ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ ԿՐԹՈՒԹՅԱՆ ԵՎ ԳԻՏՈՒԹՅԱՆ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ

ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՃԱՐՏԱՐԱԳԻՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

Կարեն Սպիրիդոնի Ամիրխանյան

ԹԵՍՏԱՎՈՐՄԱՆ ԱՎՏՈՄԱՏԱՑՎԱԾ ՀԱՄԱԿԱՐԳԻ ՄՇԱԿՈՒՄԸ ՍՏԱՏԻԿ ՀԻՇՈՂՈՒԹՅԱՆ ՍԱՐՔԵՐԻ ՀԱՄԱՐ

ԱՏԵՆԱԽՈՍՈՒԹՅՈՒՆ

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

Գիտական ղեկավար՝ ֆիզիկա-մաթեմատիկական գիտությունների թեկնածու
Վալերիկ Վարդանյան

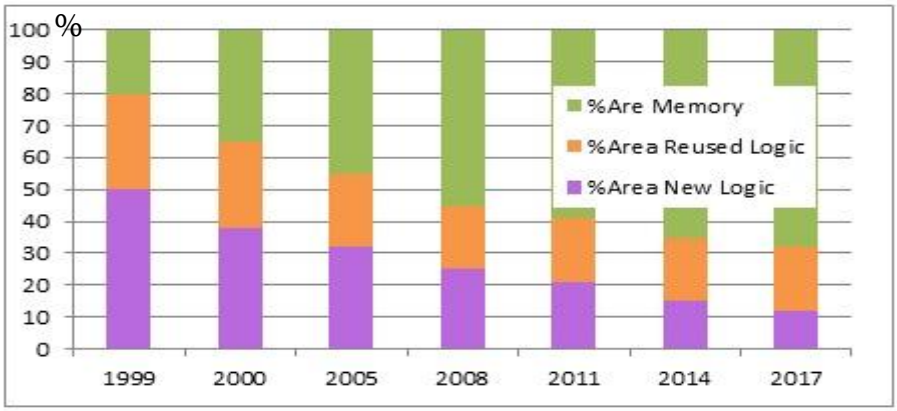
ԵՐԵՎԱՆ – 2015

Բովանդակություն

Ներածություն աշխատանքի ընդհանուր նկարագրությունը

Թեմայի արդիականությունը

Ժամանակակից կիսահաղորդչային սարքերի (առանց բացառության, բոլոր տեսակի կառավարող և հաշվիչ սարքերի) ընդգծված հատկություններից մեկն է, որ այդ սարքերը պարունակում են մեծ ծավալի ներդրված մեկ կամ ավելի, բազմակի



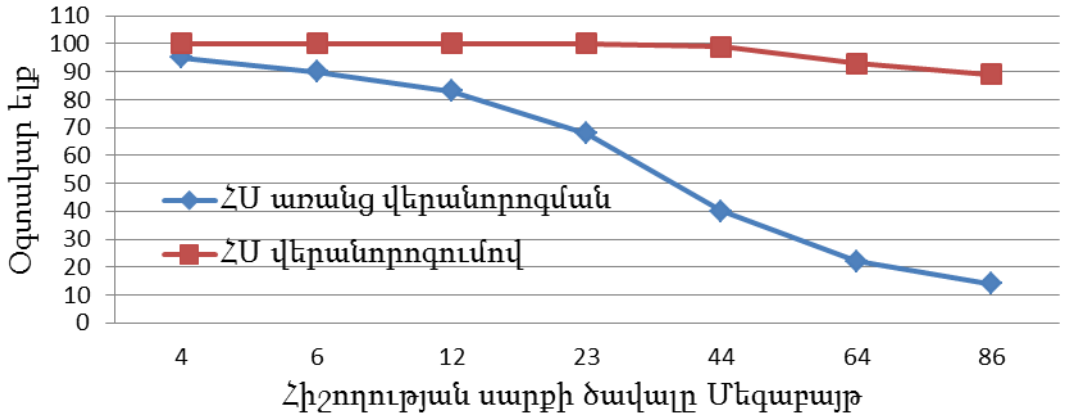
Նկար 1.1. Հիշողության մասնաբաժինը էլեկտրոնային սարքերում (կանխագուշակում՝ ըստ ITRS.net կայքի)

ֆունկցիոնալ դեր կատարող հիշողություններ: Նկար 1.1-ում տրված է էլեկտրոնային սարքերում ներդրված հիշողության սարքերի մասնաբաժինը [1]: Ներդրված հիշողության բջիջների (ՀԲ) գումարային քանակը այդ սարքերում շատ դեպքերում հասնում է հարյուրավոր միլիոնների: Ընդ որում, հիշողության բջիջների քանակը սարքերում տարեցտարի աճում է և, ըստ վիճակագրական տվյալների, 2017 թվականին ՀԲ-երը կզբաղեցնեն նախագծվող սարքերի ամբողջ ֆիզիկական մակերեսի մոտ 70 տոկոսը (Տես՝ նկ.1.1) [1]: Այդ պատճառով՝

ա) որպեսզի առավելագույնս նվազեցնենք հիշողության բջիջներով գրավված սարքերի թիթեղի մակերեսը, այդ բջիջների տոպոլոգիան (layout) նախագծվում է առավելագույնս խիտ: Այսպիսով, հիշողության բջիջների երկու առանձնահատկությունները՝ կառուցվածքի խտությունը և զբաղեցված մակերևույթը, հանդիսանում են նախագծվող սարքերի հիշողություն պարունակող տարածքում արտադրական թերությունների (դեֆեկտների) բարձր հավանականության պատճառը: Հաշվի առնելով այն փաստը, որ նույնիսկ կատարյալ պայմաններում անհնար է ստանալ արտադրվող միկրոսխեմաների 100% արտադրման օգտակար

ելք, ապա դժվար չէ կռահել, որ արտադրության խոտանի հիմնական պատճառը թիթեղում դա հիշողության սարքերի մասում առաջացած արտադրական և շահագործման թերություններն են:

բ) Ներդրված հիշողության սարքերի (ՆՀՍ) արտադրության օգտակար ելքի ավելացման նպատակով հիշողության սարքերում տեղադրվում են լրացուցիչ, այդպես կոչված, ավելցուկային հիշողության բջիջներ պարունակող տողեր և սյուներ: Հետագայում այդ ավելցուկային միավորները օգտագործվում են արտադրության կամ էլ շահագործման ժամանակ առաջացած անսարք բջիջները (տողերը կամ սյուները) փոխարինելու՝ հիշողության սարքի աշխատանքը վերականգնելու նպատակով:



Նկար 1.2. Հիշողության սարքերի ՕԵ-ի ավելացումը

գ) Հիշողության սարքերում տեղադրվում են ներկառուցված (embedded), հիշողության հետ միաժամանակ և նույն տեխնոլոգիայով պատրաստվող, լրացուցիչ էլեկտրոնային սխեմաներ, որոնց նպատակն է հիշողության հանգույցների ինքնաթեստավորումը, անսարքությունների հայտնաբերումը և, հնարավորության դեպքում, հիշողության աշխատունակության վերականգնումը: Պետք է ընդգծել, որ միկրոսխեմաների պատրաստման տեխնոլոգիական պրոցեսների բարդացման՝ տրանզիստորների չափերի փոքրացմանը, զուգահեռ ավելանում է թերությունների քանակը միկրոսխեմաների պատրաստման ընթացքում և որպես այս երևույթի արդյունք նվազում է միկրոսխեմաների օգտակար ելքը (ՕԵ): Ըստ վիճակագրական տվյալների՝ ժամանակակից տեխնոլոգիական պրոցեսների համար (22, 16, 14 նանոմետր (նմ) տեխնոլոգիաների համար) ՕԵ-ն, մոտավորապես, կազմում է 30% (Տես՝ նկ. 1.2.): Կիրառելով ներդրված հիշողության սարքերի աշխատանքի

վերականգման հատուկ մեթոդները՝ միկրոսխեմաների օգտակար ելքը, կախված հիշողության սարքերի մեծությունից հաջողվում է հասցնել, մինչև 95-99 տոկոս [2]:

Նկար 1.2-ից երևում է որ ՕԵ-ը ավելանում է այն դեպքերում, երբ հիշողության սարքերի ծավալը միկրոսխեմայում հասնում է տասնյակ մեգաբայթերի, և այդ ՀՍ-երը ունեն նորոգման սխեմա: Մինչև 44 մեգաբայթ հիշողությունների դեպքում թեստավորման (Built-in Self Test – BIST) և վերականգման (Built-in Repair Analysis-BIRA) ալգորիթմների կիրառման շնորհիվ այդ միկրոսխեմաների օգտակար ելքը հասնում է 99 տոկոսի, իսկ 44 մեգաբայթից ավել ՆՀՍ-եր պարունակելու դեպքում աշխատունակ հիշողության սարքերի օգտակար ելքը հասցվում է 90-ից 99 տոկոսի: Հասկանալի է, որ արդյունքում հաջողվում է միլիոնավոր դոլարների օգուտ ստանալ՝ հնարավոր կորուստը հասցնելով նվազագույն մեկ տոկոսի:

Պետք է նշել, որ ժամանակակից միկրոսխեմաներում, որոնք իրենցից ներկայացնում են բարդ համակարգեր (SoC - System on Chip), լայնորեն օգտագործվում են ներդրված ավտոմատ թեստավորման համակարգեր (ՆԱԹՀ) [3],[4] կառուցված BIST ներդրված գործիքների հիման վրա: ՆԱԹՀ համատարած օգտագործումը պայմանավորված է հետևյալ երեք պատճառներով [5].

1. Արտաքին թեստավորման գործիքների հետ համեմատած՝ ներդրված ատոմատացված թեստավորման գործիքները ավելի «հզոր» են: Օրինակ՝ օսցիլլոգրաֆը հնարավորությունն է տալիս դիտել սահմանափակ քանակության ազդանշանների ձևերը ժամանակային, տևողական տիրույթում: Այդ պատճառով ճարտարագետը ստիպված է լինում կատարել չափումները կարճատև կամ երկարատև ազդանշանների վատագույն՝ ծայրահեղ դեպքերի համար: Բայց այս դեպքում մեկ այլ ազդանշանների խումբ կարող է ունենալ ավելի լավ աշխատանքային պայմաններ (փոքր ունակություն, ցածր աղմուկ), արդյունքում՝ ի հայտ են գալիս ոչ լիարժեք թեստավորման ռիսկեր: Բացի այդ, չափումները կատարվում են սովորական արտաքին պայմաններում՝ PVT (Process/Voltage/Temprature), մինչդեռ թերությունները կարող են առաջանալ սովորական PVT արժեքներից դուրս պայմաններում, և այդ անսարքությունները ի հայտ կգան հետագայում՝ օգտագործողի մոտ, շահագործման ընթացքում: Ներդրված թեստավորման գործիքները զուրկ են այս սահմանափակումների հետևանքով

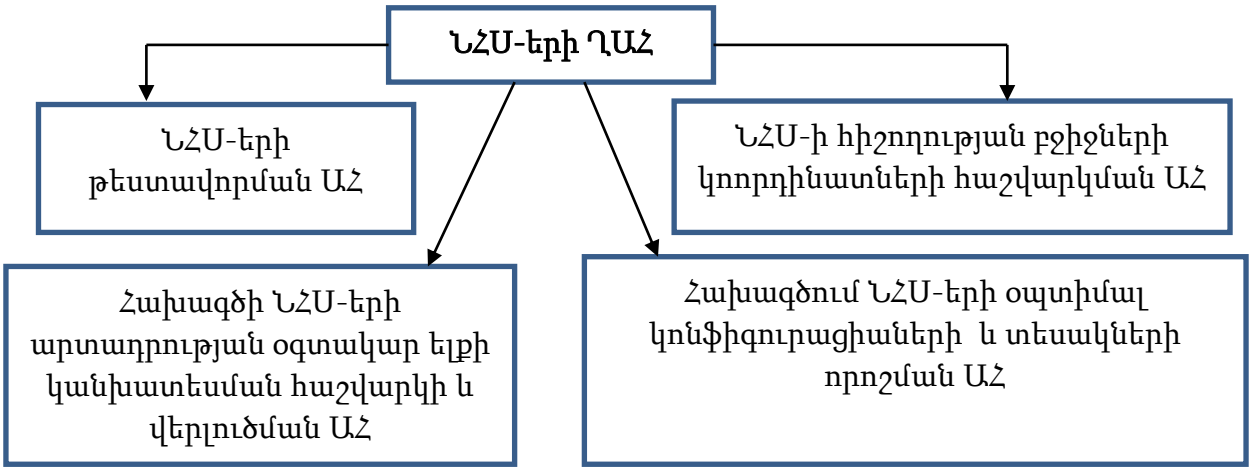
առաջացած թերություններից: Նրանք կարող են դիտարկել ցանկացած տեսակի և քանակի ազդանշաններ՝ օգտագործելով պատահական թեստավորման բիթերի հերթականություն, «վատագույն» PVT պայմանների համար:

2. Ներդրված ատոմատացված թեստավորման գործիքների նախագծումը և իրականացումը կատարվում է ավելի արագ: Օսցիլլոգրաֆի դեպքում մենք պետք է ապահովենք հետազոտվող ազդանշանների և հանգույցների հուսալի ֆիզիկական հասանելիությունը, որը իրականում երկարատև գործընթաց է: Այս գործընթացը կարող է սարքի նախագծման ժամանակին ավելացնել ևս մի քանի շաբաթներ, քանի որ պետք է ընտրել ազդանշանների ցուցակը, նախագծել տոպոլոգիայի կառուցվածքը (ազդանշանների հասանելիությունը), ստուգման թեստերը և վերջում կատարել այս ամենի համատեղ թեստավորումը:

   Ներդված գործիքների դեպքում կիրառվում է «միացրու և օգտագործիր» (plug and play) մոտեցումը՝ օգտագործելով ստանդարտ JTAG (Joint Test Action Group) թեստային միացումը և արդեն մշակված և բազմիցս ստուգված ծրագրային միջոց-ները, շահելով դրանց նախագծման համար անհրաժեշտ, շատ կարևոր շաբաթները:

3. Ներդրված գործիքները էժան են: Հաշվի առնելով այն փաստը, որ ժամանակակից PCI (Peripheral Component Interconnect) միակցիչների հիմքով թեստավորման սարքերի արժեքը կազմում է $200.000-$300.000 դոլար, գումարած միացումը ապահովող հարակից սարքերի արժեքը, ապա ակնհայտ է դառնում ներդրված գործիքների գնային առավելությունը (մի քանի անգամ) արտաքին սարքերի նկատմամբ:

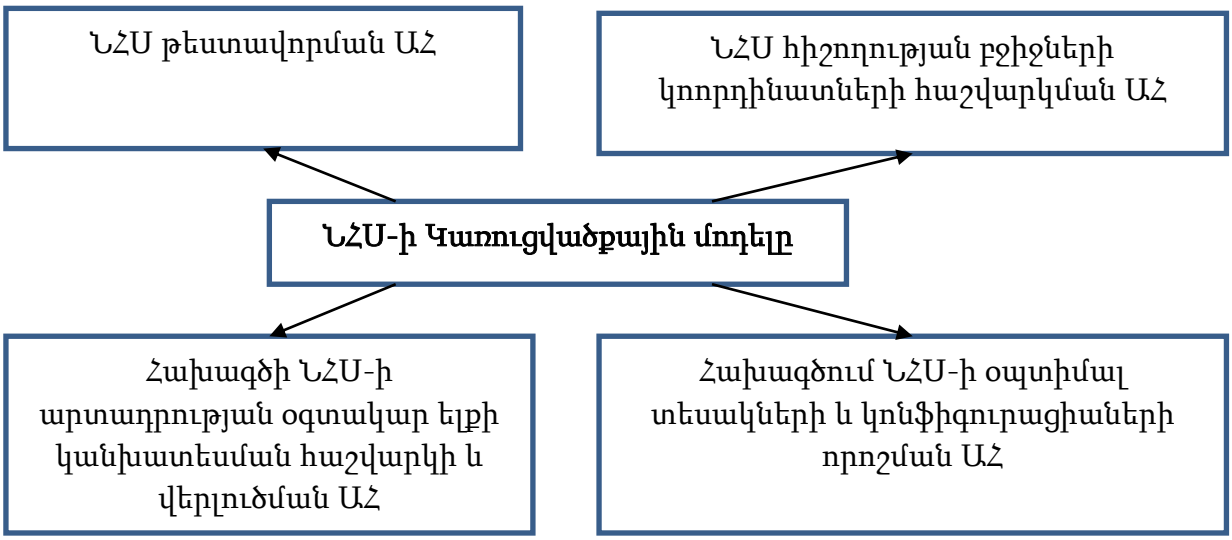
ՀՍ–երի հետ աշխատելիս ի հայտ են գալիս մի շարք ավտոմատացման խնդիրներ (Տես՝ նկ.1.3), որոնք պահանջում են միացյալ ղեկավարման ավտոմատացված համակարգի



Նկար 1.3 Ներդրված հիշողության սարքերի ՂԱՀ-ի բլոկ-սխեման

(ՂԱՀ) ստեղծումը իր ավտոմատացված ենթահամակարգերով ինչպիսիք են՝

- ներդրված հիշողության թեստավորման ավտոմատացված համակարգը (ԱՀ),
- ներդրված հիշողության բջիջների կոորդինատների հաշվարկման ԱՀ-ը,
- նախագծի մեջ օգտագործվող հիշողության սարքերի օպտիմալ տեսակների և կոնֆիգուրացիաների որոշման ԱՀ-ը,
- նախագծի և, մասնավորապես, հիշողության սարքի արտադրության օգտակար ելքի կանխատեսման հաշվարկի և, վերլուծման ԱՀ-ը:



Նկար 1.4 ՆՀՍ-ի կառուցվածքային մոդելի կիրառումը ՆՀՍ-ի ՂԱՀ-ում

Կատարված հետազոտությունները ցույցտվեցին, որ նշված խնդիրները ճարտարագետ նախագծողից պահանջում են տիրապետել և իր ծրագրային միջոցներում օգտագործել *հիշողության կառուցվածքի մասին տեղեկություններ* (Տես՝ նկ. 1.4): Հիշողության կառուցվածքային տեղեկատվությունը գտնվում է հիշողության արտադրության մուտքային ինֆորմացիա պարունակող ֆայլում, որը համակարգիչների վրա ստեղծվում և պահվում է գրաֆիկական GDSII (Graphic Database System II) ձևաչափով [6]: Նշենք, որ GDSII տեսակի գրաֆիկական ձևաչափը ծավալուն է, և այդ ֆայլի մշակումը այլ ծրագրային միջոցների կողմից անչափ ժամանակատար է:

Սահմանում հիշողության սարքի ֆիզիկական կառուցվածքի անհամապատասխանությունը կառուցվածքի տրամաբանական նկարագրությանը անվանում են կառուցվածքային խճողում (անգլերեն՝ structure scrambling):

