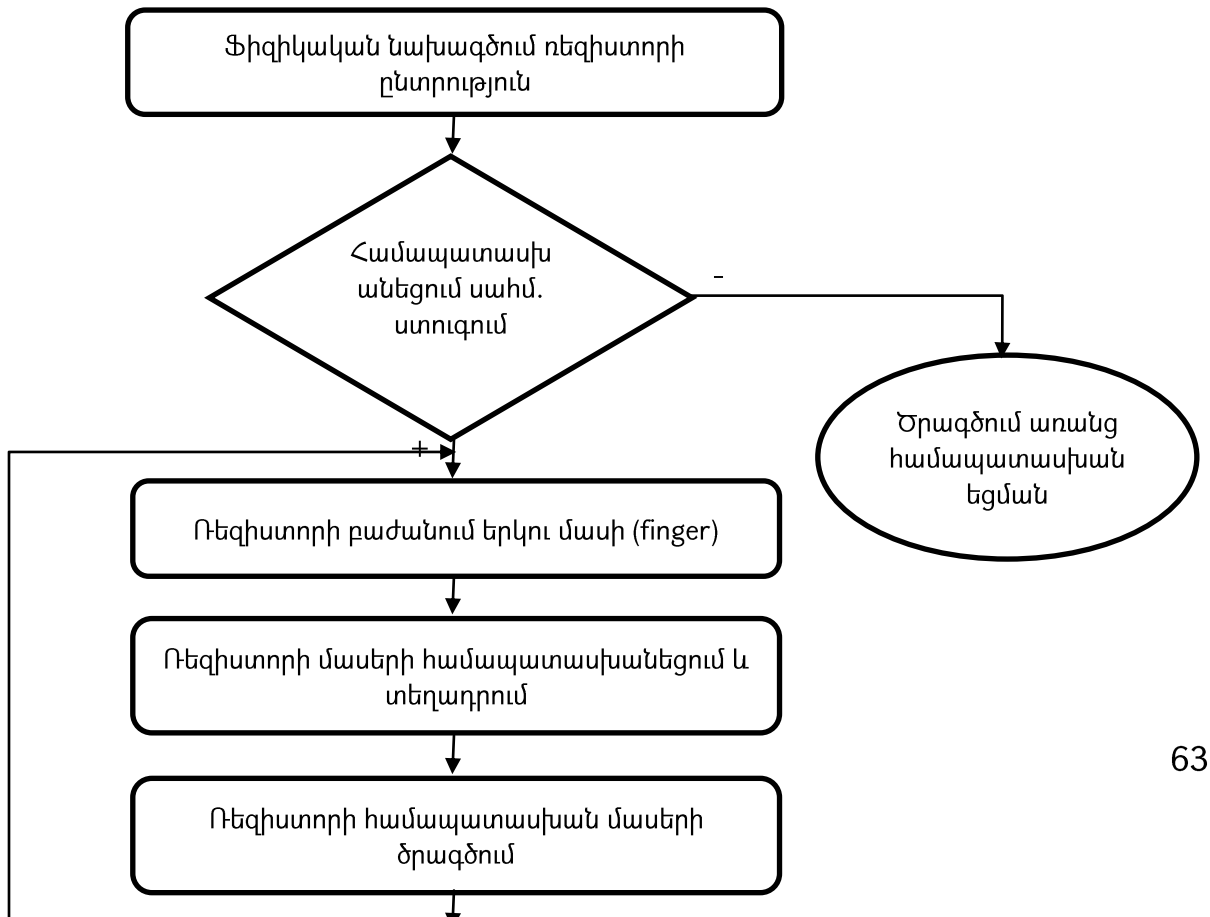
Առաջարկված եղանակում ռեզիստորների համապատասխանեցումը կատարվում է հետևյալ կերպ. RA դիմադրության երկու սեկցիաները գտնվում են T4 և T1 իզոթերմերի վրա, իսկ RB դիմադրության սեկցիաները՝ T3 և T2 իզոթերմերի վրա (նկ. 2.18):

Այս դեպքում օգտվելով (2.7) – (2.11) բանաձևերից՝ համապատասխանեցման մեջ գտնվող RB և RA դիմադրությունների տարբերության համար ստացվում է՝

$$RB - RA = 0: \tag{2.13}$$

(2.13) – ը ցույց է տալիս, որ կիրառելով առաջարկված մեթոդը, համապատասխանեցման մեջ դրված ռեզիստորների ջերմային գրադիենտով պայմանավորված դիմադրությունների տարբերությունը հավասարվում է զրոյի:

Տարրերի անհամապատասխանության նվազեցման մշակված ալգորիթմը և քայլերի հաջորդականությունը նախագծված AAR Designer և Constraint designer գործիքային միջավայրում բերված են նկ. 2.19 - ում:



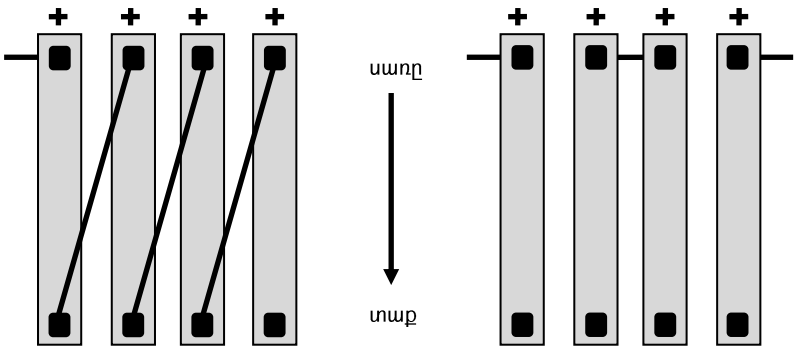
Նկ. 2.19. Անհամապատասխանեցման նվազեցման մշակված ալգորիթմը

ԻՍ-երում տրանզիստորների պարամետրերի վրա ազդող հաջորդ ջերմային երևույթը Սիբեկի երևույթն է, այն անվանում են նաև էլեկտրաջերմային երևույթ: Դրա էությունը հետևյալն է. ռեզիստորի ելուստների ջերմաստիճանների տարբերությունը հանգեցնում է ռեզիստորում պոտենցիալների տարբերության: Այդ տարբերությունը որոշվում է հետևյալ արտահայտությամբ [101, 41].

$$E_Ջ = S\Delta T_Կ \ , \qquad (2.14)$$

որտեղ S-ը Սիբեկի երևույթի գործակիցն է, իսկ $\Delta T_Կ$-ն՝ ռեզիստորի ելուստների ջերմաստիճանների տարբերությունը:

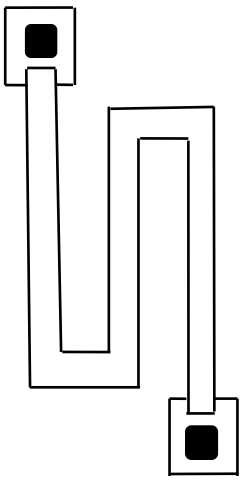
Ըստ նախագծվող ռեզիստորի տոպոլոգիայի, ջերմաէլեկտրական պոտենցիալը կարող է աճել՝ ձևավորելով յուրաքանչյուր առանձին սեգմենտից շատ ավելի մեծ ընդհանուր ջերմաէլեկտրական պոտենցիալ (նկ. 2.20 ա), կամ նվազել (նկ. 2.20 բ): Զույգ թվով ելուստների դեպում ջերմաէլեկտրական պոտենցիալը կնգտի զրոյի:



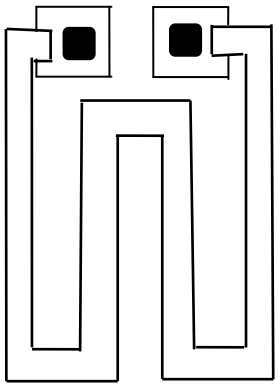
ա) բ)

Նկ. 2.20. Ջերմաէլեկտրական պոտենցիալի *ազդեցությունը ռեզիստորի վրա. ա)մեծ ջերմաէլեկտրական պոտենցիալ, բ)փոքր ջերմաէլեկտրական պոտենցիալ*

Անալոգային ԻՍ-երում օգտագործում են գալարաձև ռեզիստորներ (նկ. 2.21): Ջերմաէլեկտրական պոտենցիալի ազդեցությունը նվազարկելու համար դրանց ելուստները տեղադրվում են որքան հնարավոր է իրար մոտ (նկ. 2.22):



Նկ. 2.21 Գալարաձև ռեզիստորի տոպոլոգիան



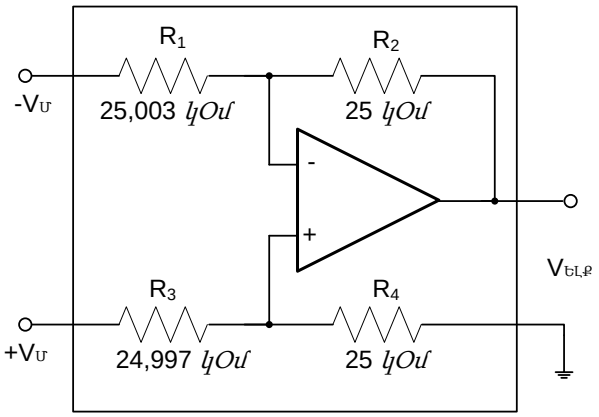
Նկ. 2.22 Ռեզիստորի ջերմային ազդեցության վերացման տոպոլոգիան

Ռեզիստորի ելուստների միջև ջերմային տարբերությունը ստացվում է մեծ (նկ. 2.21), ջերմաէլեկտրական երևույթները նվազարկելու համար ռեզիստորի ելուստները տեղադրվում են իրար մոտ (նկ. 2.22):

2.2.1. Ֆիզիկական նախագծում տարրերի անհամապատասխանության ազդեցության գնահատումը

Հետազոտվել են ռեզիստորների և կոնդենսատորների համապատասխանեցման հետ կապված արդի խնդիրները և դրանց լուծման եղանակները [109]:

Շղթայի տարրերի համապատասխանեցման անհրաժեշտությունը գնահատելու համար դիտարկվել է օպերացիոն ուժեղարարի սխեման (նկ. 2.23):



Նկ.2.23. Շղթայի տարրերի համապատասխանեցումը

Ենթադրվում է, որ ունենք իդեալական օպերացիոն ուժեղարար, և դիտարկվել են շղթայի ելքային պարամետրերը, երբ անհամապատասխանեցման հետևանքով R_1 և R_2 դիմադրությունները շեղված են անվանականից:

Ուժեղարարի ելքային լարումը որոշվում է հետևյալ բանաձևով [100, 104]

$$V_{\text{ելք}}=\frac{(R_1+R_2)R_4}{(R_3+R_4)R_1}V_{Մ}-\frac{R_2}{R_1}V_{Մ}: \qquad (2.15)$$

Մուտքային 10 Վ լարման դեպքում ուժեղարարի ելքային լարման կախվածությունը R_1 և R_3 դիմադրությունների անվանականից ունեցած շեղումից բերված է աղ. 2.3.-ում:

Աղյուսակ 2.3

Ուժեղարարի ելքային լարման կախվածությունը R_1 և R_3 դիմադրությունների անվանականից

Դիմադրությունների շեղում անվանականից	1%	0,1%	0,012%
Սխալանք, *մՎ*	99	9,9	1,2

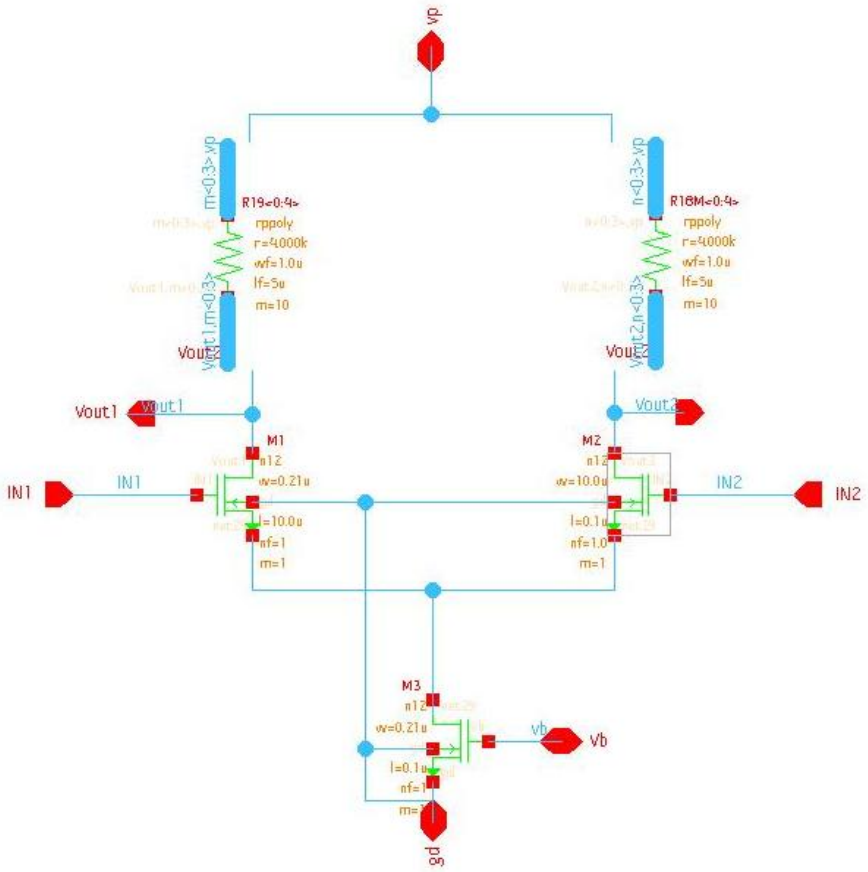
Դիտարկվել են հետևյալ դեպքերը, երբ դիմադրություններն ունեն 1%, 0,1%, 0,012% շեղում:

Ստացված արդյունքներից պարզ է դառնում, որ նույնիսկ դիմադրությունների 0,1% շեղման դեպքում ունենում ենք 9,9 *մՎ* լարման սխալանք: Այս տիրույթի սխալանքը անթույլատրելի է անալոգային ԻՍ-երի մեծ մասի համար: Հետևաբար, դիմադրությունների համապատասխանեցումը ԻՍ ֆիզիկական նախագծման կարևորագույն փուլերից մեկն է [109]:

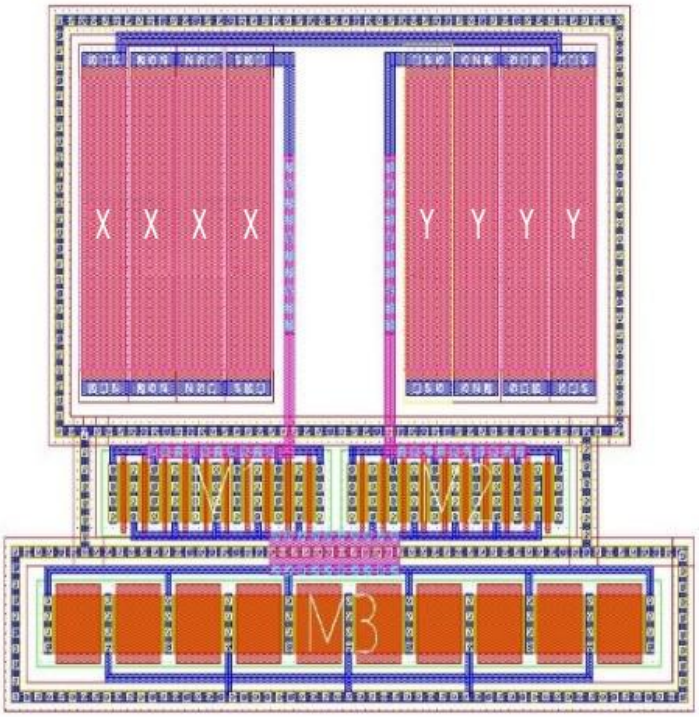
2.2.2. Տարրերի համապատասխանեցման առաջարկված եղանակի փորձնական հիմնավորումը

Անալոգային ԻՍ-երում տարրերի համապատասխանեցման առաջարկված եղանակի քանակական վերլուծության համար իրականացվել է դիֆերենցիալ ուժեղարարի սխեմատեխնիկական նախագիծ (նկ. 2.24), և ստացվել են ուժեղարարի հիմնական բնութագրական պարամետրերը [36, 74, 61]:

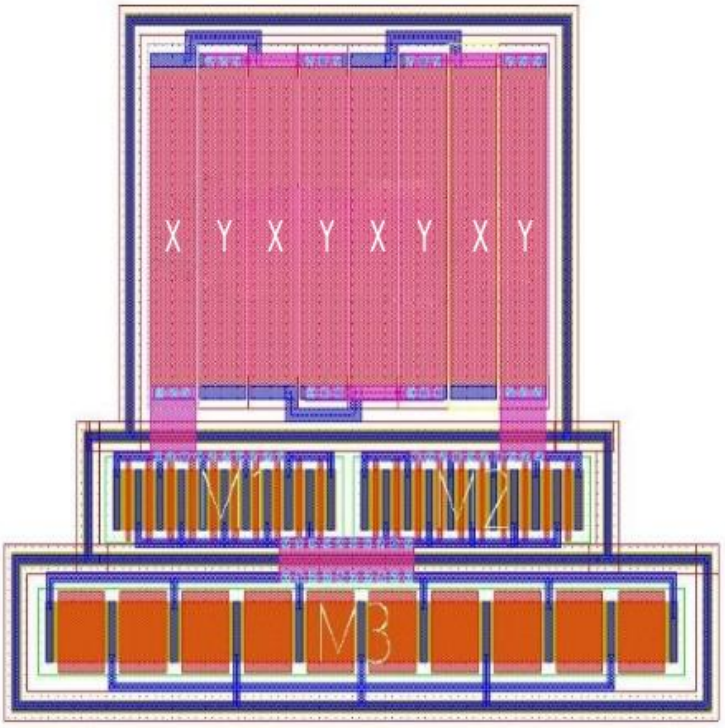
Իրականցվել է դիֆերենցիալ ուժեղարարի երեք ֆիզիկական նախագիծ՝ տարբեր մոտեցումներով. համապատասխանեցում չի կիրառվել (նկ. 2.25), կիրառվել է XYXY ռեզիստորային համապատասխանեցում (նկ. 2.26), կիրառվել է առաջարկված XYYX համապատասխանեցումը (նկ. 2.27) [112]:



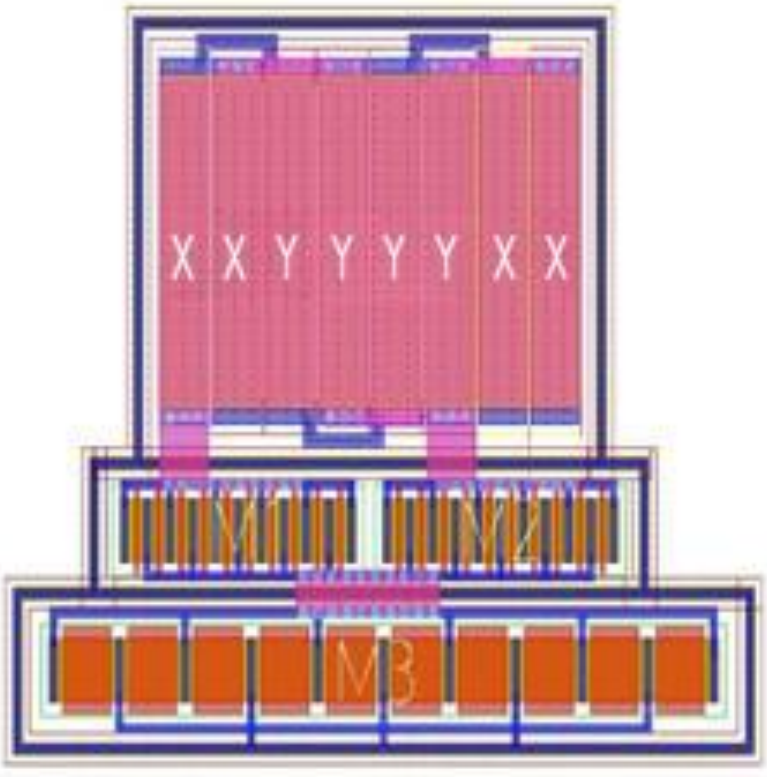
Նկ. 2.24. Դիֆերենցիալ ուժեղարարի սխեման



Նկ. 2.25. Դիֆերենցիալ ուժեղարարի ֆիզիկական նախագիծը առանց համապատասխանեցման



Նկ. 2.26. Դիֆերենցիալ ուժեղարարի ֆիզիկական նախագիծը XYXY համապատասխանեցմամբ



Նկ. 2.27. Դիֆերենցիալ ուժեղարարի ֆիզիկական նախագիծը XYYX համապատասխանեցմամբ

Փորձարարական հետազոտությունների արդյունքում ստացված տվյալները ներկայացված են աղ. 2.4.-ում:

Աղյուսակ 2.4

Համապատասխանեցման ազդեցությունը ֆիզիկական նախագծի վրա

Հիմնական պարամետրեր	Համապատասխանեցման տեսակը		
	Առանց	XYXY	XYYX
Սնման Լարում (*Վ*)	1,8		
Ուժեղացման գործակից (*դբ*)	26,2	35,7	40,2
Մուտքային շեղում (*մՎ*)	15	3	1,8
ΔR (*Օհմ*) (R1 և R2 դիմադրությունների տարբերությունը)	10,7	2,4	1,5

Աղ. 2.4-ից երևում է, որ առաջարկված համապատասխանեցման դեպքում դիմադրությունների միջև տարբերությունը ավանդական համապատասխանեցման համեմատ նվազում է 35%-ով, իսկ ուժեղարարի մուտքային շեղումը նվազում է 3 *մվ* – ից մինչև 1,8 *մվ*: Ստացված արդյունքները վկայում են համապատասխանեցման մշակված եղանակի բարձր արդյունավետության մասին:

2.3. Ֆիզիկական նախագծի ելքային պարամետրերի վրա պաշտպանիչ օղակների կիրառման ազդեցության վերլուծությունը

Անալոգային ԻՍ-երի մեջ կարևորագույն հանգույցներից են փուլահաճախական ինքնահամալարման համակարգերը (ՓԻՀ): ՓԻՀ-երը օգտագործվում են ազդանշանների դեմոդուլյացիայի, աղմուկից ազդանշանի տարանջատման, մուտքային հաճախության պատիկ ելքային հաճախության գեներացման և այլ գործողությունների համար [49]: Դրանց նախագծումը բարդ և ժամանակատար գործողություն է:

ՓԻՀ-երը հիմնականում կիրառվում են արագագործ սխեմաներում, որը բերում է դրան սխեմատեխնիկական նախագծման բարդացմանը և ֆիզիկական նախագծման

ճշգրտության վրա դրվող պահանջների խստացմանը: Նշված պատճառով անհրաժեշտ է ՓԻՀ-երի ֆիզիկական նախագծման ժամանակ հաշվի առնել երկրորդական երևույթներով պայմանավորված ՓԻՀ-ի պարամետրերի փոփոխությունը: Համակարգի կարևորագույն պարամետրերն են թրթռոցը և փուլի աղմուկը, որոնք էական դեր ունեն տվյալների փոխանցման ժամանակ: Թրթռոցով է պայմանավորված մուտք-ելք հանգույցներում տվյալների հաղորդման ճշտությունը: ՓԻՀ-երի վրա դրվող պահանջներից է.

- լայն հաճախական տիրույթներում աշխատանքի հնարավորություն,
- ելքային հաճախության նվազագույն շեղումներ,
- ցածր ծախսվող հզորություն,
- դինամիկ ծրագրավորվող և այլն:

Վերը թվարկված պահանջները բավարարելու համար ՓԻՀ-երը պետք է նախագծվեն օգտագործելով անալոգային ԻՍ-երի նախագծման հայտնի բոլոր մեթոդները, որոնք կբերեն ՓԻՀ-երի պարամետրերի լավացմանը: Քանի որ ՓԻՀ-ում առկա են ինչպես թվային, այնպես էլ անալոգային հանգույցներ, որոնք գերզգայուն են աղմուկների նկատմամբ, ապա ՓԻՀ-ում առկա բոլոր հանգույցներում պետք է կիրառել պաշտպանիչ օղակներ: ՓԻՀ-ում աղմուկների նկատմամբ ամենազգայուն հանգույցը փուլահաճախական դետեկտորն է, հետևաբար այն պետք է առնել պաշտպանիչ օղակի մեջ [111]:

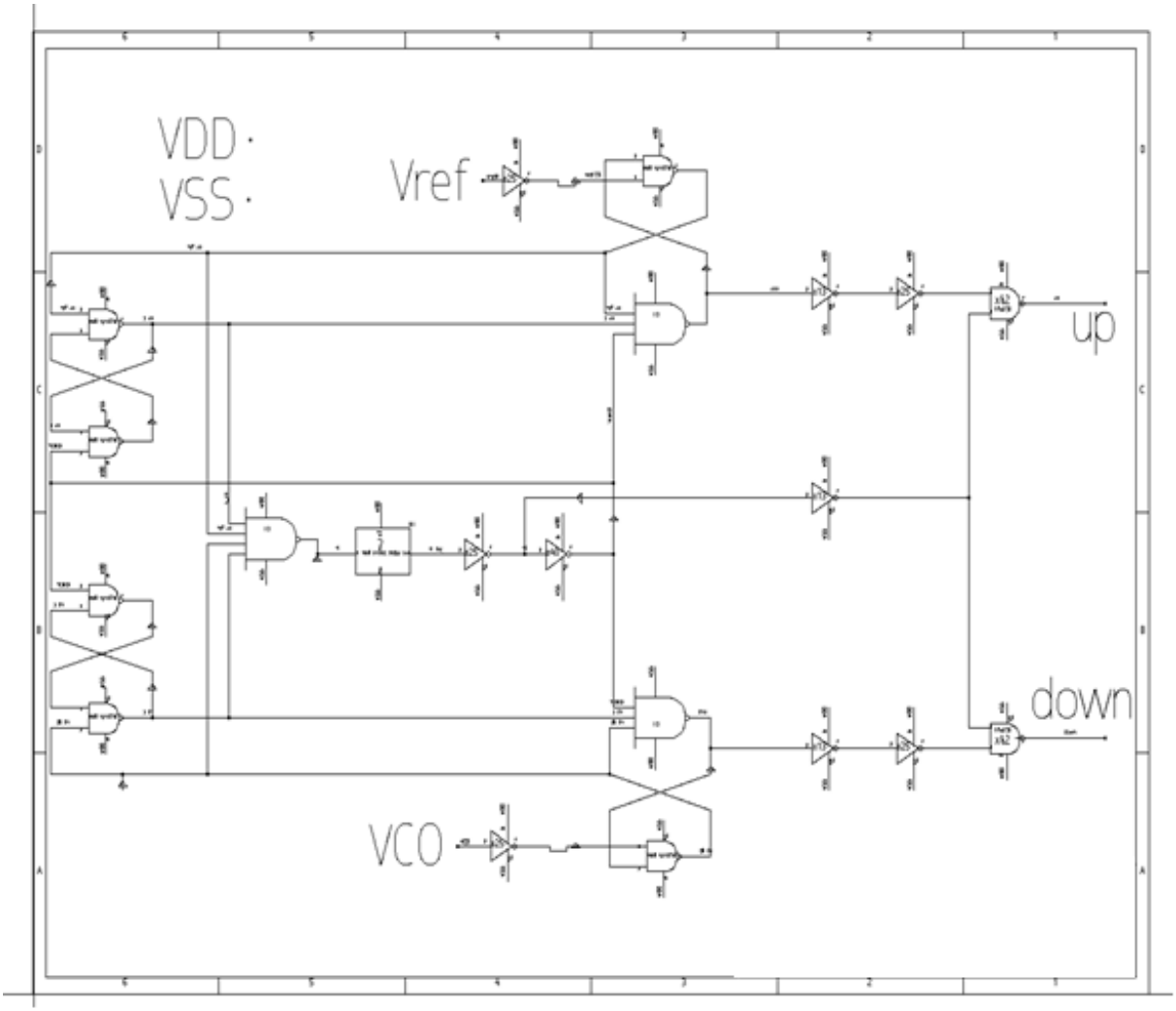
Պաշտպանիչ օղակը P տիպի հարթակի մեջ տեղադրված N տիպի գրպանիկ է: Խոռոչների կողմից պաշտպանիչ օղակի շրջանցումը չթույլատրելու համար այն պետք է միացվի P հարթակին [46]:

Դիտարկենք պաշտպանիչ օղակների ազդեցությունը փուլահաճախական դետեկտորի (ՓՀԴ) վրա: ՓՀԴ-ն, որն այլ կերպ կոչվում է փուլի համեմատիչ, ելքում գեներացնում է մուտքին տրվող ազդանշանների փուլի և/կամ հաճախությունների տարբերությանը համեմատական լարում: Այն ՓԻՀ-ի կարևորագույն հանգույցներից է [48]:

ՓԻՀ-երում օգտագործվող ՓՀԴ – ները կարելի է բաժանել երկու տեսակի: Առաջին տեսակի ՓՀԴ – ները մուտքում ստանում են անալոգային կամ թվային

ուղղանկյունաձև ազդանշաններ և գեներացնում են հաճախությունների տարբերությանը համարժեք ազդանշան: Այս տեսակի ՓՀԴ – ների ելքում ստացված ազդանշանը փոխանցվում է ՓԻՀ-ի լարումով ղեկավարվող գեներատոր հանգույցին: Երկրորդ տեսակի ՓՀԴ – ները զգայուն են միայն ազդանշանի ճակատի նկատմամբ և գեներացնում են ազդանշանների փուլերի տարբերությանը համեմատական ազդանշան, եթե դրանց հաճախությունները համընկնում են [48]:

Պաշտպանիչ օղակների ազդեցության վերլուծման համար ընտրվել է առաջին տեսակի ՓՀԴ –ի սխեմատեխնիկական նախագիծ (նկ. 2.28):

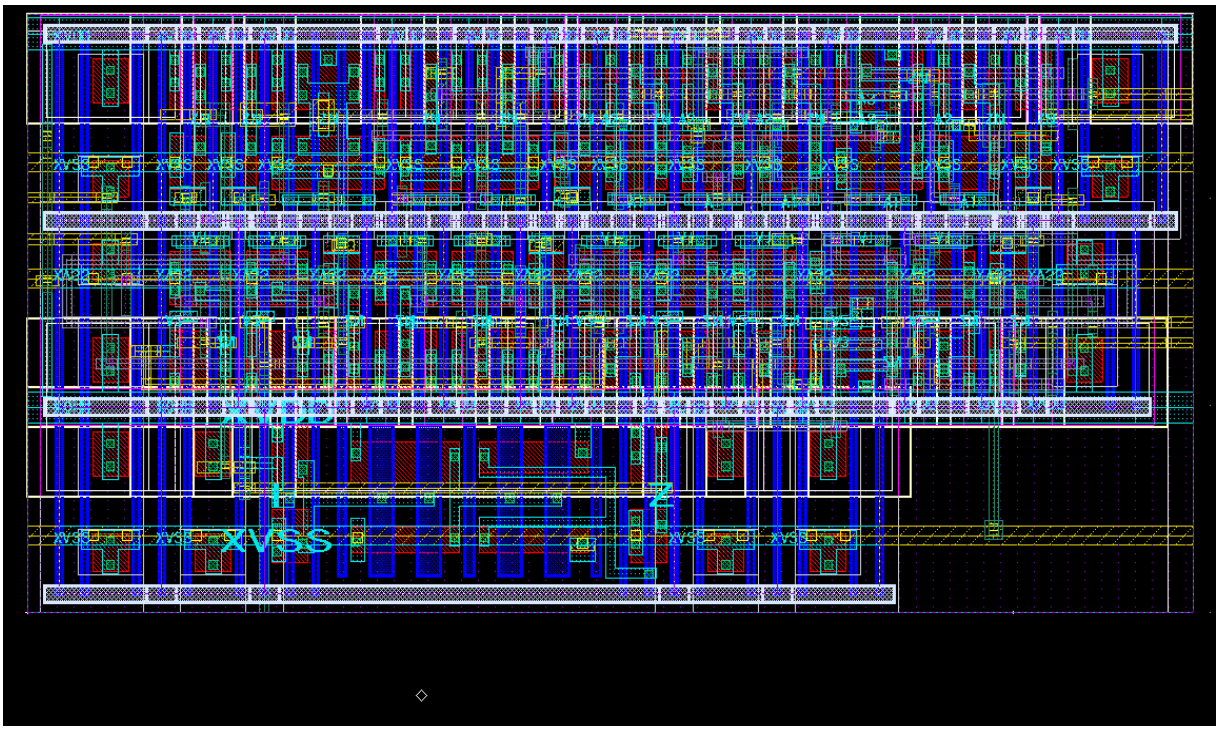


Նկ. 2.28. ՓՀԴ-ի սխեմատեխնիկական մոդելը

ՓՀԴ-ի ֆիզիկական նախագծման գործընթացում հիմնական ուշադրությունը դարձվել է նախագծման հետևյալ հնարքներին [111].

- կարճացվել են ծրագծման գծերի երկարությունները, որի արդյունքում մակաբույծ տարրերի արժեքները փոքրացել են,
- ծրագծումը կատարվել է այնպես, որ ՓՀԴ-ի up և down ճյուղերը լինեն սիմետրիկ և ունենան միևնույն երկարությունները, որը կհանգեցնի ելքային ազդանշանների փուլային շեղման նվազարկմանը:

Մշակված ՓՀԴ-ի ֆիզիկական նախագիծը բերված է նկ. 2.29 - ում:



Նկ. 2.29. ՓՀԴ-ի ֆիզիկական նախագիծը

Աղ. 2.5.-ում բերված է ՓՀԴ-ում կիրառված պաշտպանիչ օղակների ազդեցությունը ՓԻՀ-ի վերջնական ֆիզիկական նախագծի փուլային սխալի, թրթռոցի և մակերեսի վրա:

Աղյուսակ 2.5

Փուլային սխալի և թրթռոցի շեղումները ՓՀԴ-ի պատճառով

ՓԻՀ-ի ֆիզիկական նախագիծ	Փուլային սխալ	Թրթռոց	Մակերես
ՓԻՀ-ի վերջնական ֆիզիկական նախագիծ	735 *պվ*	13 *պվ*	6200 *մկմ*^{2}

Պաշտպանիչ օղակների կիրառում	-15%	-20%	1,9%

Ըստ աղ. 2.5-ի պաշտպանիչ օղակների կիրառումը ՓՀԴ-ում նվազեցնում է ՓԻՀ-ի փուլային սխալը 15% -ով, թրթռոցը՝ 20%-ով, սակայն այն նաև մեծացնում է ֆիզիկական նախագծի մակերեսը՝ 1,9%-ով:

2.4. Ջերմաստիճանային գրադիենտի ազդեցությունը ֆիզիկական նախագծի ելքային պարամետրերի վրա և դրա նկատմամբ կայուն գերճշգրիտ լիցքային պոմպի ֆիզիկական նախագծումը

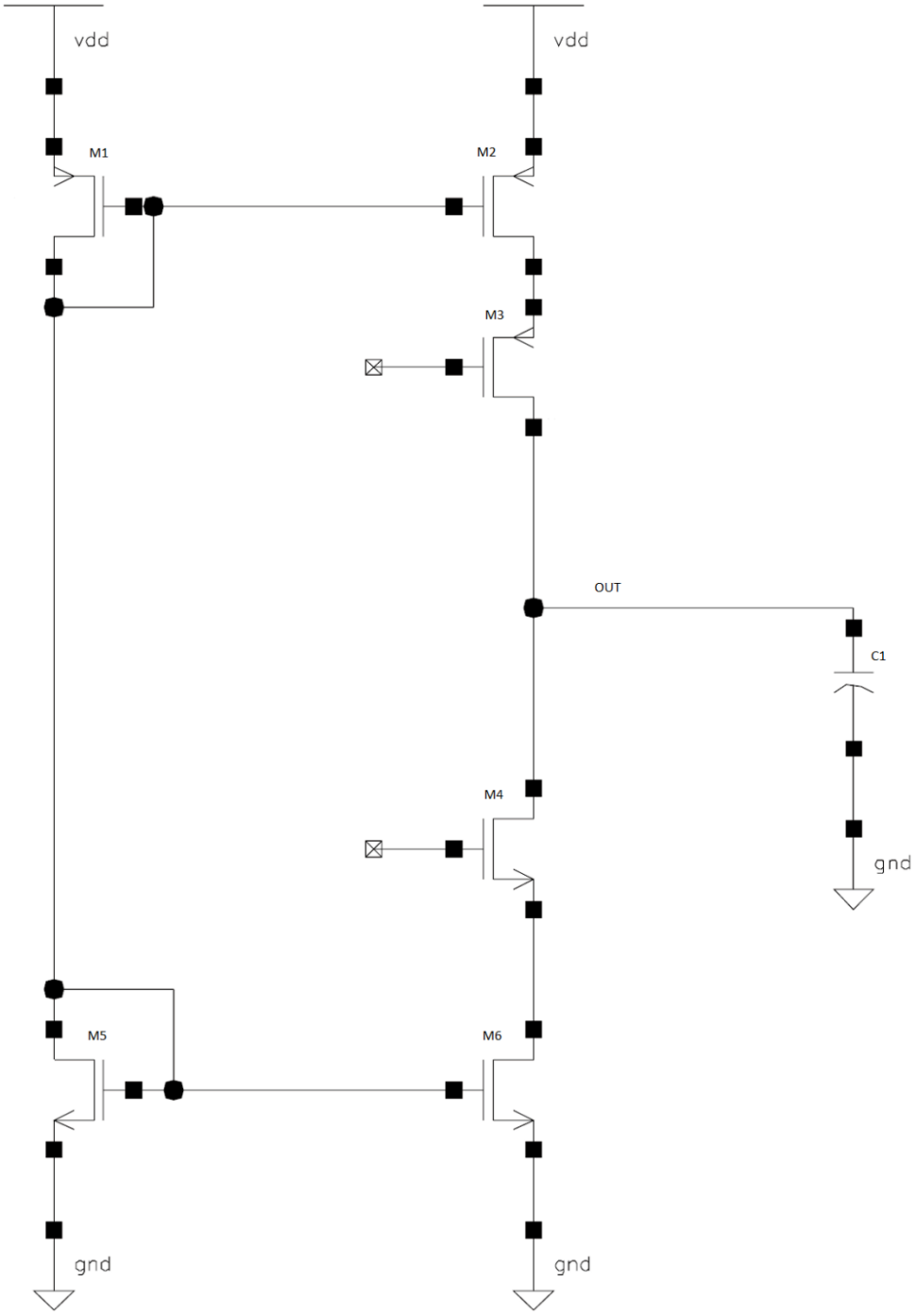
Լիցքային պոմպը (ԼՊ) նախատեսված է մի մակարդակի հաստատուն լարումը փոխակերպելու մեկ այլ մակարդակի հաստատուն լարման, որն օգտագործում է ունակություններ՝ որպես էներգիայի կուտակման աղբյուր, որը հետագայում վերածվում է բարձր կամ ցածր լարման [45]: Լիցքային պոմպերը լայն կիրառություն ունեն մակարդակի ձևափոխիչներում՝ +5 Վ և +3 Վ –ից +10 Վ և -10 Վ լարում ձևավորելու համար: ԼՊ-երը կիրառվում են նաև ՄՕԿ հիշողության տարրերում՝ 3 Վ հաստատուն հենակային լարում գեներացնելու համար: ԼՊ-երը լայնորեն կիրառվում են փուլահաճախական ինքնահամալարման համակարգերում (ՓԻՀ): ՓԻՀ-երում լիցքային պոմպի դերը այլ է. այն ծառայում է որպես կառավարվող հոսանքի աղբյուր, այսինքն՝ այն ոչ թե ձևավորում է բարձր կամ ցածր լարում, այլ ՓԻՀ-ի հաջորդ հանգույցին՝ պասիվ ֆիլտրին, տալիս է դրական կամ բացասական հոսանք: Հոսանքի ճշգրիտ արժեքը կարևորագույն դեր է կատարում ՓԻՀ-ի հաջորդ հանգույցների աշխատանքում, որի պատճառով լիցքային պոմպի ֆիզիկական նախագծի ճշգրտությունը էական ազդեցություն ունի ամբողջ ՓԻՀ համակարգի բնութագրերի վրա [45]:

ՓԻՀ-ի ճշգրիտ աշխատանքի համար ԼՊ-ն պետք է մատակարարի նույն լիցքաթափման և լիցքավորման հոսանքները, ընդ որում, ԼՊ-ն պետք է նախագծել այնպես, որ այդ հոսանքները լինեն նվազագույնը: Թվարկված պահանջները

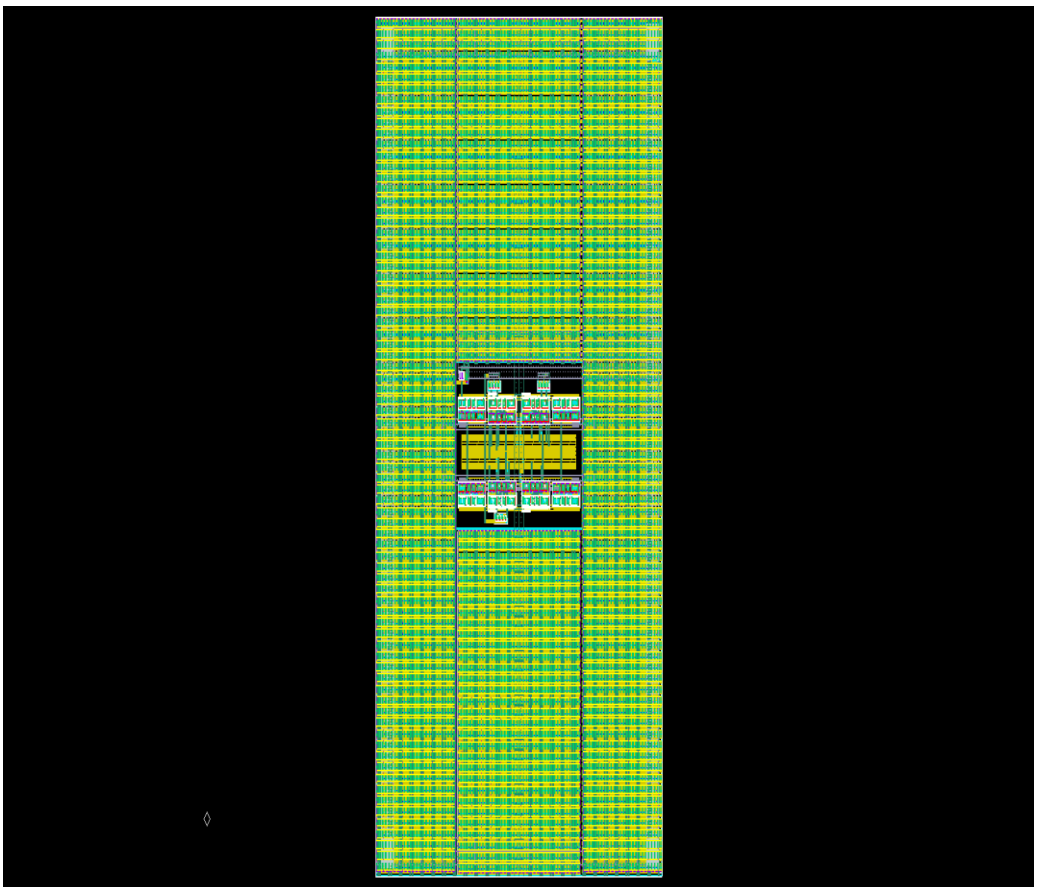
բավարարելու համար ԼՊ-ի ֆիզիկական նախագիծը պետք է բավարարի հետևյալ նախագծման կանոններին.

- ֆիզիկական նախագծում ԼՊ-ի բոլոր հանգույցներում պետք է իրականացվի համապատասխանեցում,
- ջերմաստիճանի գրադիենտների, ճնշումների և այլ երկրաչափական երևույթների ազդեցությունների նվազեցման համար ԼՊ-ի հանգույցները տեղակայվում են հնարավոր նվազագույն հեռավորության վրա [110]:

Հետազոտվող ԼՊ-ի սխեմատեխնիկական նախագիծը բերված է նկ. 2.30 - ում, իսկ ֆիզիկական նախագիծը՝ նկ. 2.31 - ում:



Նկ. 2.30. ԼՊ-ի սխեմատեխնիկական նախագիծը



Նկ. 2.31. ԼՊ-ի ֆիզիկական նախագիծը

Եզրակացություններ

1. Առաջարկվել է ԳՄԵ-ի նվազեցման տոպոլոգիական իրականացում՝ փոխարինելով նախագծում առկա P-ՄՕԿ տրանզիստորները N-ՄՕԿ տրանզիստորներով: Նախագծվել է SAED32/28 *նմ* տեխնոլոգիական գործընթացի համար հոսանքի հայելու երկու ֆիզիկական նախագիծ, առաջինը՝ P-ՄՕԿ տրանզիստորներով, երկրորդը՝ զուտ N-ՄՕԿ տրանզիստորներով: Վերը նշված մեթոդի կիրառման շնորհիվ գրպանիկի մոտիկության երևույթի ազդեցությունը նվազել է 10...100 անգամ:

2. Ֆիզիկական նախագծման տարրերի՝ ռեզիստորների, կոնդենսատորների և տրանզիստորների ճշգրիտ համապատասխանեցման համար առաջարկվել է XYYX համապատասխանեցման սխեմա: Իրականացվել է դիֆերենցիալ ուժեղարարի երեք ֆիզիկական նախագիծ՝ տարբեր մոտեցումներով առաջին դեպքում համապատասխանեցում չի կիրառվել, երկրորդ դեպքում կիրառվել է XYXY ռեզիստորային համապատասխանեցում, իսկ երրորդ նախագծում կիրառվել է

առաջարկված XYYX համապատասխանեցումը: Մեր կողմից առաջարկված համապատասխանեցման շնորհիվ ուժեղացման գործակիցն աճել է 12%-ով, իսկ համապատասխանեցման մեջ դրված դիմադրությունների տարբերությունը, դասական համապատասխանեցման դեպքի համեմատ նվազել է 35%-ով:

3. Հետազոտվել է պաշտպանիչ օղակների կիրառությունը ՓԻՀ-երի ֆիզիկական նախագծերում: Գնահատվել է պաշտպանիչ օղակների ազդեցությունը ՓԻՀ-երի ելքային բնութագրերի վրա: Նախագծվել է երկու ՓԻՀ-ի ֆիզիկական նախագիծ, որոնցից մեկում կիրառվել են պաշտպանիչ օղակներ, իսկ մյուսում՝ ոչ: Նմանակումների արդյունքում պարզ է դառնում, որ պաշտպանիչ օղակների կիրառման շնորհիվ ՓԻՀ համակարգերում ստացվում է փուլային սխալի 15%, և թրթռոցի 20% նվազում, իսկ մակերեսը աճում է ընդամենը 1,9%:

4. Հետազոտվել է ջերմաստիճանային գրադիենտի ազդեցությունը ֆիզիկական նախագծի ելքային պարամետրերի վրա, առաջարկվել է ջերմաստիճանային գրադիենտի նվազեցման եղանակ, որի շնորհիվ հնարավոր է դարձել իրականացնել ջերմաստիճանային գրադիենտի նկատմամբ կայուն լիցքային պոմպի նախագիծ:

ԳԼՈՒԽ 3. ԱՆԱԼՈԳԱՅԻՆ ԻՆՏԵԳՐԱԼ ՍԽԵՄԱՆԵՐԻ ԱՎՏՈՄԱՏԱՑՎԱԾ ՖԻԶԻԿԱԿԱՆ ՆԱԽԱԳԾՄԱՆ ԾՐԱԳՐԱՅԻՆ ՄԻՋՈՑԻ ՆԿԱՐԱԳՐՈՒԹՅՈՒՆԸ ԵՎ ՕԳՏԱԳՈՐԾՈՒՄԸ

Անալոգային ԻՍ-երի ավտոմատացված ծրագծման համար մշակվել են AAR(Automated Analog Rout) Designer և Constraint Designer ծրագրային միջոցները: Մշակված գործիքներում հաշվի են առնվել ֆիզիկական նախագծման իրականացման և երկրորդական երևույթների նվազեցման առաջարկված եղանակները: Այդ ծրագրային միջոցները իրագործվել են C++, Python, QT ծրագրավորման և սկրիպտավորման լեզուներով և նախատեսված են Linux օպերացիոն համակարգերի համար:

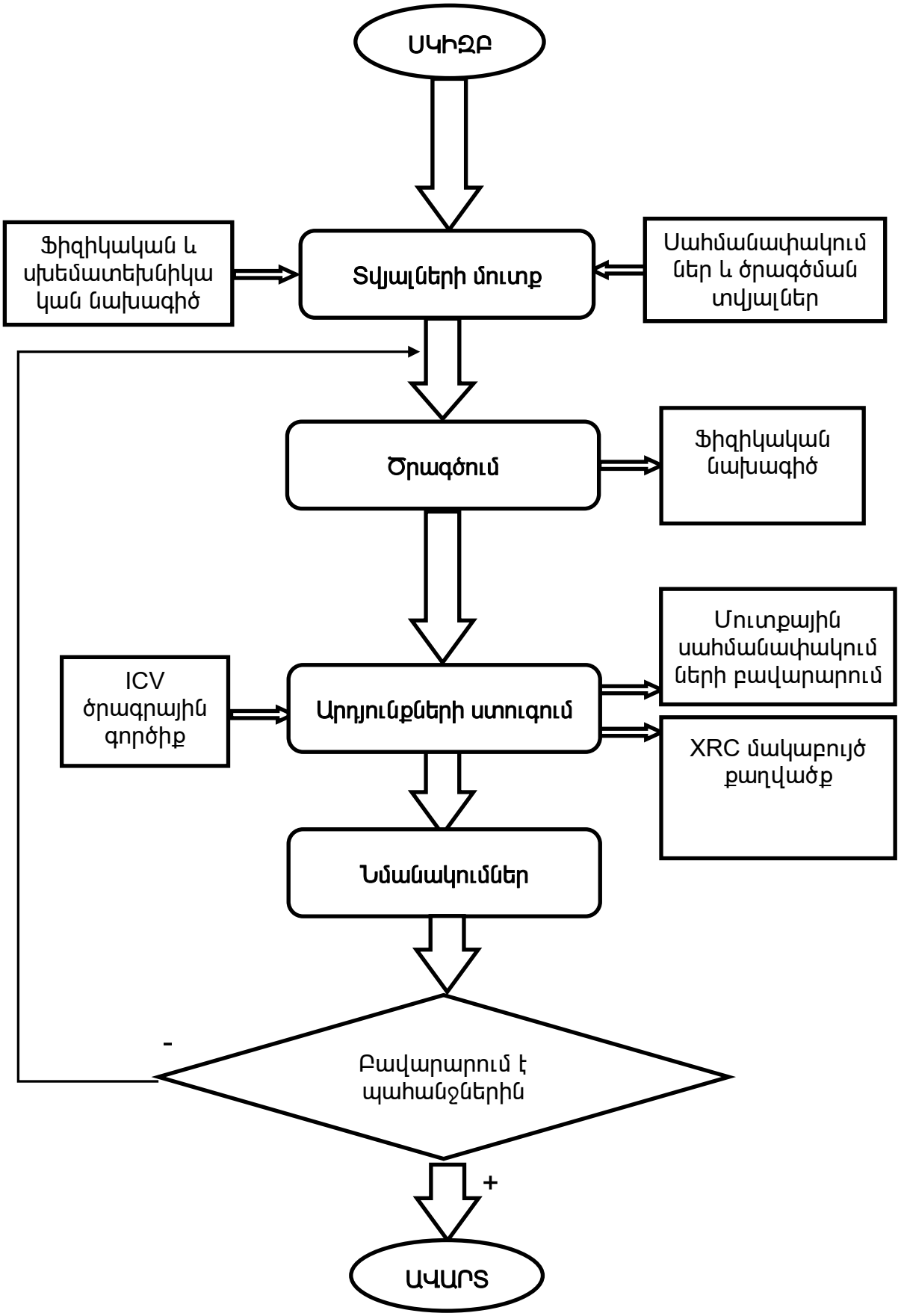
Ստեղծված գործիքային փաթեթը հնարավորություն է տալիս նախագծողի կողմից սահմանվող սահմանափակումներով անալոգային ԻՍ-երի ավտոմատացված ծրագծում՝ կիրառելով 2 գլխում առաջարկված ֆիզիկական նախագծման ժամանակ առաջացող երկրորդական երևույթների նվազեցման եղանակները:

AAR Designer ծրագրային միջոցը նախագծողի համար այնպիսի գործիք է, որով օգտագործողը արագ կարող է բացել ֆիզիկական նախագիծը, սահմանել սահմանափակումները և կատարել ծրագծում: Ծրագրի հարմարավետ և պարզ միջավայրի շնորհիվ նվազագույնի է հասցված նախագծողի կողմից մուտքային տվյալների մուտքագրումը: Մշակված ծրագրային միջոցով հնարավոր է մի քանի րոպեների ընթացքում մուտքագրել ֆիզիկական նախագծի ծրագծման համար բոլոր անհրաժեշտ մուտքային պայմանները:

Նախագծված Constraint Designer գործիքը նախագծվել է որպես C++ գրադարան, որը կարելի կլինի ներմուծել ոչ միայն նախագծված AAR Designer ծրագծման գործիքում, այլ նաև այլ ծրագծման և ֆիզիկական նախագծման գործիքներում:

Մշակված ծրագրային գործիքների փաթեթը ինտեգրվել է Սինոփսիս ընկերության ICV ծրագրային գործիքին:

AAR Designer ծրագրային միջոցի քայլերի հաջորդականությունը բերված է նկ. 3.1 - ում:


ՍԿԻԶԲ
Ֆիզիկական և սխեմատեխնիկա կան նախագիծ
Տվյալների մուտք
Սահմանափակում ներ և ծրագծման տվյալներ
Ծրագծում
Ֆիզիկական նախագիծ
ICV ծրագրային գործիք
Արդյունքների ստուգում
Մուտքային սահմանափակում ների բավարարում
XRC մակաբույծ քաղվածք
Նմանակումներ
Բավարարում է պահանջներին
-
+
ԱՎԱՐՏ

Նկ. 3.1. AAR Designer ծրագրային միջոցի աշխատանքը ներկայացնող բլոկ-դիագրամը

3.1. Անալոգային ԻՍ-երի ավտոմատացված նախագծում՝ օգտագործելով մշակված Constraint Designer և AAR Designer ծրագրային միջոցները

Մշակված գործիքային միջավայրում անալոգային ԻՍ-ի նախագծման հիմնական քայլերի հաջորդականությունը հետևյալն է:

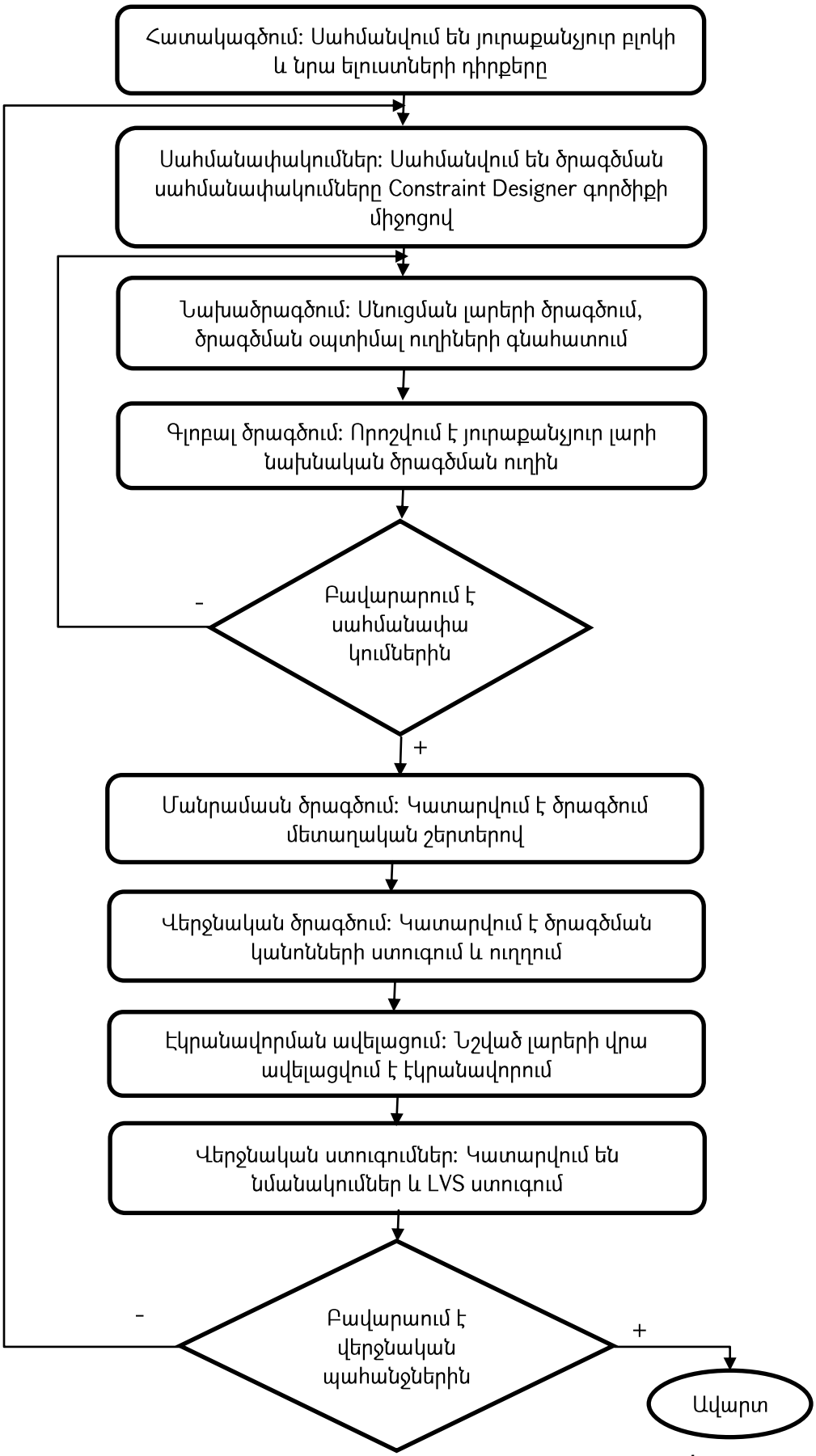
- Շղթայի նախագծողը կատարում է սխեմատեխնիկական նախագիծը:
- Նախագծողը ներմուծում է սահմանափակումները՝ օգտագործելով Constraint Designer գործիքը:
- Նախագիծը առկա սահմանափակումներով փոխանցվում է ֆիզիկական նախագծողին:
- Ֆիզիկական նախագծողը, օգտագործելով AAR Designer գործիքը, կատարում է ծրագծում:
- Կատարվում են ֆիզիկական նախագծի նմանակումներ:
- Օգտագործելով Constraint Designer գործիքը՝ ավելացվում կամ փոփոխվում են առկա սահմանափակումները:
- AAR Designer-ում կատարվում են վերջնական ծրագծում և DRC և LVS ստուգումներ:

Constraint Designer և AAR Designer գործիքների օգնությամբ անալոգային ԻՍ-երի նախագծման քայլերի հաջորդականությունը բերված է նկ. 3.2 - ում:

3.2. Սահմանափակումների ներմուծման համար մշակված Constraint Designer ծրագրային գործիքի նկարագրությունը

Անալոգային ինտեգրալ սխեմայի նախագծման համար անհրաժեշտ են մեծ քանակությամբ տվյալներ՝ այսպես կոչված սահմանափակումներ: Այդ տվյալների և սահմանափակումների քանակի աճը և բարդությունը նոր պահանջներ է առաջադրում անալոգային ԻՍ-երի ֆիզիկական նախագծման բնագավառում: Առաջանում է այդ սահմանափակումների նկարագրման և պահպանման համար մի ընդհանուր եղանակ

ունենալու անհրաժեշտություն, իսկ նախագծումը ավտոմատացնելու համար ավտոմատ գործիքը պետք է կարողանա կարդալ և վերծանել այդ տվյալները [9 ,10]:



Նկ. 3.2. Անալոգային ԻՍ-երի ավտոմատացված նախագծում՝ օգտագործելով AAR Designer և Constraint Designer գործիքները

Այս հարցի լուծմանն է ծառայում Open Access լայն տարածում գտած տվյալների հենքը: Այնտեղ պետք է պահպանվեն ֆիզիկական նախագիծը նկարագրող բոլոր սահմանափակումները:

Մշակված Constraint Designer ծրագրային գործիքը բավարարում է հետևյալ պահանջները

- այն տրամադրում է մի միջավայր, որտեղ կարելի է կատարել սահմանափակումների հետ գործողություններ՝ պահում, փոփոխություն, չեղարկում և այլն,
- երբ ընտրված է որոշակի օբյեկտ, նախագծողը կարող է բացել սահմանափակումների պատուհանը և տեղադրել որևէ սահմանափակում տվյալ օբյեկտի վրա, եթե այդ օբյեկտն արդեն ունի սահմանափակումներ, ապա այդ պատուհանում երևում են այդ բոլոր սահմանափակումները, և նախագծողը դրանք ջնջելու հնարավորություն ունի,
- սահմանափակումներին տրվող արժեքները ստուգում են անցնում և ընդունում են միայն թույլատրելի արժեքներ, եթե նախագծողը փորձում է անթույլատրելի արժեք ներմուծել որևէ սահմանափակման համար, ապա էկրանին առաջանում է համապատասխան զգուշացում,
- սահմանափակումների կիրառում մեկ ընտրված օբյեկտի վրա,
- սահմանափակումների կիրառում օբյեկտների համախմբի վրա, օրինակ, ընտրել երկու տրանզիստոր և կիրառել համապատասխանեցման համար նախատեսված սահմանափակումը,
- սահմանափակումները պահպանվում են Open Access տվյալների հենքում,
- մշակված գործիքը հնարավորություն ունի առկա սահմանափակումների ներմուծում որևէ փաստաթղթից,
- նախագծողը հնարավորություն ունի նախագծում առկա բոլոր սահմանափակումները արտածել փաստաթղթում,

- բազմամակարդակ նախագծերում հնարավորություն կա սահմանափակումների ավտոմատ կիրառում բարձր մակարդակից դեպի ներքևի մակարդակներ,
- մշակվել է հրաման, որը կատարում է սահմանափակումների ստուգում և արտածում նախագծում առկա բոլոր սահմանափակումները:

Որոշ դեպքերում նախագիծը մի նախագծողի կողմից չի կատարվում. հնարավոր է այնպիսի դեպք, որ նույն տարրի վրա սահմանված լինեն տարբեր տեսակի միմյանց հակասող սահմանափակումներ: Այս դեպքում, եթե այդ վիճակով նախագիծը փոխանցվի հաջորդ քայլին, այսինքն՝ ծրագծմանը, ապա AAR Designer գործիքը չի իմանա, թե որ սահմանափակումով է պետք կատարել ծրագծումը, ուստի այս խնդիրը պետք է լուծվի ավելի վաղ քայլերում: Մշակվել է ալգորիթմ, թե այս պարագայում որ սահմանափակմանը նախապատվություն տալ:

Դիտարկենք հետևյալ դեպքը. ենթադրենք նույն O օբյեկտի համար սահմանվել է երկու՝ C1 և C2 սահմանափակում՝ տարբեր V1 և V2 արժեքներով: Դիտարկենք այդ ալգորիթմը:

- Դիտարկվում են սահմանափակումների կիրառման շրջանակները. նախապատվությունը տրվում է ավելի նեղ շրջանակում կիրառված սահմանափակմանը:
- Սահմանափակումները համեմատվում են խմբերի տեսանկյունից: Եթե O օբյեկտը G1 և G2 խմբի անդամ է, և C1 սահմանափակումը կիրառվել է G1 խմբի վրա, իսկ C2 սահմանափակումը՝ G2 խմբի վրա, այս դեպքում, եթե G1-ը G2-ի ենթախումբն է, ապա հաղթում է C1 սահմանափակումը:
- Եթե երկու սահմանափակումն էլ պատկանում են նույն խմբին, հաղթում է վերջինը նկարագրված սահմանափակումը՝ հաշվի առնելով, որ նախագծողի վերջնական որոշումը ավելի կարևոր է:

Նախագծված Constraint Designer ծրագրային գործիքը պետք է աշխատի AAR Designer գործիքի հետ համատեղ և ներմուծի սահմանափակումներ AAR Designer գործիքի համար, որպեսզի ծրագծում կատարվի: Constraint Designer գործիքի հիմնական գործողությունը ֆիզիկական նախագծման սահմանափակումների ներմուծումն է OA տվյալների հենքի մեջ, որոնք օգտագործվելու են AAR Designer

գործիքի կողմից ծրագծման համար: Constraint Designer գործիքն ունի նաև հրամանների ներմուծման տող, որտեղ սկրիպտավորման լեզվի միջոցով կատարվում են բոլոր այն գործողությունները, որոնք հնարավոր է կատարել օգտատերի միջավայրից (GUI). Օգտատերի միջավայրը իրականացված է QT ծրագրավորման լեզվով:

3.2.1. Մշակված Constraint Designer գործիքի գրաֆիկական պատկերումը

Դիտարկենք Constraint Designer գործիքի գրաֆիկական ներկայացումը և օգտագործման եղանակները:

Constraint Designer գործիքում սահմանափակումները կարելի է բաժանել երեք խմբի.

- միայնակ սահմանափակումներ - կիրառվում են առանձին լարի վրա,
- զույգ տիպի սահմանափակումներ – սրանք կիրառվում են միանգամից լարերի զույգի վրա, օրինակ դիֆերենցիալ զույգ, համապատասխանեցում,
- խմբային սահմանափակումներ – սրանք կիրառվում են լարերի որոշակի խմբի վրա. նախօրոք սահմանվում է լարերի խումբ և կիրառվում սահմանափակում:

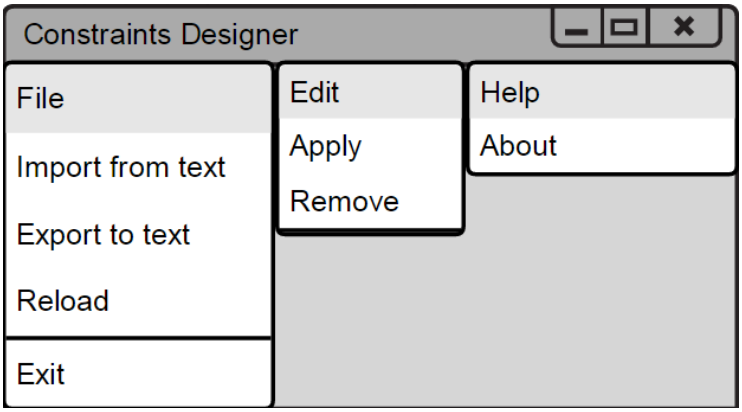
Նախագծված Constraint Designer գործիքի գրաֆիկական ներկայացումն ունի հետևյալ գործառականությունը.

- նախագծողը հնարավորություն ունի ցանկացած ֆիզիկական նախագծից բացել գործիքի գրաֆիկական ներկայացումը,
- գրաֆիկական ներկայացման մեջ պատկերվում են միայն այն սահմանափակումները, որոնք հնավոր է կիրառել տրված նախագծում,
- գրաֆիկական պատկերումը թարմացվում է ըստ ընտրված նախագծի տարրի: Երբ ոչ մի տարր ընտրված չէ, այն պատկերում է բոլոր լարերի վրա կիրառված սահմանափակումները: Երբ ընտրված է մի տարր, պատկերվում են տվյալ տարրի վրա կիրառված սահմանափակումները:

Constraint Designer գործիքի գլխավոր գործիքների վահանակը պարունակում է File, Edit, Help թափվող գործիքների վահանակները: File վահանակում գտնվում են Import from text, Export to text և Reload կոճակները: Import from text-ն օգտագործվում է

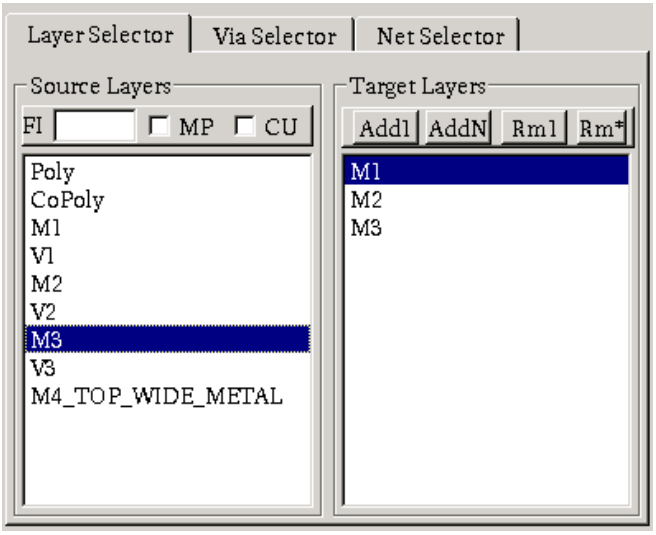
սահմանափակումների ներմուծման համար տեքստային փաստաթղթից, Export to text-ը կատարում է նախագծում գործող սահմանափակումների արտահանում և պահպանում տեքստային փաստաթղթում: Reload հրամանը տվյալների հենքից թարմացնում է նախագծում գտնվող բոլոր սահմանափակումները և չեղարկում այն սահմանափակումները, որոնք վերջնականապես չեն պահպանվել: Edit վահանակում գտնվում են Apply, Remove կոճակները: Apply կոճակը նախատեսված է տվյալների հենքում ներմուծված սահմանափակման պահպանման համար: Remove հրամանը նախատեսված է տվյալ լարի վրայից սահմանափակման հեռացման համար: Help վահանակում գտնվում են գործիքի համառոտ նկարագրությունը և հրամանների ցանկը՝ իրենց նկարագրություններով:

Constraint Designer գործիքի գլխավոր վահանակների տեսքը բերված է նկ. 3.3 – ում:



Նկ. 3.3. Constraint Designer գործիքի գլխավոր վահանակների գրաֆիկական պատկերումը

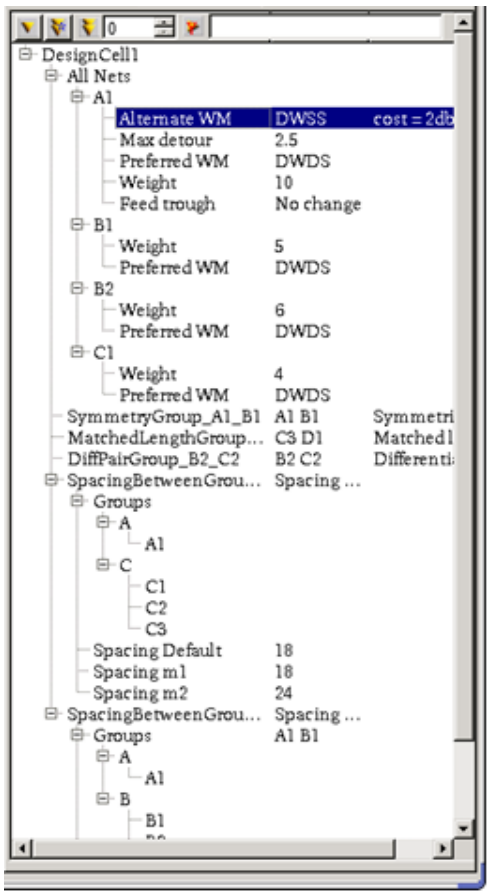
Նախագծված գործիքում ծրագծման շերտերի ընտրության համար մշակվել է Layer Selector պատուհանը,որի գրաֆիկական ներկայացումը բերված է նկ. 3.4 – ում:



Նկ. 3.4. Layer Selector պատուհանի գրաֆիկական պատկերումը

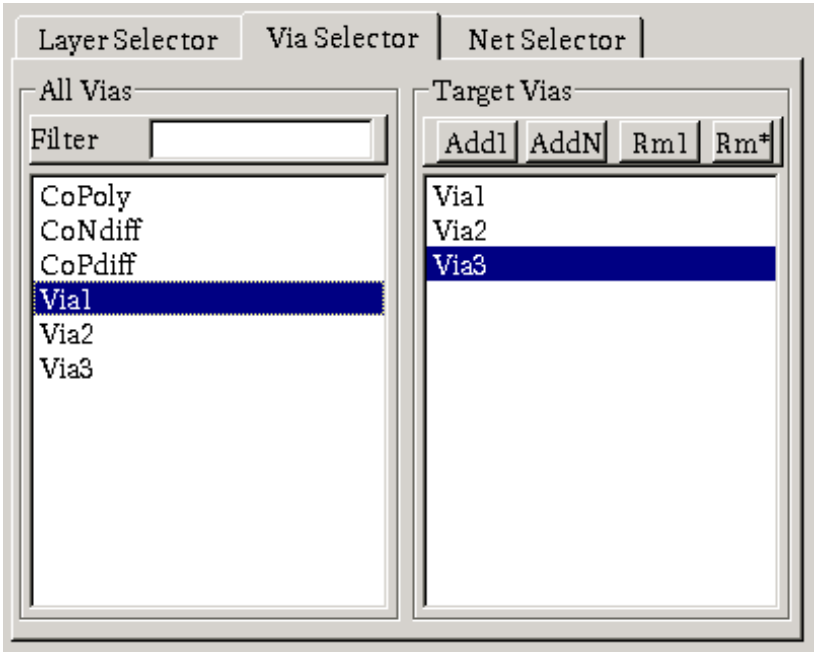
Layer Selector պատուհանի Source Layers ենթապատուհանում գտնվում են տվյալ նախագծում ծրագծման համար թույլատրելի բոլոր շերտերը: Իսկ Target Layers պատուհանում նախագծողն ինքն է ավելացնում այն շերտերը, որոնք նախագծողին անհրաժեշտ են: Add կոճակի օգնությամբ նա ավելացնում է շերտ, իսկ Rm կոճակի օգնությամբ հեռացնում է ընտրված շերտերը:

Սահմանափակումները Constraint Designer գործիքում պատկերվում են ծառի տեսքով: Նկ. 3.5 - ում պատկերված է այդ սահմանափակումների ծառը:



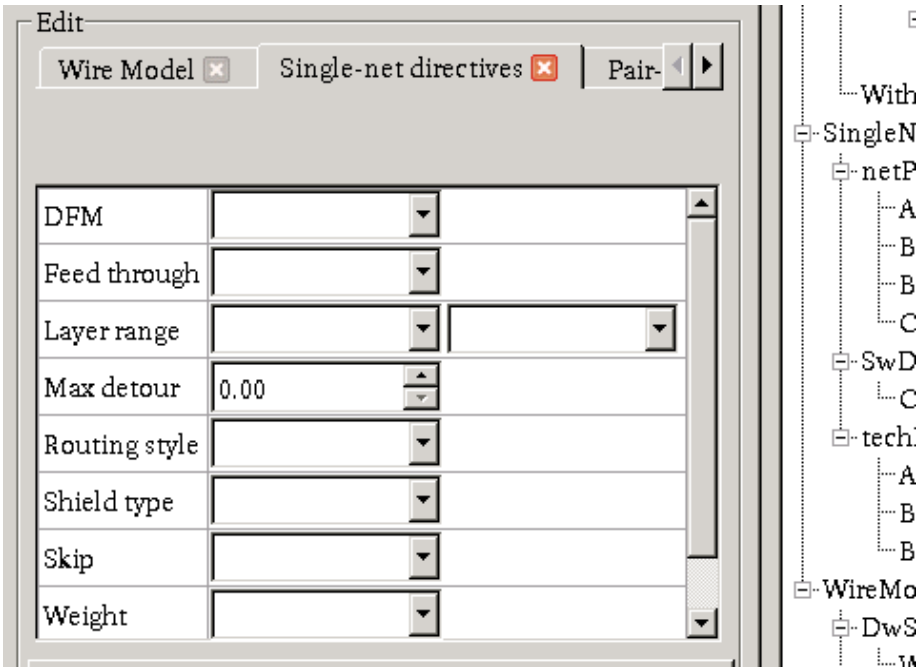
Նկ. 3.5. Constraint Designer գործիքում սահմանափակումների ծառի գրաֆիկական ներկայացումը

Հաջորդ պատուհանը միջմիացումների ընտրման Via Selector պատուհանն է: Այն բաղկացած է երկու ենթապատուհանից: All Vias ենթապատուհանում բերված են տվյալ ֆիզիկական նախագծում հասանելի բոլոր միջմիացումները, իսկ Target Vias ենթապատուհանում գտնվում են նախագծողի կողմից ավելացված միջմիացումները: Նախագծողը կարող է ավելացնել կամ հեռացնել միջմիացումներ՝ օգտագործելով Add և Rm կոճակները: Նկ. 3.6.-ում բերված է այդ պատուհանի գրաֆիկական պատկերումը:



Նկ. 3.6. Միջմիացումների ընտրման պատուհանը

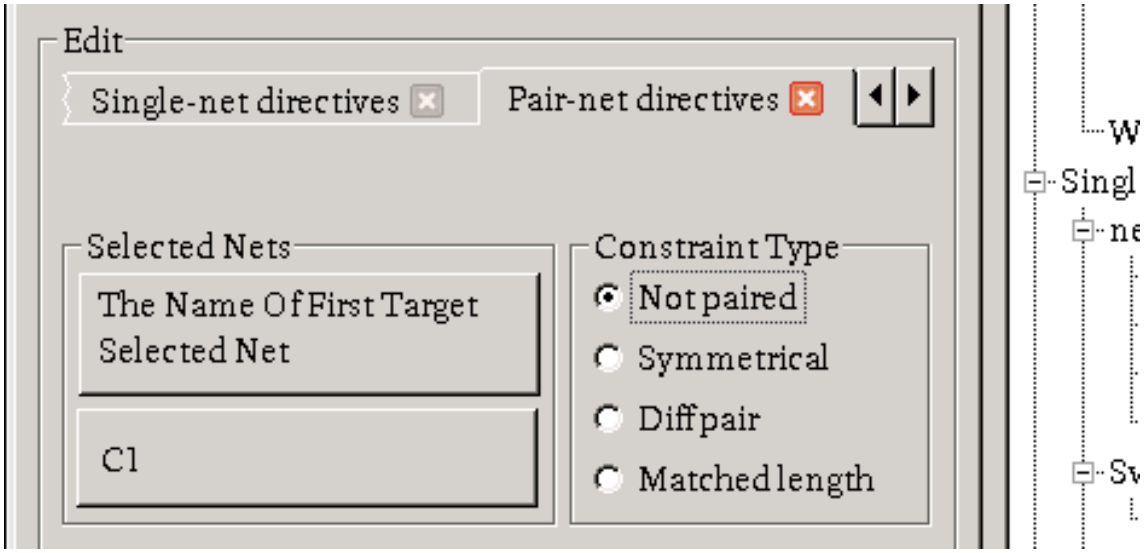
Constraint Designer գործիքի Edit պատուհանում գտնվում են սահմանափակումների ընտրման և պահպանման համար անհրաժեշտ բոլոր գործիքները: Edit պատուհանի ենթապատուհաններից է Single-Net Directives ենթապատուհանը (նկ. 3.7), որը նախատեսված է միայնակ սահմանափակումների ստեղծման համար:



Նկ. 3.7. Single-net directives պատուհանը

Սկզբում նախագծից կամ վերևում դիտարկված Net Selector պատուհանից ընտրվում է մեկ կամ մի քանի լար, այնուհետև կիրառվում է սահմանափակումը:

Edit պատուհանի երկրորդ ենթապատուհանը լարերի զույգի վրա սահմանափակում կիրառելու համար նախատեսված պատուհանն է: Այստեղ կարելի է ստեղծել Symmetrical, Diff pair կամ Matched length սահմանափակումներից մեկը: Այս պատուհանը ակտիվանում է միայն այն դեպքում, երբ Net Selector պատուհանից ընտրված է ճիշտ երկու լար: Նկ. 3.8 - ում բերված է այդ ենթապատուհանի գրաֆիկական ներկայացումը

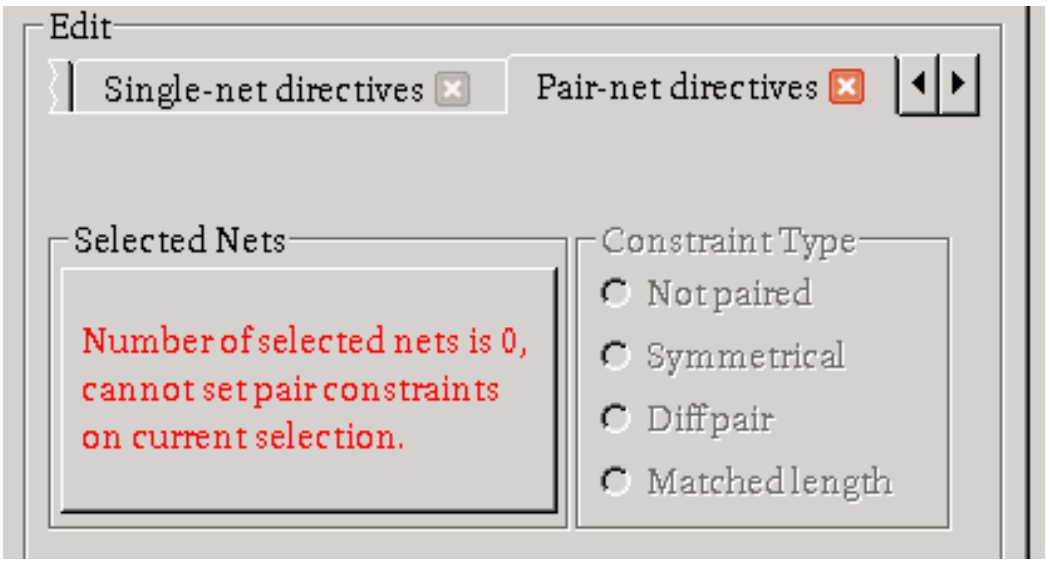


Նկ. 3.8. Pair-net Directives պատուհանի գրաֆիկական ներկայացումը

Եթե ընտրված լարերի քանակը հավասար չէ երկուսի, ապա հայտնվում է զգուշացում այդ մասին: Նկ. 3.9 - ում բերված է այդ զգուշացումով պատուհանի տեսքը, այս դեպքում ընտրված լարերի քանակը հավասար է զրոյի:

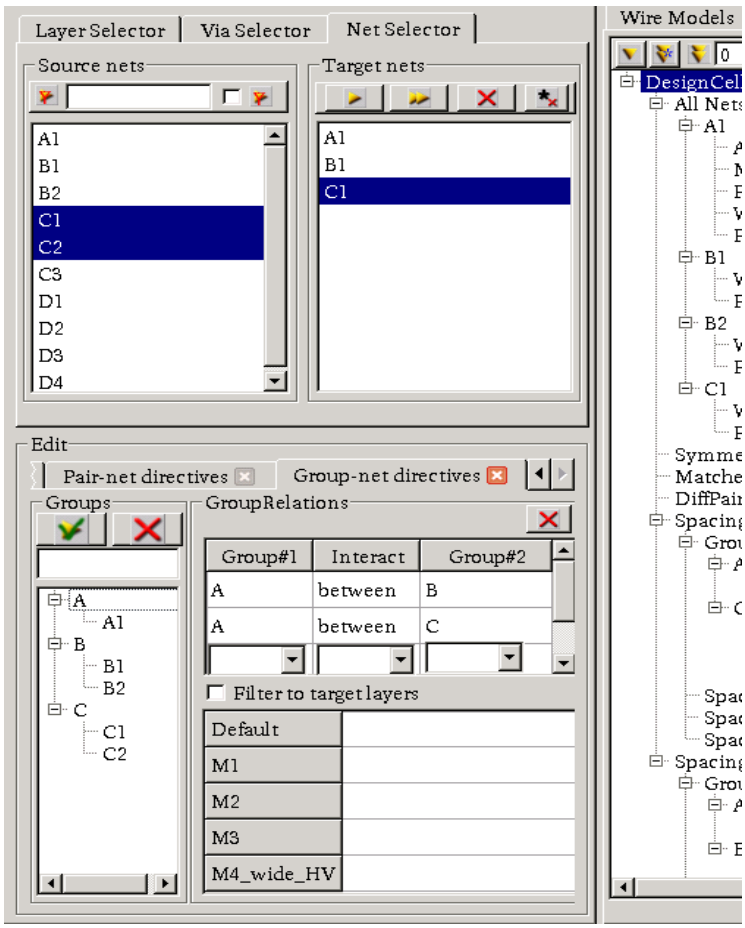
Edit պատուհանի երրորդ ենթապատուհանը լարերի խմբի վրա սահմանափակման տեղադրումն է:

Constraint Designer գործիքը հնարավորություն է տալիս խմբավորել մի քանի լար իրար հետ կամ սահմանել լարերի խումբ և այդ խմբի վրա կիրառել սահմանափակում, կամ սահմանափակումներ սահմանել երկու խմբերի միջև: Նկ. 3.10 - ում բերված է Group-net directives ենթապատուհանը:



Նկ. 3.9. Զգուշացման տեսքը լարերի քանակը երկուս չլինելու դեպքում

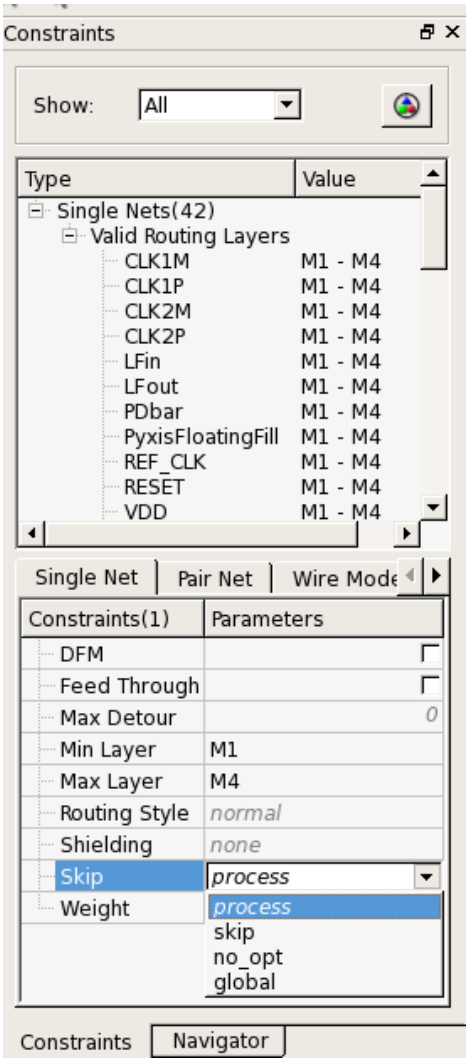
Նկ. 3.10 -ում “Group Relations” աղյուսակում պատկերվում են առկա սահմանափակումները, իսկ վերջին դատարկ տողը նախատեսված է նոր սահմանափակում ստեղծելու համար:



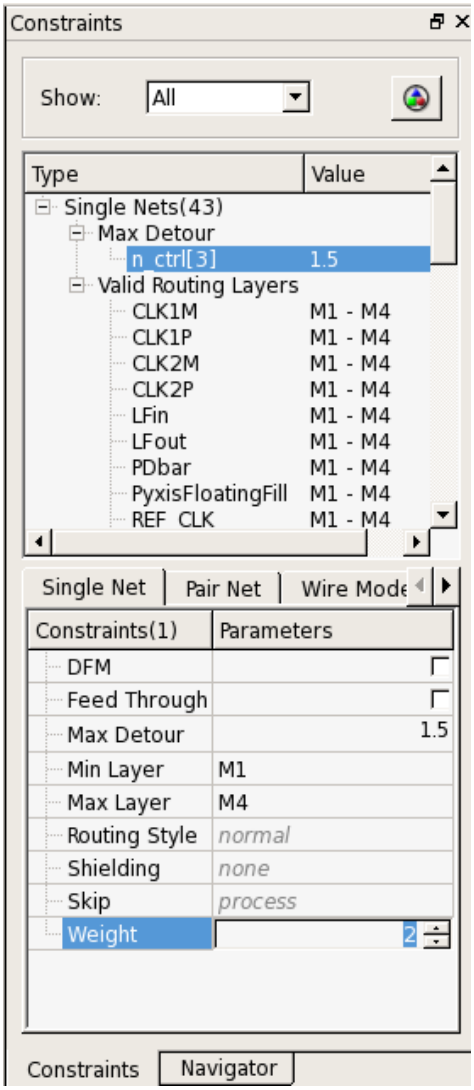
Նկ. 3.10. Խմբային սահմանափակումների սրեղծման Group-net directives ենթապատուհանը

Որոշ դեպքերում նախագծողը հնարավոր է՝ ցանկություն ունենա ինչ-որ լարերի ծրագծումը կատարել ձեռքով, որի համար անհրաժեշտ է AAR Designer գործիքին հուշել, որ չպետք է այդ լարերի ավտոմատ ծրագծում կատարել, այդ նպատակով տվյալ լարերի վրա կիրառվում է «անտեսել» (skip) սահմանափակումը (նկ. 3.11):

Սովորական վիճակում AAR Designer-ը սկզբում կատարում է ամենաերկար լարերի ծրագծումը, այնուհետև՝ լարերի երկարության նվազման հաջորդականությամբ: Որոշ դեպքերում նախագծողը ցանկանում է ծրագծում կատարել որևէ այլ հաջորդականությամբ: Այդ պատճառով պետք է սահմանել «լարի առաջնայնություն» (weight) սահմանափակումը: Սկզբում ծրագծվում են այն լարերը, որոնք ունեն այդ սահմանափակման ավելի մեծ արժեք: Նկ. 3.12 - ում պատկերված է լարի առաջնայնություն սահմանափակման կիրառման եղանակը Constraint Designer գործիքում:

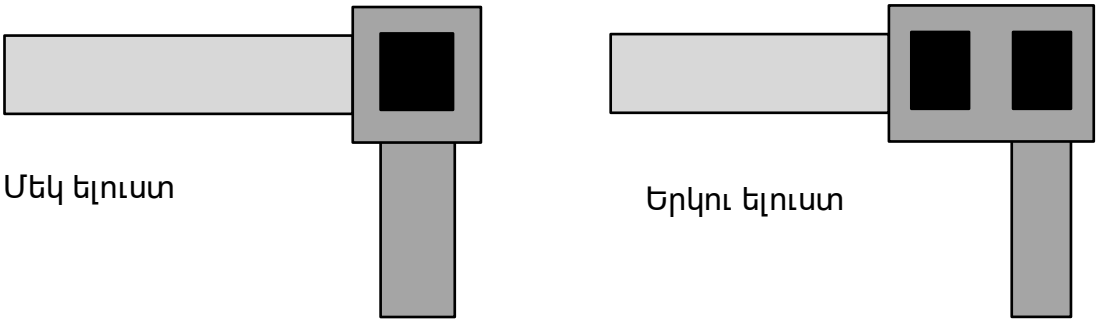


Նկ. 3.11. Անտեսել (skip) տիպի սահմանափակման կիրառումը



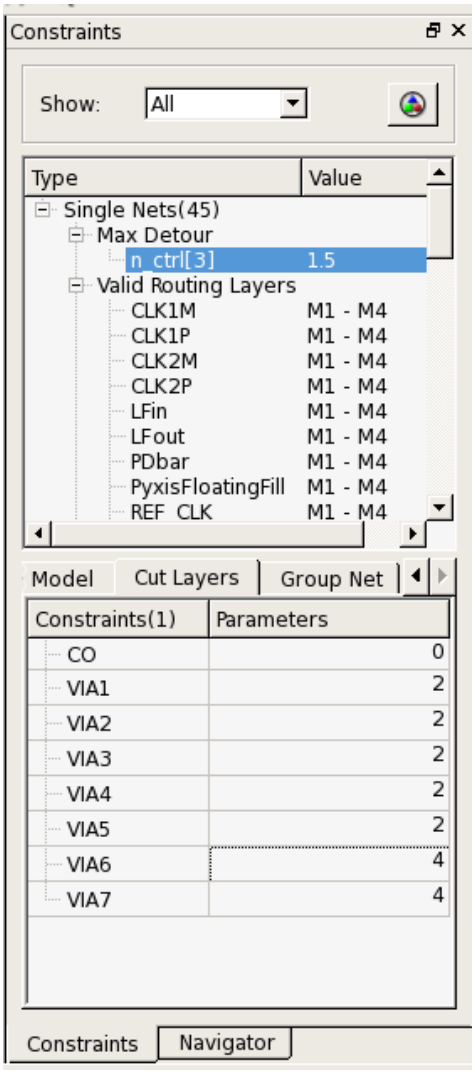
Նկ.3.12. Ծրագծման առաջնայնություն սահմանափակման կիրառումը

Հաջորդ տիպի սահմանափակումը ելուստների քանակների տեղադրման սահմանափակումն է: Այս սահմանափակումով AAR Designer գործիքը տեղեկանում է, թե որ լարի վրա քանի հատ ելուստ է պետք տեղադրել: Եթե որևէ արժեք չի տրվել, ապա լռելյայն արժեքը նրա մեկ ելուստն է (նկ. 3.13):



Նկ. 3.13. Տարբեր քանակությամբ ելուստների կիրառումը

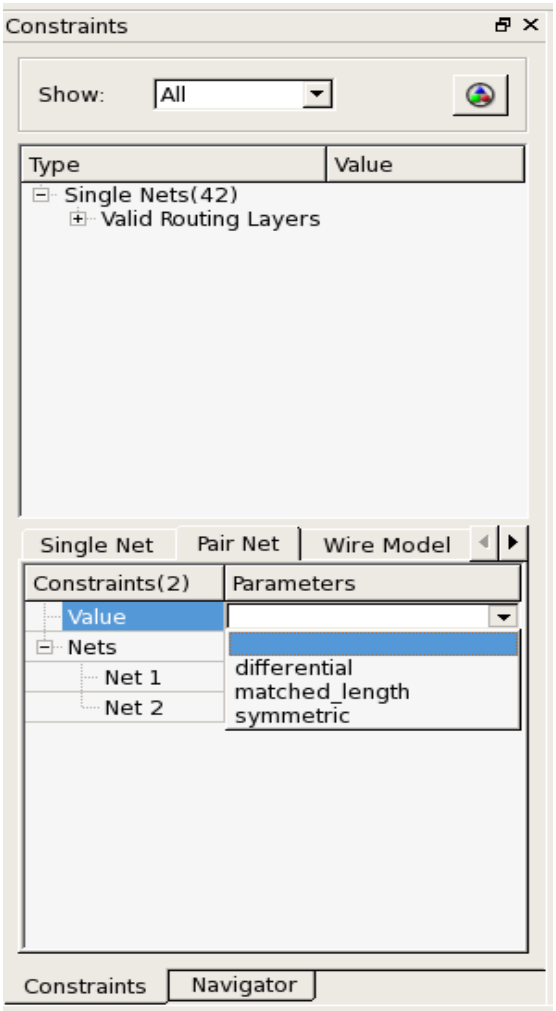
Յուրաքանչյուր տիպի ելուստի համար Constraint Designer գործիքում տրվում է համապատասխան ելուստների քանակը ծրագծման ժամանակ (նկ.3.14):



Նկ. 3.14. Constraint Designer գործիքում ելուստների քանակի սահմանումը

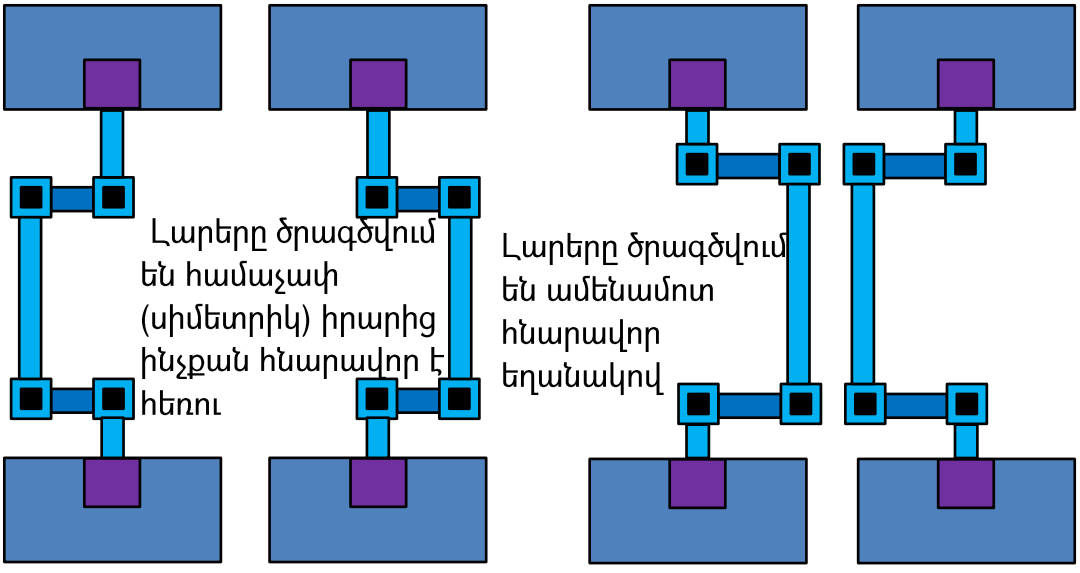
Նախ Net Selector պատուհանից ընտրվում է այն լարը, որի վրա պետք է կիրառել սահմանափակումը, այնուհետև Cut Layers ենթապատուհանում լրացվում է տվյալ լարի համար յուրաքանչյուր տիպի ելուստի քանակը: AAR Designer գործիքը հնարավորություն ունի ծրագծում կատարելիս օգտագործել մեկ կամ զույգ թվով ելուստներ:

Սահմանափակումների հաջորդ տիպը զույգ լարերի վրա կիրառվող սահմանափակումներն են: Constraint Designer գործիքում պետք է անցնել Pair Net ցյունակ, ընտրել երկու լար և դրանց վրա կիրառել զույգ տիպի սահմանափակում (նկ. 3.15):

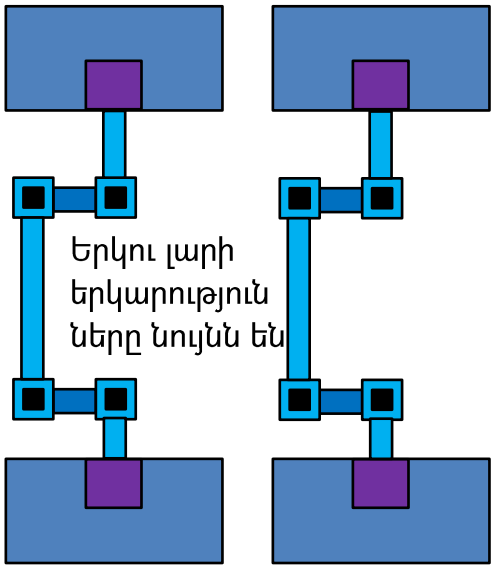


Նկ. 3.15. Զույգ տիպի սահմանափակման կիրառումը Constraint Designer գործիքում

Զույգ տիպի սահմանափակումը լինում է երեք տիպի՝ differential, matched_length, symmetrical (նկ. 3.16):



ա) բ)



գ)

Նկ. 3.16. Լարերի զույգի վրա կիրառվող սահմանափակումները. ա) symmetrical բ) differential գ) matched_length

Նախագծված Constraint Designer գործիքում ներդրվել է սխալ մուտքային տվյալների մշակման ալգորիթմ: AAR Designer գործիքն օգտվում է Constraint Designer գործիքի սահմանված սահմանափակումներից, և եթե ինչ-որ պատճառով ներմուծվել են անթույլատրելի սահմանափակումներ, ապա AAR Designer գործիքը չի կարողանա աշխատել դրանց հետ, և գործիքի աշխատանքը կընդհատվի: Այդ պատճառով մուտքային սխալների ստուգումը արվում է Constraint Designer գործիքում: Սխալ մուտքային տվյալների մշակման ալգորիթմը ստուգում է հետևյալ դեպքերը.

- երբ ստեղծվում է արդեն գոյություն ունեցող անվանմամբ սահմանափակում,
- երբ ստեղծվում է սահմանափակում, որի արժեքը գտնվում է թույլատրելի արժեքներից դուրս,
- երբ ստեղծվում է զույգ սահմանափակում այնպիսի լարի վրա, որը մասնակցում է այլ զույգ սահմանափակման մեջ:

Այս և շատ այլ դեպքերում “Apply” կիրառման կոճակը պասիվ վիճակում է, և նախագծողը հնարավորություն չունի պահպանելու արված փոփոխությունը, բացի դրանից, գրաֆիկական պատկերման մեջ առաջանում է հատուկ զգուշացում սխալի վերաբերյալ:

3.2.2. Մշակված Constraint Designer գործիքի սկրիպտավորման միջավայրը և հրամանների ցուցակը

Constraint Designer գործիքն ապահովում է Python սկրիպտավորման լեզուն: Ներկայացնենք Constraint Designer գործիքի հաճախակի օգտագործվող հրամանների ցանկը:

- create_net_group - նախագծում ստեղծում է լարերի խումբ:

Օգտագործման եղանակը՝ create_net_group(group_name='groupName', net_names=['netNames']), այստեղ group_name – ը ստեղծվող խմբի անունն է, իսկ net_names –ը խմբում ընդգրկվող լարերի անունները, որոնք տրվում են ցուցակի տեսքով:

- add_net_to_group_objects - ավելացնում է լարեր նախօրոք արդեն գոյություն ունեցող խմբին:

Օգտագործման եղանակը՝ add_net_to_group_objects(group_name='groupName', net_names=['netNames']), այստեղ group_name –ը առկա խմբի անվանումն է, իսկ net_names –ը խմբում ընդգրկվող լարերի անունները, որոնք տրվում են ցուցակի տեսքով:

- create_net_pair - տրված երկու լարի համար ստեղծում է լարերի զույգ սահմանափակումը:

Օգտագործման եղանակը՝ create_net_pair(net1='net1Name', net2='net2Name', pair_name='netPairName'), այստեղ net1-ը առաջին լարի, իսկ net2-ը երկրորդ լարի անվանումներն են, իսկ pair_name-ը ստեղծվելիք սահմանափակման անվանումն է:

- delete_net_from_group_objects - ջնջում է լարեր նախօրոք գոյություն ունեցող խմբից: Օգտագործման եղանակը՝

delete_net_from_group_objects(group_name='groupName', net_names=[‘netNames’]), այստեղ group_name –ը առկա խմբի անվանումն է, իսկ net_names –ը՝ խմբում ընդգրկված այն լարերի անունները, որոնք պետք է հեռացվեն այդ խմբից:

- delete_spacing_constraint_between_net_groups - հեռացնում է երկու խմբի միջև սահմանված «տարածություն» սահմանափակումը:

Օգտագործման եղանակը՝ delete_spacing_constraint_between_net_groups(group_name1='groupName1', group_name2 = 'groupName2'), այստեղ group_name1-ը առաջին, իսկ group_name2-ը՝ երկրորդ խմբի անվանումներն են:

- report_net_pairs - արտածում է նախագծում առկա բոլոր լարերի զույգերը: Օգտագործման եղանակը՝ report_net_pairs(net=’netName’), այստեղ net-ը այն լարի անվանումն է, որի համար պետք է վերադարձվեն լարերի զույգերը, որոնց մեջ այն գտնվում է, եթե հրամանը կանչում ենք առանց արգումենտի, ապա վերադարձվում են նախագծում առկա բոլոր լարերի զույգերը:
- report_capacitance_constraint_of_net - վերադարձնում է լարի վրա սահմանված ունակություն սահմանափակումը:

Օգտագործման եղանակը՝ report_capacitance_constraint_of_net(net_name='netName',capacitance_unit='capacitanceU nit'), այստեղ net_name-ը լարի անունն է, իսկ capacitance_unit-ն՝ ունակության չափման միավորը. այն կարող է ստանալ *Ֆ, նՖ, պՖ* և *ֆՖ* արժեքներ, նրա լռելյայն արժեքը *նՖ* է:

- report_electrical_constraint_of_nets - վերադարձնում է լարի վրա սահմանված էլեկտրական սահմանափակումները:

Օգտագործման եղանակը՝ report_electrical_constraint_of_nets (capacitance_unit = 'capacitanceUnit', net_names = ['netNames'], resistance_unit= 'resistanceUnit'), այստեղ capacitance_unit-ը ունակության չափման միավորն է, net_names-ը՝ լարերի անվանումների ցանկը, որոնց համար պետք է արտածվեն էլեկտրական սահմանափակումները, resistance_unit-ը դիմադրության չափման միավորն է, որի լռելյայն արժեքը *օհմ* է:

- report_inst_group_objects - վերադարձնում է խմբում առկա բոլոր լարերի ցուցակը: Օգտագործման եղանակը՝ report_inst_group_objects(group_name='groupName'), այստեղ group_name-ը այն խմբի անվանումն է, որին պատկանող լարերը պետք է արտածվեն էկրանին:

- report_max_layer - վերադարձնում է նախագծում օգտագործվող ամենաբարձր մակարդակի մետաղը. այս հրամանը մուտքային արգումենտներ չի ստանում:
- report_min_layer - վերադարձնում է նախագծում օգտագործվող ամենացածր մակարդակի մետաղը. այս հրամանը մուտքային արգումենտներ չի ստանում:
- report_net_groups - վերադարձնում է նախագծում առկա բոլոր լարերի խմբերը, այս հրամանը մուտքային արգումենտներ չի ստանում:
- report_resistance_constraint_of_net - վերադարձնում է տրված լարի առավելագույն դիմադրություն սահմանափակումը:

Օգտագործման եղանակը՝ report_resistance_constraint_of_net(net_name='netName', resistance_unit='resistanceUnit'), այստեղ net_name-ը լարի անվանումն է, իսկ resistance_unit-ը՝ վերադարձվող դիմադրության չափման միավորը:

- report_routing_layers - վերադարձնում է ծրագծման համար թույլատրելի բոլոր շերտերը: Այս հրամանը արգումենտներ չի ստանում:
- report_shielding - վերադարձնում է այն լարերի ցանկը, որոնց վրա կիրառված է էկրանավորում: Այս հրամանը արգումենտներ չի ստանում:
- report_spacing_constraint_between_net_groups - վերադարձնում է երկու խմբի միջև գոյություն ունեցող հեռավորության սահմանափակումը:

Օգտագործման եղանակը՝ report_spacing_constraint_between_net_groups (group_name1 = 'groupName1', group_name2 = 'groupName2'), այստեղ group_name1-ը առաջին խումբն է, իսկ group_name2-ը՝ երկրորդ խումբը:

- delete_net_pair - հեռացնում է տրված լարերի վրա կիրառված <<լարերի զույգ>> սահմանափակումը:

Օգտագործման եղանակը՝ delete_net_pair(netPairConstraint = 'netPairName'), այստեղ netPairConstraint-ը սահմանափակման անվանումն է, որը պետք է հեռացնել:

- delete_capacitance_constraint_on_net - ջնջում է լարի վրա կիրառված ունակություն սահմանափակումը:

Օգտագործման եղանակը՝ delete_capacitance_constraint_on_net (net_names = 'netNames'), այստեղ net_names-ը այն լարերի ցուցակն է, որոնց համար պետք է կիրառվի հրամանը:

- delete_resistance_constraint_on_net - ջնջում է լարի վրա կիրառված դիմադրության սահմանափակումը:

Օգտագործման եղանակը՝ delete_resistance_constraint_on_net (net_names='netNames'), այստեղ net_names-ը այն լարերի ցուցակն է, որոնց համար հեռացվում է դրված սահմանափակումը:

- set_capacitance_constraint_on_net - լարի վրա սահմանում է ունակության սահմանափակումը և այն պահում տվյալների հենքում:

Օգտագործման եղանակը՝ set_capacitance_constraint_on_net(max_c='Cmax', net_names='netNames'), այստեղ max_c-ն ունակության չափն է՝ *նՖ*-ներով, իսկ net_names-ը՝ լարերի ցանկը, որոնց վրա պետք է կիրառվի սահմանափակումը:

- set_resistance_constraint_on_net - լարի վրա սահմանում է «դիմադրություն» սահմանափակումը և այն պահում տվյալների հենքում: Այն AAR Designer – ը օգտագործելու է լարի ծրագծման ժամանակ, որպես լարի առավելագույն դիմադրություն:

Օգտագործման եղանակը՝ set_resistance_constraint_on_net(max_r='Rmax', net_names='netNames'), այստեղ max_r-ը դիմադրության առավելագույն չափն է՝ *օհմ*-երով, իսկ net_names-ը՝ լարերի ցանկը, որոնց վրա պետք է կիրառվի սահմանափակումը:

- set_spacing_constraint_between_net_groups - սահմանում և պահում է տվյալների հենքում երկու լարի խմբերի միջև գոյություն ունեցող «նվազագույն հեռավորություն» սահմանափակումը:

Օգտագործման եղանակը՝ set_spacing_constraint_between_net_groups(group_name1='groupName1', group_name2='groupName2',min_spacing=’spacing’), այստեղ group_name1-ը առաջին խմբի անվանումն է, group_name2-ը՝ երկրորդ, իսկ min_spacing-ը՝ «նվազագույն հեռավորություն» սահմանափակումը՝ արտահայտված *նմ*-ներով:

- set_vias_on_net - սահմանում է այն բոլոր Via-ների բազմությունը, որով ծրագծման գործիքը կարող է կատարել տրված լարի ծրագծումը:

Օգտագործման եղանակը՝ set_vias_on_net(net='netName', via_names=['viaNames']), այստեղ net-ը տրված լարի անվանումն է, իսկ via_names-ը՝ այդ լարի ծրագծման համար թույլատրելի ելուստների անվանումները:

3.2.3. Constraint Designer գործիքում մշակված սահմանափակումների տիպերը և դրանց նկարագրությունը

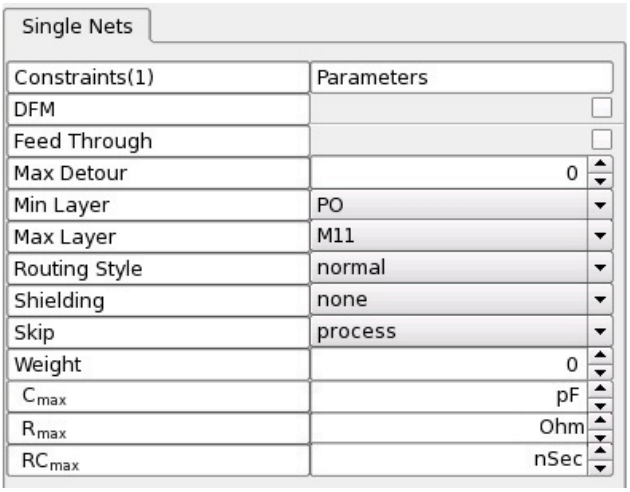
Constraint Designer ծրագրային գործիքում ներկայացված սահմանափակումներից մեկը “Avoid” տիպի սահմանափակումն է: Avoid սահմանափակումն օգտագործվում է AAR Designer ծրագրի կողմից ծրագծման ժամանակ, որպեսզի տրվի տարրերի կամ լարերի խմբերի միջև եղած նվազագույն հեռավորությունը: Այս տիպի սահմանափակումը կարելի է կիրառել, եթե կա լարերի խումբ կամ երկու, կամ ավելի տարրեր են ընտրված: Սահմանափակումն ամրագրվում է OA տվյալների հենքում: Avoid սահմանափակման լռելյայն արժեքը 0 է: Avoid սահմանափակումը ջնջվում է, եթե տարրերից կամ լարերից մեկը, որի վրա այն կիրառված է, ջնջվում է ֆիզիկական նախագծից: Բացի դրանից, այն կարելի է ջնջել՝ սեղմելով ստեղնաշարի delete կոճակը:

Constraint Designer գործիքում սահմանված սահմանափակումների տիպերից հաջորդ կարևորագույնը էլեկտրական սահմանափակումներն են:

Էլեկտրական սահմանափակումները լինում են երեք տիպի.

- C_{max} - առավելագույն ունակություն,
- R_{max} - առավելագույն դիմադրություն,
- RC_{max} - առավելագույն դիմադրության և ունակության արտադրյալ:

Նկ. 3.17 - ում բերված է Constraint Designer գործիքի պատուհանը, որը ցույց է տալիս ընտրված լարի վրա կիրառված բոլոր սահմանափակումները, այդ թվում նաև՝ էլեկտրական սահմանափակումները:



Նկ. 3.17. Էլեկտրական սահմանափակումների ներմուծման պատուհանը

C_{max} սահմանափակումը տալիս է լարի առավելագույն «ունակություն սահմանափակումը». այն ստիպում է AAR Designer-ին այդ լարի ծրագծում կատարել ավելի փոքր ունակությամբ, քան տրված է: C_{max} սահմանափակումը կարող է ստանալ միայն դրական արժեքներ, այն լռելյայն արժեք չունի, այսինքն, եթե C_{max} սահմանափակում չենք սահմանում, ապա ծրագծման ընթացքում լարը կարող է ունենալ շատ մեծ ունակություն: C_{max} սահմանափակման չափման միավորը պիկոֆարադն է:

R_{max} սահմանափակումը տալիս է լարի առավելագույն դիմադրությունը, այն ստիպում է AAR Designer-ին այդ լարի ծրագծում կատարել ավելի փոքր դիմադրությամբ, քան տրված է: R_{max} սահմանափակումը կարող է ստանալ միայն դրական արժեքներ, այն լռելյայն արժեք չունի, R_{max} սահմանափակման չափման միավորը *օհմ*-ն է:

RC_{max} սահմանափակումը տալիս է լարի առավելագույն հապաղումը, այն ստիպում է AAR Designer-ին այդ լարի ծրագծում կատարել ավելի փոքր դիմադրության և կոնդենսատորի արտադրյալի արժեքով, քան տրված է: RC_{max} սահմանափակումը կարող է ստանալ միայն դրական արժեքներ, այն լռելյայն արժեք չունի: RC_{max} սահմանափակման չափման միավորը *նվ*-ն է:

3.3. Անալոգային ինտեգրալ սխեմաների ֆիզիկական նախագծի ավտոմատացված ծրագծման համար մշակված գործիքի նկարագրությունը

Անալոգային ԻՍ-երի ավտոմատացված ֆիզիկական ծրագծման համար նախատեսված AAR Designer գործիքը բավարարում է հետևյալ պահանջները, և այն ունի հետևյալ գործառականությունը:

- AAR Designer գործիքն ինտեգրվում է ձեռքով ֆիզիկական նախագծման համար նախատեսված գործիքին:
- AAR Designer-ը կատարում է ծրագծում՝ հաշվի առնելով կամ ֆիզիկական նախագծի սխեմատեխնիկական նկարագրությունը, կամ նախագծողը ինքն է նշում, թե որ հանգույցը որին պետք է միացնել, և տալիս՝ այդ միացման համար նախատեսված սահմանափակումները:
- Մշակված AAR Designer գործիքը հաջողությամբ կարող է ծրագծում կատարել բազմամակարդակ ֆիզիկական նախագծերում և հետևաբար ֆիզիկական նախագիծը հարթեցնելու և մի մակարդակի բերելու անհրաժեշտություն չի առաջանում. այն անցնում է մակարդակից մակարդակ և կատարում անհրաժեշտ ծրագծումը բոլոր մակարդակներում:
- AAR Designer գործիքն աշխատում է ինտերակտիվ ռեժիմում. սա նշանակում է, որ ֆիզիկական նախագծում ձեռքով կատարված ցանկացած փոփոխություն տեսանելի է գործիքին, և հետագա ծրագծումը կատարվում է՝ այդ փոփոխությունները հաշվի առնվելով:
- Երբ ծրագծման ավարտից հետո նախագծողն ավելացնում է որոշակի հանգույցներ և կատարում ձեռքով մասնակի ծրագծում, ապա AAR Designer գործիքը հնարավորություն ունի շարունակելու այդ ծրագծումը:
- AAR Designer գործիքն ունի հրամանների կատարման տող, որտեղից Python սկրիպտավորման լեզվով տրվում է հրամանների հաջորդականությունը, եթե նախագծողը չի ուզում աշխատել ինտերակտիվ ռեժիմում:
- Ներկա պահին AAR Designer-ն աշխատում է 28 նմ տեխնոլոգիայով, սակայն գործիքը այնպես է նախագծված, որ հեշտությամբ կարելի է անցնել ավելի փոքր տեխնոլոգիաների:

- AAR Designer-ը հնարավորություն ունի աշխատելու նախագծողի կողմից տրվող սահմանափակումների հիման վրա: Այդ սահմանափակումները լրացվում են նախագծի OA տվյալների հենքում, և հետագա ծրագծման ժամանակ AAR Designer-ը օգտվում է դրանցից:
- AAR Designer-ը հնարավորություն ունի ինտեգրվել Սինոփսիս ընկերության ICV գործիքին և ծրագծման ավարտին կատարել մակաբույծ քաղվածքի ստացում, ինչպես նաև DRC և LVS ստուգումներ:

3.3.1. AAR Designer գործիքի աշխատանքի համար պահանջվող մուտքային փաստաթղթերը

AAR Designer գործիքին աշխատանքի համար անհրաժեշտ են որոշակի մուտքային փաստաթղթեր, և նախագիծը պետք է բերված լինի որոշակի վիճակի: Ենթադրվում է, որ ծրագծման կանոնները և սահմանափակումները պահպանված են OA տեխնոլոգիական տվյալների հենքում: Ենթադրվում է, որ տվյալների հենքում է պահվում նաև միջմիացումների մասին ինֆորմացիա: Անհրաժեշտ է display.drf փաստաթուղթը, որը պարունակում է ծրագծման շերտերի մասին ինֆորմացիա և նախագծում առկա բջիջների գրադարանը նկարագրող lib.defs փաստաթուղթը:

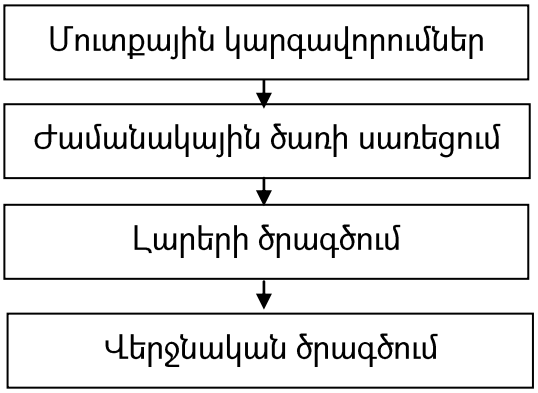
Կարևորագույն փաստաթղթերից մեկը AAR.py փաստաթուղթն է, որն անհրաժեշտ է գործիքի աշխատանքի սկսման և մուտքային փոփոխականների փոխանցման համար:

3.3.2. Ավտոմատ ծրագծում AAR Designer գործիքում մշակված run_route_auto հրամանից օգտվելու դեպքում

AAR Designer-ում նախագծվել է մի հրաման, որն ավտոմատ կերպով կատարում է նախնական ծրագծում, DRC և LVS ստուգումներ և վերջնական ծրագծում:

Ավելի մանրամասն ծանոթանանք AAR Designer-ի run_route_auto հրամանին: Այս հրամանը նախատեսված է աշխատելու ցանկացած նախագծի համար՝ անկախ հատակագծից, անկախ սխեմատեխնիկական նկարագրությունից և տեխնոլոգիական գրադարանից, և այն բավականաչափ կարճ ժամանակում կատարում է

բավականաչափ բարձր որակի ծրագծում: run_route_auto հրամանի հիմնական գործողությունների հաջորդականությունը բերված է նկ. 3.18 - ում:



Նկ. 3.18. run_route_auto հրամանի հաջորդական քայլերի բլոկ սխեման

Ավելի մանրամասն դիտարկենք ամեն քայլում կատարվող գործողությունները:

- Մուտքային կարգավորումներ քայլում տրվում են համապատասխան մակաբուծային և հապաղման կարգավորումները:
- «Ժամանակային ծառի սառեցում» քայլում ստուգվում է ժամանակային լարերի ամբողջականությունը: Եթե կան ոչ լիարժեք միացումներ, այս քայլում սկզբում կատարվում են այդ միացումները: Եթե ժամանակային ծառն ամբողջական է, այս փուլը բաց է թողնվում և անցում է կատարվում հաջորդ փուլին:
- «Լարերի ծրագծում» քայլում կատարվում է ազդանշանային գծերի ծրագծում: Այս քայլում run_route_auto հրամանը կանչում է route_track հրամանին: Վերջինս կատարում է ժամանակային ծրագծում, կրիտիկական լարերը տեղաբաշխվում են միմյանցից մեծ հեռավորությունների վրա, և խուսափում է երկար զուգահեռ լարերի հատվածներից:
- Վերջնական ծրագծում կատարվում է, որպեսզի ուղղվեն LVS և DRC սխալները: Այս փուլում կատարվում է հետևյալ հրամանների հաջորդականությունը. route_final -mode full_drc, check_drc, check_lvs:

run_route_auto հրամանը կանչելուց առաջ պետք է համոզված լինել, որ ապահովված են հետևյալ պայմանները.