Ատենախոսությանը ոլորտին վերաբերող գրականությունը ուսումնասիրելիս ի հայտ եկավ այն հետաքրքիր փաստը, որ չնայած նրան որ ստատիկ հիշողության սարքի

(ՀՍ) կառուցվածքի մասին տեղեկությունը բացարձակ կարևոր դեր է խաղում ՀՍ-ի ավտոմատ թեստավորման գործընթացում, այդ ասպարեզը մնացել էր քիչ հետազոտված և բառացիորեն մեկ - երկու տպագրություններ են հրապարակվել այդ թեմայի վերաբերյալ: Պետք է նշել, որ այդ տենդենցը պահպանվում է մինչ այժմ: Տպագրությունների քիչ լինելու պատճառներից է նաև այն փաստը, որ հիշողության կոմպիլյատորներ նախագծող և արտադրող ընկերությունները ձգտում են գաղտնի պահել իրենց կողմից նախագծված սարքերի կառուցվածքի մասին ինֆորմացիան, մրցակից էնկերությունների կողմից այդ ՀՍ-ի կառուցվածքային գաղափարները, գաղտնիքները իրենց նախագծերում օգտագործելու հնարավորությունից հնարավորինս զերծ պահելու նպատակով:

ՀՍ-ի կառուցվածքին վերաբերող առաջին տպագրություններից է 1998թ. լույս տեսած և 1999-ին վերահրատարակված, ՀՍ-ի թեստավորման և թեստ ալգորիթմների խնդիրներին նվիրված, Վան դե Գուրի գիրքը [16]: Այս գրքի 2 գլխի 4 բաժնում ներկայացված են SRAM տիպի հիշողությունների թեստավորման ալգորիթների տեսակները, ալգորիթներում օգտագործվող թեստային տվյալների օրինակները և դրանց հետ կապված հիմնախնդիրները: Գրքի եզրակացությունում (գլուխ 13-ում) հեղինակը ներկայացրել է հիշողության տվյալների և հասցեների խճողման ազդեցությունը ՀՍ-ի թեստավորման ալգորիթմի վրա:

Վան դե Գուրի մեկ այլ տպագրությունում [26] 2002թ., որը ըստ էության [16] –ի տրամաբանական զարգացումն է, ներկայացված են հասցեների և բիթային գծերի խճողումների առաջացման պատճառները և դրանց ազդեցությունը թեստ ալգորիթմի աշխատանքի արդյունավետության վրա:

Մեկ այլ տպագրությունում [32] 2003թ., ներկայացված են հիշողության բջիջների մի քանի տոպոլոգիաների օրինակներ և բջիջների այդ տոպոլոգիաներում դիմադրության թերությունների իրատեսական տարբերակները: Բերված է այդ թերությունների հետևանքով առաջացած անսարքությունների 18 տեսակների առաջացման հավանականությունը հիշողության բջիջներում և դրանց հայտնաբերումը 17N քայլային թեստ ալգորիթմի միջոցով: Այս հոդվածում արտացոլված է ՀՍ-ի GDSII-ում իրատեսական թերությունների հետազոտման խնդրի կարևորությունը:

Ստատիկ ՀՍ-երի խճողումների նկարագրությանն է նվիրված 2006թ. հրապարակված ատենախոսության [17] 2-րդ գլխի առաջին մասը, որտեղ ներկայացված են ստատիկ ՀՍ-ում առկա խճողումների տեսակները: Իսկ ատենախոսության 2-րդ գլխի երկրորդ մասում նկարագրված է խճողումների ստուգումը իրականացնող ավտոմատ ծրագրային հոսքի ընդհանուր աշխատանքը: Հարկ է նշել, որ վերոնշյալ ատենախոսության 2-րդ գլխի հետազոտությունների արդյունքները հեղինակի՝ Կ. Ալեքսանյանի հետ համատեղ կատարված աշխատանքներն են, որոնք և համահեղինակության իրավունքով, հրապարակվել են արտոնագրերի տեսքով [11], [12]: [17] ատենախոսությունում տրված է SRAM հիշողությունների խճողումների նկարագրման գրաֆիկական, ավտոմատ ծրագրային գործիքի (MIG) նկարագրությունը: MIG գործիքը օգտագործվում է հիշողության կոմպիլյատորի խճողումների ֆայլի ավտոմատ գեներացման համար:

Աշխատանքի նպատակն է՝

1. մշակել և դասակարգել հիշողության սարքերի կառուցվածքային մոդելը (ՀՍԿՄ),
2. ստեղծել ավտոմատացված համակարգ ՀՍԿՄ-ի գեներացման և ստուգման համար,
3. ստեղծել ավտոմատացված համակարգ՝ ավտոմատ ծրագրային հոսք, որի միջոցով հնարավոր կլինի ՀՍԿՄ-ը ավտոմատ ձևով դուրս բերել (to extract) հիշողության սարքի գրաֆիկական GDSII ձևաչափից,
4. ստեղծել ավտոմատացված ծրագրային միջավայր, որը հնարավորություն կտա ա) հիշողության սարքերում ներդնել տարբեր տեսակի ֆիզիկական թերություններ (defects) կամ վարքագծային անսարքություններ (faults) և բ) ստեղծել և ստուգել այդ անսարքությունը հայտնաբերող թեստավորման ալգորիթմը:

Գիտական նորությունը

Հետազոտվել են բազմակի Ստատիկ պատահական դիմումով հիշողության (ՍՊԴՀ) սարքերի (անգլերեն՝ Static Random Access Memory - SRAM) կառուցվածքներ: Հետագայում, ամբողջ տեքստում, ասելով հիշողության սարքեր մենք միշտ նկատի կունենաք միայն ՍՊԴՀ տիպի հիշողությունները:

Ստեղծվել է ՍՊԴՀ-երի կառուցվածքը նկարագրող ծրագրային լեզու՝ հիշողության սարքի կառուցվածքային մոդելը (ՀՍԿՄ) [11]:

Մշակվել են՝

- ալգորիթմ, որը թույլ է տալիս *ավտոմատ ձևով ստուգել* հիշողության կոմպիլյատորի կառուցվածքային մոդելը՝ այն համեմատելով հիշողության գրաֆիկական ներկայացման ֆայլերի՝ GDSII-ի, կառուցվածքի հետ [12]
- ալգորիթմ, որը թույլ է տալիս նմուշի գրաֆիկական ներկայացման GDSII ֆայլից *ավտոմատ ձևով ստանալ* հիշողության նմուշի կառուցվածքային մոդելը [14]
- ալգորիթմ, որը հնարավորություն է տալիս. ա) հիշողության սարքերում ներդնել տարբեր տեսակի ֆիզիկական և վարքագծային անսարքություններ և բ) մշակել այդ անսարքությունը հայտնաբերող թեստավորման ալգորիթմը [9]:

Հետազոտությունների առարկան

Աշխատանքում հետազոտությունների առարկա են հանդիսացել 250 նանոմետրից մինչև 14 նանոմետր չափսերի հիշողության կոմպիլյատորների կողմից գեներացված հիշողության նմուշները, այդ նմուշների կառուցվածքը, նմուշների հիշողության բջիջների ֆիզիկական կառուցվածքը, հիշողության բջիջներում և հիշողության մաս կազմող այլ հանգույցներում բազմազան ֆիզիկական և տրամաբանական անսարքությունները, ինքնաստուգող և նորոգվող հիշողության ավտոմատացված համակարգերը, հիշողությունների թեստավորման և նորոգման ընթացակարգը, հիշողության նմուշի կառուցվածքային մոդելը և մոդելի ճշտությունն ավտոմատ ձևով ստուգումը, հիշողության թեստավորման ալգորիթմների գեներացման և ավտոմատ ստուգման ձևերը, հիշողության կառուցվածքային մոդելի կիրառումը անսարքությունների հայտնաբերման, ախտորոշման և նորոգման համակարգերի մեջ:

Հետազոտությունների մեթոդները

Աշխատանքում օգտագործվել են. հանրահաշվական մինիմալացման և մաթեմատիկական մոդելավորման մեթոդներ, թվային և անալոգային կիսահաղորդչային սարքերի նախագծման և արտադրման, ԱՀ-երի հիմունքները:

Աշխատանքի կիրառական նշանակությունը

Աշխատանքի ընթացքում.

- Մշակվել է ծրագրային լեզու, որը հնարավորություն է տալիս նկարագրել ՀՍ-ի կառուցվածքային մոդելը, ընդհանուր դեպքում, հիշողության կոմպիլյատորների համար և, մասնավոր դեպքերում, հիշողության սարքերի նմուշի համար:

- Նախագծվել է ավտոմատ համակարգ, որի միջոցով կատարվում է ստացված կառուցվածքային մոդելի ստուգումը:
- Մշակվել է ավտոմատ համակարգ, որը թույլ է տալիս ավտոմատ ձևով դուրս բերել հիշողության նմուշի կառուցվածքային մոդելը GDSII ձևաչափից:
- Մշակվել է ավտոմատ համակարգ, որը թույլ է տալիս ավտոմատ ձևով տարբեր տեսակի ֆիզիկական թերություններ և վարքագծային անսարքություններ ներդնել հիշողության նմուշների մեջ և մշակել թեստային ալգորիթմներ, որոնք կհայտնաբերեն այդ անսարքությունները:

Աշխատանքի գիտական նշանակությունը հաստատվում է Ամերիկայի Միացյալ Նահանգներում ստացված չորս արտոնագրերով [11-14]:

Աշխատանքի կիրառական նշանակությունը հաստատվում է համապատասխան ներդրման արձանագրությամբ՝ հավելված 1:

Պաշտպանության են ներկայացվում

- հիշողության նմուշի մշակված կառուցվածքային մոդելը, որը թույլ է տալիս իրականացնել հիշողության սարքերի առավել արդյունավետ թեստավորումը, հիշողության սարքերում ավտոմատ ձևով անսարքությունների հայտնաբերման, ախտորոշման և նորոգման էլեկտրոնային և ծրագրային արդյունավետ համակարգերի ստեղծումը
- հիշողության կառուցվածքային մոդելի ավտոմատ ստուգման մեթոդը
- հիշողության նմուշից կառուցվածքային մոդելը ավտոմատ դուրսբերման մեթոդը
- հիշողության նմուշում ֆիզիկական թերություններ և վարքագծային անսարքություններ ներդնող և համապատասխան թեստային ալգորիթմներ ստուգող ավտոմատ ծրագրային հոսքը:

Տպագրված աշխատանքները

Աշխատանքի հիմնական արդյունքները և դրույթները զեկուցվել են և քննարկվել «Միկրոէլեկտրոնային սխեմաներ և համակարգեր» միջֆակուլտետային ամբիոնում, միջազգային on-line թեստավորման գիտաժողովում (IOLTS'2005) [7], միջազգային հաշվողական գիտությունների և տեղեկատվական տեխնոլոգիաների կոնֆերանսում

(CSIT'2011) [8], միջազգային նախագծման և թեստավորման խնդիրներին նվիրված կոնֆերանսում (EWDTS'2012) [9], ԵԱՊՀ գիտական սեմինարում 2014թ.[10]:

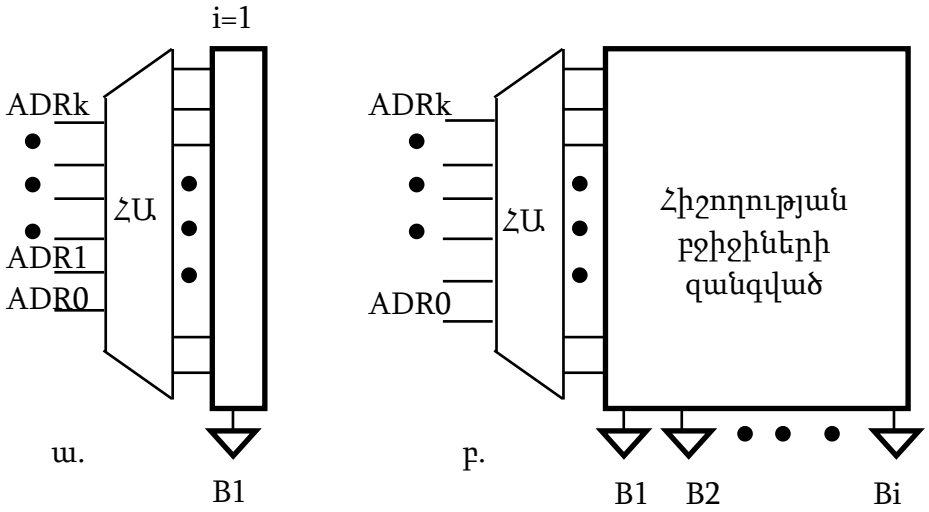
Աշխատանքի հիմնական արդյունքները տպագրված են 4 գիտական հոդվածներում [7-10] և 4 Ամերիկայի Միացյալ Նահանգների արտոնագրերի միջազգային կազմակերպությունում հաստատված արտոնագրերում [11-14]:

Աշխատանքի կառուցվածքը

Աշխատանքը կազմված է ներածության բաժնից, երեք գլուխներից, ամփոփումից, գրականության ցանկից և 6 հավելվածներից: Այն կազմում է 148 էջ, ունի 68 նկար և 16 աղյուսակ: Գրականության ցանկը զբաղեցնում է 10 էջ և ներառում է 103 աշխատություն, հավելվածները զբաղեցնում են 12 էջ:

ԳԼՈՒԽ 1 – ՍՏԱՏԻԿ ՀԻՇՈՂՈՒԹՅԱՆ ՆՄՈՒՇՆԵՐԻ ՆԱԽԱԳԾՄԱՆ ԱՎՏՈՄԱՏԱՑՎԱԾ ՀԱՄԱԿԱՐԳԻ ԿԱՌՈՒՑՎԱԾՔԱՅԻՆ ՄՈԴԵԼԻ ՏԱՐՐԵՐԸ

1.1 Ներդրված հիշողության նմուշների թեստավորման ավտոմատացված համակարգի արդյունավետության բարձրացման ձևերը



Նկար 1.5. B1 և Bi տեսակի հիշողության սարքերը, որտեղ

ա. B1 տեսակի հիշողության սարքի բլոկ-սխեման

բ. Bi տեսակի հիշողության սարքի բլոկ-սխեման

ADR0-ADRk – Մուտքային հասցեներն են

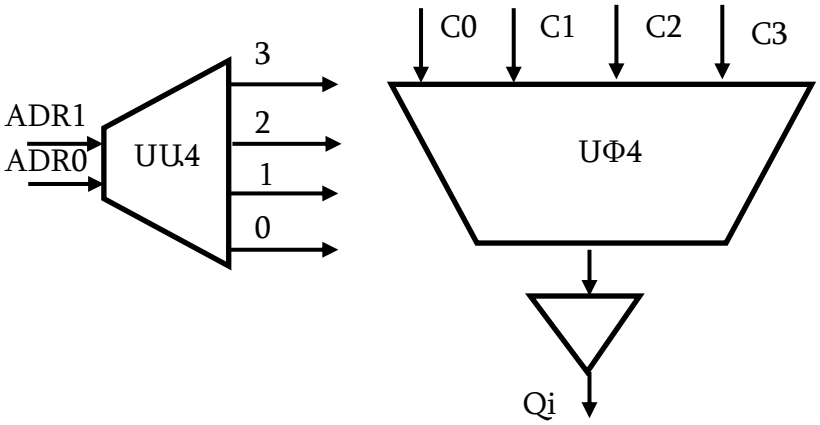
ՀԱ – Հասցեների Ապակոդավորման սխեմա

Թեզում կատարված աշխատանքների հիմնական հետազոտման առարկա են հանդիսանում ներդրված հիշողության սարքերը: Մասնագիտացված գրականությունում [15-18] ՀՍ-երը դիտարկում են որպես B1 և Bi տեսակի հիշողության սարքեր: Նկար 1.5-ի ա. մասում բերված է B1 տեսակի հիշողության կառուցվածքային սխեման: B1 տեսակի կոչվում են այն հիշողության սարքերը, որոնց հիշողության զանգվածը հասցեավորման ողջ տիրույթում հասցեավորում է մեկ բջիջ: Այլ ձևով ասած, B1 տեսակի հիշողության սարքերում յուրաքանչյուր հասցեին վերագրվում է բջիջների զանգվածում գտնվող

բջիջներից միայն մեկ հիշողության բջիջ: Այդ տեսակի հիշողությունների բջիջների դաշտը մեկ չափանի է, մինչդեռ էլեկտրոնային սարքերում տարածված են և գերակշռող մաս են կազմում բազմաբիթանի՝ Bi տեսակի, բառային կառուցվածք ունեցող սարքերը (Տես՝ նկ. 1.5-ի բ.): Bi բառային կառուցվածք ունեցող հիշողության սարքերում գրվող, կարդացվող տվյալները (բառերը) ունեն i քանակության բիթեր: Bi տեսակի հիշողության բջիջների դաշտը (զանգվածը) երկչափանի է: Նկար 1.5-ի բ.-ից երևում է որ այն դեպքում, երբ բառի բիթերի քանակը հավասար է հիշողության ֆիզիկական տողերի քանակին, ապա այդ դեպքում հիշողության բջիջների զանգվածի մակերեսը կստանա քառակուսի ձև: Հակառակ դեպքում զանգվածի մակերևույթը ստանում է ուղղանկյան տեսք, որտեղ ուղղանկյան կողերի հարաբերությունը համապատասխանում է հիշողության հասցեի բիթերի քանակի և հիշողության բառի բիթերի քանակի հարաբերությանը: Բջիջների զանգվածի մակերևույթի տեսքը կարևոր դեր է խաղում հիշողության նմուշի աշխատանքի հուսալիության, արագագործության, ջերմակայունության և նմուշում օգտագործված էլեկտրական հզորությունը ցրման հաշվակներում: Մյուս կողմից, ներդրված հիշողության սարքի տոպոլոգիայի ֆիզիկական տեսքը կարևոր դեր է խաղում այն դեպքում երբ անհրաժեշտ է այդ հիշողությունը (իսկ ընդհանուր դեպքում հիշողությունների զանգվածը) օպտիմալ ձևով տեղադրել նախագծվող էլեկտրոնային սարքի թիթեղում, լուծելով սարքի հանգույցների տոպոլոգիական բաշխման լավարկման խնդիրը: Հիշողության զանգվածի կողերի հարաբերությունը էլ ավելի կառավարելի դարձնելու նպատակով և, որպես անմիջական հետևանք, հիշողության բջիջների զանգվածի մակերեսի ձևի օպտիմալությունը ապահովելու նպատակով ՀՍ-ի ճարտարագետները, կիրառում են հետևյալ մեթոդը՝ օգտագործելով էլեկտրոնային հասցեավորվող փոխանջատիչ (անգլերեն multiplexser), հիշողության բառի յուրաքանչյուր բիթին միացվում է մեկից ավելի հիշողության զանգվածի սյուներ:

ՀՍ-ի աշխատանքի ընթացքում սյուների հասցեները որոշող հասցեների բիթերը, սյուների հասցեների Ապակոդավորման սխեմայի միջոցով ապակոդավորվում են՝ ակտիվացնելով տող ընտրող ազդանշաններից մեկը, որն էլ Սյուների Փոխանջատիչ հանգույցի օգնությամբ միացնում է ակտիվացված սյունը հիշողության ելքին՝ Qi-ին: Հիշողության սարքի այս բնութագիրը նկարագրվում է «Սյուների Փոխանջատիչ» (ՍՓ) պարամետրով (անգլերեն column multiplexer CM):

Նկար 1.6-ում ներկայացված է Սյուների Փոխանջատման սխեմայի բլոկ-սխեման: Այսպիսով, Ստատիկ Պատահական Դիմումով Հիշողության (ՍՊԴՀ) կառուցվածքի հիմնական հանգույցները երկուսն են՝ ա) հիշողության բջիջները պարունակող զանգվածը, և բ) ՀՍ-ի աշխատանքը ապահովող տրամաբանական մասը:



Նկար 1.6. Սյուների Փոխանջատման սխեմայի բլոկ-սխեման

ADR[0,1] – Հասցեական բիթեր

ՍԱ4 – Սյուների հասցեների Ապակոդավորման սխեմա 2x4

ՍՓ4- Սյուների Փոխանջատման սխեմա 4x1

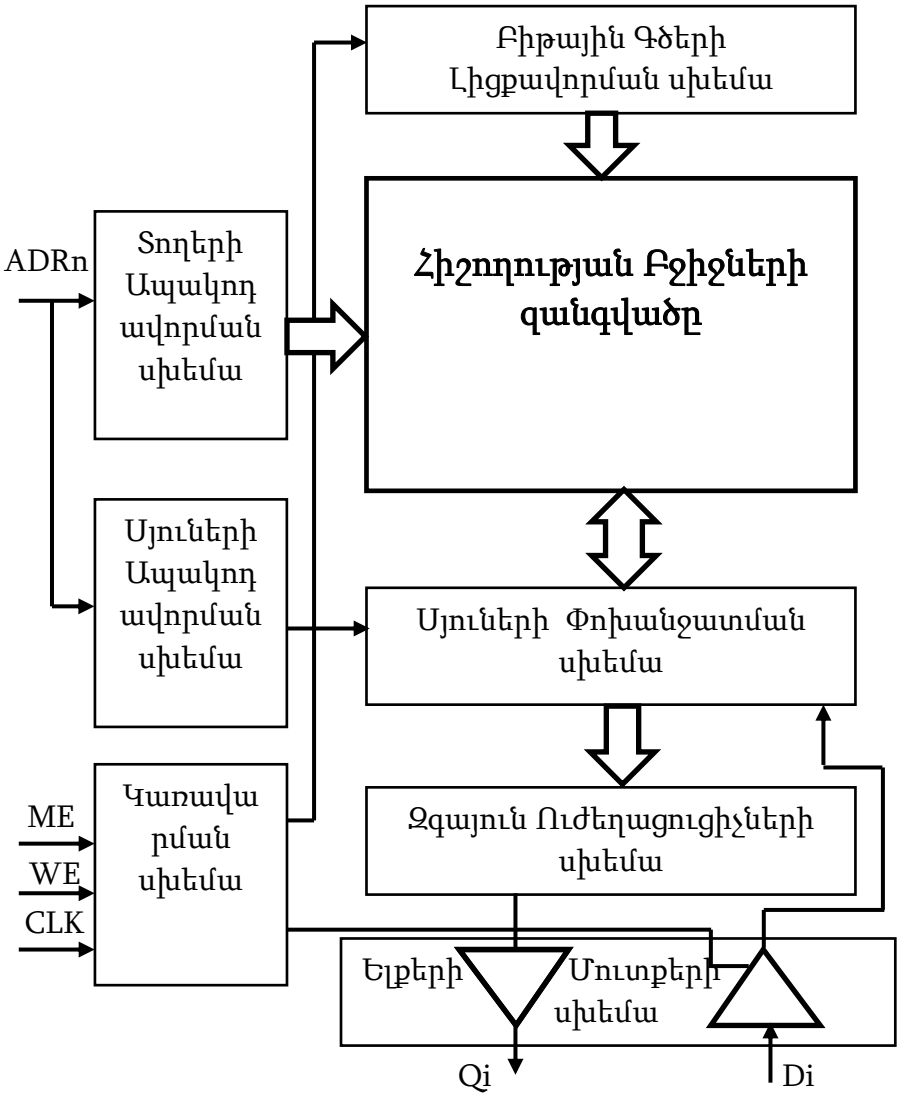
C0-C3 – Հարևան սյուներ հիշողության զանգվածում

Qi – «i» Ելքը

Կախված հիշողության սարքերի տեսակից՝ (մեկ մուտքանի կամ բազմամուտքանի) և ֆունկցիոնալ առանձնահատկություններից (արագագործ ՀՍ, ցածր լարումով ՀՍ, մեծ խտությամբ և այլն) ՀՍ-երի կառուցվածքները կարող են էականորեն տարբերվել մեկը մյուսից: Հիշողության սարքի նմուշը բաղկացած է հիմնական ութ հանգույցներից.

1. Հիշողության Բջիջների զանգված
2. Տողերի Ապակոդավորման սխեմա
3. Սյուների Ապակոդավորման սխեմա
4. Կառավարման և սինխրոնացման սխեմա
5. Զգայուն Ուժեղացուցիչների սխեմա
6. Սյուների Փոխանջատման սխեմա
7. Բիթային Գծերի Լիցքավորման սխեմա
8. Մուտքերի և Ելքերի սխեմա

Նկար 1.7-ում ներկայացված է ժամանակակից ՍՊԴՀՍ-ի մանրամասնված բլոկ-սխեման:



Նկար 1.7. Ստատիկ Պատահական Դիմումով Հիշողության բլոկ-սխեման

որտեղ

ADRn – Հասցեական բիթերը

ME – Հիշողության աշտանքը թույլատրող ազդանշան

WE – Գրել թույլատրող ազդանշան

CLK – Հիշողության աշտանքային հաճախականություն

Qi – Ելքային ազդանշանները

Di – Մուտքային ազդանշանները

Ինչպես կտեսնեք հետագայում, նմուշի այս հանգույցների ֆիզիկական կառուցվածքը կարևոր դեր ունի հիշողության բջիջների ֆիզիկական կոորդինատների

հաշվարկման ժամանակ, էականորեն ազդում է հիշողությունը ստուգող թեստային ալգորիթմի արդյունավետության վրա, և ՀՍ-ի օգտակար ելքի գնահատման աշխատանքների ժամանակ:

Դիտարկենք ՍՊԴՀ նմուշը բնութագրող հիմնական տրամաբանական և ֆիզիկական պարամետրերը.

- NW (անգլերեն՝ number of words) – **Բառերի քանակը** հիշողությունում
- NB (անգլերեն՝ number of bits) - **Բիթերի քանակը** հիշողության բառի մեջ
- BK (անգլերեն՝ number of banks) – Հիշողության նմուշում **հորիզոնական բանկերի** քանակը
- CM (անգլերեն՝ column mux) – **Սյուների փոխանջատման** քանակը
- PR (անգլերեն՝ physical rows) – **Ֆիզիկական տողերի** քանակը բջիջների զանգվածում
- PRB (անգլերեն՝ physical rows of the bank)– **Ֆիզիկական տողերի** քանակը նմուշի բջիջների զանգվածի մեկ բանկում
- PC (անգլերեն՝ physical columns) - **Ֆիզիկական սյուների** քանակը բջիջների զանգվածում:

Ներկայացնենք ՍՊԴՀ-րը նկարագրող և լայն գործածություն ունեցող բանաձևերից մի քանիսը.

$$PR=NW/CM \quad (1)$$

$$PRB=PR/BK=NW/(CM*BK) \quad (2)$$

$$PC=NB*CM \quad (3)$$

Հիշողության բջիջների քանակը նմուշի զանգվածում (NCell) հավասար է

NCell= PR*PC, կիրառենք (1) և (2) և կստանաք

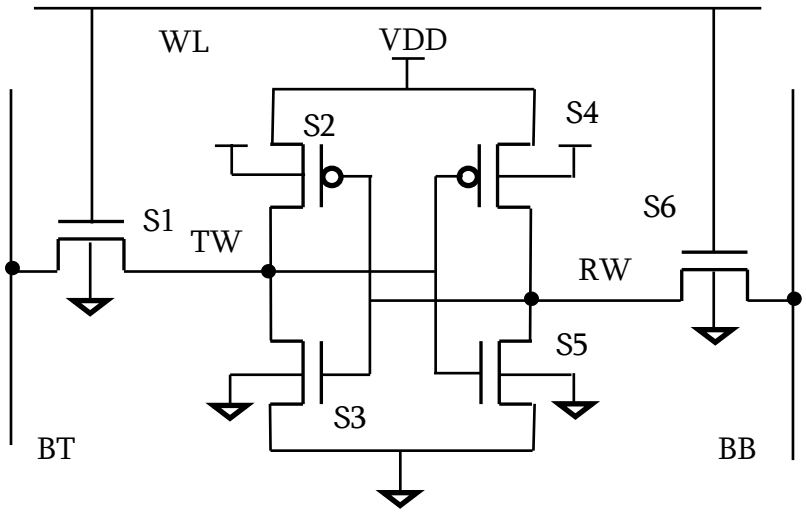
$$NCell=NW*NB \quad (4)$$

Եթե հիշողությունը ունի բանկային կառուցվածք, ապա բջիջների քանակը մեկ բանկի զանգվածում (NCellBK) հաշվարկվում է հետևյալ պարզ բանաձևով,

$$NCellBK=NW*NB/BK \quad (5)$$

Ակնհայտե որ իմանալով հիշողության բջջի բարձրությունը և լայնությունը և օգտվելով բանաձև (5) –ից, պարզագույն դեպքի համար, կարելի է հաշվել հիշողության զանգվածի մակերևույթը:

Նկար 1.8-ում տրված է ստատիկ հիշողության, արդեն դասական դարձած, վեց տրանզիստորային, էլեկտրական սխեման: Հիշողության սարքում տվյալները գրվում են և պահպանվում են հիշողության բջիջում: Իսկ Նկար 1.9-ում տրված է հիշողության



Նկար 1.8. Հիշողության բջիջի էլեկտրական սխեման, որտեղ

BT (Bit-line True) – Ուղիղ արժեքով բիթային գիծը

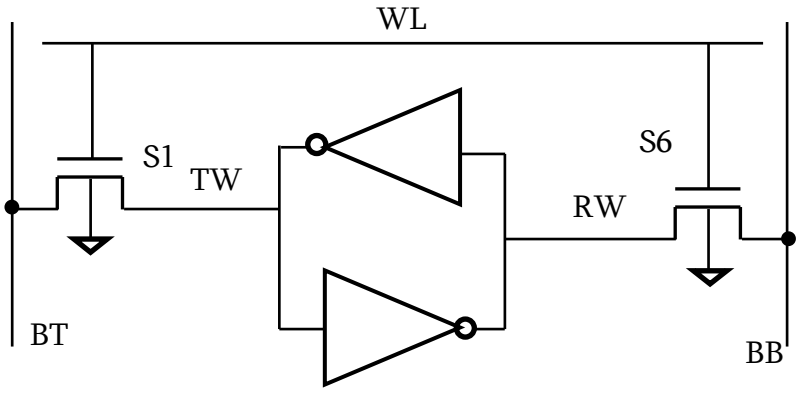
BB (Bit-line Bar) – Հակառակ արժեքով բիթային գիծը

WL (Word Line) – Բառային գիծը

TW (True Wire) – Ճիշտ արժեքով մետաղալարը

RW (Reverse Wire) – Հակառակ արժեքով մետաղալարը

բջիջի պարզեցված, պայմանական սխեման:



Նկար 1.9. Հիշողության բջջի պարզեցված սխեման

Վերլուծենք այդ բջջի աշխատանքը: Ինչպես երևում է 1.8 և 1.9 նկարներից, հիշողության բջիջը ունի բացարձակապես սիմետրիկ կառուցվածք: Պայմանականորեն կարելի է

ասել, որ այն բաղկացած է «աջ» և «ձախ» մասերից: Յուրաքանչյուր մասը իր մեջ ընդգրկում է մեկ ինվերտոր՝ կառուցված S2 և S3 տրանզիստորներով ձախ մասի համար (համապատասխանորեն S4 և S5 տրանզիստորներով աջ մասի համար) և բջիջը Բիթային BT գծին միացնող, այդպես կոչված, անցումային S1 տրանզիստորից (S6 տրանզիստորը՝ բջջի աջ BB Բիթային գծի համար): ՀՍ-ի սնուցման լարումը անմիջապես միացնելուց հետո աջ և ձախ մասերը անցնում են հակառակ տրամաբանական վիճակ՝ «0» կամ «1»: Այդ վիճակը պահպանում է այնքան ժամանակ, մինչև, արտաքին սխեմաների ազդեցության ներքո բջջի տրամաբանական արժեքը շրջվում է՝ ստանալով նախորդին հակառակ արժեքը: Աշխատանքի ընթացքում բջջի աջ և ձախ մասերը միշտ գտնվում են հակառակ տրամաբանական վիճակում: Ընդ որում, BT գծի վրայի տրամաբանական արժեքը միշտ համապատասխանում է բջջի տրամաբանական արժեքին:

Հիշողության բջիջը ունի աշխատանքային երեք ռեժիմ: Հիշողության բջջի ռեժիմները ներկայցված են Աղյուսակ 1.1-ում, որտեղ {BTi} գծին համապատասխանում է գործողության ընթացիկ արժեքը, իսկ {i-1}-ին համապատասխանում է բջջում կատարված նախորդ գործողության արժեքին:

Աղյուսակ 1.1

Հիշողության բջջի աշխատանքային ռեժիմները

TW	RW	BT	BB	Բջջում կատարվող գործողությունը՝
TWi-1	RW i-1	x	x	Վիճակի պահպանում
1	0	1	0	Գրել «1» արժեք
0	1	0	1	Գրել «0» արժեք
TW i-1	RW i-1	TWi-1	RW i-1	Կարդալ բջջի արժեքը

Ներկայումս հիշողության սարքերի նմուշները և նմուշների հետ պահանջվող հարակից ֆայլերը որոնցից են՝ գրաֆիկական GDSII ֆայլը, նմուշի նկարագրությունը, նմուշի Verilog և SPICE ֆայլերը, վարքագծային և կառուցվածքային մոդելները, նմուշի էլեկտրական բնութագրերը և մի շարք այլ ֆայլեր և կաղապարներ գեներացվում են ավտոմատ ձևով՝ «*Հիշողության Կոմպիլյատոր*» կոչվող ծրագրային գործիքի միջոցով: Հիշողության Կոմպիլյատորը պարունակում է նախապես նախագծված գրադարանային

տարրեր: Այդ տարրերը իրենցից ներկայացանում են որոշակի տեխնոլոգիայի համար նախագծված պարզագույն՝ չտրոհվող, բջիջների նկարագրությունները, բջիջների տոպոլոգիան, էլեկտրական սխեման: Աշխատեցնելով Հիշողության Կոմպիլյատորը և փոխանցելով նրան (ծրագրային փաթեթին) գեներացվող հիշողության նմուշը նկարագրող մուտքային տվյալները՝ NW, NB, CM, BK և մի շարք այլ անհրաժեշտ ֆունկցիոնալ և կառուցվածքային պարամետրներ, ստանում ենք հիշողության նմուշը ավարտուն, ամբողջական փաթեթի տեսքով: Գեներացված ֆայլերի փաթեթը ընդգրկում է նմուշը արտադրելու և ստուգելու համար անհրաժեշտ ամբողջական ֆայլերը, հիշողության նմուշի նկարագրությունը, նմուշը բնութագրող տվյալները պատվիրատուին օգտագործման համար հարմար ֆայլերի տեսքով: Հիշողության նմուշի գեներացումը կարելի է ներկայացնել հետևյալ արտահայտությամբ՝

$M_{inst} = F_{comp}\{k1,k2,\ldots,kn\}$, որտեղ (6)

M_{inst} – memory instance, հիշողության նմուշի փաթեթն է,

F_{comp} – կառուցողական ֆունկցիան է՝ մեր դեպքում Հիշողության կոմպիլյատոր գործիքը,

{k1,k2,…kn} – մուտքային պարամետրերն են:

Մուտքային պարամետրները լինում են երկու տեսակի. ա) պարտադիր {NW, NB, CM}, և բ) լրացուցիչ {BK, VBK, cd_enable, reduncy_enable, bist_enable, frequency, instan_orientation և այլն }: Այդպիսով՝

$M_{inst} = F_{comp}\{NB, NW, CM, p1, p2,\ldots pn\}$, որտեղ (7)

{ p1, p2,…pn} – լրացուցիչ պարամետրերն են:

Ինտեգրացված սխեմաների արտադրության շահութաբերությունը կախված է միկրոսխեմայի արտադրության օգտակար ելքից, որը սահմանվում է որպես արտադրված սխեմաներից աշխատունակների հարաբերությունը ամբողջին: Հարկ է նշել, որ արտադրության պրոցեսի նույնիսկ ամենալավ պայմաններում, հնարավոր չէ ստանալ 100% արտադրության օգտակար ելք: Գոյություն ունեն արտադրության պրոցեսի բազմակի թերություններ, որոնց պատճառով առաջանում է արտադրական խոտան: Արտադրության տեխնոլոգիաների շարունակական զարգացման շնորհիվ՝ միկրոսխեմաներում ակտիվ տարրերի ֆիզիկական չափերի կրճատման հետևանքով,

արտադրական խոտանը զգալիորեն ավելանում է՝ հասնելով 70-80 տոկոսի : Դրան զուգահեռ մեկ բյուրեղի մակերևույթի վրա գտնվող «միջուկների» քանակը և խտությունը անընդհատ ավելանում է, որը բերում է բյուրեղի ընդհանուր չափերի մեծացմանը՝ դրանով էլ ավելի, նվազեցնելով սխեմայի արտադրության օգտակար ելքը: Միկրոսխեմաների արտադրության օգտակար ելքի մեծացումը մնում է արդիական խնդիր, որն արդյունավետ լուծում է պահանջում [19]:

ՀՍ-ի թեստավորման արդյունավետության բարձրացումը հանդիսանում է ՀՍ-ի օգտակար ելքի ավելացման միջոցներից կարևորներից մեկը: Լավացնելով միկրոսխեմայի թեստավորման ԱՀ-ի արդյունավետությունը՝ ավելացնում ենք միկրոսխեմաների օգտակար ելքը [20-22]: Հիշողության սարքերի թեստավորման հետ կապված խնդիրները էականորեն տարբերվում են տրամաբանական սարքերի թեստավորումից: Դրա հիմնական պատճառն այն է, որ հիշողության սարքում առաջացած անսարքության վարքագիծը համապատասխանում է տրամաբանական սարքերում առաջացող անսարքության վարքագծին, մինչդեռ օգտագործվող անսարքությունների մոդելը, հիմնականում, համապատասխանում է թվային մոդելին (օրինակ կարճ միացում, կամ տրամաբանական «0» կամ «1»): Բացի այդ, ՀՍ-ի ներքին գործողությունները իրականացվում են որպես տրամաբանական սխեմաներում կատարվող գործողություններ: Արդյունքում անսարքության գումարային վարքագիծը դառնում է շատ բարդ, խճճված և դժվար վերլուծելի [15], [16]:

Ժամանակակից ներդրված հիշողության սարքերի օգտակար ելքի ավելացման նպատակով այդ սարքերում, դեռ նախագծման փուլում, հիշողության բջիջների զանգվածում, տեղադրվում են պահեստային սյուներ կամ տողեր կամ էլ միաժամանակ պահեստային և՛ տողեր, և՛ սյուներ: ՀՍ-ում ներդրված այս պահեստային տարրերը օգտագործվում են հետևյալ ձևով՝ ներդրված ՀՍ-ի սնուցման լարումը անմիջապես միացնելուց հետո կատարվում է այդ սարքի աշխատանքի ստուգումը, որը կատարվում է հիշողության սարքի հետ միաժամանակ պատրաստված հատուկ էլեկտրոնային սխեմայի միջոցով: Ժամանակակից սարքերում այդ սխեմայի դեր է կատարում հիշողությունը ստուգող, միայն այդ նպատակի համար նախագծված և միկրոսխեմայում ներդրված պրոցեսորը [19-22]: Այդ պրոցեսորը աշխատեցնում է ներդրված ինքնաթեստավորման ալգորիթմը՝ BIST: Այդ BIST ալգորիթմի աշխատանքի ընթացքում

հայտնաբերված սխալի մասին տեղեկությունը փոխանցվում է պրոցեսորին, որն էլ ի պատասխան հայտնաբերված սխալի աշխատեցնում է ներդրված՝ այդ սխալը «վերացնող» ալգորիթմը - BIRA: BIRA-ի նպատակն է օպտիմալ ձևով՝ պահեստային սյուների, տողերի միջոցով «փոխարինել» անսարք բջիջը, իսկ ընդհանուր դեպքում, հիշողության զանգվածում անսարք բջիջներ պարունակող սյուները կամ էլ տողերը: Անսարք բջիջներ պարունակող սյունը, շնորհիվ կատարվող սյուների շեղումների, փոխանակվում է պահեստային սյունով, իսկ անսարք տողը անջատվում է հիշողության զանգվածի հասցեավորման տիրույթից, փոխարենը վնասված տողի հասցեն փոխանակվում է՝ վերահասցեավորվում է, պահեստային տողի հասցեով [2], [19]: BIST-ի և BIRA-ի ալգորիթմների աշխատանքի շնորհիվ հնարավոր է դառնում լիովին վերականգնել հիշողության սարքի աշխատանքը:

ՀՍ-ի թեստավորման ընթացքում անսարքությունների հայտնաբերումը, անսարքությունների պատճառների վերլուծումը և արդյունքում այդ պատճառների վերացումը, լինեն դրանք տեխնոլոգիական պատճառներ, կամ սխեմատիկական սխալ լուծումների հետևանք, կամ էլ արտադրական վատ պայմանների պատճառով առաջացած անսարքություններ, հանդիսանում է թեստավորման հիմնախնդիրն: Թեստավորման աշխատանքների ողջ գործընթացը բերում է հիշողության սարքերի ՕԵ-ի բարձրացմանը և այդ գործընթացի կարևորագույն մասն է կազմում հիշողության աշխատանքը ստուգող թեստային ալգորիթմը: Ներկայումս, ներդրված ՀՍ-ի թեստավորման նպատակով կիրառվող թեստային ալգորիթմներից ամենատարածվածը «Մարշ» (անգլերեն՝ March) տեսակի ալգորիթմներն են: «Մարշ» ալգորիթմները, իրենց պարզության շնորհիվ, իրականցման համար պահանջում են նվազագույն միջոցներ: Միաժամանակ լինելով շատ արդյունավետ՝ «Մարշ» տեսակի ալգորիթմները լայնորեն կիրառվում են որպես ներդրված ՍՊԴՀ-ի ԱՀ թեստավորման ալգորիթմներ [8], [23], [24]:

«Մարշ» ալգորիթմը բաղկացած է մարշ էլեմենտների վերջավոր հաջորդականությունից՝

$$M = \{M_1, M_2, ..., M_k\} : \qquad (8)$$

Մարշ էլեմենտը կազմված է հասցեավորման ուղղությունը որոշող նիշից (⇑՝ աճող, ⇓՝ նվազող) և վերջավոր թվով մարշ գործողություններից (W- գրել բառը, R - կարդալ բառը և D - հապաղում):

$$M_i = A_i\{O_1, O_2, \ldots, O_m\}\ , \quad \text{որտեղ} \qquad (9)$$

$$A_i \in \{\Uparrow, \Downarrow, \Updownarrow\};\ j \in \{1,m\} \qquad (10)$$

$$O_j \in \{R0, R1, W0, W1, Dn, n\};\ i \in \{1,k\}\ n \in \{1,h\} \qquad (11)$$

Մարշ թեստի կիրառության ժամանակ, յուրաքանչյուր բջջի վրա համապատասխան հերթականությամբ կիրառվում են մարշ գործողությունների հաջորդականությունները: Ընդ որում՝

$\Uparrow$ – մարշ տարրի գործողությունները կիրառվում են հիշողության բոլոր հասցեներում աճման կարգով՝ սկսած զրոյից մինչև հիշողության վերջին հասցե,

$\Downarrow$ – մարշ տարրի գործողությունները կիրառվում են հիշողության բոլոր հասցեներում նվազման կարգով՝ սկսած հիշողության վերջին հասցեից մինչև զրոյական հասցե,

$\Updownarrow$ - մարշ տարրի գործողությունները կիրառվում են հիշողության բոլոր բջիջներում՝ հասցեների նվազման կամ աճման կարգով,

R0 – ընթացիկ հասցեից կատարվում է կարդալ (անգլերեն՝ Read) գործողությունը՝ ակնկալելով տրամաբանական «զրո» արժեք: Ոչ «զրո» արժեք կարդալու դեպքում արձանագրվում է սխալ,

R1 – ընթացիկ հասցեից կատարվում է կարդալ գործողությունը՝ ակնկալելով տրամաբանական «մեկ» արժեք: Ոչ «մեկ» արժեք կարդալու դեպքում արձանագրվում է սխալ,

W0 – ընթացիկ հասցեում կատարվում է տրամաբանական «զրո» արժեքի գրել (անգլերեն՝ Write) գործողությունը,

W1 – ընթացիկ հասցեում կատարվում է տրամաբանական «մեկ» արժեքի գրել գործողությունը,

Dn – հապաղում (անգլերեն Delay) գործողություն է, որի տևողությունը հաշվարկվում է n*Tclk բանաձևով, որտեղ «n»-ը դա ամբողջ թիվ է իսկ Tclk հիշողության պարզագույն գործողության տևողությունը:

Մարշ ալգորիթմը բնութագրվում է «Մարշ» ալգորիթմի բարդությունը բնութագրող (N) պարամետրով: Մարշ ալգորիթմների բարդությունը՝ N պարամետրը, ցույց է տալիս Մարշ ալգորիթմի աշխատանքի ընթացքում, մեկ բջջի վրա կատարվող գործողությունների քանակը:

Նկար 1.10-ում՝ որպես օրինակ, ներկայացված է «Մարշ C» ալգորիթմի տարրերը իրենց պարզաբանմամբ (ըստ [8], [24]):

$$\{\Updownarrow(W0);\quad \Uparrow(R0,W1);\quad \Uparrow(R1,W0);\quad \Updownarrow(R0);\quad \Downarrow(R0,W1);\quad \Downarrow(R1,W0);\quad \Updownarrow(R0)\}$$

M0 M1 M2 M3 M4 M5 M6

Նկար 1.10. Մարշ C ալգորիթմը

Որտեղ M0-M6 մարշ տարրն են Այս ալգորիթմի բարդությունը՝ N=11:

Ինչպես տեսնում ենք պարզագույն մարշ տարրը M_i բաղկացած է երկու մասերից՝ հասցեավորման ուղղությունը ցույց տվող մասից A_i և բջիջների վրա կատարվող գործողություններից O_m: Իր հերթին կատարվող գործողությունը բաղկացած է գործողությունից (W, R, D) և գործողության տվյալից: Մեկ բիթային (B1) հիշողությունների համար գործողության տվյալը՝ գրականությունում այն նաև անվանում են «գործողության օրինակ» (Background Pattern –BP), ստանում է «զրո» կամ «մեկ» արժեքը: Իսկ բառային (Bi) հիշողություններում այն ունի «i» երկարություն: Հիշողության թեստավորման արդյունավետությունը բարձրացնելու նպատակով թեստավորման ընթացքում օգտագործվում է գործողության տվյալների (ԳՏ) այնպիսի հաջորդականություն, որի արդյունքում հիշողության զանգվածում գրանցված տվյալները կկազմեն որոշակի ֆիզիկական պատկեր: Սովորաբար, հիշողության սարքը թեստավորելիս օգտագործվում են մի քանի տեսակի ֆիզիկական պատկերներ (անգլերեն՝ physical Background Patterns (PBP)) ապահովելով ՀՍ-ի լիարժեք թեստավորումը: Աղյուսակ 1.2-ում ներկայացված են թեստավորման գործողության ժամանակ օգտագործվող տվյալների ֆիզիկական օրինակներից հիմնականները [8], [25]:

Աղյուսակ 1.2

Թեստավորման գործողության տվյալների ֆիզիկական տվյալները

1.	Solid (SO) 00000000 00000000 00000000 00000000	Solid bar (˜SO) 11111111 11111111 11111111 11111111

Աղյուսակ 1.2 (շարունակություն)

Թեստավորման գործողության տվյալների ֆիզիկական տվյալները

2.	Column Stripe (CS) 01010101 01010101 01010101 01010101	Column Stripe bar (~CS) 10101010 10101010 10101010 10101010
3.	Row Stripe (RS) 00000000 11111111 00000000 11111111	Row Stripe bar (~RS) 11111111 00000000 11111111 00000000
4.	Checkerboard (CB) 01010101 10101010 01010101 10101010	Checkerboard bar (~CB) 10101010 01010101 10101010 01010101
5.	Double Column Stripe (DCS) 00110011 00110011 00110011 00110011	Double Column Stripe bar (~DCS) 11001100 11001100 11001100 11001100
6.	Double Row Stripe (DRS) 00000000 00000000 11111111 11111111	Double Row Stripe bar (~DRS) 11111111 11111111 00000000 00000000
7.	Double Checkerboard (DCB) 00110011 11001100 00110011 11001100	Double Checkerboard bar (~DCB) 11001100 00110011 11001100 00110011
8.	Bit-line Checkerboard (TCB)	Bit-line Checkerboard bar (~TCB)

Ներդրված ՀՍ-ի թեստավորման գործընթացում ժամանակի և տեղի խիստ սահմանափակության պատճառով օգտագործվում են BP-ից երկուսը կամ, լավագույն դեպքում, երեքը: Իսկ մնացած BP-ները, որպես լրացուցիչ տարբերակ, օգտագործվում են միայն ՀՍ-ի լայնածավալ թեստավորման, հետազոտման աշխատանքներ իրականացնելիս: Առանձնահատուկ հետաքրքրություն է ներկայացնում հիշողության

բջիջում Բիթային գծերի կառուցվածքային հերթականությունը հաշվի առնող Bit-line Checkerboard (TCB) գործողության Օրինակը: Ինչպես ցույց կտրվի հետագայում, TCB-ի օրինակը ապահովում է անսարքությունների հայտնաբերման ամենաբարձր հավանականությունը և դրանով է բացատրվում թեստային ալգորիթմում այս BP-ի օգտագործման կարևորությունը: TCB-ի օրինակը կիրառման համար պետք է իմանալ հիշողության զանգվածի բջիջներում բիթային գծերի ֆիզիկական բաշխվածությունը, որն էլ իր հերթին հիշողության սարքերի կառուցվածքային ինֆորմացիայի՝ **խճողման** (անգլերեն՝ scrambling), տարրերի տեսակներից մեկն է: Ակնհայտ է կառուցվածքային խճողումների անմիջական ազդեցությունը ՀՍ-երի թեստավորման արդյունավետության վրա: Հետազոտենք այդ ազդեցության պատճառները:

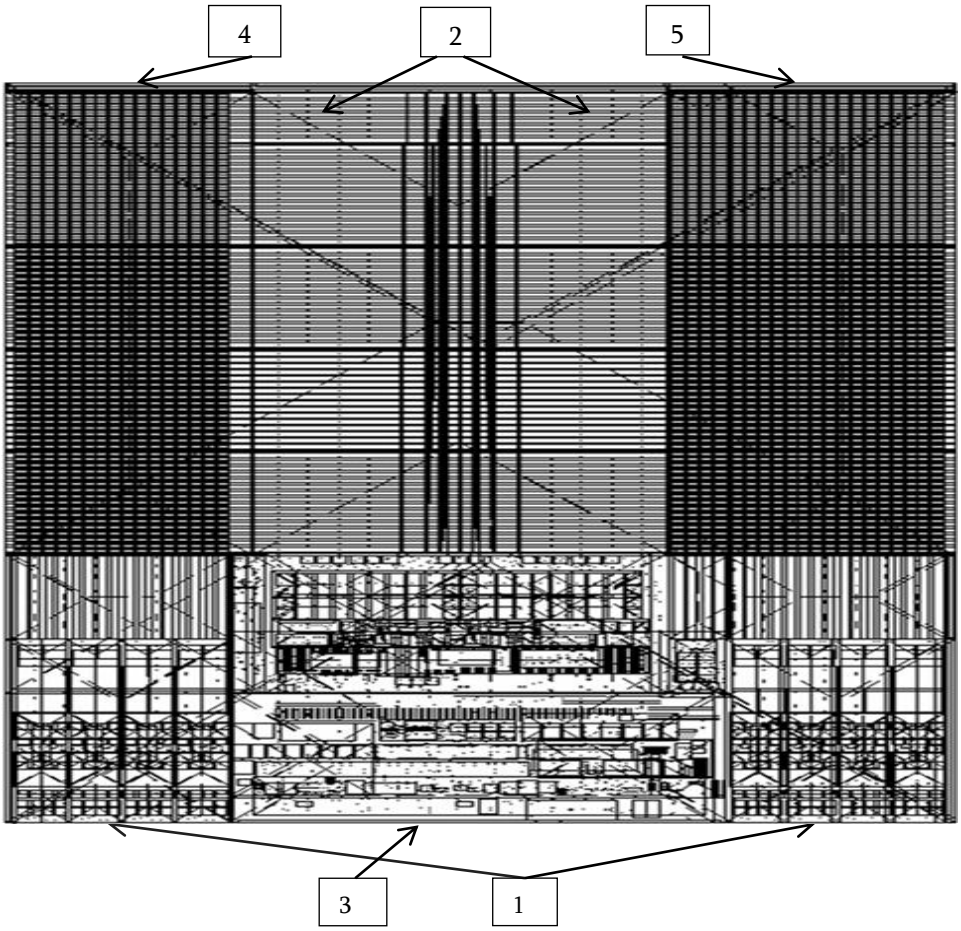
2.1 Հիշողության կառուցվածքային խճողումների տարրերը

ՍՊԴՀ նմուշը պատվիրատուին տրվում է արտադրության մեջ օգտագործվող GDSII ձևաչափով ներկայացված ֆայլով: Այն պարունակում է ավարտուն և ամբողջական տեղեկություն ՆՀՍ-ի մասին: GDSII տեսակի գրաֆիկական ձևաչափը ծավալուն է, և այդ ֆայլի մշակումը այլ ԱՀ ծրագրային միջոցների կողմից անչափ անհարմար է և ժամանակատար: ՍՊԴՀ սարքերի նմուշի GDSII ֆայլը նախագծելիս ճարտարագետը կատարում է հիշողության հանգույցների լավարկումը՝ ապահովելով թիթեղի օգտագործվող մակերևույթի մինիմալացումը և բջիջների տոպոլոգիայի օպտիմալ տեղադրումը: Որպես այդ լավարկման աշխատանքների հետևանք, հիշողության նմուշում ձևավորվում են կառուցվածքային խճողումները, որոնք, ինչպես ցույց կտանք հետագայում, մեծ ազդեցություն ունեն նմուշի թեստավորման արդյունավետության վրա:

<u>Սահմանում.</u> ՀՍ-ի նմուշը բաղկացած է կառուցվածքային երկու մասերից՝ հիշողության բջիջները պարունակող զանգվածից և հիշողության բջիջներ չպարունակող մասից՝ գոտիներից:

Հիշողության գոտիները, հիշողության GDSII ֆայլում այն տարածքներն են, որտեղ իրականացված են հիշողության ֆունկցիոնալ այլ հանգույցների՝ օրինակ հասցեների

ապակոդավորման սխեման, զգայուն ուժեղացուցիչների սխեման, սնուցման լարումների միացման և փոխանջատման սխեման և այլ հանգուցների , տոպոլոգիան:



Նկար 1. 11. Հիշողության սարքի նմուշի՝ GDSII ձևաչափի օրինակը

1. Մուտքային/Ելքային բջիջներ
2. Տողերի ապակոդավորման սխեմա
3. Սյուների ապակոդավորման սխեմա
4. Հիշողության բջիջներ՝ խմբավորված ՀՍ-ի աջ մասում
5. Հիշողության բջիջներ՝ խմբավորված ՀՍ-ի ձախ մասում

Նկար 1.11-ում տրված է հիշողության սարքի GDSII ձևաչափի օրինակը: Նկար 1.11-ում 4 և 5 միասնաբար կազմում են հիշողության բջիջների զանգվածը, իսկ նմուշի մնացած մասերը (1-3 մասերը)՝ համաձայն մեր սահմանմանը, կառուցվածքային գոտիներն են: Հիշողության մոդելը նկարագրելիս գոտիները ունեն միայն տարածքային և տեղայնացման նշանակություն, և չեն պարունակում որևէ տեղեկություն իրենց ֆունկցիոնալության մասին:

Կիրառական ծրագրային միջոցների հետ աշխատանքի ժամանակը էապես պակասեցնելու նպատակով առաջակվեց ստեղծել և կիրառել հիշողության նմուշի պարզեցված կառուցվածքային մոդելը: Հիշողության կառուցվածքային մոդելը բաղկացած է երկու հիմնական մասերից՝ հիշողության գոտիների բաշխումը նկարագրող մասից և հիշողության կառուցվածքային խճողումները նկարագրող մասից: Հիշողության նմուշի մոդելը ստեղծվում և պահվում է տեքստային ձևաչափով, իսկ հիշողության կոմպիլյատորի դեպքում՝ TCL ծրագրային լեզվով գրված ձևաչափով: Հիշողության կառուցվածքային մոդելը համապատասխան ստուգումից հետո [12] օգտագործվում է այլ կիրառական ծրագրերում [8,13,14]:

Իհարկե, կան բարդ և անչափ ծավալուն թեստավորման ալգորիթմներ, որոնց արդյունավետության համար էական չէ նմուշի կառուցվածքային խճողումների մասին տեղեկության առկայությունը: Բայց այդպիսի ալգորիթմների աշխատանքի տևողություն ժամանակը և իրականացման համար անհրաժեշտ՝ անընդունելի մեծ տարածքը թիթեղի վրա, անհնար են դարձնում դրանց օգտագործումը ներդրված սարքերում:

1.2.1 ՀՍ-ի խճողումների պատճառները և ազդեցությունը հիշողության թեստավորման վրա

Դիտարկենք ՀՍ-ի հասցեների և տվյալների խճողումները, դրանց հայտնվելու պատճառները և ազդեցությունը հիշողության թեստավորման վրա: Հիշողության սարքի կառուցվածքային խճողումների հիմնական պատճառներն են.