- նախագիծը պետք է հատակագծման փուլը վերջացրած լինի,
- նախագծում պետք է վերջացված լինի սնուցման ազդանշանների ծրագծումը,

- բոլոր մուտքի-ելքի օղակները պետք է տեղադրված լինեն նախագծի եզրագծին:

Դիտարկենք run_route_auto հրամանի ստացվող արգումենտները և օգտագործման մի քանի օրինակ:

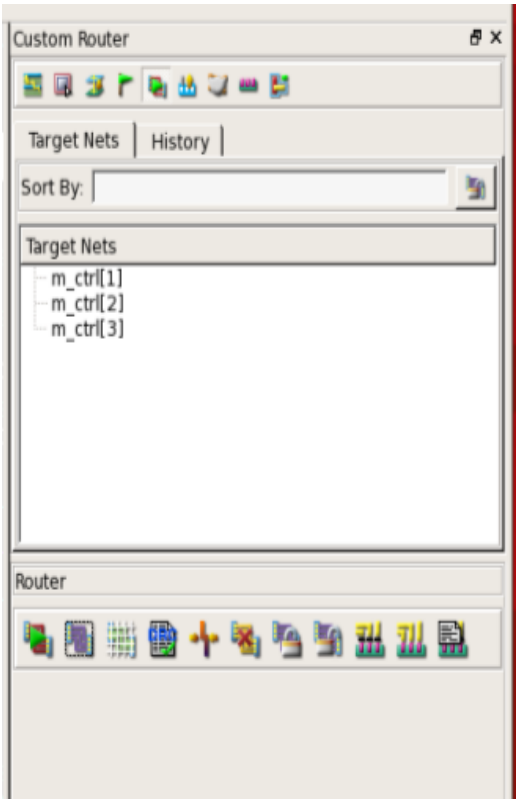
run_route_auto : [-cpus] [-preserve_clocks], այստեղ cpus արգումենտը որոշում է, թե քանի պրոցեսոր օգտագործել, և նրա լռելյայն արժեքը 4 է, իսկ preserve_clocks արգումենտը որոշում է՝ արդյոք պետք է կատարել ժամանակի ազդանշանների ծրագծում, թե ոչ, այն ստանում է {false, true, auto} արժեքները, և նրա լռելյայն արժեքը auto –ն է:

Դիտարկենք run_route_auto հրամանի օգտագործման հետևյալ օրինակները:

- Կատարել ծրագծում 8 պրոցեսորով և միժամանակ կատարել ժամանակային շղթայի ծրագծում՝ run_route_auto –cpus 8 – preserve_clocks false:
- Կատարել ծրագծում 4 պրոցեսորով և ժամանակային շղթային ձեռք չտալ՝ run_route_auto -preserve_clocks true:

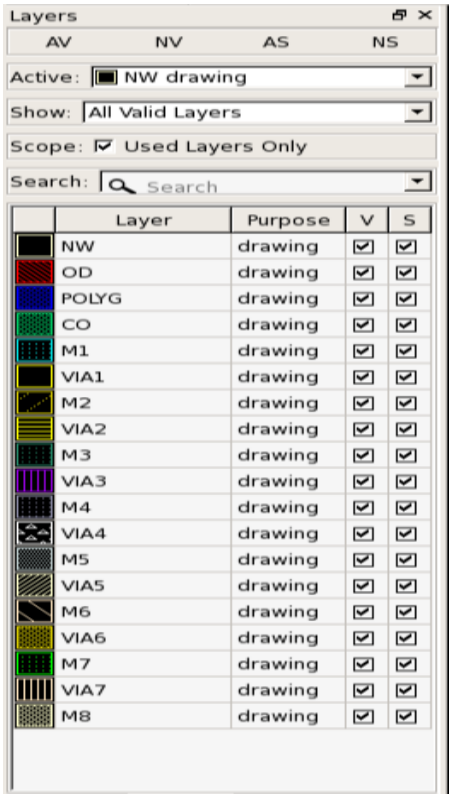
3.3.3. Անալոգային ԻՍ-երի ավտոմատացված ծրագծման համար մշակված AAR Designer գործիքի գրաֆիկական պատկերումը

AAR Designer-ի գրաֆիկական պատկերումը ներբեռնելուց հետո բացվում է նկ. 3.19 - ում պատկերված գործիքների վահանակը: Այստեղ Target Nets պատուհանում երևում են նախագծում ընտրված լարերը, որոնց ծրագծումը պետք է կատարի AAR Designer – ը, պատուհանում բերված են նաև հիմնական գործողություններ կատարող բոլոր կոճակները:



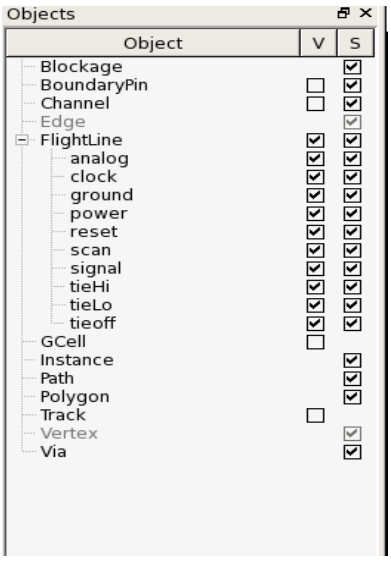
Նկ. 3.19. AAR Designer գործիքի ծրագծման հիմնական վահանակը

Հաջորդ կարևոր վահանակը շերտերի վահանակն է (նկ. 3.20). այստեղ պատկերվում են display.drf փաստաթղթում նկարագրված ծրագծման շերտերը: Ծրագծում կատարելու համար նախագծողն այդտեղից է ընտրում որևէ շերտ: Շերտերի գույները այստեղ հնարավոր չէ փոխել. դրանք ներբեռնվում են միայն մեկ անգամ, երբ ներբեռնում ենք display.drf փաստաթուղթը: Օգտագործելով շերտերի վահանակը՝ նախագծողը հնարավորություն ունի նախագծում շերտը դարձնել անտեսանելի կամ անընտրելի:



Նկ. 3.20. Շերտերի ընտրման վահանակը

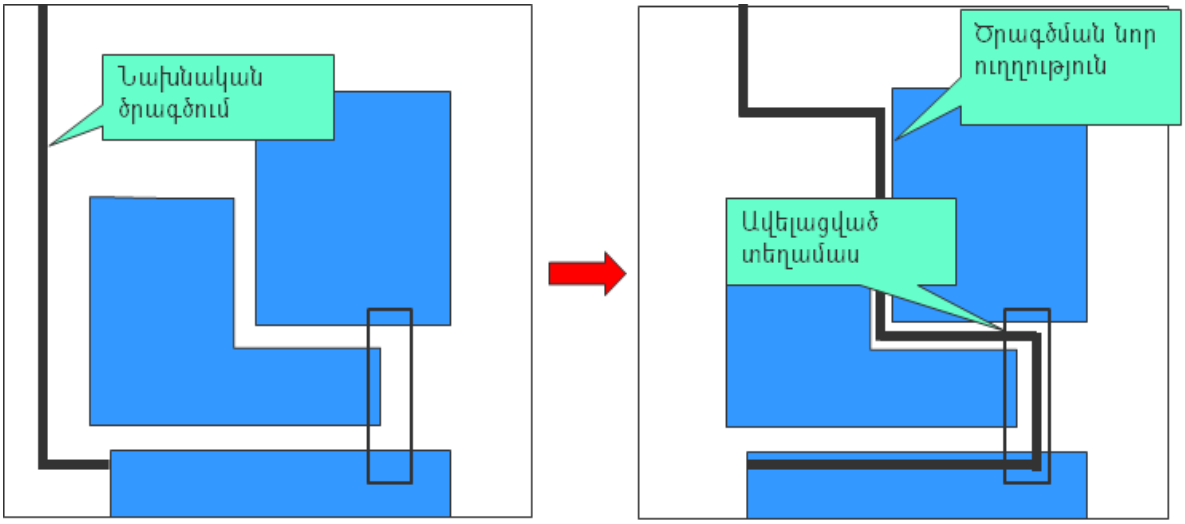
Հաջորդ վահանակը օբյեկտների վահանակն է. այստեղ պատկերված են նախագծում օգտագործվող բոլոր տիպի օբյեկտները, և նախագծողը հնարավորություն ունի փոխել օբյեկտների տեսանելիությունը, ընտրելով յուրաքանչյուր օբյեկտի համար տեսանելիություն ընտրման կոճակը (նկ. 3.21): Նախագծողը հնարավություն ունի նաև փոխել օբյեկտների ընտրման ակտիվացումը, նշելով “S” կոճակը ընտրված օբյեկտի համար:



Նկ. 3.21. Օբյեկտների վահանակի գրաֆիկական պատկերումը

Ծրագծում կատարելիս ավտոմատ ծրագծման գործիքները պետք է հետևեն որոշակի կանոնների [80, 81]: Այդ կանոնների լռելյայն արժեքները տրվում են տեխնոլոգիական փաստաթղթում: Սակայն հնարավոր է, որ նախագծողին անհրաժեշտ լինի այն լարերը, որոնցով անցնում է մեծ հոսանք, ծրագծվեն ավելի հաստ մետաղով: Այս դեպքում նախագծողը յուրաքանչյուր ծրագծման լարի համար սահմանում է լարի հաստություն սահմանափակումը և AAR Designer – այդ լարերի ծրագծումը կատարում է տրված հաստություններով:

AAR Designer – ը լարի ծրագծում կատարում է այն հատվածում, որն ըստ ծրագրում ներդրված ալգորիթմի լավագույնն է, սակայն որոշ դեպքերում ելնելով նախագծի առանձնահատկություններից, նախագծողը հնարավորություն ունի նաև ստիպել AAR Designer գործիքին կոնկրետ լարի համար ծրագծում կատարել այն ֆիզիկական նախագծի տեղամասով, որտեղով նա ցանկանում է: Դրա համար պետք է նախագծի այդ հատվածում ավելացնել ծրագծման հատված տարրը (նկ. 3.22):



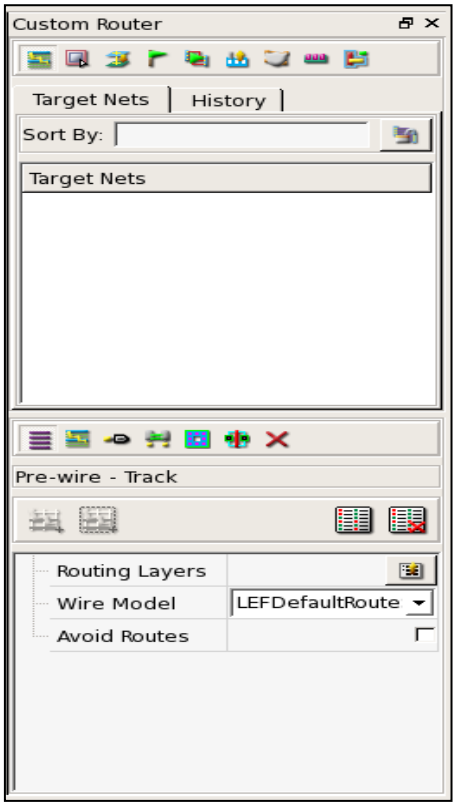
Նկ. 3.22. Նախագծողի կողմից ծրագծման հետագծի ընտրումը

AAR Designer գործիքը նախքան ծրագծումը կատարում է նախածրագծում կոչվող գործողությունը: Այս քայլի ժամանակ որոշվում են յուրաքանչյուր լարի օպտիմալ ծրագծման ուղիները: Նախածրագծման ժամանակ յուրաքանչյուր լար ծրագծվում է՝ առանց հաշվի առնելու նախագծման կանոնները և սահմանափակումները և յուրաքանչյուր լարի համար անկախ մնացած լարերից գտնում է լավագույն ծրագծման ուղին: Քանի որ ծրագծման ալգորիթմի կատարման ժամանակի մեծ մասը զբաղեցնում

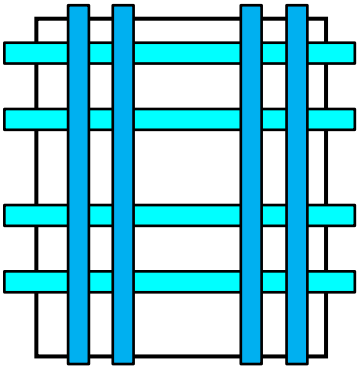
է սահմանափակումների և նախագծման կանոնների ստուգումը, հետևաբար նախածրագծումը կատարվում է շատ արագ, և այն չի պահանջում համակարգչային մեծ ռեսուրսներ:

AAR Designer գործիքում նախածրագծում կատարվում է հատուկ կոճակի օգնությամբ, որով բացվում է նախածրագծման վահանակը (նկ. 3.23):

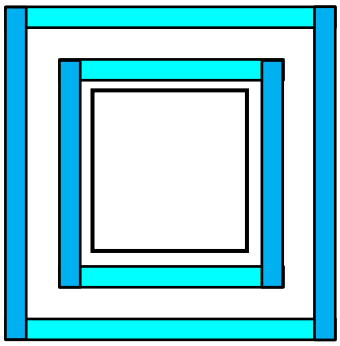
AAR Designer գործիքում նախածրագծման փուլում կատարվում է նաև սնուցման շղթայի կառուցում: Այն կատարվում է այս փուլում, համոզված լինելու համար, որ սնուցման շղթայի կառուցմանը ոչ մի այլ լար չի խանգարում: Կարելի է կառուցել երկու տիպի սնուցման շղթա՝ ցանցաձև (նկ. 3.24) և օղակաձև (նկ. 3.25):



Նկ. 3.23. Նախածրագծման համար նախատեսված վահանակը



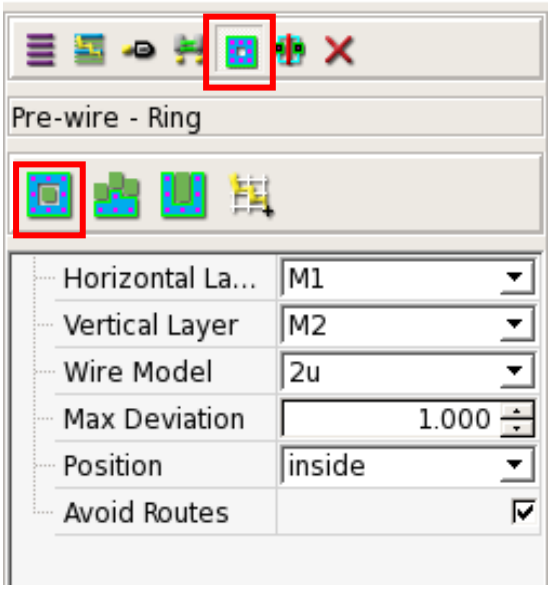
Նկ. 3.24. Ցանցաձև սնուցման շղթայի կառուցումը



Նկ. 3.25. Օղակաձև սնուցման շղթայի կառուցումը

Եթե բջջում սնուցման VDD և VSS ելուստներ չեն հայտնաբերվել, տվյալ բջջում սնուցման շղթա չի կառուցվում:

Դիտարկենք ավելի մանրամասն օղակաձև տիպի սնուցման շղթայի կառուցումը: Նախածրագծման գործիքների վահանակից պետք է ընտրել «օղակ» կոճակը, այնուհետև՝ համապատասխան պարամետրերը և սեղմել «կառուցել» կոճակը (նկ. 3.26):



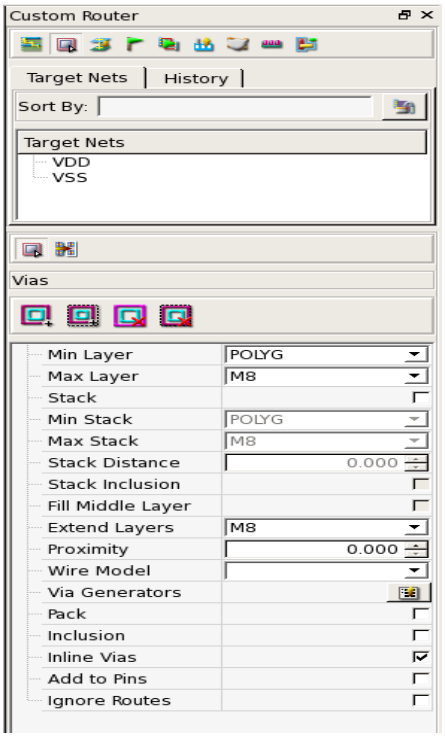
Նկ. 3.26. Օղակաձև սնուցման շղթայի կառուցումը

AAR Desigher գործիքը հատուկ պատուհան ունի միջմիացումների տեղադրման համար: Դրա համար պետք է կատարել հետևյալ հաջորդական քայլերը (նկ. 3.27).

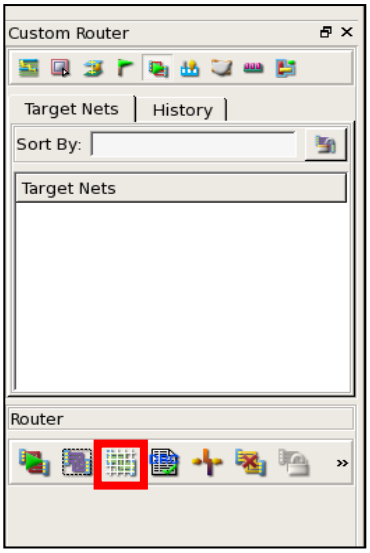
- ընտրել այն լարերը, որոնց վրա պետք է տեղադրվեն միջմիացումները,
- AAR Designer գործիքում սեղմել միջմիացումների կոճակը,

- սեղմել «ներդնել» կոճակը,
- ընտրել տեղադրվող միջմիացումների համապատասխան պարամետրերը,
- սեղմել տեղադրել միջմիացումներ կոճակը:

Մինչ մանրամասն ծրագծման (detail routing) քայլին անցնելը պետք է կատարել գլոբալ ծրագծում (global routing): Այս քայլը կարելի է նաև բաց թողնել, սակայն երբ արդեն կատարված է գլոբալ ծրագծում, մանրամասն ծրագծումը կատարվում է շատ արագ և արդյունավետ: Գլոբալ ծրագծման համար պետք է AAR Designer գործիքում սեղմել «գլոբալ ծրագծում» կոճակը (նկ. 3.28):



Նկ. 3.27. Միջմիացումների տեղադրում՝ օգտագործելով AAR Designer գործիքը



Նկ. 3.28. AAR Desginer-ի գլոբալ ծրագծման վահանակը

Գլոբալ ծրագծման ավարտին AAR Designer-ը վերադարձնում է գործողության ավարտի մասին տեղեկատվություն:

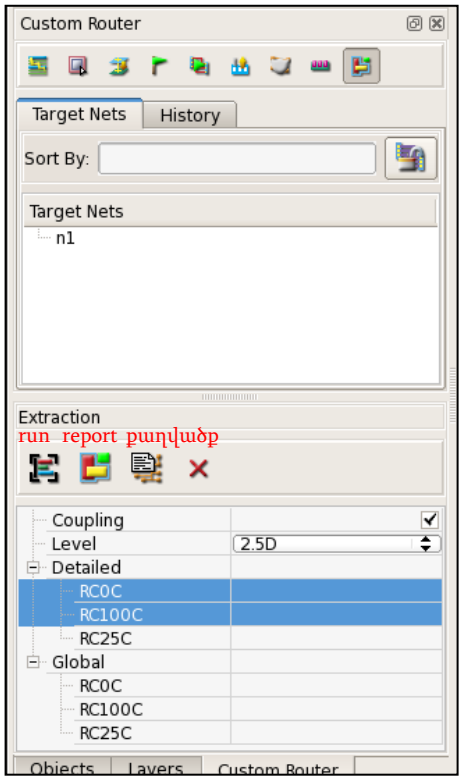
<2015-10-28 T 10:45:38>: Processing 172/172 nets, 420/420 connections

<2015-10-28 T 10:47:12>: Fully routed nets: 172

<2015-10-28 T 10:47:13>: Finished global routing:

Ֆիզիկական ծրագծումն առաջացնում է մակաբույծ ունակություններ և դիմադրություններ, որոնք բացասաբար են ազդում նախագծի պարամետրերի վրա: Այդ մակաբույծ տարրերը հաշվում են՝ ելնելով ծրագծման մետաղների պարամետրերից և ամեն լարի երկարությունից:

AAR Designer գործիքում մակաբույծ տարրերի քաղվածք ստանալու համար պետք է կատարել հետևյալ քայլերը. ընտրել այն լարերը, որոնց համար պետք է ստանալ քաղվածքը, այնուհետև AAR Designer գործիքում սեղմել «քաղվածք» կոճակը (նկ. 3.29): Բացված ենթապատուհանում պետք է ներմուծել քաղվածքի անունը և սեղմել OK կոճակը: Այնուհետև պետք է սեղմել run կոճակը, որը կսկսի մակաբույծ տարրերի քաղվածքի դուրսբերումը: Գործողության ցանկացած պահի կարելի է տեսնել արդյունքները՝ սեղմելով «report» կոճակը:



Նկ. 3.29. AAR Designer գործիքում մակաբույծ քաղվածքի ստացումը

Մակաբույծ տարրերի քաղվածքն ունի նկ. 3.30- ում բերված տեսքը:

```
########################################################################
Extraction Statistics  Thu Jul 18 16:24:38 2015

  cell:              div_by_m
  view:              routed
  analysis_point:    final_route_typical
  Total nets:        41
  Extracted nets:    41
  Resistors:         481
  Nodes:             522
  Total CouplingCap: 0 pf
  Total GroundedCap: 0.136015 pf
  Total Capacitance: 0.136015 pf
  Total Resistance:  306.227 ohms

########################################################################

NET                          Ctotal(pF)     Ccoupling(pF)  Cgnd(pF)       R(ohms)
------------------------------------------------------------------------------------
4-BIT_COMPARE1/N$144          0.00055977     0              0.00055977     0.76203
4-BIT_COMPARE1/N$3            0.00089271     0              0.00089271     1.5861
4-BIT_COMPARE1/N$46           0.00089271     0              0.00089271     1.5861
4-BIT_COMPARE1/N$47           0.0021214      0              0.0021214      3.6252
4-BIT_COMPARE1/N$58           0.001909       0              0.001909       3.9913
4-BIT_COMPARE1/N$59           0.0034775      0              0.0034775      7.9707
4-BIT_COMPARE1/N$6            0.0018973      0              0.0018973      3.5842
4-BIT_COUNTER1/Din[0]         0.0002521      0              0.0002521      0.57257
4-BIT_COUNTER1/Din[1]         0.0021379      0              0.0021379      4.5076
4-BIT_COUNTER1/Din[2]         0.002287       0              0.002287       4.8496
4-BIT_COUNTER1/Din[3]         0.0014771      0              0.0014771      2.9265
4-BIT_COUNTER1/N$471          0.0087739      0              0.0087739      20.203
4-BIT_COUNTER1/N$475#0        0.0077454      0              0.0077454      19.061
4-BIT_COUNTER1/N$475#1        0.002988       0              0.002988       6.7446
4-BIT_COUNTER1/N$475#2        0.0029885      0              0.0029885      6.77
4-BIT_COUNTER1/N$475#3        0.0046932      0              0.0046932      11.278
4-BIT_COUNTER1/N$793          0.00090204     0              0.00090204     1.8676
```

Նկ. 3.30. Մակաբույծ տարրերի քաղվածքի տեսքը

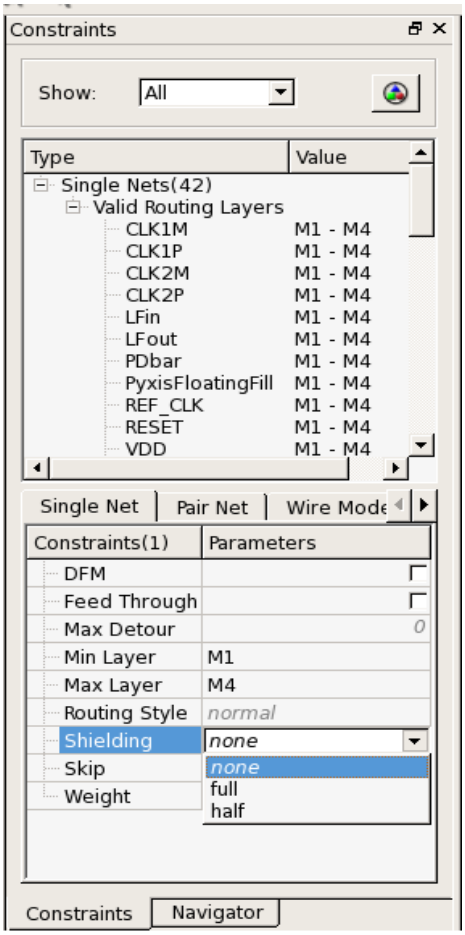
3.4. Անալոգային ԻՍ-երի ֆիզիկական նախագծման ժամանակ երկրորդական երևույթների նվազեցումը AAR Designer գործիքի օգնությամբ

Առաջին և երկրորդ գլուխներում հետազոտված երկրորդական երևույթների նվազեցման լուծումները և ալգորիթմները ներդրվել են AAR Designer գործիքի մեջ: Ստորև դիտարկենք այդ լուծումները մշակված գործիքային միջավայրում:

3.4.1. Էկրանավորման ավելացումը

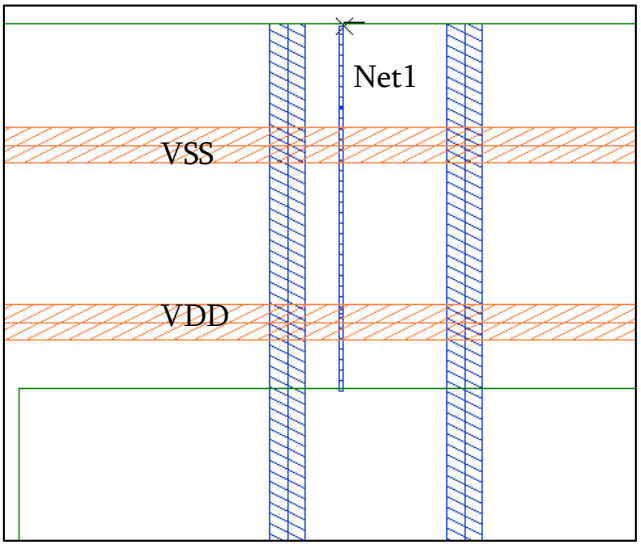
Նախագծողը հնարավորություն ունի սահմանելու «էկրանավորում» սահմանափակում և հրահանգ տալ AAR Designer գործիքին, որ այն ազատ տարածք հատկացնի լարերի էկրանավորման համար: Էկրանավորումը կատարվում է այն

բանից հետո, երբ բոլոր լարերը ծրագծվել են: Նկ. 3.31 – ում պատկերված է էկրանավորման սահմանափակման սահմանումը Constraint Designer գործիքում:



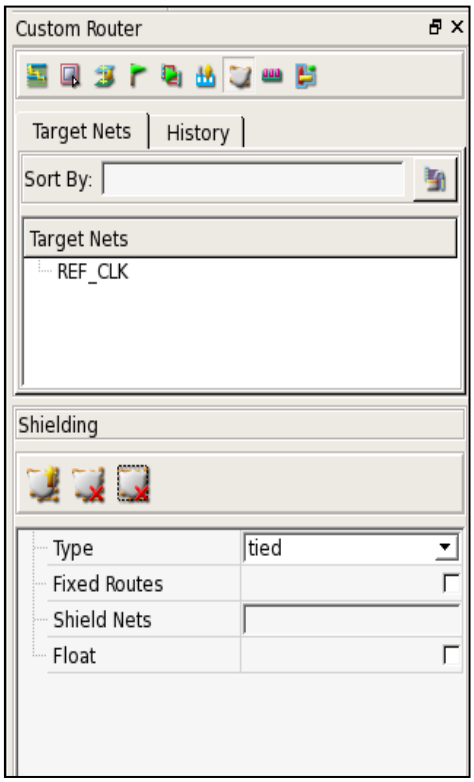
Նկ. 3.31. «Էկրանավորում» սահմանափակման կիրառումը

Դիտարկենք նկ. 3.32 - ում պատկերված ֆիզիկական նախագծի տեղամասը, որտեղ Net1 լարի վրա պետք է ավելացնել էկրանավորում:



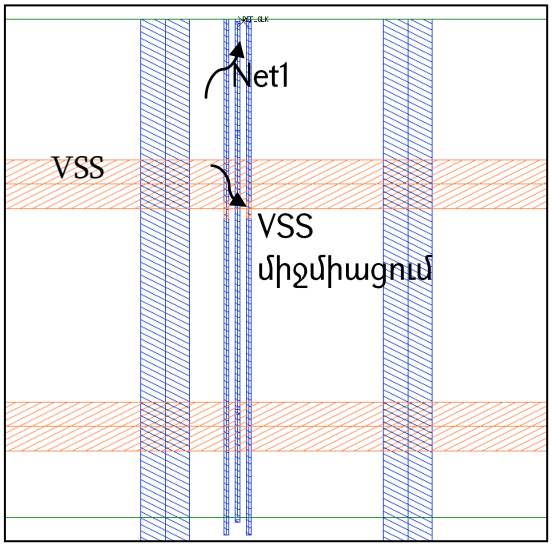
Նկ. 3.32. Էկրանավորում կատարելու Ֆիզիկական նախագծի տեղամասը

Էկրանավորում ավելացնելու համար AAR Designer գործիքում պետք է ընտրել էկրանավորում ենթապատուհանը, ընտրել այն լարը, որի վրա պետք է ավելացնել էկրանավորում և սեղմել «կատարել» կոճակը (նկ. 3.33):



Նկ. 3.33. AAR Designer գործիքում էկրանվորում ավելացնելու պատուհանը

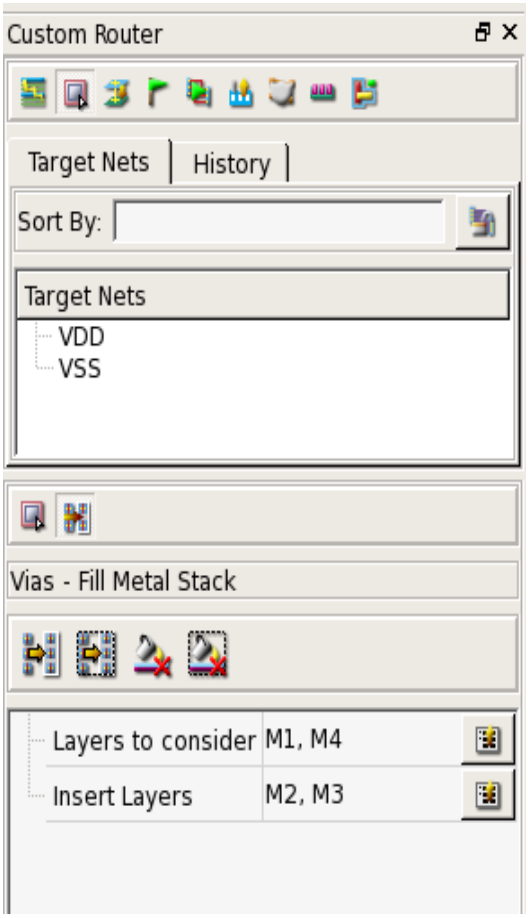
Էկրանավորում ավելացնելուց հետո Net1 լարի երկու կողմում ավելանում են VSS սնուցման լարին միացված երկու լար (նկ. 3.34):



Նկ. 3.34. Net1 լարի էկրանավորման ավելացումը

3.4.2. Մետաղի լցնում ջերմաստիճանային գրադիենտից ազատվելու համար

ԻՍ-երի նախագծման ժամանակ ջերմաստիճանային գրադիենտից և նմանատիպ այլ երևույթներից խուսափելու համար պետք է ամբողջ ԻՍ-ի ծավալով մետաղի խտությունը նույնը լինի [58]: Դրա համար AAR Designer-ում նախատեսված է գործիքների վահանակ դատարկ մետաղի լցման համար (նկ. 3.35):



Նկ. 3.35. Մետաղի լցման վահանակը

AAR Designer գործիքը, ունենալով ծրագծված լարերի երկարությունները, մոտավորապես հաշվում է յուրաքանչյուր մետաղական շերտի խտության բաշխվածությունը ֆիզիկական նախագծի ամբողջ մակերեսով: Այնուհետև ընտրվում է ֆիզիկական նախագծի այն տեղամասը, որտեղ պետք է կատարել մետաղի լցնում, ընտրվում է մետաղական շերտը և պետք է սեղմել «կատարել» կոճակը, որից հետո գործիքը ավտոմատ կերպով կատարում է մետաղի լցնում գործողությունը:

3.4.3. Ֆիզիկական նախագծում անտենա-երևույթի ուղղումը AAR Designer ծրագրային գործիքի միջոցով

Անտենա-երևույթը փականի օքսիդի շերտի բարակելն է, որի պատճառը լիցքակիրների կուտակումն է մակաբույծ կոնդենսատորներում, որոնք միացված են այդ փականի օքսիդի շերտին: Սա տեղի է ունենում միայն ԻՍ-ի արտադրման ժամանակ, երբ օքսիդը դեռևս միացված չէ դիֆուզիոն շերտին [44]:

Անտենա կանոնները ստուգվում են AAR Designer գործիքի կողմից ծրագծման վերջնական փուլում, երբ կատարվում են վերջնական ստուգումները: Այս կանոնները սահմանվում են որպես մետաղի մակերեսի և փականի մակերեսի հարաբերություն:

Ստորև բերված է տեխնիկական փաստաթղթի անտենա-երևույթի սահմանման օրինակ.