1. Տվյալների տարածքային դասավորումը նմուշում, տվյալների դասավորման խճողումը, նմուշի երկրաչափական կառուցվածքի օպտիմալացումը,
2. Հասցեների՝ տողերի և սյուների ապակոդավորիչների տոպոլոգիայի լավարկումը,
3. Բջիջների՝ մեկը մյուսին միացնող կապերի և «N» և «P» տիպի լիգերացման գոտիների համատեղ օգտագործումը՝ կիսումը,
4. Նմուշի աշխատանքի արագության և կայունության լավարկումը բիթային գծերի ոլորապտումների միջոցով,

5. Հասցեների կամ տվյալների գծերի տեղափոխումը՝ օգտագործելով Մուտք/Ելք միացումների համատեղելիությունը
6. Արտադրության օգտակար ելքի օպտիմալացումը պահեստային տողերի և սյուների միջոցով:

Տվյալների տարածքային դասավորումը՝տվյալների խճողում

Նկար 1.12-ում ներկայացված են հիշողության զանգվածում տվյալների բաշխման երկու հնարավոր տարբերակների օրինակները: Դիտարկենք մի տարբերակ, երբ հիշողության բջիջների զանգվածի տողը կազմված է 16 սյուներից (col0 – col15), իսկ յուրաքանչյուր հիշողության բառը բաղկացած է 4 բիթերից՝ bit0-bit3: Այս դեպքում հիշողության ֆիզիկական մեկ տողը կարող է պարունակել 4 բառ: Նկար 1.12-ի ա.-ում բերվում է մի տարբերակ, երբ մեկ տողում տվյալները՝ բառերի բիթերը, խմբավորված են հաջորդաբար բառ առ բառ՝ word0, word1, word2, word3: Այս դեպքում, հիշողության բջիջների զանգվածում տվյալների խճողում չի կատարվում:

row1										
row0	word0 **bit0**	word0 **bit1**	word0 **bit2**	word0 **bit3**	word1 **bit0**	word1 **bit1**	word1 **bit2**	word1 **bit3**		word3 **bit3**
	col0	col1	col2	col3	col4	col5	col6	col7	-	col15

ա. Բառերի հարևան բաշխումը (տվյալները խճողված չեն)

row1										
row0	word0 **bit0**	word1 **bit0**	word2 **bit0**	word3 **bit0**	word0 **bit1**	word1 **bit1**	word2 **bit1**	word3 **bit1**		word3 **bit3**
	col0	col1	col2	col3	col4	col5	col6	col7	-	col15

բ. Բիթերի հարևան բաշխումը (տվյալները խճողված են)

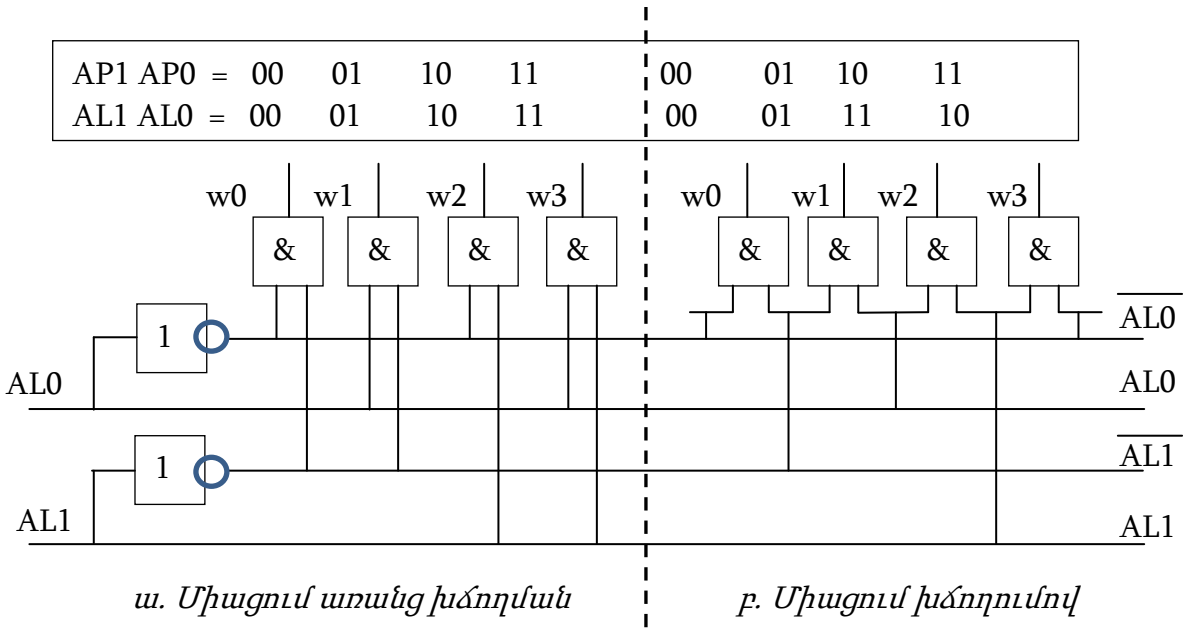
Նկար 1.12. ՆՀՄ-ի տվյալների խճողումը

Նկ. 1.12 բ.-ում պատկերված է մի տարբերակ, երբ բառերը տողում բաշխված են ըստ բիթերի հարևանության: Հարկ է նշել, որ ա. տարբերակում նկարագրված բաշխման տոպոլոգիան կառուցելիս հանդիպում ենք՝ Մուտք/Ելք հանգույցների միացումների

իրականացման մեծ բարդությունների առաջ: Այդ պատճառով ա. տարբերակը ՆՀՍ-երում չի կիրառվում: Կիրառվում է բ. տարբերակը, ունենալով տվյալների խճողում, կարողանում ենք օպտիմալացնել Մուտք/Ելք հանգույցների դասավորումը տոպոլոգիայում՝ մասնավորապես բաշխել դրանք հաջորդաբար աճման կամ էլ նվազեցման կարգով: ՆՀՍ-ի բառերի բաշխումը լինում է երկու տեսակի՝ ներդրված (interleaved) և հարևան բառերով (adjacent): Այս աշխատանքում դիտարկվում են միայն ներդրված բառային կառուցվածք ունեցող հիշողության սխեմաները:

Հասցեների ապակոդավորիչի լավարկումը

Հիշողության սարքերի տողերի և սյուների ապակոդավորիչների տոպոլոգիան նախագծելիս ևս հաճախ պահանջում է օգտագործել խճողումը նմուշի տոպոլոգիայի լավարկման ընթացքում: Նկար 1.13 – ում ներկայացված է երկուսը չորսի ապակոդավորիչի սխեման, որը իրականացնում է հետևյալ ձևափոխման ֆունկցիան՝ AP0=AL0 xor AL1, AP1= AL1:



Նկար 1.13. Հասցեների ապակոդավորիչի միացումների լավարկումը

Նկար 1.13-ի ա.-ում տրված է տարբերակ, երբ ապակոդավորիչը լավարկված չէ, և այս դեպքի համար խճողում չի առաջանում: Նկար 13-ի բ.-ում նույն ապակոդավորիչի համար կատարվել է միացնող մուտքային գծերի լավարկում՝ երկար միացումներ ունեցող կապերի քանակը կրճատելու նպատակով: Այս դեպքում, տոպոլոգիայի

ֆիզիկական կառուցվածքում առաջանում է, միացումների խճողումը: ա. և բ. տարբերակների խճողման պարզաբանումը ներկայացված է աղյուսակ 1.3-ում:

Աղյուսակ 1.3

Նկար 1.13-ի հասցեների միացումների խճողման պարզաբանումը

AL1	AL0	Տարբերակ (ա)	Տարբերակ (բ)
0	0	w0	w0
0	1	w1	w1
1	0	w2	w3 (խճողում)
1	1	w3	w2 (խճողում)

<u>Հասցեների ապակոդավորիչի միացումների լավարկում</u>

Դիտարկենք հասցեների խճողման ազդեցության մի քանի օրինակ: Հասցեների խճողումը զգալի ազդեցություն ունի թեստավորման ժամանակ օգտագործվող, BP-ի (Տես՝ աղ. 1.2) վրա: Ակնհայտ է որ հասցեների խճողումը չի ազդում «Solid» և «Solid bar» BP-ի ժամանակ, քանի որ հիշողության զանգվածի բոլոր բջիջներում գրվում է միայն մեկ տեսակի տվյալ «0» կամ «1»: *Տողերի* հասցեների խճողումը ազդում է Row Stripe, Double Row Stripe, Chacker Board, Bitline Chacker Board թեստային օրինակների դեպքերում և չի ազդում Column Stripe, Double Column Stripe BP-ների ժամանակ: Նկար 1.14-ում տրված է տողերի հասցեների խճողման ազդեցությունը Rows Stripe BP-ի վրա: Նկար

row3	0	0	0	0
row2	1	1	1	1
row1	0	0	0	0
row0	1	1	1	1
	col0	col1	col2	col3

ա. Տողերի խճողում չկա {0,1,2,3}

row2	1	1	1	1
row3	0	0	0	0
row1	0	0	0	0
row0	1	1	1	1
	col0	col1	col2	col3

բ. Տողերի խճողում {0,1,3,2}

Նկար 1.14. Rows Stripe BP-ի վրա տողերի խճողման ազդեցությունը

1.14 ա.-ում տողերի ֆիզիկական դասավորվածությունը կանոնավոր է {0,1,2,3}, իսկ բ.-ում երկրորդ և երրորդ տողերի ֆիզիկական դիրքերը փոխանակված են {0,1,3,2}: Տողերի դիրքերի այդ փոփոխումը բացասական ազդեցություն ունի հիշողության թեստավորման

վրա, քանի որ դրա արդյունքում ի հայտ է գալիս մի իրավիճակ, երբ չնայած նրան, որ այս թեստավորման ընթացքում օգտագործվում է տրամաբանական Rows Stripe BP-ը (Տես՝ նկ. 1.14 ա.) հիշողության զանգվածի ֆիզիկական մակարդակում ստացվում է մեկերի և զրոների բաշխման մեկ այլ պատկեր (Տես՝ նկ. 1.14 բ.): Արդյունքում ստանում ենք մի մասնավոր դեպք, երբ ֆիզիկական մակարդակում՝ որոշակի հարևան տողերում հիշողության բջիջները տողից տող անցնելիս ունեն միանման արժեք: Այս իրավիճակը զգալիորեն նվազեցնում է անսարքությունները հայտնաբերելու հավանականությունը խճողում ունեցող տողերի տարածքում, քանի որ անսարքությունը իրեն հստակորեն բացահայտում է, երբ հիշողության հարևան բջիջները ունենում են հակադարձ արժեքներ:

row3	1	0	1	0
row2	1	0	1	0
row1	1	0	1	0
row0	1	0	1	0
	col0	col1	col2	col3

ա. Սյուների խճողում չկա {0,1,2,3}

row3	0	1	1	0
row2	0	1	1	0
row1	0	1	1	0
row0	0	1	1	0
	col1	col0	col2	col3

բ. Սյուների խճողում {1,0,3,2}

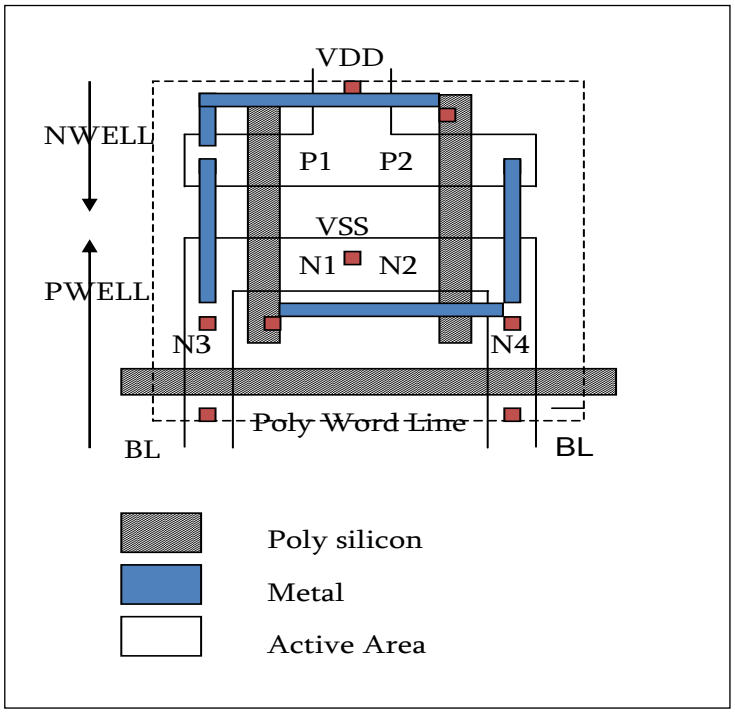
Նկար 1.15. Columns Stripe BP-ի վրա սյուների խճողման ազդեցությունը

Նկար 1.15-ում տրված է Սյուների խճողման ազդեցությունը Columns Stripe BP-ի վրա: Այս դեպքի մեկնաբանումը համընկնում է նկ. 1.14 դեպքին, միայն մեկ տարբերությամբ, որ ՀՍ-ի խճողումը կատարված է սյուներում, և այն ազդում է սյուների կողմնորոշում ունեցող BP-ի վրա: Սյուների հասցեների խճողումը ազդում է Column Stripe, Double Column Stripe, Checker Board, Bitline Checker Board թեստային ալգորիթմում թեստային օրինակների դեպքերում և չի ազդում Row Stripe, Double Row Stripe BP-ի վրա:

Զանգվածի բջիջներում բջջային միացումների և լիցերացման գոտիների համատեղ օգտագործումը

Հիշողության բջջի էլեկտրական սխեման պատկերված է նկ. 1.8-ում: Այդ էլեկտրական սխեմայի տոպոլոգիան ունի իրականացման բազմազան տարբերակներ, որոնցում ճարտարագետը իրականացնում է յուրաքանչյուր տեխնոլոգիային բնորոշ լավարկման աշխատանքներ: Մի դեպքում անհրաժեշտ է լինում փոքրացնել

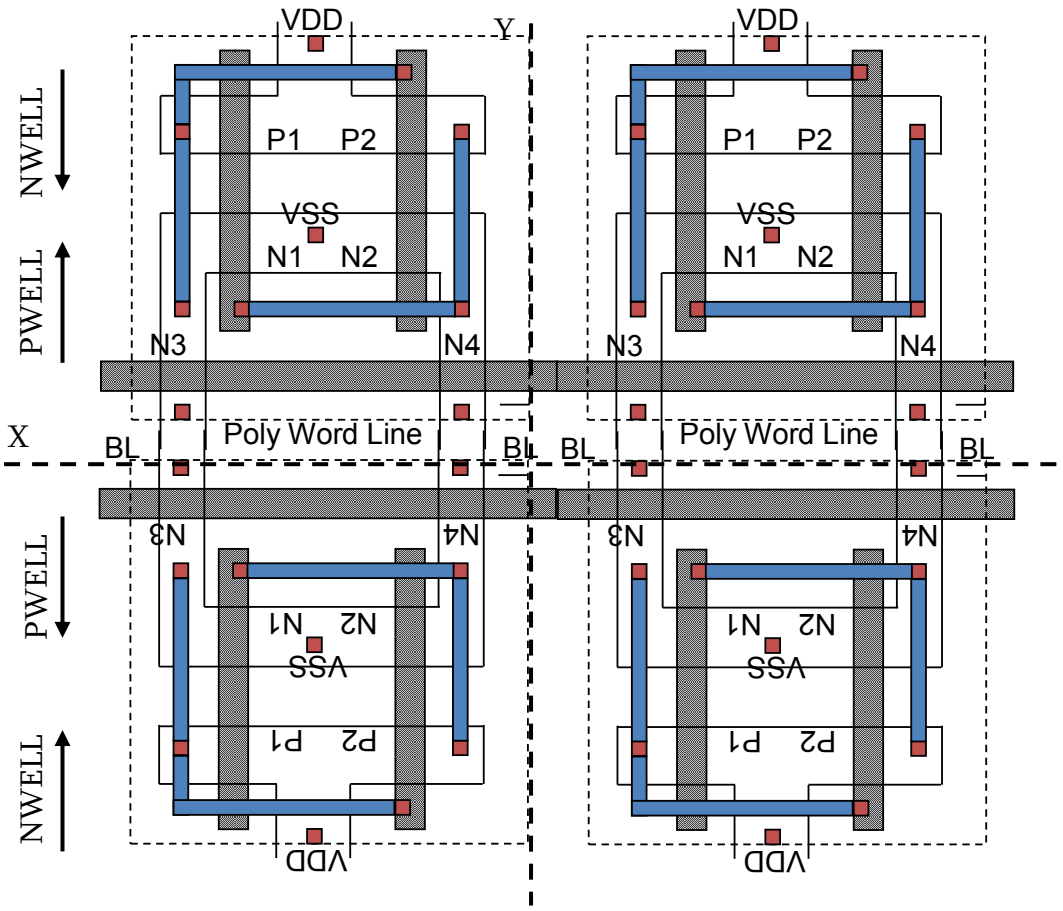
հիշողության բջջի չափսերը, մեկ այլ դեպքում՝ ապահովել բարձր արագործությունը, իսկ մեկ այլ դեպքում ապահովել տեխնոլոգիայի մեկ այլ պահանջ:



Նկար 1.16. Հիշողության բջջի տոպոլոգիայի օրինակը

Նկար 1.16-ում պատկերված է հիշողության բջջի տոպոլոգիայի օրինակը: Տոպոլոգիայի այդ օրինակում NWELL-ը և PWELL-ը համապատասխանաբար N և P լեգիրացման գոտիների բաշխումներն են հիշողության բջջի տոպոլոգիայում: ՆՀՍ-ի տոպոլոգիայում այդ լեգիրացման գոտիների համատեղ օգտագործումը տարբեր բջիջների համար առաջացնում է բջիջների բիթային գծերի տրամաբանական և ֆիզիկական բաշխման խաճողում՝ բիթային գծերի տարածական կանոնավոր կողմնորոշվածության խախտումը «X» առանցքում :

Նկար 1.17-ում ներկայացված է հիշողության զանգվածից երկու տող և երկու սյուն ունեցող մի հատված: Նկարից երևում է, որ վերևի և ներքևի տողերում գտնվող բջիջները հայելաձև շուռ են տված հորիզոնական «X» առանցքի նկատմամբ, իսկ աջ և ձախ սյուների բջիջները շուռ են տված ուղղահայաց «Y» առանցքում: Բջիջների այս տեղաբաշխումը պատճառ է դառնում բիթային գծերի (ուղղահայաց) և սնուցման գծերի (հորիզոնական) խաճողումների:

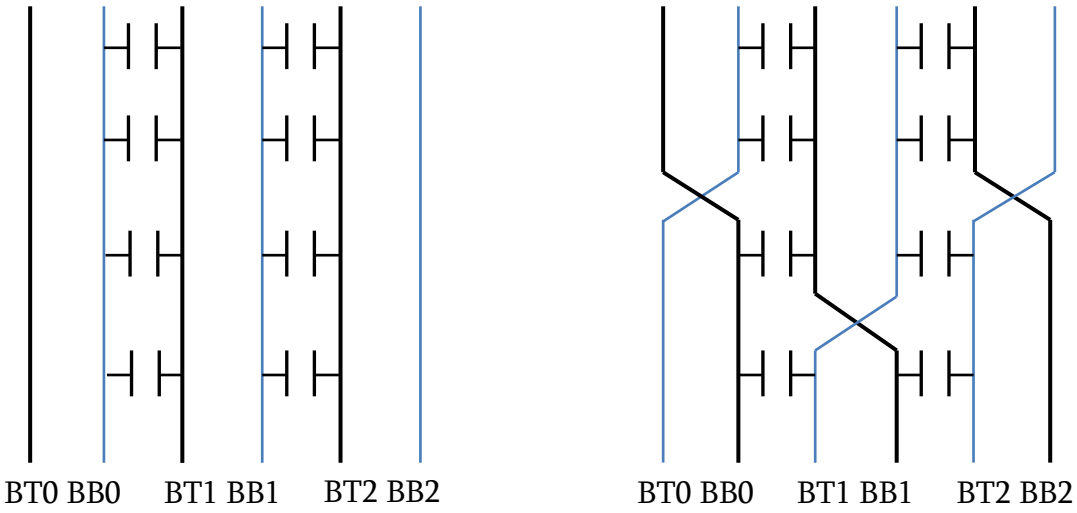


Նկար 1.17. Հիշողության զանգվածում լեգիրացման գոտիների և կապերի բաշխման օրինակը

<u>Հիշողության զանգվածի բիթային գծերի խճողման պատճառները</u>

Քանի որ հիշողության զանգվածի բիթային գծերի բաշխման մասին տեղեկությունը կարևոր դեր է կատարում թեստավորման արդյունավետության վրա, ապա նպատակահարմար է ավելի մանրամասն դիտարկել բիթային գծերի խճողման առաջացման պատճառները: Բիթային գծերի խճողման առաջացման պատճառ է հանդիսանում նմուշի աշխատանքի արագության և կայունության լավարկումը բիթային գծերի դասավորվածության կարգավորման և գծերի ոլորապտույտների կիրառման միջոցով [18], [25]՝ դրանով նվազեցնելով բիթային գծերում միջ-գծային ունակությունը: Այդ ունակությունը մեծ արգելք է հանդիսանում հիշողության աշխատանքի համար, նամանավանդ շատ տողեր, այսինքն՝ երկար բիթային գծեր ունեցող, հիշողության նմուշների դեպքերում: Բիթային գծերի միջև մեծ ունակությունը միաժամանակ

նվազեցնում է հիշողության աշխատանքային հաճախականությունը և հիշողության աշխատանքի հուսալիությունը (Տես՝ նկ. 1.18): Ունակության այդ բացասական ազդեցությունը պակասեցնելու նպատակով հիշողության սարքերում կիրառվում են բիթային գծերի ոլորապտումները:



ա. Բիթային գծերը առանց ոլորապտումների

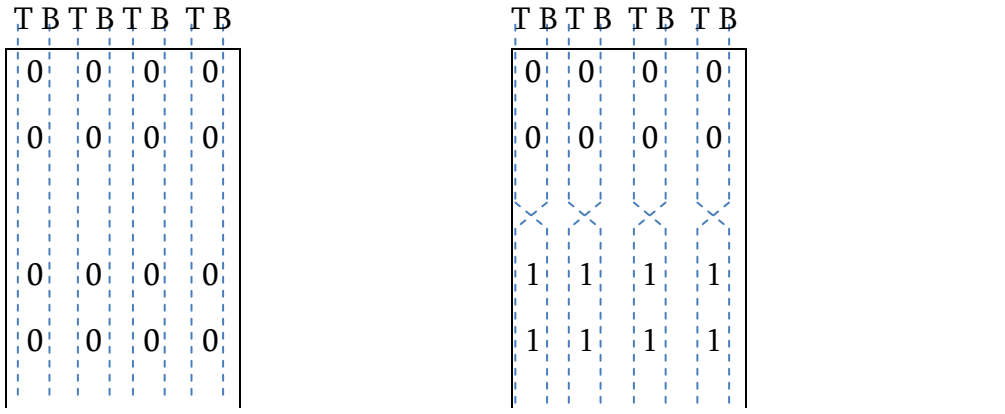
բ. Բիթային գծերը ոլորապտումով

Նկար 1.18. Հիշողության զանգվածում բիթային գծերի բաշխումը

Նկար 1.16 ա.-ից երևում է, որ BB0 և BT1 գծերի միջև (նմանապես BB1 և BT2 միջև) ընկած գումարային ունակությունը ավելանում է գծերի երկարությանը զուգահեռ: Նկար 1.18 բ.-ից երևում է, որ գծերի ոլորապտման շնորհիվ գծերի երկարությամբ բաշխված ունակությունը վերաբաշխվում է, որի հետևանքով հարևան բիթային գծերի գումարային ունակությունը նվազում է: Ակնհայտ է, որ ոլորապտման արդյունքում ստացված գծերի ունակության գումարային արժեքը կախված է հիշողության զանգվածում կատարված, գծերի ոլորապտման ֆիզիկական դիրքից: Այդ ոլորապտումը բաժանում է հիշողության զանգվածը հատվածների, ընդ որում, այդ հատվածներում տվյալների ֆիզիկական և տրամաբանական տվյալները ստանում են հակադիր արժեքներ:

Հարկ է նշել, որ հիշողության ողջ զանգվածում հիշողության բջիջների բիթային գծերի բաշխման մասին ինֆորմացիան կարևոր դեր է կատարում թեստավորման ալգորիթմում կիրառվող (աշխատանքի ընթացքում) տվյալների՝ BP-ի արդյունավետ ձևով օգտագործման համար [16]:

Նկար 1.19-ում, որպես օրինակ, ներկայացված է մի դեպք, երբ հիշողության զանգվածի բոլոր բջիջները տրամաբանական «0»-ով լցնելուց հետո հիշողության զանգվածում գրանցված ֆիզիկական արժեքը տարբեր հատվածներում հակադիր է (Տես՝ նկ. 1.19 բ.): Հիշողության բջիջներում գրանցված ֆիզիկական և տրամաբանական արժեքների տարբերությունը պատճառ է հանդիսանում բիթային գծերի խճողման ազդեցության բջիջների պարունակության վրա:



ա. Զանգվածում բջիջների ֆիզիկական արժեքը առանց բիթային գծերի ոլորապտումների

բ. Զանգվածում բջիջների ֆիզիկական արժեքը բիթային գծերի ոլորապտումների դեպքում

Նկար 1.19. Բիթային գծերի խճողման ազդեցությունը հիշողության բջիջներում գրանցված ֆիզիկական արժեքի վրա

Բազմիցս նշել ենք, որ անսարքությունը հայտնաբերելու համար անհրաժեշտ է ապահովել հիշողության տոպոլոգիայում ֆիզիկապես հարևան բջիջների բիթային գծերի վրա հակադիր արժեք ունեցող լարումներ (տրամաբանական {0,1} կամ {1,0})՝ միայն այդ միջոցով կարելի է հայտնաբերել անսարքությունը բջիջում: Հակառակ դեպքում (տրամաբանական {1,1}, {0,0} արժեքների դեպքում), թեստավորման ընթացքում, անսարքությունը կմնա չհայտնաբերված: Թեստավորման ընթացքում «ճիշտ» թեստային տվյալների օգտագործումը մոտ 30% բարձրացնում է թեստավորման արդյունավետությունը [16], [27] միաժամանակ պակասեցնելով այդ ալգորիթմի իրականացման համար անհրաժեշտ սարքավորումը, որն իր հերթին շատ կարևոր է ներդրված սարքերի մակերևույթի լավարկման դեպքում: Մի քանի օրինակների միջոցով ցույց տանք հիշողության բջիջների բիթային գծերի բաշխման մասին տվյալների ազդեցությունը թեստավորման վրա: Նկար 1.20-ում տրված է մի դեպքի

նկարագրություն, երբ թեստային ալգորիթմում օգտագործելով լավագույնը համարվող տրամաբանական «Շախմատաձև» Checkerboard BP-ը (Տես՝ նկ.1.20 ա.), կարող է հիշողության զանգվածում չհայտնաբերել անսարքությունը: Նկար 1.20 բ.-ում ներկայացված է նույն BP-ի ֆիզիկական պատկերը, որտեղ յուրաքանչյուր բջիջի

row2	0	1	0
row1	1	0	1
row0	0	1	0
	col0	col1	col2

row2	T B 0 0 1	T B 1 1 0	T B 0 0 1
row1	T B 1 1 0	T B 0 0 1	T B 1 1 0
row0	T B 0 0 1	T B 1 1 0	T B 0 0 1
	col0	col1	col2

T B T B T B

ա. Տրամաբանական BP *բ. Ֆիզիկական BP*

Նկար 1.20. Շախմատաձև BP-ի տրամաբանական և նրա համապատասխան ֆիզիկական ձևերը

կենտրոնում տրված է բջիջի տրամաբանական արժեքը, բջիջի վերևի անկյունամասերում բջիջի բիթային գծերի խաճողումը (T, B), իսկ ներքևի անկյունամասերում բիթային գծերի արժեքները (0, 1): Զանգվածի բիթային գծերի բաշխման յուրահատկության պատճառով {T B, T B, T B} հարևան բիթային գծերի արժեքները հակադիր չեն, չնայած որ տրամաբանական արժեքները հակադիր են: Պարզ է, որ այս բիթային գծերի խաճողման դեպքում հիշողության թեստավորման ալգորիթմը օգտագործելով տրամաբանական լավագույնը թվացող շախմատաձև BP-ը իհայտ կգա մի իրավիճակ, երբ անսարքությունը չի ակտիվանա և դրա արդյունքում թերությունը չի հայտնաբերվի թեստավորման ընթացքում:

Նկար 1.21- դեպքին հակառակը, ոչ օպտիմալ թվացող «Տողերի գոտիներ» Row Stripe տրամաբանական BP-ը կարող է բացահայտել (սինթեզել) անսարքությունը նմուշի զանգվածի բիթային գծերի խաճողման շնորհիվ, որի պատճառով բջիջում ֆիզիկական մակարդակով հարևան բիթային գծերի արժեքները հակադարձ են՝ թեստավորման համար առավելագույն բարենպաստ վիճակում (Տես՝ նկ. 1.21 բ.):

row2	0	0	0
row1	1	1	1
row0	0	0	0
	col0	col1	col2

ա. Տրամաբանական BP

row2	T B 0 0 1	T B 0 0 1	T B 0 0 1
row1	T B 1 1 0	T B 1 1 0	T B 1 1 0
row0	T B 0 0 1	T B 0 0 1	T B 0 0 1
	col0	col1	col2
	T B	T B	T B

բ. Ֆիզիկական BP

Նկար 1.21. Շախմատաձև՝ ֆիզիկական BP-ը

Քանի որ բիթային գծերի ոլորապտումները փոխում են բջիջների բիթերի բաշխումը հիշողության զանգվածում, ակնհայտ է դառնում, որ բիթային գծերի ոլորապտումները ևս էական ազդեցություն ունեն թեստավորման ալգորիթմի արդյունավետության վրա: Միանշանակորեն կարելի պնդել, որ հիշողության թեստավորման համար լավագույն թեստային BP է հանդիսանում «*Բիթային Շախմատաձև*» (TCB) *BP-ը*: Այն ապահովում է թեստավորման համար արժեքների լավագույն բաշխումը ֆիզիկական մակարդակի վրա՝ հիշողության զանգվածում բջիջների բիթային գծերի մակարդակում: Նկար 1.21 բ.-ի բջիջների բիթային գծերի արժեքների պատկերը համապատասխանում է «Բիթային Շախմատաձև» BP-ին: TCB BP-նի շնորհիվ հիշողության զանգվածում յուրաքանչյուր բջիջը, բջջի բիթային գծերը միաժամանակ ստանում են հակադարձ արժեքները և՛ հարևան տողերում, և՛ հարևան սյուներում (Տես՝ նկ. 1.21 բ.):

<u>Մուտք/Ելք հանգույցների խճողումը</u>

Մուտք/Ելք միացումների՝ հասցեական բիթերի, տվյալների Մուտքերի և Ելքերի միացումների, ՀՍ-ի ղեկավարման ազդանշանների համատեղելիության ապահովումը արտադրության կողմից ներկայացված Մուտք/Ելք «բարձիկների» տեղաբաշխման նկատմամբ, հիշողության նախագծման կարևոր խնդիրներից է: Որպես Մուտք/Ելք-րի տրամաբանական ներկայացում, հիմնականում՝ հիշողության սարքերում օգտագործվում է ելքերի կանոնավոր բաշխումը: Օրինակ՝ մուտքային տվյալները {D0, D1, D2…Dn}, կամ ելքերը {Q0, Q1, Q2…Qn} ներկայացված են կանոնավոր բաշխումով

(0,1,...,n) (Տես՝ նկ. 1.5բ., նկ.1.7), մինչդեռ՝ ֆիզիկական բաշխումը միկրոսխեմայի թիթեղի վրա (GDSII ֆայլում, տես՝ նկ. 1.11), կարող է էականորեն տարբերվել՝ օրինակ լինել հակառակ ուղղությամբ {n, (n-1)… 2, 1, 0}, կամ ել հակադարձ հայելային {n, (n-1)… (n/2), 0, 1, … (n/2-1)}: ՊԴՀՍ-ի պատվիրատուի համար Մուտք/Ելք միացումների խճողման վերաբերյալ տեղեկատվությունը նույնպես կարևոր մաս է կազմում ՀՍ-ի մասին տեղեկատվության բազմության մեջ և պետք է արտացոլվի ՆՀՍ-ի կառուցվածքային մոդելում:

1.2.2 ՄՊԴՀ կառուցվածքային խճողումների ներկայացման ձևերը

Վերոհիշյալ օրինակներից պարզ է դառնում, որ հիշողության նմուշի խճողման մասին ինֆորմացիան կարևոր դեր է խաղում այդ նմուշի թեստավորման ԱՀ-ի արդյունավետության բարձրացման խնդրում: Ինչպես ցույց է տրված [16] և [25]-ում, հիշողության խճողման կիրառումը ԱՀ-ի թեստավորման ալգորիթմում կարող է մինչև 30% բարձրացնել այդ ալգորիթմի արդյունավետությունը: Բացի այդ կան մի շարք ԱՀ-ի կիրառություններ, որոնք չեն կարող գործածվել առանց նմուշի խճողման մասին ինֆորմացիայի: Այդ կիրառություններից են՝

ա) ՀՍ-ի նմուշում հիշողության բջիջների ֆիզիկական դիրքերի կոորդինատների հաշվարկումը: Բջիջի {X,Y} կոորդինատները անհրաժեշտ է լինում որոշել մի շարք կիրառությունների աշխատանքի ընթացքում: Օրինակ՝ ՀՍ-ի թեստավորման ընթացքում թեստային ալգորիթմը հայտնաբերում է անսարքությունը և անսարքության պատճառը հայտնաբերելու և հետազոտելու նպատակով անհրաժեշտ է լինում պարզել անսարք բջջի ֆիզիկական դիրքը (կոորդինատները) հիշողության նմուշի թիթեղում: Այս դեպքում թեստային ալգորիթմը վերադարձնում է անսարք բջջի տրամաբանական հասցեն և «Ելքի» տրամաբանական համարը անսարք բառում, պետք է մուտքային այս երկու տվյալների հիման վրա հաշվարկել անսարք բջջի ֆիզիկական դիրքի կոորդինատները: Հասկանալի է, որ այս դեպքում պետք է հաշվի առնել նմուշի տողերի, սյուների, Մուտք/Ելքեր, գոտիների և պահեստային սյուների/տողերի խճողումները:

բ) Նմուշների օգտակար ելքի հաշվարկի ժամանակ մուտքային տվյալներից են նմուշի ողջ մակերեսը և հիշողության զանգվածի մակերեսը: Այս տվյալները հաշվարկվում են գոտիների բաշխման և հիշողության բջջի չափերի և քանակի մասին տվյալների հիման վրա:

Բոլոր նշված տվյալների մասին ավարտուն և հավաստի ինֆորմացիան գտնվում է նմուշի GDSII գրաֆիկական ֆայլում (Տես՝ նկ. 1.11), որի հետ աշխատելը, ինչպես արդեն նշել էինք, շատ ժամանակատար է և պահանջում է մեծ քանակի հաշվողական ռեսուսներ: Բազմաքանակ հիշողության կոմպիլյատորներ և հիշողության նմուշներ հետազոտելուց հետո հասկանալի դարձավ, որ

1. անհրաժեշտ է հետազոտել, դասակարգել և նկարագրել հիշողության սարքերում խճողման բոլոր տեսակները և դրանց պատճառները,
2. մշակել նկարագրման մի լեզու, որը հնարավորություն կտա լիարժեք նկարագրել և կիրառական ծրագրերի համար հարմար ձևաչափով տրամադրել հիշողության սարքերի խճողման ինֆորմացիան կոմպիլյատորների և նաև նմուշների մակարդակով: Խնդիր էր դրվում ստեղծել ՀՍ-ի կառուցվածքային մոդելը և այն ներկայացնող լեզուն:

Հետազոտման աշխատանքների ընթացքում հիշողության կոմպիլյատորների միջոցով գեներացվել և հետազոտվել են հիշողության նմուշներ հարյուրավոր տարբեր չափսերի և տեխնոլոգիական պրոցեսների համար (250, 180, 130, 90, 65, 55, 45, 28, 16, 14 նանոմետր), զանազան տեսակի (արագագործ, բարձր խտության, ցածր հզորության), մեկ և ավելի մուտքանի ՊԴՀՍ-եր: Հիշողության նմուշների կառուցվածքների հետազոտման արդյունքում իրականացվել է հիշողության սարքում կառուցվածքային խճողումների տեսակների նկարագրումը և դասակարգումը: Աշխատանքի այս փուլի արդյունքները ներկայացված են արտոնագրում [11]: Ժամանակակից ՀՍ-ի նկարագրված կառուցվածքային խճողումների քանակը տասնինն է: Բայց խճողումների տեսակների այդ քանակը կարող է մեծանալ՝ ՀՍ-ի նախագծման ընթացքում, հիշողության սարքերում կառուցվածքային նոր լուծումների կիրառմանը զուգահեռ: Ընդհանուր դեպքում, հիշողությունը նկարագրող մոդելը բաղկացած է երկու մասերից՝ հիշողության վարքագծային և կառուցվածքային մասերից [11]: Այս աշխատանքի սահմաններում մենք կդիտարկենք միայն հիշողության մոդելի կառուցվածքային

մոդելի գեներացման հետ կապված խնդիրները: Կառուցվածքային մոդելը կազմող կառուցվածքային խճողման տիպերը և դրանց հատկությունները մանրամասնորեն ներկայացված են [11]-ում: Թվարկենք կառուցվածքային խճողումներից մի քանիսը՝

1. հասցեների գծերի խմբային խճողում,
2. հիշողության բանկերի ապակոդավորիչներ խճողում,
3. տողերի ապակոդավորիչների խճողում,
4. սյուների ապակոդավորիչների խճողում,
5. Մ/Ե հանգույցների բաշխման խճողում,
6. բազմապորտանի հիշողությունների մուտքերի խճողում,
7. Մ/Ե-ի **T** և **B** բիթային գծերի խճողում,
8. բիթային գծերի ոլորապտումների բաշխվածություն,
9. սյուների ոլորապտումների բաշխվածություն,
10. բառային գծերի ոլորապտումների բաշխվածություն,
11. տողերի ոլորապտումների բաշխվածություն,
12. տրամաբանական սխեմաների գոտիների բաշխվածություն,
13. բջիջների զանգվածում սնուցման և հոզակցման գծերի բաշխվածությունը բառային և բիթային գծերի միջև,
14. մետաղական շերտերի խճողում,
15. ավելցուկային տողերի և սյուների բաշխվածություն,
16. այսպես կոչված, «պարապ» տողերի և սյուների բաշխվածություն:

Նշենք, որ հիշողության մոդելի կառուցվածքային խճողումների տրոհումը տեսակների թույլ է տալիս արդյունավետ կերպով լուծել հիշողությունների թեստավորման, ախտորոշման և նորոգման խնդիրները՝ օգտագործելով միայն այդ պահին պահանջվող, որոշակի կառուցվածքային միավորների խումբը:

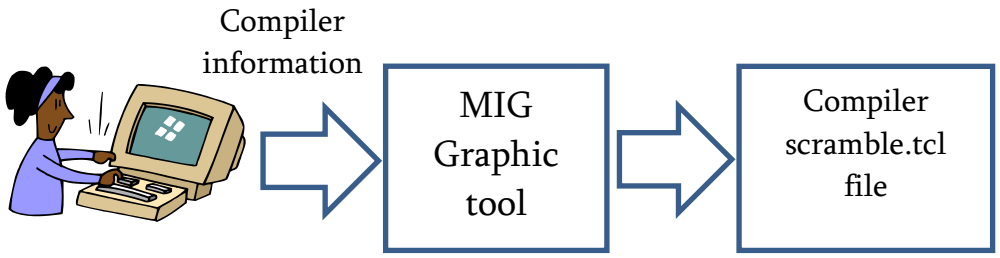
Մեկ այլ կարևոր խնդիր է հիշողության մոդելի ներկայացման ձևաչափի ընտրությունը: Մոդելի ձևաչափը պետք է բավարարի հետևյալ պահանջներին.

ա) մոդելի ձևաչափը պետք է հնարավորություն ընձեռի դյուրին կերպով նկարագրել ինչպես ներկայումս օգտագործվող, այնպես էլ մոտակա ապագայում հիշողություններում տեսականորեն կանխատեսելի կառուցվածքային խճողումների տեսակները,

բ) ձևաչափը պետք է հնարավորություն տա պարամետրացված ձևով նկարագրել խճողումների տեսակները,

գ) այդ ձևաչափով ներկայացված ինֆորմացիան պետք է հեշտությամբ մշակվի և ինտեգրացվի կիրառական այլ ծրագրային միջոցների կողմից, տարբեր ԱՀ-երում:

Հետազոտությունների արդյունքում՝ ՀՍ-ի *կոմպիլյատորներում* կառուցվածքային մոդելի ներկայացման ձևաչափ է ընտրվել սցենարների նկարագրման նպատակով ստեղծված TCL (Tool Command Language) լեզուն, իսկ հիշողության *նմուշների* կառուցվածքային մոդելի նկարագրության համար օգտագործել է տեքստային ձևաչափը [11]: TCL ծրագրավորման լեզուն մեծ տարածում ունի էլեկտրոնային սարքեր նախագծող ընկերություններում և լայնածավալ օգտագործվում է սարքերի թեստավորման բնագավառում՝ որպես թեստավորման սցենարներ նկարագրող գործիք [28]: Մինչ TCL լեզուն օգտագործելը հիշողության կոմպիլյատորներում խճողման մասին ինֆորմացիան ներկայացվում էր բազմաթիվ աղյուսակների տեսքով՝ DOC կամ TXT ֆայլերում: Ընդհանուր առմամբ, մուտքային տվյալների DOC ֆայլերում աղյուսակային ներկայացումը անհարմար ձևաչափ է հանդիսանում ավտոմատացման



Նկար 1.22. Հիշողության կոմպիլյատորի խճողման ֆայլի գեներացումը

համար: Բայց մյուս կողմից TCL լեզվով հիշողության խճողումների տեսակները նկարագրելը (ծրագրավորելը), այնուհետև այդ մոդելում ծրագրային սխալների վերացումը իր հերթին, բարդ գործընթաց է և նախագծող ճարտարագետից պահանջում է բարձր որակավորում և հատուկ գիտելիքների առկայություն: Հիշողության խճողման ինֆորմացիայի ստանդարտացման և ներդրման աշխատանքների սահմաններում՝ ճարտարագետի աշխատանքը հեշտացնելու նպատակով ստեղծվել է կառուցվածքային մոդելի գեներատոր ԱՀ MIG (անգլերեն՝ Memory Information Generater) ծրագրային գործիքը [17] (Տես՝ նկ. 1.22), որը հնարավորություն է տալիս գրաֆիկական

ինտերակտիվ միջավայրում նկարագրել հիշողության կոմպիլյատորի խճողման տեսակները և այնուհետև ավտոմատ ձևով գեներացնել նկարագրությանը համապատասխան TCL կոդը՝ scramble.tcl ֆայլը: MIG ծրագրային ԱՀ-ի նկարագրումը դուրս է այս աշխատանքի սահմաններից, բայց հարկ է նշել, որ այն (MIG ԱՀ-ը) հիշողության խճողման ինֆորմացիայի ստանդարտացման [11] և կիրառման առաջին արդյունքն է հանդիսանում:

MIG գործիքը էականորեն հեշտացնում է հիշողության կոմպիլյատորի խճողման նկարագրման գործընթացը և բացառում է սխալները *TCL կոդի* գեներացման ընթացքում: Իհարկե, կոմպիլյատորի խճողման TCL կոդի վերջնական տարբերակը՝ հիշողության սարքերի կառուցվածքային մոդելը, հետագայում անցնում է՝ բոլոր տեսակի մոդելների համար պարտադիր պայման հանդիսացող [12], [28], [29], համապատասխանության ստուգման փուլերով:

Դիտարկենք կառուցվածքային խճողումների տեսակներից մի քանիսը: Մոդելների ներկայացման ձևերն են. աղյուսակային, թվային արտահայտություններ, բանաձևերով, գրաֆիկների տեսքով: Մինչ խճողումների դասակարգումը և նկարագրման ստացման ավտոմատացումը, հիշողության կոմպիլյատորներում, հիշողության կառուցվածքի խճողման մասին ինֆորմացիան ներկայացվում էր տարբեր տեսակի աղյուսակների միջոցով:

<u>Հիշողության հասցեների խճողումը</u>

Հիշողության այն հասցեն, որի նկատմամբ կիրառվում է արժեքի ընթերցման կամ պահպանման գործողությունը, կանվանենք *տրամաբանական* հասցե: Հիշողության բջջի (բառի) *ֆիզիկական* հասցե (դիրքը) կանվանենք նրա ֆիզիկական դիրքը՝ ֆիզիկական տողի և սյան համարները, հիշողության բջիջների երկչափանի զանգվածում: Ավանդական «Մարշ» տեստային ալգորիթմները բաղկացած են «Մարշ» վերջավոր թվով հաջորդականություններից, որոնք կատարվում են հիշողության բջիջներից կազմված բառերի համար՝ հասցեների որոշակի աճման (0-ական հասցեից մինչև մաքսիմալ հասցե), նվազման (մաքսիմալ հասցեից մինչ 0-ական հասցե) հաջորդականությամբ: Անսարքությունների դասերը [25], որոնց համար նախագծվում էին այդ ալգորիթմները, դրսևորվում են հիշողության հասցեներին դիմելիս կանոնավոր հաջորդականությամբ: [24]-

ում ժամանակակից հիշողության սխեմաներում առաջացող անսարքությունների որոշակի դասերի հայտնաբերման համար առաջարկվում է ալգորիթմ, որը պետք է կիրառի արժեքի պահպանման և ընթերցման գործողությունները ֆիզիկապես զույգ (կենտ) համարներ ունեցող տողերի (սյուների) վրա գտնվող բջիջներում: Նման տիպի ալգորիթմների ճշգրիտ իրականացման համար անհրաժեշտ է իմանալ հիշողության տրամաբանական և ֆիզիկական հասցեների համապատասխանությունը: Հիշողության հասցեի խմբի (address bus) մուտքերին միացած ազդանշանները տրամաբանորեն բաժանվում են երկու մասի. մի մասը մուտք են հանդիսանում բջիջների երկչափ զանգվածում *ֆիզիկական տողերի*, իսկ մյուս մասը՝ *ֆիզիկական սյան* ապակոդավորման սխեմաների համար: Տրամաբանական հասցեի մյուս մասը, որը մուտք է հանդիսանում ֆիզիկական սյան ապակոդավորող սխեմայի համար, կանվանենք տրամաբանական սյան հասցե: Հասցեների խճողումը նկարագրող ֆունկցիան կարելի է ներկայացնել հետևյալ տեսքով (12).

$$ADR(BKj, CMi, a_1, a_2, \ldots a_{max}) = F_{adr}\{N, C, P1, P2, \ldots, Pm\} \quad , \text{որտեղ} \qquad (12)$$

BKj – բանկերի քանակն է՝ հիշողության կոմպիլյատորում,

CMi - սյուների ապակոդավորիչի գործակիցներն են՝ հիշողության կոմպիլյատորում,

$a_1, a_2, \ldots a_{max}$ - հիշողության տրամաբանական հասցեներն են

P1,P2,…,Pm - հիշողության ապակոդավորիչի Ֆիզիկական հասցեներն են:

Աղյուսակ 1.4-ում տրված է հասցեների խճողումը ընդհանուր դեպքում աղյուսակային տեսքով ներկայացումը: Լրացնելով աղյուսակի դաշտերը հիշողության բանկերի (BK) և սյուների ապակոդավորման (CM) արժեքների համար կստանաք հիշողության հասցեների խճողումը պատկերող աղյուսակը:

Աղյուսակ 1.4

Հասցեների խճողումը ներկայացնող աղյուսակային ձևը

BK	-	-	-	-
CM	-	-	-	-
ADDR0	-	-	-	-

Աղյուսակ 1.4 (շարունակություն)

Հասցեների խճողումը ներկայացնող աղյուսակային ձևը

ADDR1	-	-	-	-
ADDR2	-	-	-	-
…	-	-	-	-
ADDRmax	-	-	-	-

Որպես օրինակ, աղյուսակ 1.6-ում տրված է հասցեների խճողման մի օրինակ, որտեղ մուտքային պարամետրները ունեն հետևյալ արժեքներ՝ BK {1,2}, CM {2,4} և ADDRmax=ADDR5: Աղյուսակում -YA0 և YA1՝ սյուների ապակոդավորիչի

Աղյուսակ 1. 5

Հասցեների խճողումը ներկայացնող աղյուսակի օրինակը

BK	1	1	2	2
CM	2	4	2	4
ADDR0	YA0	YA0	YA0	YA0
ADDR1	XA0	YA1	XA0	YA1
ADDR2	XA1	XA0	XA1	XA0
ADDR3	XA2	XA1	XA2	XA1
ADDR4	XC0	XA2	XBK0	XA2
ADDR5		XC0		XBK0

մուտքային հասցեներն են, իսկ XA0-XA2, XC0՝ տողերի ապակոդավորիչի մուտքերն են, և XBK0՝ բանկերի ապակոդավորիչի մուտքն է: Պետք է նշել, որ աղյուսակում բերված տվյալները երկկողմանի են՝ ներկայացնում են ինչպես տրամաբանականից դեպի ֆիզիկական, այնպես էլ ֆիզիկականից դեպի տրամաբանական անցումները: Հասցեների խճողումը ներկայացնող TCL կոդը իրենից ներկայացնում է պրոցեդուրա, որի աշխատանքի շնորհիվ վերադարձվում է մասնավոր դեպքի՝ հիշողության նմուշի, հասցեների խճողումը բնորոշող ազդանշանների անուններով ցուցակը, օրինակ {YA0 YA1 XA0 XA1 XA2 XBK0}:

```
proc address_scramble { } {
    global scramble_db
    …
}
```

Տողերի խճողումը

Հիշողության կոմպիլյատորի տողերի խճողումը նկարագրող ֆունկցիայի տեսքն է՝

$$Ri = F_{row}\{Pn\}, \quad \text{որտեղ} \qquad (13)$$

Ri – նմուշի տրամաբանական տողի հասցեն է և $i \in \{0,1,\ldots, max\}$,

Pn - նմուշի ֆիզիկական տողի հասցեն է և $n \in \{0,1,\ldots, max\}$,

Աղյուսակ 1.6-ում տրված է տողերի խճողման ներկայացման աղյուսակային ձևը:

Աղյուսակ 1.6

Տողերի խճողման աղյուսակային ներկայացման ձևը

Տրամաբանական համարը	Ֆիզիկական համարը
Row 0	-
Row 1	-
Row 2	-
…	-
…	-
Row Maximum	-

Այն իրենից ներկայացնում է տրամաբանական և ֆիզիկական տողերի համարների պարզագույն արտապատկերումը: Ինչպես արդեն նշել ենք, տողերի խճողման պատճառ են հանդիսանում տողերի ապակոդավորիչի կառուցվածքային յուրահատկությունները: Տողերի խճողման աղյուսակային ներկայացման ձևաչափը նույնպես երկկողմանի է (տրամաբանականից դեպի ֆիզիկական և հակառակը): Ի տարբերություն աղյուսակային ներկայացմանը, TCL տրվող պրոցեդուրան միակողմանի է և հետևաբար պետք է ունենալ տողերի խճողումը նկարագրող երկու իրարից անկախ՝ տրամաբանականից դեպի ֆիզիկական և ֆիզիկականից դեպի տրամաբանական, անցման պրոցեդուրաներ.

ա) Տողերի խճողման տրամաբանականից դեպի ֆիզիկական TCL (row_scramble_logical_to_physical) պրոցեդուրան վերադարձնում է հիշողության

նմուշի ֆիզիկական տողերը ներկայացնող համարների հաջորդականություն՝ ենթադրելով, որ տողերի տրամաբանական համարները կանոնավոր ձևով աճում են 0-ից մինչև առավելագույնը, իսկ ֆիզիկական «0» տողին համապատասխանում է հիշողության զանգվածի ֆիզիկական ներքնի մասի առաջին տողը օրինակ {0 1 2 3 **7 6 5 4** 8 9 10 11}:

```
proc row_scramble_logical_to_physical { log_row } {
    global scramble_db
    …
}
```

բ) Տողերի խճողման ֆիզիկականը դեպի տրամաբանականը ներկայացնող TCL (row_scramble_physical_to_logical) պրոցեդուրան ֆիզիկական տողի համարի հիման վրա գեներացվում է տրամաբանական տողի համարը:

```
proc row_scramble_physical_to_logical { phys_row } {
    global scramble_db
    …
}
```

Սյուների խճողումը

Սյուների խճողում հասկացության տակ մենք հասկանում ենք հիշողության զանգվածի սյուների տրամաբանական և ֆիզիկական համարների արտապատկերումը: Սյուների խճողման պատճառ կարող են դառնալ սյուների ապակոդավորիչի կառուցվածքը կամ էլ սյուների ոլորապտույտը հիշողության զանգվածում: Հիշողության զանգվածի տարբեր մասեր կարող են ունենալ տարբեր սյուների խճողում, արդյունքում թեստավորման BP-ները այդ մասերում մեկը մյուսից կտարբերվեն: Հիշողության կոմպիլյատորի սյուների խճողումը նկարագրող ֆունկցիայի տեսքն է՝

$$C_k = F_{col}\{I_n, CM_c\}, \quad \text{որտեղ} \qquad (14)$$

C_k - դա նմուշի տրամաբանական սյան հասցեն է և $k \in \{0,1,\ldots, (\#PC-1)\}$,

I_n – դա նմուշի ֆիզիկական Մ/Ե հասցեն է և $n \in \{0,1,\ldots, (\#NB-1)\}$,

CM_c - դա սյուների ֆիզիկական համարն է և $c \in \{0,1,\ldots, (\#CM-1)\}$,

Սյուների խճողումը սահմանվում է յուրաքանչյուր Մուտք/Ելք հանգույցի սյան համար անհատականորեն: Աղյուսակ 1.7-ում տրված է սյուների խճողումը աղյուսակային ներկայացման ընդհանուր ձևը:

Աղյուսակ 1.7

Սյուների խճողման աղյուսակային ներկայացման ձևը

I/On			
Col 0	Col 1	…	Col (CM-1)