LAYER m1

TYPE ROUTING; DIRECTION HORIZONTAL; PITCH 0.14; WIDTH 0.07; OFFSET 0.07; AREA 0.0215; ANTENNACUMDIFFAREARATIO PWL ((0 4900) (0.059 4900) (0.060 43027) (0.5 43228) (1 43456) (1.5 43684));

END m1:

Անտենա-երևույթից խուսափելու համար կա երկու հիմնական մեթոդ.

- ծրագծման մետաղների հաջորդականության փոփոխման եղանակով: Փականներին միացված լարերի հատվածները միացվում են ամենաբարձր մակարդակի ծրագծման մետաղական շերտին,
- դիոդների ներմուծման մեթոդ: Փականին մոտ գտնվող հատվածներում դիոդներ են տեղադրվում: Այս լուծումը կիրառվում է միայն այն դեպքում, երբ առաջին մեթոդով լուծել խնդիրը հնարավոր չէ:

AAR Designer-ում կա ընտրելու հնարավորություն. մեթոդներից որն է կիրառելի տվյալ հանգույցի համար:

Անտենա կանոններ սահմանելու համար պետք է օգտվել հետևյալ հրամանից՝ Set_antena_rule(layers=m1,value=5000,oxide=1,cululative=true):

Էկրանին բոլոր անտենա կանոնները պատկերելու համար օգտվում են Report_tech(display=antenna_rule) հրամանից:

Տեխնոլոգիական փաստաթղթում ֆիզիկական նախագծի բոլոր հանգույցները պետք է ունենան նկարագրված անտենա կանոններ: Ստորև բերված է այդպիսի նկարագրություններից մեկը:

PIN z

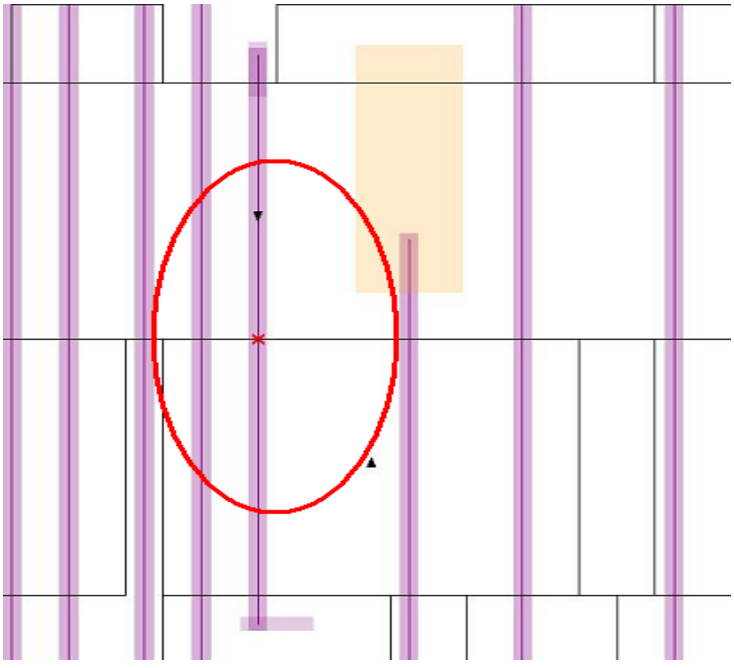
DIRECTION OUTPUT; LAYER m1; ANTENNADIFFAREA 0.2072

END z

Դիտարկենք AAR Designer-ում անտենա-երևույթի խախտումները ուղղելու քայլերի հաջորդականությունը.

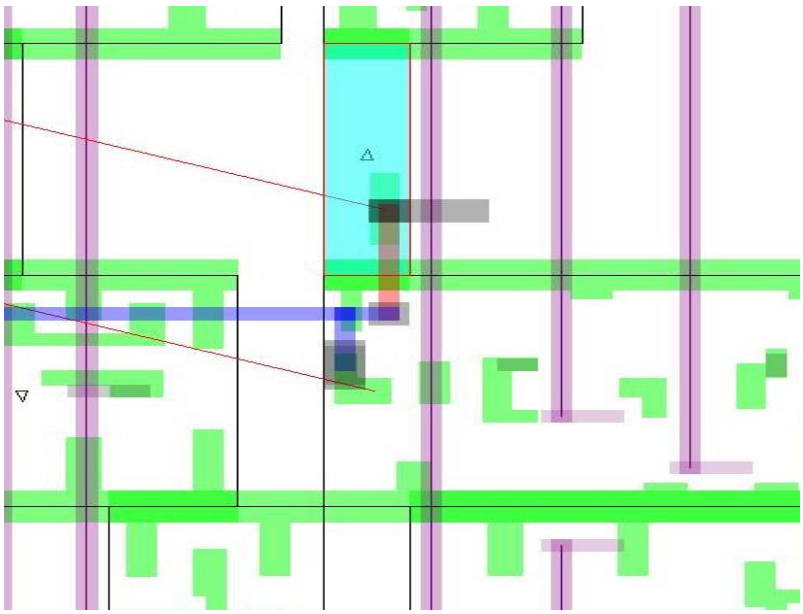
- սահմանվում են անտենա-երևույթի սահմանափակումները,
- ստուգվում են՝ արդյոք առկա են անտենա-երևույթի խախտումներ Check_antenna() հրամանով,
- ուղղվում են այդ խախտումները Fix_antenna(method=diode) հրամանով,
- կատարվում է վերջնական ծրագծում Route_final() հրամանով:

Նկ. 3.36.-ում բերված է ֆիզիկական նախագծի մի հատված, որտեղ առկա է անտենա-երևույթի խախտում:



Նկ. 3.36. Անտենա-երևույթի խախտմամբ ֆիզիկական նախագիծը

Այս խախտումը կարելի է ուղղել՝ տեղադրելով դիոդներ մուտքային հանգույցի մոտ: Նկ. 3.37 – ում բերված է անտենա-երևույթի խախտման ուղղման օրինակ AAR Designer գործիքի fix_antenna(method=diode) հրամանով:



Նկ. 3.37. Դիոդների տեղադրման օգնությամբ անտենա-երևույթի խախտման վերացում

3.4.4. Մշակված AAR Designer գործիքի սկրիպտավորման միջավայրը և հրամանների ցուցակը

AAR Designer գործիքն ապահովում է Python սկրիպատավորման լեզուն: Ներկայացնենք AAR Designer գործիքում հաճախակի օգտագործվող հրամանների ցանկը:

- report_nets_parasitics - վերադարձնում է նշված լարերի համար մակաբույծ քաղվածքը: Օգտագործման եղանակը՝ report_nets_parasitics (net_names = ['netNames']), այստեղ net_names - այն լարերի անվանումներն են, որոնց համար պետք է գեներացնել քաղվածքը. այն տրվում է ցուցակի տեսքով:
- export_to_gds – արտահանում է ֆիզիկական նախագիծը՝ GDSII տեսքով:

Օգտագործման եղանակը՝ export_to_gds (cell='cellName', gds='gdsFile', lib='libName'), այստեղ 'cellName'-ը նախագծի անվանումն է, 'libName'-ը գրադարանի անվանումը, իսկ 'gdsFile' –ը արտահանվող փաստաթղթի անվանումն է:

- import_display – ներմուծում է ծրագծման շերտերի մասին ինֆորմացիա պարունակող display.drf փաստաթուղթը:

Օգտագործման եղանակը՝ import_display (display= 'display.drf'), այստեղ 'display.drf' - ը ներմուծվող փաստաթղթի անվանումն է:

- create_power_ring – ավելացնում է օղակաձև սնուցման շղթա:

Օգտագործման եղանակը՝ create_power_ring (horizontal_layer='horizLayer', location_rect=['x0', 'y0', 'x1', 'y1'], vertical_layer='vertLayer'), այստեղ horizontal_layer –ը սնուցման շղթայի հորիզոնական շերտն է, location_rect –ը շղթայի կոորդինատներն են, իսկ vertical_layer-ը՝ ուղղահայաց շերտը:

- close_design – փակում է նախագիծը և մաքրում հիշողությունը:

Օգտագործման եղանակը՝ close_design (design='design', lib='lib'), այստեղ design-ը նախագծի անվանումն է, իսկ lib-ը այն գրադարանի, որտեղ գտնվում է նախագիծը:

- generate_missed_vias – ֆիզիկական նախագծում գտնում է այն լարերը, որտեղ ելուստները ինչ-որ պատճառով ջնջնվել են, և ավելացնում է դրանք: Այս հրամանը արգումենտներ չի ստանում:

- load_design – AAR Designer – ի հիշողություն է ներբեռնվում նախագիծը հետագա ծրագծման համար:

Օգտագործման եղանակը՝ load_design (design='design', lib='lib', mode=['read|write'), այստեղ design-ը նախագծի անվանումն է, իսկ lib-ը այն գրադարանի, որտեղ գտնվում է նախագիծը, mode – ը սահմանում է նախագիծը բացելու ռեժիմը, այն կարող է լինել միայն կարդալու կամ փոփոխություններ կատարելու թույլատրման ռեժիմ:

- report_statistics – նշված փաստաթղթում լրացնում է նախագծի առկա վիճակը. քանի լար է ծրագծվել, ընդհանուր մակերես, ժամանակ և այլ անհրաժեշտ տվյալներ:

Օգտագործման եղանակը՝ report_statistics (file='reportFile'), այստեղ file-ը այն փաստաթղթի անվանումն է, որտեղ պետք է գրվեն տվյալները:

- save_design – պահպանում է նախագծում կատարված մանրամասն ծրագծման և լարերի ավելացման կատարված քայլերը Open Access տվյալների հենքում:

Օգտագործման եղանակը՝ save_design (design='design', lib='lib'), այստեղ design-ը նախագծի անվանումն է, իսկ lib-ը այն գրադարանի, որտեղ գտնվում է նախագիծը:

- set_routing_area – սահմանում է նախագծի այն հատվածը որտեղ պետք է կատարել ծրագծում:

Օգտագործման եղանակը՝ set_routing_area (area =['x0', 'y0', 'x1', 'y1'), այստեղ area – ն այն նախագծի տեղամասի կոորդինատներն է, որտեղ պետք է կատարել ծրագծումը:

- connect_terms – կատարում է տրված երկու հանգույցի ծրագծում:

Օգտագործման եղանակը՝ connect_terms (connector_layer='connLayer', term1='term1' , term2='term2'), այստեղ connector_layer – ը ծրագծման շերտն է, իսկ term1 և term2 –ը՝ միմյանց միացվող հանգույցները:

- report_drc_count – վերադարձնում է նախագծում առկա բոլոր լարերի վրա նախագծման կանոնների խախտումը:

Օգտագործման եղանակը՝ get_drc_count (), այս հրամանը արգումենտներ չի ստանում:

- report_failed_nets – վերադարձնում է այն լարերի ցուցակը որոնց ծրագծումը ձախողվել է:

Օգտագործման եղանակը՝ report_failed_nets (file='fileName') , այստեղ file-ը այն փաստաթղթի անվանումն է որտեղ պետք է գրվեն այն լարերի տվյալները որոնց ծրագծումը ձախողվել է:

- run_detailed_router – կատարում է նախագծի մանրամասն ծրագծում:

Օգտագործման եղանակը՝ run_detailed_router (area=['x1','y1','x2','y2'], fail_file='fail_file'), այստեղ area – ն նախագծի այն հատվածն է, որտեղ պետք է կատարվի մանրամասն ծրագծում, fail_file – ը այն փաստաթղթի անվանումն է, որտեղ մանրամասն ծրագծման ձախողման դեպքում կգրվեն ձախողման պատճառները:

- run_global_router – կատարում է նախագծի գլոբալ ծրագծում. այս հրամանը արգումենտներ չի ստանում:
- run_net_shielding – կատարում է ընտրված լարերի էկրանավորում:

Օգտագործման եղանակը՝ run_net_shielding (nets='NetNames', type='Type'), այստեղ nets –ը այն լարերն են, որոնց վրա AAR Designer գործիքը պետք է ավելացնի էկրանավորում, իսկ type – ը՝ էկրանավորման տիպը:

- check_connectivity – ստուգում է լարերի ճշգրիտ միացումները:

Օգտագործման եղանակը՝ check_connectivity (nets='netNames'), այստեղ nets – ը այն լարերն են, որոնց վրա կատարվում է ստուգում և յուրաքանչյուր լարի համար վերադարձվում է PASS կամ FAIL:

- set_routing_layer – AAR Designer-ին ստիպում է ծրագծում կատարել տրված ծրագծման շերտով:

Օգտագործման եղանակը՝ set_routing_layer (layer='name', net='netName', spacing='minSpacing', width='minWidth'), այստեղ layer-ը ծրագծման շերտն է, net-ը այն լարն է, որի ծրագծումը պետք է կատարվի, spacing-ը տվյալ շերտով ծրագծման լարերի միջև նվազագույն հեռավորությունն է, իսկ width –ը՝ ծրագծման լարի նվազագույն լայնությունը:

- delete_router_constraint – ընտրված լարի վրայից հեռացնում է սահմանափակումը:

Օգտագործման եղանակը՝ delete_router_constraint (constraint='ConstraintType', net='netNames'), այստեղ constraint-ը այն սահմանափակումն է որը պետք է հեռացվի net լարի վրայից, եթե constraint –ին արժեք չի տրվում, ապա AAR Designer – ը հեռացնում է տրված լարի վրա սահմանված բոլոր սահմանափակումները:

- delete_shielded_wires – նախագծում հեռացնում է տրված լարի վրա կիրառված էկրանավորումը:

Օգտագործման եղանակը՝ delete_shielded_wires (area='area'), այստեղ area – ն ֆիզիկական նախագծի այն տեղամասն է, որտեղ գտնվող լարերի վրայից պետք է կատարել էկրանավորման հեռացումը:

- delete_routing – նախագծի տրված հատվածում հեռացնում է լարերի և ելուստների ծրագծումը:

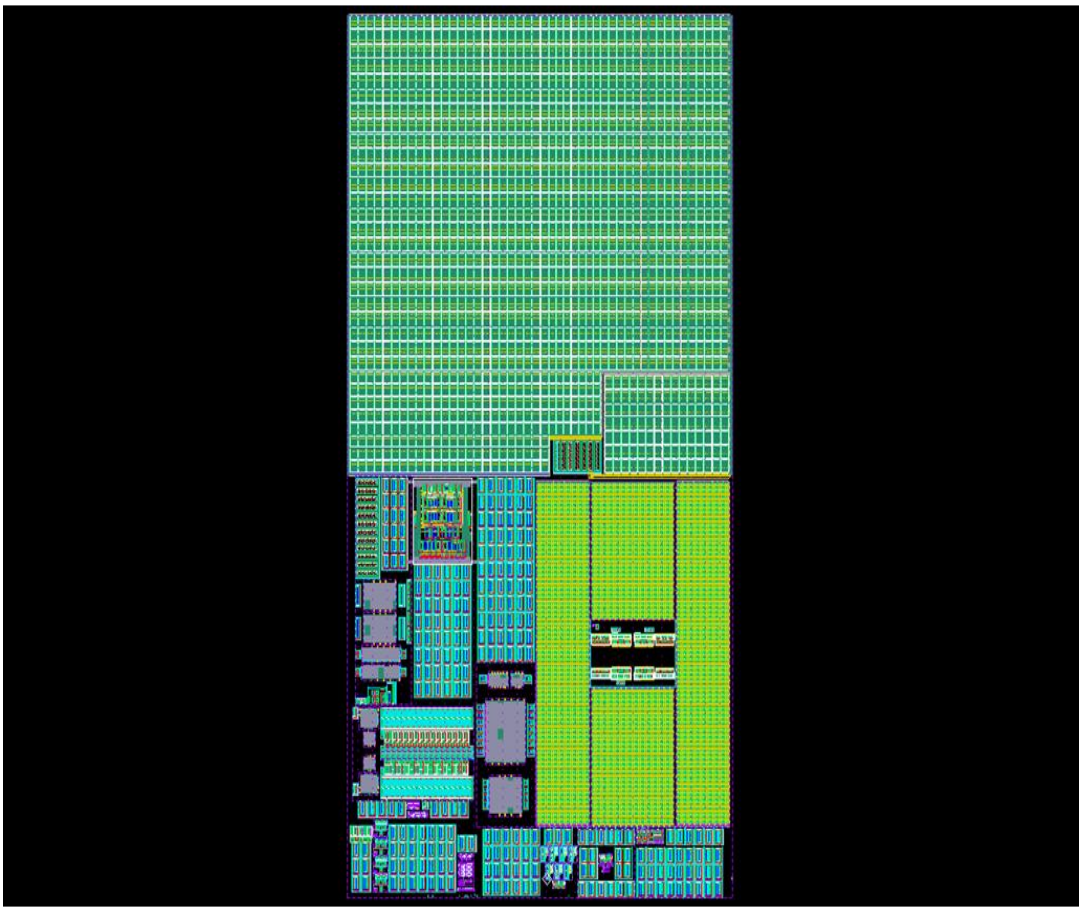
Օգտագործման եղանակը՝ delete_routing (design='cellName', lib='libName', area='cellArea'), այստեղ design-ը նախագծի անվանումն է, իսկ lib-ը այն գրադարանի, որտեղ գտնվում է նախագիծը, area – ն ֆիզիկական նախագծի այն տեղամասն է, որտեղից պետք է հեռացվի կատարված ծրագծումը:

- ignore_blockages – AAR Designer – ը անտեսում է սահմանված տեղամասում արգելափակումները:

Օգտագործման եղանակը՝ ignore_blockages (cell='cellName', lib='libName', area='cellArea'), այստեղ design-ը նախագծի անվանումն է, իսկ lib-ը այն գրադարանի, որտեղ գտնվում է նախագիծը, area – ն՝ ֆիզիկական նախագծի այն տեղամասը, որի ծրագծումը AAR Designer – կատարում է հաշվի չառնելով արգելափակումները:

3.5. Անալոգային ինտեգրալ սխեմաների ավտոմատացված ֆիզիկական նախագծման համար մշակված ծրագրային միջոցի արդյունավետության գնահատումը

Առաջարկված AAR Designer ծրագրային միջոցի արդյունավետությունը գնահատելու համար այն թեստավորվել է ՓԻՀ-ի ֆիզիկական նախագծի վրա: Վերցվել է նախօրոք տեղադրում անցած ՓԻՀ նախագիծ, կիրառվել են համապատասխան սահմանափակումները և կատարվել է ձեռքով և ավտոմատացված ծրագծում՝ օգտագործելով AAR Designer գործիքը: Այնուհետև AAR Designer գործիքի արդյունավետությունը գնահատելու համար կատարվել է համեմատական վերլուծություն այդ երկու ֆիզիկական նախագծերի միջև: AAR Designer գործիքով ծրագծված ՓԻՀ-ի ֆիզիկական նախագիծը բերված է նկ. 3.38 - ում:



Նկ. 3.38. ՓԻՀ-ի ֆիզիկական նախագիծը

Աղ. 3.1.-ում բերված է երկու ֆիզիկական նախագծերի և սխեմատեխնիկական նախագծման արդյունքների համեմատությունը:

Աղյուսակ 3.1

Ավտոմատ և ձեռքով նախագծման տարբերությունները

Բնութագրեր	Սխեմատեխնիկական	Ձեռքով ծրագծում	AAR Designer ծրագծում	Միավոր
Ելքային հաճ.	1200	1200	1200	*ՄՀց*
Մուտքային հաճ.	25	25	25	*ՄՀց*
Փուլային սխալ	700	734	720	*պվ*
Թրթռոց	12	14	13	*պվ*
Թվային սնման հոսանք	0,3	0,3	0,3	*մԱ*
Անալոգային սնման հոսանք	0,12	0,12	0,12	*մԱ*
Բաց թողնման շերտ	1	1	1	*ՄՀց*
Հաստատման ժամանակ	1,9	2,7	2,9	*մկվ*
Մուտքային ազդանշանի լցման գործակից	40	50	50	*%*
Ելքային ազդանշանի լցման գործակից	45	50	50	*%*
Մակերես	-	82900	93700	*Մկմ²*

Եզրակացություններ

1. Մշակվել է C++ ծրագրավորման միջավայրում անալոգային ԻՍ-երի ավտոմատացված նախագծման համար նախատեսված սահմանափակումների ներմուծման Constraint Designer գործիքը՝ հիմնված Open Access տվյալների հենքի վրա, որի շնորհիվ կարելի է կատարել ֆիզիկական նախագծում առկա սահմանափակումների հետ գործողություններ՝ պահում, փոփոխություն, չեղարկում և այլն:

2. Մշակվել է սահմանափակումների գերակայությունը որոշող ալգորիթմ, որի շնորհիվ Constraint Designer գործիքը ավտոմատ կերպով որոշում է կայացնում, թե որ սահմանափակմանը նախապատվություն տալ. որոշ դեպքերում նախագիծը մի

նախագծողի կողմից չի կատարվում. հնարավոր է այնպիսի դեպք, որ նույն տարրի վրա սահմանված լինեն տարբեր տեսակի միմյանց հակասող սահմանափակումներ, այս դեպքում գործի է դրվում մշակված ալգորիթմը:

3. Անալոգալին ԻՍ-երի ավտոմատացված նախագծման եղանակները և մեթոդները իրագործվել են AAR Designer ծրագրային գործիքային միջավայրում, որն օժտված է օգտագործողի համար նախատեսված հեշտ և պարզ միջավայրով, ունի թեստավորմանը հնարավորինս հարմարեցված կառուցվածք: Շնորհիվ գործիքում կիրառված արագագործ ալգորիթմների, նախագծման գործընթացի տևողությունը կրճատվում է միջինը 20...40 անգամ:

4. AAR Designer ծրագրային միջոցի փորձարկումը ՓԻՀ համակարգերում հաստատում է դրա բարձր արդյունավետությունը՝ ծրագծման ժամանակը նվազել է մոտ 35 անգամ: Շնորհիվ տարրերի համապատասխանեցման, պաշտպանիչ օղակների կիրառման և ԳՄԵ-ի նվազեցման առաջարկված մեթոդների կիրառման, ապահովվել է նախագծի անհրաժեշտ ճշգրտությունը, կիսահաղորդչային բյուրեղի վրա զբաղեցրած մակերեսի ընդամենը 10,3% աճի հաշվին:

ԵԶՐԱՀԱՆԳՈՒՄ

1. Մշակվել է ռեզիստորների և կոնդենսատորների համապատասխանեցման նոր մեթոդ, որը ֆիզիկական նախագծում տարրերի հատվածների վրա արտաքին ազդեցությունների հավասարաչափ բաշխման շնորհիվ, ապահովում է ռեզիստորների և կոնդենսատորների համապատասխանեցման աճ: Մեր առաջարկած մեթոդը փորձարկվել է դիֆերենցիալ ուժեղարարի ֆիզիկական նախագծի վրա, և համապատասխանեցման մեջ դրված դիմադրությունների տարբերությունը նվազել է 35%-ով համեմատած դասական համապատասխանեցման մեթոդների հետ:

2. Առաջարկվել է ԳՄԵ-ի նվազեցման տոպոլոգիական իրականացում՝ փոխարինելով նախագծում առկա P-ՄՕԿ տրանզիստորները N-ՄՕԿ տրանզիստորներով: Նախագծվել է SAED32/28 *նմ* տեխնոլոգիական գործընթացի համար հոսանքի հայելու երկու ֆիզիկական նախագիծ, առաջինը P-ՄՕԿ տրանզիստորներով, երկրորդը՝ զուտ N-ՄՕԿ տրանզիստորներով: Վերը նշված մեթոդի կիրառման շնորհիվ գրպանիկի մոտիկության երևույթի ազդեցությունը նվազել է 10...100 անգամ:

3. Հետազոտվել է պաշտպանիչ օղակների կիրառությունը ՓԻՀ-երի ֆիզիկական նախագծերում: Գնահատվել է պաշտպանիչ օղակների ազդեցությունը ՓԻՀ-երի ելքային բնութագրերի վրա: Նախագծվել է երկու ՓԻՀ-ի ֆիզիկական նախագիծ, որոնցից մեկում կիրառվել են պաշտպանիչ օղակներ, իսկ մյուսում՝ ոչ: Նմանակումների արդյունքում հաստատվել է, որ պաշտպանիչ օղակների կիրառման շնորհիվ ՓԻՀ համակարգերում ստացվում է փուլային սխալի 15%, և թրթռոցի 20% նվազում, իսկ մակերեսն աճում է ընդամենը 1,9%:

4. Հետազոտվել է ջերմաստիճանային գրադիենտի ազդեցությունը ֆիզիկական նախագծի ելքային պարամետրերի վրա, առաջարկվել է ջերմաստիճանային գրադիենտի ազդեցության նվազեցման եղանակ, որի շնորհիվ հնարավոր է դարձել իրականացնել ջերմաստիճանային գրադիենտի նկատմամբ կայուն լիցքային պոմպի ֆիզիկական նախագիծ:

5. C++ ծրագրավորման միջավայրում մշակվել է անալոգային ԻՍ-երի ավտոմատացված նախագծման համար նախատեսված սահմանափակումների ներմուծման Constraint Designer գործիքը՝ հիմնված Open Access տվյալների հենքի վրա, որի շնորհիվ կարելի է կատարել ֆիզիկական նախագծում առկա սահմանափակումների հետ գործողություններ՝ պահում, փոփոխություն, չեղարկում և այլն:

6. Մշակվել է անալոգալին ԻՍ-երի ավտոմատացված ծրագծման համար նախատեսված AAR Designer ծրագրային գործիքային միջավայրը, որում ներդրված են անալոգային ԻՍ-երի ֆիզիկական նախագծման ժամանակ երկրորդական երևույթների ազդեցության թուլացման առաջարկված սկզբունքները: Մշակված AAR Designer գործիքն ապահովում է նախագծման գործնական պահանջներին բավարարող էներգասպառում, կիսահաղորդչային բյուրեղի վրա զբաղեցրած փոքր մակերես և արդյունքների անհրաժեշտ ճշտություն, այն օժտված է օգտագործողի համար նախատեսված պարզ միջավայրով, ունի թեստավորմանը հնարավորինս հարմարեցված կառուցվածք: Շնորհիվ գործիքում կիրառված արագագործ ալգորիթմների, նախագծման գործընթացի տևողությունը կրճատվում է միջինը 20...40 անգամ:

7. Մշակված AAR Designer ծրագրային միջոցի փորձարկումը ՓԻՀ համակարգերում վկայում է դրա բարձր արդյունավետության մասին. ծրագծման ժամանակը նվազել է մոտ 35 անգամ: Շնորհիվ տարրերի համապատասխանեցման, պաշտպանիչ օղակների կիրառման և ԳՄԵ-ի նվազեցման առաջարկված մեթոդների կիրառման, ապահովվել է նախագծի անհրաժեշտ ճշգրտությունը, կիսահաղորդչային բյուրեղի վրա զբաղեցրած մակերեսի ընդամենը 10,3% աճի հաշվին:

ՕԳՏԱԳՈՐԾՎԱԾ ԳՐԱԿԱՆՈՒԹՅՈՒՆ

1. Agarwal K., Nassif S. Characterizing process variation in nanometer CMOS // Proceedings of the 44th Design Automation Conference.- 2007.- P. 396-399.
2. Alpaydin G., Balkir S., Dundar G. An Evolutionary Approach to Automatic Synthesis of High-Performance Analog Integrated Circuits // IEEE Transactions.- 2003.- Vol 7.- P. 240-252.
3. Antreich K., Eckmueller J., Graeb H., Pronath M. et al WiCkeD: analog circuit synthesis incorporating mismatch // Proceedings of the IEEE Custom Integrated Circuits Conference.- May, 2000.- P. 511-514.
4. Baker J.R. CMOS Circuit Design, Layout, and Simulation. - John Wiley and Sons, 2004.- P. 1080.
5. Barros M., Guilherme J., Horta N. Analog circuits optimization based on evolutionary computation techniques // Integration.- 2009.- Vol. 43, no. 1.- P. 136-155.
6. Barros M., Guilherme J., Horta N. State-of-the-art on analog design automation // Studies in Computational Intelligence.- 2010.- Vol. 294.- P. 19-47.
7. Borkar S., Karnik T., Tschanz J., et al Variations and impact on circuit and microarchitecture // IEEE Micro.- 2006.- P. 30-39.
8. Castro-Lopez R., Guerra O., Roca E., Fernandez F. An integrated layout-synthesis approach for analog ICs // IEEE Transactions on Computer-Aided Design of Integrated Circuits.- July, 2007.- P. 1179-1189.
9. Chang H., et al Top-Down Constraint Driven Methodology for Analog Integrated Circuits.- Kluwer, 1997.- P. 369.
10. Charbon E., Malavasi E., Sangiovanni-Vincentelli A. Generalized constraint generation for analog circuit design // Proceedings of the IEEE/ACM International Conference on Computer-Aided Design.- November, 1993.- P. 408-414.
11. Chen D., Sheu B. Generalised approach to automatic custom layout of analogue ICs, Circuits, Devices and Systems // IEE Proceedings G - Circuits, Devices and Systems.- August, 1992.- P. 481–490.

12. Chien Y., Huang L., Chen W., Mukherjee T., et al SPEED: Synthesis of High-Performance Large Scale Analog/Mixed Signal Circuit // IEEE Int'l Symp. on Technology, System and Applications, Design, Automation and Test (VLSI-TSADAT '05).- April, 2005.- P. 112-115.
13. Choudhury U., Sangiovanni-Vincentelli A. Automatic generation of parasitic constraints for performance-constrained physical design of analog circuits // IEEE Transactions on Computer-Aided Design of Integrated Circuits.- February, 1993.- P. 208–224.
14. Cohn J., et al KOAN/ANAGRAM II: New Tools for Device-Level Analog Placement and Routing // IEEE Journal of Solid-State Circuits.- March, 1991.- Vol. 26, No. 3.- P. 330-342.
15. Cohn J., Garrod D., Rutenbar R., Carley L. Analog Device-Level Layout Automation.- Boston, MA, Kluwer Academic Publishers, 1994.- P. 285.
16. Cook T., Brown J., Cottrell P., et.al Lateral ion implant straggle and mask proximity effect // IEEE Trans. Electron. Devices.- 2006.- Vol. 50.- P. 1946-1951.
17. Das A., Vemuri R. An Automated Passive Analog Circuit Synthesis Framework using Genetic Algorithms // IEEE Computer Society Annual Symposium on VLSI.- 2007.– P. 145-152.
18. De Smedt B., Gielen G. WATSON: Design Space Boundary Exploration and Model Generation for Analog and RF IC Design // Proc IEEE.- February 2003.- P. 213–224.
19. Dessouky M., Louerat M., Porte J. Layout-Oriented Synthesis of High Performance Analog Circuits // Design Automation and Test in Europe.- 2000.- P. 53–57.
20. Dimov B., Boos V., Reich, T., et al A novel technique for CAD-optimization of analog circuits with bipolar transistors // Advances in Radio Science.- 2009.- Vol. 7.- P. 219-223.
21. Doboli A., et al Behavioral Synthesis of Analog Systems using Two - Layered Design Space Exploration // Proc. Design Automation Conference.- 1999.– P. 951-957.
22. Doboli A., Vemuri R. A Regularity Based Hierarchical Symbolic Analysis Method for Large scale Analog Networks // IEEE Trans. Circuits & Systems – II.- 2001.- Vol. 48.- P. 1054-1068.

23. Drennan P., Locascio D. Implications of proximity effects for analog design // Custom Integrated Circuits Conference.- 2006.- P. 169-176.
24. Eissa H., ElMously A. Physical aware design methodology for analog and mixed signal integrated circuits // IEEE International Design and Test Workshop.- 2009.- P. 1-5.
25. El-Turky F., Perry E. BLADES: an artificial intelligence approach to analog circuit design // IEEE Transactions on Computer-Aided Design of Integrated Circuits.- June, 1989.- P. 680-692.
26. Fakhfakh M. A novel Alienor-based heuristic for the optimal design of analog circuits // Microelectronics Journal.- 2009.- Vol. 40.- P. 141-148.
27. Faricelli J. Layout-Dependent Proximity Effects in Deep Nanoscale CMOS // IEEE CICC.- 2010.- P. 1-8.
28. Gielen G, Wambacq P, Sansen W. Symbolic analysis methods and applications for analog circuits // Proceedings of the IEEE.- 1994.- Vol. 82, P. 287–304.
29. Gielen G., et. al An Analogue Module Generator for Mixed Analogue/Digital ASIC Design // International Jorunal of Circuit Theory and Applications.- 1995.- Vol. 23.- P. 269-283.
30. Gielen G., McConaghy T., Eeckelaert T. Performance Space Modeling For Hierarchical Synthesis Of Analog Integrated Circuits // Proc. ACM/IEEE DAC.- 2005.- P. 881-886.
31. Gielen G., Rutenbar R. Computer-Aided Design Of Analog And Mixed-Signal Integrated Circuits // Proceedings of the IEEE.- 2000.- Vol. 88.- P. 1825-1854.
32. Giuseppe N., Salvatore R., Eva S. An evolutionary algorithm-based approach to robust analog circuit design using constrained multi-objective optimization // Knowledge-Based Systems.- April, 2008.- Vol 21.- P. 175-183.
33. Goldman R., Bartleson K., Wood T., Kranen K., Melikyan V., Babayan E. 32/28nm Educational Design Kit: Capabilities, Deployment and Future // Proceedings of the Asia-Pacific Conference on Postgraduate Research in Microelectronics & Electronics.- India, Visakhapatnam.- 2013.- P. 284-288.

34. Graeb H., Balasa F., Castro-Lopez R., et al Analog layout synthesis—recent advances in topological approaches // Proceedings on Design, Automation and Test in Europe.- 2009.- P. 274-279.
35. Graeb Helmut E. Analog Layout Synthesis, A Survey of Topological Approaches.- Springer, 2010.- P. 302.
36. Gray Paul R., Hurst Paul J., Lewis Stephen H., Meyer Robert G. Analysis and Design of Analog Integrated Circuits.- Wiley, 2009.- P. 896.
37. Habal H., Graeb H. Constraint-based layout-driven sizing of analog circuits // IEEE Transactions on Computer-Aided Design of Integrated Circuits.- August, 2011.- P. 1089–1102.
38. Hao Q., Dong S., Chen S., Hong X., et al Constraints generation for analog circuits layout // IEEE International Conference on Communications, Circuits and Systems.- June, 2004.- P. 1339–1343.
39. Harjani R., et. al OASYS: A Framework for Analog Circuit Synthesis // IEEE Trans. on Computer-Aided Design.- December 1989.- Vol. 8, No. 12.- P. 1247-1266.
40. Harjani R., Rutenbar R., Carey L. A prototype framework for knowledge-based analog circuit synthesis // Proc. 24th Design Automation Conf.- 1987.- P. 42–49.
41. Hastings A. The Art of Analog Layout.- Prentice Hall, 2001.- P. 648.
42. Hershenson M. Efficient Description of the Design Space of Analog Circuits // DAC 2003.- June, 2003.- P. 970–973.
43. Hook T., Brown J., Tian X. Proximity Effects and VLSI Design // ICICT.- 2007.- P. 167-170.
44. http://en.wikipedia.org/wiki/Antenna_effect
45. https://en.wikipedia.org/wiki/Charge_pump
46. https://en.wikipedia.org/wiki/Driven_guard
47. https://en.wikipedia.org/wiki/Electromagnetic_shielding
48. https://en.wikipedia.org/wiki/Phase_detector
49. https://en.wikipedia.org/wiki/Phase-locked_loop

50. Ismail M., Fiez T. Analog VLSI Signal and Information Processing.- Mc-Graw-Hill, 1994.-P. 741.
51. Jarafi A., Sadri S. Zekri M. Design optimization of analog integrated circuits by using artificial neural networks // International Conference of Soft Computing and Pattern Recognition.- 2010.- P. 385-388.
52. Jianhai Y., Zhigang M. Automated design method for parameters optimization of CMOS analog circuits based on adaptive genetic algorithm // 7th International Conference on ASIC.- 2007.- P. 1217-1220.
53. Johns D., Martin K. Analog Integrated Circuit Design.- New York, Wiley, 2011.- P. 816.
54. Kleine U., Zhang L., Wolf M. Mismatch Optimization for Analog Circuits Using the Design Assistant // Proc. IEEE International Conference on Electronics, Circuit and System.- September, 2002.- P. 1135-1138.
55. Krasnicki M., Phelps R., Rutenbar R., Carley L. MAELSTROM: Efficient Simulation-Based Synthesis for Custom Analog Cells // Proc. ACMIIEEE DAC.- June, 1999.- P. 945-950
56. Kumar D. V., et al Evaluation of the Impact of Layout on Device and Analog Circuit Performance with Lateral Asymmetric Channel MOSFETs // IEEE Trans. Elec. Dev.- July, 2005.- P. 1603-1609.
57. Lampaert K., Gielen G., Sansen W. Analog Layout Generation for Performance and Manufacturability.- Kluwer, 1999.- P. 175.
58. Lee B., Boning D., Hetherington D. Using smart dummy fill and selective reverse etchback for pattern density equalization // Chemical-Mechanical Polish for VLSI/ULSI Multilevel Interconnection Conference.- 2000.- P. 255-258.
59. Lohn J., Colombano S. A Circuit Representation Technique for Automated Circuit Design // IEEE Transactions.- September, 1999.- Vol. 3.- P. 205–219.
60. Malavasi E., Charbon E., Felt E., Sangiovanni-Vincentelli A. Automation of IC layout with analog constraints // IEEE Transactions on Computer-Aided Design of Integrated Circuits.- August, 1996, P. 923–942.

61. Marsik J., Subrt O., Martinek P. Developing automated design procedure for operational amplifier blocks // International Conference on Signals and Electronic Systems.- 2008.- P. 269-272.
62. Meduri P., Dhali S. A framework for automatic CMOS opamp sizing // 53rd IEEE International Midwest Symposium on Circuits and Systems.- 2010.- P. 608-611.
63. Melikyan V., Goldman R., Babayan E. Method of Electro-Thermal Co-Simulation of Integrated Circuits // Proceedings of the 8th International Conference of "Semiconductor Micro- and Nanoelectronics", Yerevan, Armenia, 2011.- P. 207-213
64. Michael C., Ismail M. Statistical modeling of device mismatch for analog MOS integrated circuits // IEEE Journal of Solid-State Circuits.- February, 1992.- P. 154–166.
65. Miyamoto M., et al Impact of Reducing STI-Induced Stress on Layout Dependence of MOSFET Characteristics // IEEE Trans. Elec. Dev.- March, 2004.- P. 440-443.
66. Naiknaware R. Fiez T. Automated Hierarchical CMOS Analog Circuit Stack Generation with Intramodule Connectivity and Matching Considerations // IEEE JSSC.- March, 1999.- P. 304–317.
67. Nassif S., Modeling and analysis of manufacturing variations // IEEE Pmc. Of CICC.- 2001.- P. 223-228.
68. Nassif S., Agarwal K., Nowka K. Characterization and design for variability and reliability // IEEE Custom Integrated Circuits Conference.- September, 2008.- P. 341-346.
69. Ochotta E.S., Rutenbar R.A., Carley L.R. Synthesis of highperformance analog circuits in ASTRX/OBLX // IEEE Trans. Computer-Aided Design.- 1996.- Vol. 15.- P. 273–294.
70. Owen B., et al. An Analog Layout Language // IEEE Custom Integrated Circuits Conference.- 1995.- P. 41-44.
71. Pelgrom M., Duinmijer A., Welbers A. Matching Properties of MOS Transistors // IEEE JSSC.- October, 1989.- P. 1433–1440.

72. Phelps R., Krasnicki M., Rutenbar R., et al Anaconda: simulation-based synthesis of analog circuits via stochastic pattern search // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.- June, 2000.- P. 703–717.
73. Ranjan M., Verhaegen W., Agarwal A., et al Fast, Layout-Inclusive Analog Circuit Synthesis using Pre-Compiled Parasitic-Aware Symbolic Performance Models // Design Automation and Test in Europe.- February, 2004.- P. 604–609.
74. Razavi B. Design of Analog CMOS Integrated Circuits.- Tata McGraw-Hill, 2002.- P. 684.
75. Rijmenants J., et al ILAC: An Automated Layout Tool for Analog CMOS Circuits // IEEE Journal of Solid-State Circuits.- April, 1989.- Vol. 24, No. 2.- P. 417-425.
76. Roy J., Markov I. High-performance routing at the nanometer scale // IEEE Transactions on Computer-Aided Design of Integrated Circuits.- June, 2008.- P. 1066–1077.
77. Roychowdhury J., Gielen G., Rutenbar R. Hierarchical Modeling, Optimization and Synthesis for System-Level Analog and RF Designs // Proceedings of the IEEE.- 2007.- P. 640-669.
78. Rutenbar R., Gielen G., Antao B., et al Computer-Aided Design of Analog Integrated Circuits and Systems.- Wiley-IEEE Press, April, 2002.- P. 754.
79. Sabat S., Kumar K., Udgata S. Differential evolution and swarm intelligence techniques for analog circuit synthesis // World Congress on Nature & Biologically Inspired Computing.- 2009.- P. 469-474.
80. Schardein W., Andersch M., Hosticka B., et al Analog Module Generator for Effective Design Assistance // Proc. ESSCIRC'94.- September, 1994.- P. 160-163.
81. Schwencker R., Eckmueller J., Graeb H., Antreich K. Automating the sizing of analog CMOS circuits by consideration of structural constraints // Proceedings of the IEEE Conference on Design, Automation and Test in Europe.- 1999.- P. 323–327.
82. Sheu Y., Sheng-Jier Y., Hsien-Te C., et al Modeling Well Edge Proximity Effect on Highly-Scaled MOSFETs // IEEE CICC.- 2005.- P. 831-834.

83. Somani A., Chakrabarti P., Patra A. An evolutionary algorithm-based approach to automated design of analog and RF circuits using adaptive normalized cost functions // IEEE Transactions on Evolutionary Computation.- 2007.- Vol. 11.- P. 336-353.
84. Tang H., Doboli A. Employing Layout Templates for Synthesis of Analog Systems // Midwest Symposium on Circuits and Systems.- 2002.- P. 505-508.
85. Tang H., Zhang H., Doboli A. Layout-Aware Analog System Synthesis Based on Symbolic Layout Description and Combined Block Parameter Exploration, Placement and Global Routing // Annual Symposium on VLSI.- 2003.- P. 266-271.
86. Tlelo-Cuautle E., Guerra-Gomez I., Duarte-Villasenor M., et al Applications of evolutionary algorithms in the design automation of analog integrated circuits // Journal of Applied Sciences.- 2010.- Vol. 10.- P. 1859-1872.
87. Vancorenland P. et al A Layout-Aware Synthesis Methodology for RF Circuits // Proc. of ICCAD.- 2001.- P. 358-362.
88. Veselinovic P., Leenaerts D., van Bokhoven W., et al A flexible topology selection program as part of an analog synthesis system // Proceedings of the European Design and Test Conference.- 1995.- P. 119–123.
89. Wai-Kai Chen Analog and VLSI Circuits. Third Edition.- CRC Press, 2009.- P. 702.
90. Wang F., Li Y., Li L. Li K. Automated analog circuit design using two-layer genetic programming // Applied Mathematics and Computation.- 2007.- P. 1087-1097.
91. Wolf M., et al A Novel Analog Module Generator Environment // Proc. The European Design & Test Conference.- March, 1996.- P. 388-392.
92. Wolf M., et al New Description Language and Graphical User Interface for Module Generation in Analog Layouts // Proc. ISCAS.- June, 1998.- Vol. VI.- P. 290-293.
93. Wolf S. Silicon Processing for the VLSI Era, Vol. 4: Deep-Submicron Process Technology.- Lattice Press, 2002.- P. 822.
94. Yeh W., Chen S., Fang Y., Chao C. Effect of extrinsic impedance and parasitic capacitance on figure of merit of RF MOSFET // IEEE Transactions on Electron Devices.- September, 2005.- P. 2054–2060.

95. Yilmaz E., Dundar G. Analog layout generator for CMOS circuits // IEEE Transactions on Computer-Aided Design of Integrated Circuits.- January, 2009.- P. 32-45.
96. Zhang L., Kleine U. A Genetic Approach to Analog Module Placement with Simulated Annealing // Proc. IEEE International Symposium on Circuits and Systems.- May, 2002.- P. 345-348.
97. Zhang L., Kleine U., Rudolph T., Wolf M. A New Design Rule Description for Automated Layout Tools // Proc. 7th IEEE International Conference on Electronics, Circuits and System.- December, 2000.- P. 988-992.
98. Атаев Д.И., Болотников В.А. Аналоговые интегральные микросхемы.– Москва, Изд-во «МЭИ», 1991.– 237 с.
99. Бессонов Л.А. Теоретические основы электротехники: Электрические цепи.– Москва, 1978.– 528 с.
100. Волович Г.И. Схемотехника аналоговых и аналого-цифровых устройств.– Москва Издательский дом, Додека-XXI, 2005.– 528 с.
101. Пригожин И., Кондепуди И. Современная термодинамика.- , Мир, 2002.- 461 с.
102. Титце У., Шенк К. Полупроводниковая схемотехника.– Москва, Мир, 1982.– 512 с.
103. Федорков Б.Г., Телец В.А. Микросхемы ЦАП и АЦП.– Москва, Энергоатомиздат, 1990.– 320 с.
104. Хоровиц П.Т., Хилл У.К. Исскуство Схемотехники.– Москва, Мир, 1998.– 704с.
105. Шалимова К.В. Физика полупроводников.- СПб, Лань, 2010.- 392 с.
106. Якубовский С.В. Цифровые и аналоговые интегральные микросхемы.– Москва, Радио и связь, 1990.– 496 с.
107. Կասարջյան Հ.Գ. 32/28նմ տեխնոլոգիական գործընթացում գրպանիկի մոտիկության երևույթի ազդեցությունը P – MOԿ տրանզիստորի պարամետրերի վրա //ՀՃԱ Լրաբեր.- 2015.- Հ. 12, N 2.- էջ 359-363:
108. Կասարջյան Հ.Գ. 32/28նմ տեխնոլոգիական գործընթացում գրպանիկի մոտիկության երևույթի ազդեցության նվազեցման եղանակ: //ՀՀ ԳԱԱ և ՀԱՊՀ Տեղեկագիր.- 2015. Հատոր LXVIII, No 4.- էջ 491-496:

109. Կասարջյան Հ.Գ. Անալոգային ինտեգրալ սխեմաների ֆիզիկական նախագծման ժամանակ ՄՕԿ տրանզիստորների համապատասխանեցման մեթոդները և դրա անհրաժեշտությունը // ՀԱՊՀ Լրաբեր. Գիտական հոդվածների ժողովածու. - Երևան, 2016. - Մաս 1. - էջ 274-278:
110. Կասարջյան Հ.Գ. Ջերմաստիճանային գրադիենտի նկատմամբ կայուն գերճշգրիտ լիցքային պոմպի ֆիզիկական նախագծում //ՀՃԱ Լրաբեր.- 2016.- Հ. 13, N 1.- էջ 135-137:
111. Կասարջյան Հ.Գ. Փուլահաճախական դետեկտորում պաշտպանիչ օղակների կիրառման ազդեցությունը փուլահաճախական ինքնահամալարման համակարգերի ֆիզիկական նախագծի ելքային պարամետրերի վրա: //ՀՃԱ Լրաբեր.- 2015.- Հ. 12, N 4.- էջ 744-747:
112. Մելիքյան Վ.Շ., Կասարջյան Հ.Գ. Անալոգային ինտեգրալ սխեմաների ռեզիստորային և կոնդենսատորային պատահական անհամապատասխանությունների նվազեցման եղանակ //ՀՃԱ Լրաբեր.- 2015.- Հ. 12, N 1.- էջ 125-129:

ՀԱՎԵԼՎԱԾ 1

SYNOPSYS®

№ 252/16 "4" 04 2016

Հաստատում եմ՝
«ՍԻՆՈՓՍԻՍ ԱՐՄԵՆԻԱ» ՓԲԸ տնօրեն՝
Հ. Մուսայելյան

Հովհաննես Գևորգի Կասարջյանի «Անալոգային ինտեգրալ սխեմաների ավտոմատացված ֆիզիկական նախագծման միջոցների մշակումը և հետազոտումը» թեմայով թեկնածուական ատենախոսության արդյունքների

ՆԵՐԴՐՄԱՆ ԱԿՏ

ԵԿՏԱ «ավտոմատացման համակարգեր» մասնագիտությամբ 3-րդ կուրսի ասպիրանտ Հ. Գ. Կասարջյանի կողմից մշակված անալոգային ինտեգրալ սխեմաների ավտոմատացված ֆիզիկական նախագծման AAR Designer և Constraint Designer ծրագրային միջոցների փաթեթը ներդրված է «ՍԻՆՈՓՍԻՍ ԱՐՄԵՆԻԱ» ՓԲԸ -ում: Այդ ծրագրային միջոցի կիրառումը հնարավորություն է տվել արդյունավետ կերպով և կարճ ժամանակում իրականացնել անալոգային ԻՍ-երի ավտոմատացված ֆիզիկական նախագծումը, միաժամանակ ապահովելով սխեմայի ընդունելի չափով էներգասպառում և կիսահաղորդչային բյուրեղի վրա զբաղեցրած մակերես: Ծրագրային միջոցը կազմում է անալոգային ԻՍ-երի նախագծման գործընթացի կարևոր մասը և բավարարում է ժամանակակից էլեկտրոնային նախագծման բնագավառում կիրառվող ծրագրային գործիքային միջոցին ներկայացվող բոլոր պահանջներին:

Թվային և մուտքի/ելքի հանգույցների
նախագծման դեպարտամենտի տնօրեն՝ Վ. Մովսիսյան

"ՍԻՆՈՓՍԻՍ ԱՐՄԵՆԻԱ" ՓԲԸ
0026, ՀՀ, ԵՐԵՎԱՆ, ԱՐՇԱԿՈՒՆՅԱՑ 41
ՀԵՌ.՝ (+374) 10 492100, ՖԱՔՍ՝ (+374) 10 492696
ՀՎՀՀ 02236362

"SYNOPSYS ARMENIA" CJSC
41 ARSHAKUNYATS AVE., YEREVAN, ARMENIA, 0026
TEL.: (+374) 10 492100, FAX: (+374) 10 492696
TAX PAYER'S ID 02236362

ՀԱՎԵԼՎԱԾ 2

- **Անտենա խախտումների սահմանման և ուղղման սկրիպտ**

```
# SETUPS
# =======
set ANT_D [get_lib_cell r2hd_n_nwellsp4_1dx]
set CPUS 8
# CONFIGS
# ========
config_app -cpu 8
config_name_rules -cell_prefix ANTENNA
config_name_rules -net_prefix ANTENNA
config_antenna -check all -diodes $ANT_D
config_antenna -gate_default_area 12000000
config_antenna -skip_incomplete_nets true
config_antenna -report
report_tech -display antenna_rules

# CREATE PROPERTY
# ===============
create_property -name antenna -object_type lib_cell -data_type bool -init_value false -storage sparse -
persistent true
set_property -name antenna -value true  [get_lib_cell $ANT_D]
set_property -name dont_use -value false [get_lib_cell $ANT_D]

check_antenna
config_antenna
fix_antenna -method jumper
fix_antenna -method {jumper diode}
place_detail -driven_by {timing cells_moved}

# Routing S&R
# ===========
route_final -cpu $CPUS -mode accurate_drc -effort expensive
check_drc -cpu $CPUS
check_lvs -cpu $CPUS

# Antenna check
# =============
check_antenna
```

- **AAR.py ֆայլի օրինակ**

```
set_property("route.bias_connections_to_center",False)
set_property('route.single_connection_per_pin',True)
set_property('route.fix_net_topology',False)
set_property('router.special_nets_implicitly_connected',False)
set_property('route.inline_multicut_vias', True)
set_property('router.flatten_query', False)
set_property('router.clip_to_port_query', True)
set_property('route.inline_multicut_vias', True)
set_property('route.single_connection_per_pin',True)

__first_pre_load = 1
def my_pre_load(lib, cell, view):
  print "Start pre loading..."
  global __first_pre_load
  if (__first_pre_load == 1 and cell != 'sdp1' ):
    set_diffusion_layers(layers=['OD'])
    set_visible_layer_range(min_layer='POLYG', max_layer='M6')
    define_auxiliary_routing_layer(layer="POLYG",pitch="400",
width="120",spacing="180",direction="vertical")
    set_property("droute.layer.POLYG.via_access_layer",True)
  if ( cell == 'chip' or cell == "cordic" ):
    set_property('groute.cap_obs_multiplier',0.95)
    set_property('droute.pin_access_caching','false')
    set_property('droute.fix_net_topology','true')
    set_propert0y('droute.pin_include_wire','true')
    set_property('droute.pin_protect_amount',8)
  __first_pre_load = 0
  print "End pre loading..."

def my_post_load(lib, cell, view):
  print "Start post loading..."
  set_property('router.special_nets_implicitly_connected',False)
  if ( cell == 'chip' ):
    clear_routing_tracks()
    set_router_constraint('-nets','*', '-min_layer','M1', '-max_layer','M3')
    set_router_constraint('-nets','CHIP_CLK', '-min_layer','M5', '-max_layer','M6')
    create_gate_blockages(block_layer=['POLYG','M1','M2'], poly_layer='POLYG', diffusion_layers=['OD'])
  print "End post loading..."
```

ՀԱՎԵԼՎԱԾ 3

- **Constraint Desginer GUI փաստաթղթի հատված**

```
#ifndef CONSTRAINTAPPMODULE_H
#define CONSTRAINTAPPMODULE_H

// Qt includes
#include <QObject>
#include <QtPlugin>

class QAction;
class QMenu;
class QPixmap;
class QWidget;

// application includes
#include <oapActionMgrIfc.h>
#include <oapModulePluginIfc.h>
#include <oapModuleToolIfc.h>
#include <oapStatus.h>
#include <ConstraintDockable.h>

// system includes

namespace PxConstraint {

class oap::ToolBarIfc;
class oap::IconMgrIfc;

class ConstraintAppModule :
   public QObject,
   public oap::ModulePluginIfc,
   public oap::ModuleToolIfc
{
Q_OBJECT
Q_INTERFACES(oap::ModulePluginIfc oap::ModuleToolIfc)

public : // methods
  /*!
   * The public constructor for the ConstraintAppModule.
   */
  ConstraintAppModule();

  /*!
   * The public destructor for the ConstraintAppModule.
   */
  virtual ~ConstraintAppModule();

  // -----------------------
  // module loader interface
  // -----------------------
  /*!
   * Starts the loading of the Constraint Application Module and initializes
   * module static variables.
   * \return an oap::Status to indicate the success or failure of the
   * intial load steps.
   */
  oap::Status init(const QString & modulePkgDir);

  /*!
```

```
   * installs example Python commands.
   * \return an oap::Status to indicate any problems during installation.
   */
  //oap::Status initPython();

  // --------------
  // tool interface
  // --------------
  /*!
   * Provides the means to create an application banner.
   * \return a pair with the name of the application and a description of
   * the module.
   */
  QPair<QString, QString> aboutTool() const;

  /*!
   * Identifies where the application module images are located
   * \return a directory path to the application modules images
   */
  QString getIconPath() const;

  /*!
   * install toolbar, menu, etc.
   */
  void declareInterface(
      oap::ToolInstallerIfc *installer,
      oap::IconMgrIfc       *iconMgr,
      oap::ActionMgrIfc     *actionMgr
      );

  /*!
   * Creates all of the sub-menus that will be unique to this application
   * module.
   * \param [in] actionInst The instance pointer to the Action Manager.
   * \returns a list of pairs that identify the target menu name and the
   * action that will be inserted into the named target.
   * \warning Fatal assertion: \a actionInst is null.
   */
  QList<QPair<QString, QAction*> >
    createSubMenus(oap::ActionMgrIfc *actionInst);
  /*!
   * Creates all of the application module's dockables.
   * \returns a list of pairs that identify the initial location of the
   * dockable and a pointer to the actual dockable itself.
   */
  QList<QPair<Qt::DockWidgetArea, oap::DockableIfc*> > createDockables();

private:
  ConstraintDockable* constraintEditorDockable;

private slots:
  void Action_OpenConstraintEditor();

};

} // namespace PxConstraint

#endif // CONSTRAINTAPPMODULE_H
```

- **Constraint Designer.cxx փաստաթուղթ**

```
#include <Python.h>

#include <oaDesignDB.h>
#include <oaDesignObserver.h>

#include <iDomain.h>
#include <oapCollection.h>
#include <oapPublicAPIMgr.h>
#include "proxyNDRTreeItem.h"
#include <ConstraintMgrExceptions.h>

#include <QDebug>
#include <QTimer>

#include "ConstraintMgrStateMachine.h"
#include "ConstraintMgrStates.h"


#include "ConstraintMgr.h"
#include <Navigator.h>
#include <GroupObservers.h>
#include <IconMgr.h>
#include <oapDesignContextMgr.h>
#include <oapSelectManager.h>
#include <oapIconUtils.h>
#include <InputLineDialog.h>


#include <QObject>
#include <QAction>
#include <QComboBox>
#include <QGroupBox>
#include <QLabel>
#include <QList>
#include <QMenu>
#include <QMenuBar>
#include <QPushButton>
#include <QStandardItemModel>
#include <QToolButton>
#include <QTreeWidget>
#include <QTreeWidgetItem>
#include <QTreeView>
#include <QSplitter>
#include <QVBoxLayout>
#include <QSpinBox>
#include <QMessageBox>
#include <QEventTransition>
#include <QHistoryState>

#include <iostream>
#include <iomanip>
#include <algorithm>
#include <vector>
#include <map>
#include <boost/assign/list_of.hpp>
#include <boost/shared_ptr.hpp>

#include <QApplication>
```

```
//USE_OA_NAMESPACE

#include <oapSingletonCorral.h>
#include <oapComponentPtr.h>
#include <oapMessageMgrIfc.h>
#include <oapNavigatorIfc.h>
#include <oapReportMgrIfc.h>

#include "ConstraintBrowser.h"
#include "ConstraintEditorIfc.h"
#include "SingleNetConstraintEditor.h"
#include "WireModelConstraintEditor.h"
#include "CutLayersConstraintEditor.h"
#include "PairedNetsConstraintEditor.h"
#include "GroupNetConstraintEditor.h"

namespace
{
oa::oaNativeNS sns;
}

namespace PxConstraint
{
CreatorFunction ProxyNDRBase::proxyCreator = fakeProxyCreator;
}

namespace cmgui
{

ConstraintMgr* ConstraintMgr::instance_ = 0;
ConstraintMgr* ConstraintMgr::instance(QWidget* parent)
{
  if(0 == instance_)
  {
    Q_ASSERT(0 != parent);
    instance_ = new ConstraintMgr(parent);
  }
  return instance_;
}


// extern function prototype
bool initConstraintMgrModule();

ConstraintMgr::ConstraintMgr(QWidget* parent)
  : QWidget(parent)
  , DesignContextObserver(1, true)
  , SelectionObserver(1, true)
  , oa::oaObserver<oa::oaDesign>(1, true)
  , NavigatorObserver(1, true)
  //, activeBlock_(0)
  , currentSelectionSource_(UnknownSelectionSource)
  , constraintBrowser_(new ConstraintBrowser(this))
  , netGroupEditor_(0)
  , splitter_(new QSplitter(Qt::Vertical))
  , activeDesign_(0)
  , ndrProxy_(0)
  , creator_(new EditorCreator(this))
  , postSelectionCM_(new PostSelectionCM())
  , designUndoObserver_(new DesignUndoObserver(this))
  , activeDesignIsInUndo_(false)
  , undoOperationCount_(0)
```

```
  , isDirtySelection_(false)
  , mouseButtonClicked_(false)
{
  groupObserver_ = new MgrGroupObserver(this);
  groupMemObserver_ = new MgrGroupMemberObserver(this);

  connect(this, SIGNAL(selectionUpdated()), this, SLOT(slotUpdateSelection()), Qt::QueuedConnection);

  constraintEditorTab_ = new QTabWidget;
  constraintEditorTab_->setObjectName("ConstraintEditorTab");
  constraintEditorTab_->addTab(creator_->getEditor(SingleNetEditorType, ndrProxy_), singleNetTabLabel);
  constraintEditorTab_->addTab(creator_->getEditor(PairedNetsEditorType, ndrProxy_), pairNetTabLabel);
  constraintEditorTab_->addTab(creator_->getEditor(WireModelEditorType, ndrProxy_),
wireModelTabLabel);
  constraintEditorTab_->addTab(creator_->getEditor(CutLayersEditorType, ndrProxy_), cutLayerTabLabel);
  constraintEditorTab_->addTab(creator_->getEditor(GroupNetEditorType, ndrProxy_), groupNetTabLabel);

  QGroupBox* filterGroup = new QGroupBox(this);
  QHBoxLayout* filterLayout = new QHBoxLayout;
  filterLayout->addWidget(new QLabel("Show:", this));
  filterLayout->addSpacerItem(new QSpacerItem(20, 20, QSizePolicy::Fixed));
  filters_ = new QComboBox(this);
  filters_->setObjectName("showComboBoxFilterConstraint");
  filters_->addItems(QStringList() << "All" << singleNetLabel << pairedNetsLabel << groupLabel);
  connect(filters_, SIGNAL(currentIndexChanged(QString)), this, SLOT(slotShowNets(QString)));

  filterLayout->addWidget(filters_);
  filterLayout->addSpacerItem(new QSpacerItem(0/*98*/, 20, QSizePolicy::Expanding,
QSizePolicy::Minimum));
  addGrButton_ = new QPushButton(QIcon(":/oap/icons/CreateGroup24.png"), "", this);
  addGrButton_->setObjectName("netGroupButton");
  addGrButton_->setToolTip("Create net group");
  connect(addGrButton_, SIGNAL(clicked()), this, SLOT(slotAddGroup()));
  filterLayout->addWidget(addGrButton_);
  filterGroup->setLayout(filterLayout);

  connect(constraintBrowser_, SIGNAL(signalSelectionChanged()), this,
SLOT(slotUserSelectionChanged()));
  connect(constraintBrowser_, SIGNAL(signalItemsDeleted()), this, SLOT(slotItemsDeletedFromBrowser()));
  connect(constraintEditorTab_, SIGNAL(currentChanged(int)), this, SLOT(slotEditorTabChanged(int)));

  QVBoxLayout* layout = new QVBoxLayout(this);
  layout->addWidget(filterGroup);
  splitter_->addWidget(constraintBrowser_);
  splitter_->addWidget(constraintEditorTab_);
  layout->addWidget(splitter_);

  setLayout(layout);
  setFocusPolicy(Qt::StrongFocus);

  const PxConstraint::EnumToStringMap singleNetTypeEnumMap = boost::assign::map_list_of
    ( PxConstraint::DFMTypes, dfmStr )
    ( PxConstraint::WMPrimaryValues, wireModelStr)
    ( PxConstraint::MaxDetourTypes, maxDetourStr )
    ( PxConstraint::CMaxTypes, cMaxStr )
    ( PxConstraint::RMaxTypes, rMaxStr )
    ( PxConstraint::FeedThroughTypes, feedThroughStr )
    ( PxConstraint::LayerMaxValues, maxLayerStr )
    ( PxConstraint::LayerMinValues, minLayerStr )
    ( PxConstraint::RoutingStyleTypes, routingStyleStr )
    ( PxConstraint::ShieldTypes, shieldingStr )
    ( PxConstraint::SkipTypes, skipStr )
```

```
    ( PxConstraint::WeightTypes, weightStr );
  singleNetTypeEnumMap_ = singleNetTypeEnumMap;

  initStateMachine();

  hiddenStateButton_ = new QPushButton(this);
  hiddenStateButton_->resize(1,1); // make button not visible , as it is used only for gui logging mechanism to
change the state of CM GUI
  hiddenStateButton_->setObjectName("ConstraintMgrStateButton");

  connect( hiddenStateButton_, SIGNAL( clicked(bool) ),
      this, SIGNAL( selectionUpdated())
      );

  static bool ConstraintInitialiazed = false;
  if(false == ConstraintInitialiazed)
  {
    ConstraintInitialiazed = initConstraintMgrModule();
    ConstraintInitialiazed = true;
  }
```

- **AAR Designer GUI header փաստաթուղթ**

```
#ifndef ROUTERASSISTANTEXECUTOR_H
#define ROUTERASSISTANTEXECUTOR_H

#include <oapSelectionObserver.h>
#include <iSelectionVisitorIfc.h>
#include <oapNet.h>
#include <iDomain.h>

#include <QAbstractItemModel>
#include <QLineEdit>
#include <QSortFilterProxyModel>
#include <QWidget>
#include <QSet>
#include <QComboBox>
#include <QToolBar>
#include <QAction>
#include <QHBoxLayout>
#include <QTreeWidget>
#include <QStandardItemModel>
#include <oapSelectMgrIfc.h>

#include <vector>
#include <map>
#include <set>
#include <Arg.h>
#include <CommandExecutor.h>

#include <oapRectCommand.h>
#include <oapPolygonCommand.h>
#include <oapPlylineCommand.h>

#include <RouterAssistant.h>
#include <RouterAssistantExtraction.h>
#include <RouterAssistantRouter.h>
#include <RouterAssistantRoutingGuides.h>
#include <RouterAssistantPropagateWires.h>
#include <RouterAssistantShielding.h>
```

```
#include <RouterAssistantPhysicalPorts.h>
#include <RouterAssistantViasFill.h>
#include <RouterAssistantViasInsert.h>
#include <RouterAssistantPreWireDelete.h>
#include <RouterAssistantPreWireDevices.h>
#include <RouterAssistantPreWireRing.h>
#include <RouterAssistantPreWireMatchPin.h>
#include <RouterAssistantPreWireToPin.h>
#include <RouterAssistantPreWireStripes.h>
#include <RouterAssistantPreWireTrack.h>
#include <RouterAssistantDFM.h>
#include <RouterAssistantDefineGrid.h>
#include <RouterAssistantDeleteGrid.h>
#include <RouterAssistantAddStage.h>
#include <RouterAssistantTargetNets.h>
#include <RouterAssistantHistory.h>

BEGIN_OA_NAMESPACE
class oaBlock;
class oaBlockObject;
class oaDesign;
class oaNet;
END_OA_NAMESPACE

namespace ra {

class PostSelectionRA : public oap::SelectionVisitorIfc {

private:
   //fields
   vector<string> selectedNets;
   vector<string> selectedTerms;

public:
   virtual bool visit(oap::IUnknown *o)
   {
     oap::IDomain<oa::oaBlockObject> *oaDomain = dynamic_cast<oap::IDomain<oa::oaBlockObject>
*>(o);
      if (oaDomain && oaDomain->get())
      {
        oa::oaNet *net = NULL;
        oa::oaTerm *term = NULL;
        oa::oaBlockObject *obj = oaDomain->get();
        oaType objType = obj->getType();

        if (objType == oacPathType || objType == oacPathSegType ||
            objType == oacPolygonType || objType == oacRectType ||
            objType == oacDotType || objType == oacCustomViaType ||
            objType == oacStdViaType)
        {
          oaPinFig *fig = static_cast<oaPinFig*>(obj);
          net = fig->getNet();
          oaPin *pin = fig->getPin();
          if (pin)
          {
            term = pin->getTerm();
          }
        }
        else if (objType == oacScalarNetType ||
                 objType == oacBusNetBitType)
        {
          net = static_cast<oaNet*>(obj);
```

```
    }
    else if(objType == oacScalarTermType ||
          objType == oacBusTermBitType)
    {
      term = static_cast<oa::oaTerm*>(obj);
      net = term->getNet();
    }

    oaNativeNS ns;
    if(net)
    {
      oa::oaString netName;
      net->getName(ns, netName);
      bool found = false;
      for(size_t i = 0; i < selectedNets.size(); i++)
      {
        if(std::string(netName).compare(selectedNets[i]) == 0)
        {
          found = true;
          break;
        }
      }
      if(!found)
      {
        selectedNets.push_back(std::string(netName));
      }
    }
    if(term)
    {
      oa::oaString termName;
      term->getName(ns, termName);
      bool found = false;
      for(size_t i = 0; i < selectedTerms.size(); i++)
      {
        if(std::string(termName).compare(selectedTerms[i]) == 0)
        {
          found = true;
          break;
        }
      }
      if(!found)
      {
        selectedTerms.push_back(std::string(termName));
      }
    }
  }
  return true;
}

vector<string> getSelectedNets()
{
  return selectedNets;
}

void clearSelectedNets()
{
  selectedNets.clear();
}

vector<string> getSelectedTerms()
{
  return selectedTerms;
```

```
    }

    void clearSelectedTerms()
    {
       selectedTerms.clear();
    }

    PostSelectionRA()
    {}

    virtual ~PostSelectionRA()
    {}

};

class RouterAssistantExecutor: public QObject,
   public oap::SelectionObserver
{
  Q_OBJECT
   public:
      RouterAssistantExecutor();
      ~RouterAssistantExecutor();

  private:
    enum shapeType{
    Rect = 0, Polygon = 1, Polyline = 2
    };
    shapeType type_;

  private slots:
    void saveDefaultWireModel();

  private slots:
      void doDisplayAddStage();
      void doRunExtraction();
      void doReportParasitics();
      void doRemoveStage();

  private slots:
      void doRouteNets();
      void doRouteNetsByArea();
      void doGlobalRoute();
      void doRouterDrcChecker();
      void doMinimizeJogs();
      void doInvalidateRoutes();
      void doFixRoutes();
      void doUnfixRoutes();
      void doDeferredRouting();
      void doNoDeferredRouting();
      void doReportDeferredPorts();

  private slots:
      void doPropagateWires();
      void doPropagateWiresByArea();
      void doDeletePropagatedWires();

 private slots:
      void doAddBoundaryConnection();
      void doDeleteBoundaryConnection();
      void doAddChannel();
      void doAddGuide();
      void doRemoveRoutingGuide();
```

```
 private slots:
    void doAddNetShielding();
    void doDeleteNetShielding();
    void doDeleteNetShieldingByArea();

private slots:
    void doAddPhysicalPorts();
    void doAddPhysicalPortsByArea();
    void doDeletePhysicalPorts();

private slots:
    void doFillStack();
    void doFillStackByArea();
    void doDeleteFill();
    void doDeleteFillByArea();


private slots:
    void doInsertVias();
    void doInsertViasByArea();
    void doDeleteVias();
    void doDeleteViasByArea();

private slots:
    void doDeleteStripes();
    void doDeleteStripesByArea();

private slots:
    void doAddStripesDevices();
    void doAddStripesDevicesByArea();
    void doActivateActionsDevices();

private slots:
    void doAddShape();
    void doAddRectangle();
    void doAddPolygon();
    void doAddPolyline();
    void doActivateActionsRing();

 private slots:
    void doAddStripesMatchedPin();
    void doAddStripesMatchedPinByArea();

private slots:
    void doAddStripesToPin();
    void doAddStripesToPinByArea();

private slots:
    void doAddStripes();
    void doAddStripesByArea();

private slots:
    void doAddTracks();
    void doAddTracksByArea();
    void doDisplayDefineGrid();
    void doDisplayDeleteGrid();

private slots:
    void doCommitDFM();
    void doCommitDFMByArea();
```

```
private slots:
     void doDefineGridOk();
     void doDeleteGridOk();
     void doAddStageOk();

private slots:
     void targetNetsFilterChange(const QString&);
     void lockTargetNets();

 private:
     void CommitDFM(bool areaBased);
     void runAddStripesCmd(bool areaBased);
     void runPropCmd(bool areaBased);
     void runFillCmd(bool areaBased);
     void runAddMatchPinCmd(bool areaBased);
     void runAddToPinCmd(bool areaBased);
     void runInsertCmd(bool areaBased);
     void runViaDeleteCmd(bool areaBased);
     void runDeleteCmd(bool areaBased);
     void doAddPowerDevice(bool areaBased);

  public: // SelectionObserver
   virtual void postSelection(oap::SelectionGroup *selectSet);

  private:
   std::vector<std::string> getSelectedNets();
   std::vector<std::string> getSelectedPorts();
   void setTargetSelectedNets();
   void enableDisableActionButtons();
   void getDirectivesHandles();
   void createConnect();
   void createConnectForExtraction();
   void createConnectForRouter();
   void createConnectForPropagateWires();
   void createConnectForRoutingGuides();
   void createConnectForShielding();
   void createConnectForPhyPort();
   void createConnectForViaFill();
   void createConnectForViaInsert();
   void createConnectForPreWireDelete();
   void createConnectForPreWireDevice();
   void createConnectForPreWireRing();
   void createConnectForPreWireMatchPin();
   void createConnectForPreWireToPin();
   void createConnectForPreWireStripes();
   void createConnectForPreWireTracks();
   void createConnectForDFM();
   void createConnectForTargetNets();

  private:
   PostSelectionRA* postSelectionRA_;
   RouterAssistant *rassist_;
   PxForms::CommandExecutor *executor_;
   RouterAssistantExtraction *extractionDirective_;
   RouterAssistantRouter *routerDirective_;
   RouterAssistantPropagateWires *preWireDirective_;
   RouterAssistantRoutingGuides *guidesDirective_;
   RouterAssistantShielding *shieldingDirective_;
   RouterAssistantPhysicalPorts *phyPortDirective_;
   RouterAssistantViasFill *viaFillDirective_;
   RouterAssistantViasInsert *viaInsertDirective_;
   RouterAssistantPreWireDelete *preWireDeleteDirective_;
```

```
    RouterAssistantPreWireDevices *preWireDeviceDirective_;
    RouterAssistantPreWireRing *preWireRingDirective_;
    RouterAssistantPreWireMatchPin *preWireMatchPinDirective_;
    RouterAssistantPreWireToPin *preWireToPinDirective_;
    RouterAssistantPreWireStripes *preWireStripesDirective_;
    RouterAssistantPreWireTrack *preWireTrackDirective_;
    RouterAssistantDFM *dfmDirective_;
    RouterAssistantDefineGrid  *defineGridform_;
    RouterAssistantDeleteGrid  *deleteGridform_;
    RouterAssistantAddStage  *addStageForm_;
    RouterAssistantTargetNets *targetNetsDirective_;
    RouterAssistantHistory *historyDirective_;
    QAction *insertViasAction_;
    QAction *fixRoutesAction_;
    QAction *unFixRoutesAction_;
    QAction *deferredRoutingAction_;
    QAction *noDeferredRoutingAction_;
    QAction *propagateWiresAction_;
    QAction *propagateWiresByAreaAction_;
    QAction * addBoundaryConnectionAction_;
    QAction *deleteBoundaryConnectionAction_;
    QAction *addChannelAction_;
    QAction *addGuideAction_;
    QAction *removeRoutingGuideAction_;
    QAction *fillStackAction_;
    QAction *fillStackByAreaAction_;
    QAction *deleteFillAction_;
    QAction *deleteFillByAreaAction_;
    QAction *deleteStripesAction_;
    QAction *deleteStripesByAreaAction_;
    QAction *addRectangleAction_;
    QAction *addPolygonAction_;
    QAction *addPolylineAction_;
    QAction *addToLastGeometryAction_;
    QAction *addToPinAction_;
    QAction *addToPinByAreaAction_;
    QAction *addStripesAction_;
    QAction *addStripesByAreaAction_;
    QAction *addTracksAction_;
    QAction *addTracksByAreaAction_;
    QAction *addStripesDevicesAction_;
    QAction *addStripesDevicesByAreaAction_;
    QStandardItemModel *netTreeModel_;
    std::vector<std::string> selectedTargetNets;
    oap::SelectMgrIfc *selectMngr_;
    bool locked_;
};

  /// Helper class which represents an Arg vector and deletes its
  /// context on leaving the scope.
  class ManagedArgs : public PxFramework::ArgVec
  {
  public:
    ManagedArgs()
    { }
    ~ManagedArgs()
    {
      for (size_t i = 0; i < this->size(); i++)
        delete (*this)[i];
    }
  private:
  };
```

```
}

#endif // ROUTERASSISTANTEXECUTOR_H
```

- **AAR Designer GUI cxx փաստաթղթի հատված**

```
#include <RouterAssistantExecutor.h>
#include <RouterAssistantUtil.h>
#include <QString>
#include <QAction>
#include <QPushButton>
#include <QObject>
#include <QVBoxLayout>
#include <QMessageBox>
#include <iostream>
#include <map>
#include <Arg.h>
#include "Properties.h"
#include <oapLayoutCmdSystemIfc.h>
#include <oapComponentPtr.h>
#include <oapMessageMgrIfc.h>
#include <oapReportMgrIfc.h>
#include <otmTechManagerInterface.h>
#include <oapPublicAPIMgr.h>


using namespace std;
using namespace oap;
using namespace ra;
using namespace PxFramework;
using namespace PxForms;
using namespace Px;

RouterAssistantExecutor::RouterAssistantExecutor(): SelectionObserver(1, true), type_(Rect)
{
  postSelectionRA_ = new PostSelectionRA();
  rassist_ = RouterAssistant::getRouterAssistant();
  getDirectivesHandles();
  createConnect();
  executor_ = new CommandExecutor;
  locked_ = false;

  selectMngr_ = oap::PublicAPIMgr::instance()->selectMgr();
  postSelection(selectMngr_->activeSelectSet());

  // Install the various commands to gather points from the user.
  oap::ComponentPtr<oap::LayoutCmdSystemIfc> cmdSystem;
  pxassert(cmdSystem.get() != NULL);
  WithRectCommand::ArgType ringType = WithRectCommand::RING_ARG;
  boost::function<oap::Command*()> ringFactory = WithRectCommandFactory(ringType);
  cmdSystem->registerCmd(WithRectCommand::getName(ringType), ringFactory);

  boost::function<oap::Command*()> polygonFactory = WithPolygonCommandFactory();
  cmdSystem->registerCmd(WithPolygonCommand::getName(), polygonFactory);

  boost::function<oap::Command*()> polylineFactory = WithPolylineCommandFactory();
  cmdSystem->registerCmd(WithPolylineCommand::getName(), polylineFactory);
}
```

```
void RouterAssistantExecutor::saveDefaultWireModel()
{
  PxFramework::Properties *props = PxFramework::Frame::getProperties();
  PxFramework::Property *prop = props->getProp("app.default_wire_model_name");
  if(prop)
  {
    RouterAssistantUtil::getRouterAssistantUtil()->setDefaultWireModel(prop->getStringValue());
  }
}

void RouterAssistantExecutor::getDirectivesHandles()
{
  QVBoxLayout *extraction = rassist_->getEditor("Extraction");
  extractionDirective_ = dynamic_cast<RouterAssistantExtraction*>(extraction);

  QVBoxLayout *Router = rassist_->getEditor("Router");
  routerDirective_ = dynamic_cast<RouterAssistantRouter*>(Router);

  QVBoxLayout *propWire = rassist_->getEditor("PropWires");
  preWireDirective_ = dynamic_cast<RouterAssistantPropagateWires*>(propWire);

  QVBoxLayout *guides = rassist_->getEditor("Guides");
  guidesDirective_ = dynamic_cast<RouterAssistantRoutingGuides*>(guides);

  QVBoxLayout *shielding = rassist_->getEditor("Shielding");
  shieldingDirective_ = dynamic_cast<RouterAssistantShielding*>(shielding);

  QVBoxLayout *phyPort = rassist_->getEditor("PhysPorts");
  phyPortDirective_ = dynamic_cast<RouterAssistantPhysicalPorts*>(phyPort);

  QVBoxLayout *viaFill = rassist_->getEditor("Fill Metal Stack");
  viaFillDirective_ = dynamic_cast<RouterAssistantViasFill*>(viaFill);

  QVBoxLayout *viaInsert = rassist_->getEditor("Insert");
  viaInsertDirective_ = dynamic_cast<RouterAssistantViasInsert*>(viaInsert);

  QVBoxLayout *preWireDelete = rassist_->getEditor("Delete");
  preWireDeleteDirective_ = dynamic_cast<RouterAssistantPreWireDelete*>(preWireDelete);

  QVBoxLayout *preWireDevice = rassist_->getEditor("Devices");
  preWireDeviceDirective_ = dynamic_cast<RouterAssistantPreWireDevices*>(preWireDevice);

  QVBoxLayout *preWireRing = rassist_->getEditor("Ring");
  preWireRingDirective_ = dynamic_cast<RouterAssistantPreWireRing*>(preWireRing);

  QVBoxLayout *preWireMatchPin  = rassist_->getEditor("Match Pin");
  preWireMatchPinDirective_ = dynamic_cast<RouterAssistantPreWireMatchPin*>(preWireMatchPin);

  QVBoxLayout *preWireToPin  = rassist_->getEditor("To Pin");
  preWireToPinDirective_ = dynamic_cast<RouterAssistantPreWireToPin*>(preWireToPin);

  QVBoxLayout *preWireStripes  = rassist_->getEditor("Stripes");
  preWireStripesDirective_ = dynamic_cast<RouterAssistantPreWireStripes*>(preWireStripes);

  QVBoxLayout *preWireTrack  = rassist_->getEditor("Track");
  preWireTrackDirective_ = dynamic_cast<RouterAssistantPreWireTrack*>(preWireTrack);

  QVBoxLayout *dfm  = rassist_->getEditor("DFM");
  dfmDirective_ = dynamic_cast<RouterAssistantDFM*>(dfm);

  targetNetsDirective_ = rassist_->getTargetNetsEditor();
  historyDirective_  = rassist_->getHistoryEditor();
```

```
}

void RouterAssistantExecutor::createConnect()
{
  QObject::connect(RouterAssistantUtil::getRouterAssistantUtil(), SIGNAL(updateDefWireModel()), this,
SLOT(saveDefaultWireModel()));

  createConnectForExtraction();
  createConnectForRouter();
  createConnectForPropagateWires();
  createConnectForRoutingGuides();
  createConnectForShielding();
  createConnectForPhyPort();
  createConnectForViaFill();
  createConnectForViaInsert();
  createConnectForPreWireDelete();
  createConnectForPreWireDevice();
  createConnectForPreWireRing();
  createConnectForPreWireMatchPin();
  createConnectForPreWireToPin();
  createConnectForPreWireStripes();
  createConnectForPreWireTracks();
  createConnectForDFM();
  createConnectForTargetNets();
}

void RouterAssistantExecutor::createConnectForExtraction()
{
  QAction *displayAddStageAction = extractionDirective_->getDisplayAddStageAction();
  QObject::connect(displayAddStageAction, SIGNAL(triggered()), this, SLOT(doDisplayAddStage()));

  QAction *runExtractionAction = extractionDirective_->getRunExtractionAction();
  QObject::connect(runExtractionAction, SIGNAL(triggered()), this, SLOT(doRunExtraction()));

  QAction *reportParasiticsAction = extractionDirective_->getReportParasiticsAction();
  QObject::connect(reportParasiticsAction, SIGNAL(triggered()), this, SLOT(doReportParasitics()));

  QAction *removeStageAction = extractionDirective_->getRemoveStageAction();
  QObject::connect(removeStageAction, SIGNAL(triggered()), this, SLOT(doRemoveStage()));
}

void RouterAssistantExecutor::createConnectForRouter()
{
  QAction *detailedRouterAction = routerDirective_->getDetailedAction();
  QObject::connect(detailedRouterAction, SIGNAL(triggered()), this, SLOT(doRouteNets()));

  QAction *detailedRouterByAreaAction= routerDirective_->getDetailedByAreaAction();
  QObject::connect(detailedRouterByAreaAction, SIGNAL(triggered()), this, SLOT(doRouteNetsByArea()));

  QAction *globalRouterAction = routerDirective_->getGlobalAction();
  QObject::connect(globalRouterAction, SIGNAL(triggered()), this, SLOT(doGlobalRoute()));

  QAction *drcCheckerAction = routerDirective_->getDrcCheckerAction();
  QObject::connect(drcCheckerAction, SIGNAL(triggered()), this, SLOT(doRouterDrcChecker()));

  QAction *minimizeJogsAction = routerDirective_->getMinimizeJogsAction();
  QObject::connect(minimizeJogsAction, SIGNAL(triggered()), this, SLOT(doMinimizeJogs()));

  QAction *invalidateRoutes= routerDirective_->getInvalidateRoutesAction();
  QObject::connect(invalidateRoutes, SIGNAL(triggered()), this, SLOT(doInvalidateRoutes()));

  fixRoutesAction_ = routerDirective_->getFixRoutesAction();
```

```
    QObject::connect(fixRoutesAction_, SIGNAL(triggered()), this, SLOT(doFixRoutes()));

    unFixRoutesAction_ = routerDirective_->getUnFixRoutesAction();
    QObject::connect(unFixRoutesAction_, SIGNAL(triggered()), this, SLOT(doUnfixRoutes()));

    deferredRoutingAction_ = routerDirective_->getDeferredAction();
    QObject::connect(deferredRoutingAction_, SIGNAL(triggered()), this, SLOT(doDeferredRouting()));

    noDeferredRoutingAction_ = routerDirective_->getNoDeferredAction();
    QObject::connect(noDeferredRoutingAction_, SIGNAL(triggered()), this, SLOT(doNoDeferredRouting()));

    QAction *reportDeferredPortsAction= routerDirective_->getReportDeferredAction();
    QObject::connect(reportDeferredPortsAction, SIGNAL(triggered()), this, SLOT(doReportDeferredPorts()));

}
```

ՀԱՎԵԼՎԱԾ 4

Նկարների ցանկ

Աղյուսակների ցանկ

Հապավումների ցանկ

ԻՍ – ինտեգրալ սխեմա

ՓԻՀ - փուլահաճախական ինքնահամալարման համակարգ

ԳՄԵ - գրպանիկի մոտիկության երևույթ

ՓՀԴ - փուլահաճախական դետեկտոր

ԼՊ - լիցքային պոմպը

ՄՕԿ - մետաղ-օքսիդ-կիսահաղորդիչ

ՄՄՀ - միավորների միջազգային համակարգ