TCL «col_scramble_logical_to_physical» պրոցեդուրան, ստանալով Մուտք/Ելք-ի համարը և հարցվող սյունի համարը, վերադարձնում է թվերի զույգ հաջորդականություն՝ առաջին թիվը Մուտք/Ելք-ի «n»-ի համարն է իսկ երկրորդ թիվը համապատասխան ֆիզիկական սյունի համարն է: Օրինակ {(0 0) (0 1) (0 2) (0 3) (1 3) (1 2) (1 1) (1 0) (2 0) (2 1) (2 2) (2 3)} նշանակում է, որ ՀՍ-ը ունի երեք Մուտք/Ելք (n={0, 1, 2}) յուրաքանչյուրում չորս սյուն (CM=4): Ընդ որում, սյուների ֆիզիկական դասավորվածությունը ըստ Մուտք/Ելք-ի հետևյալն է.I/O 0 {0 1 2 3}, I/O 1 {3 2 1 0}, I/O 2 {0 1 2 3}: Ինչպես տեսնում ենք I/O1-ի սյուները ֆիզիկապես դասավորված են հակառակ համարակալությամբ: Իսկ մեկ այլ

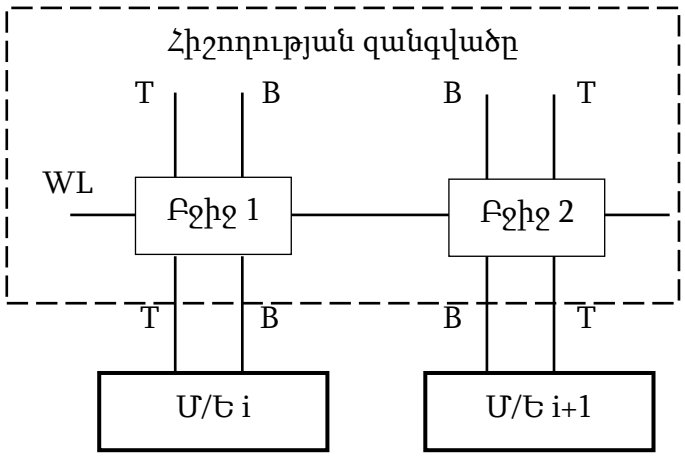
```
proc col_scramble_logical_to_physical { phys_IO log_col } {
    global scramble_db
    …
}
```

TCL պրոցեդուրա «col_scramble_physical_to_logical» ապահովում է հկադարձ՝ ֆիզիկականից դեպի տրամաբանականը, սյուների դասավորման տեղեկության տրամադրումը:

```
proc col_scramble_physical_to_logical { phys_IO phys_col } {
    global scramble_db
    …
}
```

Մ/Ե բիթային գծերի դասավորության խճողումը

Մ/Ե **T** և **B** բիթային գծերի դասավորության խճողում նկարագրում է բիթային գծերի բաշխումը հիշողության զանգվածի և Մուտքային/Ելքային հանգույցների միջև: Բիթային գծերի դասավորվածությունը հիշողության զանգվածի մեկ տողում



Նկար 1.23. Մ/Ե բիթային գծերի բաշխումը

կառուցվում է տողում բջիջների բոլոր բիթային գծերի բաշխման միջոցով (Տես՝ նկ. 1.23), որտեղ i=(0, 1,…(NB-1)): Համաձայն նկ. 1.8-ի և աղ. 1.1-ի, բիթային գծերի դասավորվածությունը ՀՍ-ի բջջում նկարագրվում է բանաձևով՝

$$F_{BL} = \begin{cases} TB, & if\ V_{logic}\ is\ equal\ to\ V_{physical} \\ BB, & if\ V_{logic}\ is\ inverse\ to\ V_{physical} \end{cases}, \text{ որտեղ} \qquad (15)$$

V_{logic} - բջիջի տրամաբանական արժեքն է

$V_{physical}$ – բջիջի բիթային գծի ֆիզիկական արժեքն է

Իսկ բիթային գծերի բաշխվածությունը զանգվածի տողում կնկարագրվի բանաձևով՝

$$R_{BL} = \{F_{BL}0, F_{BL}1,\ldots F_{BL}(\#PC-1) \qquad (16)$$

PC - դա հիշողության սյուների քանակն է և հաշվարկվում է ըստ (3) բանաձևի: ՀՍ-ի տողում **T** և **B** բիթային գծերի հնարավոր չորս տեսակի բաշխվածությամբ՝ ա) {TB TB TB TB}, բ) {BT BT BT BT}, գ) {TB BT TB BT}, դ) {BT TB BT TB}: Մ/Ե **T** և **B** բիթային գծերի բաշխվածությունը յուրահատուկ է յուրաքանչյուր հիշողության նմուշի համար: Մ/Ե **T** և

B բիթային գծերի դասավորության խճողման նկարագրման աղյուսակային ձևը ներկայացված է աղյուսակ 1.8-ում:

Աղյուսակ 1.8

*Մ/Ե **T** և **B** բիթային գծերի դասավորության աղյուսակային ներկայացումը*

I/O i			
Column0	Column1	…	Column(CM-1)
<TB/BT>	<TB/BT>	…	<TB/BT>

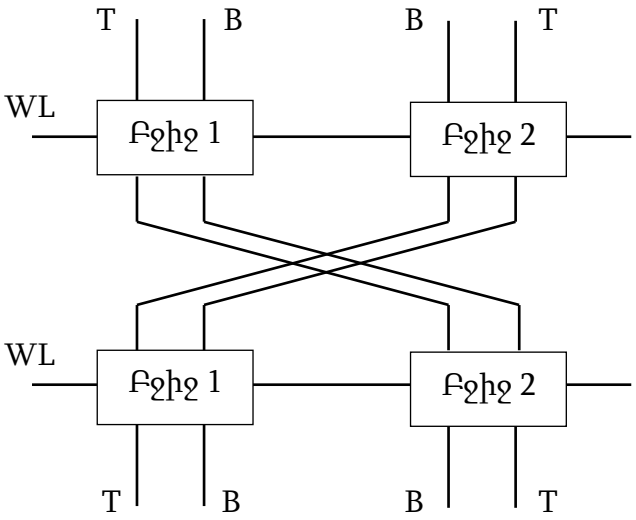
Այս խճողման աղյուսակային ներկայացման ձևը երկկողմանի չէ, քանի որ բիթերի բաշխվածությունից հնարավոր չէ միանշանակորեն ստանալ Մուտքային/Ելքային հանգույցների խճողումը:

Մ/Ե **T** և **B** բիթային գծերի խճողումը իրականացվում է TCL «io_bl_mirroring» պրոցեդուրայով: Ստանալով որպես մուտքային տվյալները Մուտք/Ելք հանգույցի և համապատասխան սյունի ֆիզիկական համարները՝ «io_bl_mirroring» պրոցեդուրան վերադարձնում է **T** և **B** բիթային գծերի դասավորվածությունը:

```
proc io_bl_mirroring { phys_IO phys_col } {
  global scramble_db
  …
}
```

<u>Սյուների ոլորապտույտների բաշխվածություն</u>

ՍՊԴՀՍ-ում սյուների ոլորապտույտների կիրառման դրդապատճառները նույնն են, ինչ որ բիթային գծերի ոլորապտույտների օգտագործումը՝ հիշողության սարքի աշխատանքի հուսալիության բարձրացումը շնորհիվ հիշողության բիթային գծերի վրայի ունակության նվազեցման (Տես նկ. 1.18): Ի տարբերություն բիթային գծերի ոլորապտումի, սյուների ոլորապտումը իրականցվում է ոչ միայն անմիջական հարևան սյուների միջև, այլև սյուների միջև, որոնք գտնվում են մի միյանց նկատմաբ մի քանի սյուն շեղվածությամբ: Լինելով ավելի արդյունավետ, քան բիթային գծերի ոլորապտումը՝ սյուների ոլորապտումի իրականացումը տոպոլոգիայում շատ ավելի բարդ է, և այդ պատճառով այն հազվադեպ է օգտագործվում:



Նկար 1.24. Սյուների պարզ ոլորապտումը

Նկար 1.24-ում ներկայացված է սյուների ոլորապտման պարզ օրինակը: Նկարում պատկերված ոլորապտումը կոչվում է պարզ, քանի որ այն կատարվում է անմիջական հարևան սյուների միջև և առանց բիթային գծերի ոլորապտման: Աղյուսակ 1.9-ում ներկայցված է սյուների ոլորապտման խճողման ներկայացման աղյուսակային ձևը,

Աղյուսակ 1.9

Սյուների ոլորապտման խճողման ներկայացման աղյուսակային ձևը

Ոլորապտվող տողի հասցեն	I/O i			
	Column0	Column1	…	Column(CM-1)
<Row$_1$>	-	-	-	-
<Row$_2$>	-	-	-	-
…	-	-	-	-
<Row$_k$>	-	-	-	-

որտեղ <Row$_1$> ցույց է տալիս այն տողը, որից վերև կատարվել է սյուների ոլորապտումը, իսկ «Columnn» դաշտում նշվում է՝ համապատասխան սյան համար՝ կ՞ա ոլորապտում, թ՞ե ոչ, և եթե ընթացիկ սյան համար կա ոլորապտում, ապա նշվում է այն սյան համարը, որի հետ կատարվել է ոլորապտումը: Հեշտության համար, սյուների ոլորապտման զույգերը կարող են նկարագրվել բանաձևով: Օրինակ՝ սյուների ոլորապտման հասցեն կարող է հաշվարկվել համաձայն բանաձևի՝ C_j, $C_j \pm \Delta$, որտեղ C_j ոլորապտըտվող սյան

համարն է, իսկ ± Δ՝ շեղման հաստատուն գործակիցն է, այդ ժամանակ $C_j \pm \Delta$ ցույց կտա այն սյան հասցեն, որի հետ կատարվել է ոլորապտումը: Հասկանալի է, որ սյուների ոլորապտումը կարող է իրականացված լինել հիշողության զանգվածի մի քանի {1, 2,…,k} տողերում: Բարդ դեպքերում, սյուների ոլորապտումը կարող է կիրառվել տարբեր Մ/Ե հանգուցներում սյուների նկատմամբ նույնպես:

Սյուների ոլորապտման խճողման TCL պրոցեդուրան «column_twist_scramble» պետք է վերադարձնի. ա) սյուների ոլորապտման տողի համարը և բ) զույգ համարների զանգվածը, որը ցույց կտա Մ/Ե-ի համարը, ոլորապտըտված սյուների համարը և համապատասխան շեղումը: Օրինակ՝ այն կարող է իրենից ներկայացնել տվյալների այսպիսի մի զանգված՝

31 { {0 0 +1} {0 1 -1} {0 2 +1}…{15 15 -1} } 95 { {0 0 +1} {0 1 -1} {0 2 +1}…{15 15 -1} } :

«column_twist_scramble» պրոցեդուրաի կաղապարը ունի հետևյալ տեսք՝

```
proc column_twist_scramble { } {
    global scramble_db
    …
}
```

<u>Հիշողության գոտիների բաշխումը</u>

Հիշողության գոտիների մասին ինֆորմացիան կարևոր դեր է խաղում ներդրված ՍՊԴՀ-ի հետ աշխատող կիրառական ծրագրերում. ա) գոտիները, բաժանելով ՀՍ-ի մակերեսը համասեռ հատվածների, հնարավորություն են տալիս կիրառել այդ հատվածի համար որոշակի BP-ը՝ դրանով էականորեն պարզեցնել հիշողության թեստավորման ալգորիթմը, նվազեցնելով այդ ալգորիթմի իրականացման համար անհրաժեշտ ներդրված սարքավորումների քանակը, բ) առանց հիշողության նմուշի GDSII ֆայլի հաշվարկել բջիջների կոորդինատները և գ) հաշվարկել նմուշի հիշողության զանգվածի և տրամաբանական մասերի մակերեսները:

$St(i) = F_{st}(D_k, P_n, S_m)$, որտեղ (17)

D_k - գոտու ուղղվածությունը՝ հորիզոնական կամ ուղղահայաց,

P_n – գոտու դիրքը հիշողության զանգվածում՝ տողի (սյունի) համարը,

S_m – գոտու չափսը (um)՝ երկարությունը (բարձրությունը)

Աղյուսակ 1.10-ում ներկայացված է հիշողության գոտիների բաշխման աղյուսակային ձևը:

Աղյուսակ 1.10

Նմուշի գոտիների բաշխումը

Գոտու Անվանումը	Ուղղվածությունը հորիզոնական/ ուղղահայաց	Դիրքը տողը/ սյունը	Չափսը (um) երկարությունը/ բարձրությունը
-	-	-	-

Ինչպես տեսնում ենք, գոտիները նկարագրող պարամետրերն են՝ անվանումը, ուղղվածությունը, դիրքը և չափսը: Որպես օրինակ դիտենք հիշողության նմուշի պարտադիր գոտիների՝ երկու ուղղահայաց, և երկու հորիզոնական գոտիները նկարագրող աղյուսակը 1.11-ը:

Աղյուսակ 1.11

Նմուշի գոտիների բաշխումը

Անվանումը	Ուղղվածությունը	Դիրքը	Չափսը (um)
Ձախ_եզր	հորիզոնական	սյուն 0	58.2
Աջ_եզր	հորիզոնական	սյուն 1024	3.1
Ներքևի_եզր	ուղղահայաց	տող 0	87.12
Վերևի_եզր	ուղղահայաց	տող 512	4.7
Գոտի_1	հորիզոնական	սյուն 256	2.5
Գոտի_2	հորիզոնական	սյուն 512	2.5
Գոտի_3	ուղղահայաց	տող 128	24.5
Գոտի_4	ուղղահայաց	տող 256	24.5

Հիշողության «strap_location» պրոցեդուրան վերադարձնում է նմուշում գոտիների բաշխվածությունը նկարագրող տվյալների մի զանգված՝

{Ձախ_եզր -1 0 58.2 Աջ_եզր -1 1024 3.1 Ներքևի_եզր 1 0 87.12 Վերևի_եզր 1 512 4.7 Գոտի_1 -1 256 2.5 Գոտի_2 -1 512 2.5 Գոտի_3 1 128 24.5 Գոտի_4 1 256 24.5 }:

```
proc strap_location { } {
  global scramble_db
        …
}
```

Այսպիսով, կոմպիլյատորի մոդելը նկարագրող բանաձևը ընդգրկում է բոլոր խճողումների նկարագրությունը և գոտիների բաշխումը՝

M_{comp} = { Scr1, Scr2,…, Scr n, St1,St2,…,Stm} , որտեղ (18)

Scr n –հիշողության կոմպիլյատորի խճողումների տարրերն են,

St m - հիշողության կոմպիլյատորի գոտիների բաշխումներն են:

Ինչպես արդեն նշել ենք, ՍՊԴՀՍ-ի հետազոտված կառուցվածքային մոդելի խճողումների քանակը տասնվեցն են: ԱՀ–ում կիրառական ծրագրերի կողմից՝ ընտրովի ձևով, ՀՍ-ի կառուցվածքային խճողումների TCL պրոցեդուրաների օգտագործումը էականորեն հեշտացնում է կառուցվածքային մոդելի կիրառումը: ՍՊԴՀՍ-ի կառուցվածքային մոդելի առաջին վեց խճողումների համապատասխան TCL պրոցեդուրաները, նրանց մուտքային և ելքային պարամետրերը նկարագրված են աղյուսակ 1.12-ում:

Աղյուսակ 1.12

Կառուցվածքային խճողումները նկարագրող TCL պրոցեդուրաների նկարագրությունը

Կառուցվածքային մոդելի միավորը	**Պրոցեդուրայի անունը**	**Մուտքային պարամետրերը**	**Ելքային պարամետրերը**
հասցեի ծածկագրումը	address_scramble	-	հասցեի բջիջների նկարագրությունը
բանկերի ապակոդավորման ֆունկցիան	bank_scramble	տրամաբանական բանկի հասցեն	ֆիզիկական բանկի հասցեն
տողերի ապակոդավորման ֆունկցիան	row_scramble	տրամաբանական տողի հասցեն	Ֆիզիկական տողի հասցեն

Աղյուսակ 1.12 (շարունակություն)

Կառուցվածքային խճողումները նկարագրող պրոցեդուրաների նկարագրությունը

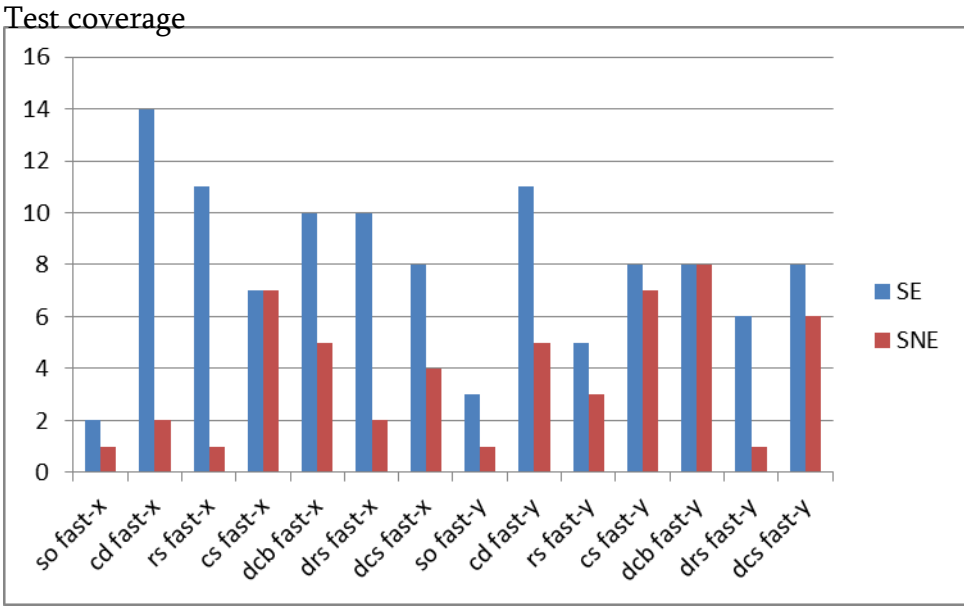
սյուների ապակոդավորման ֆունկցիան	column_scramble	տրամաբանական սյան հասցեն	ֆիզիկական սյան հասցեն
Մ/Ե հանգույցների խճողումը	io_cell_scramble	տրամաբանական Մ/Ե-ի համարը	ֆիզիկական Մ/Ե-ի համարը
մուտքերի տիպերի խճողումը	port_scramble	մուտքի անուն	մուտքի նկարագրությունը (R/W/RW)
Բիթային գծերի ոլորապտման խճողումը	bl_twist_scramble	Բիթային գծի համարը	Բիթային գծի տեսակը (TB, BT)
Մ/Ե բիթային գծերի խճողումը	io_bl_mirroring	-	«0» տողում բիթային գծերի բաշխումը

1.3 Փորձնական տվյալները

Աշխատանքի ընթացքում հետազոտվել են տարբեր տեխնոլոգիաների և տեսակների (800-ից ավելի) հիշողության կոմպիլյատորների կառուցվածքները, և ստացված տվյալների հիման վրա գեներացվել են այդ կոմպիլյատորների կառուցվածքային մոդելները: Ստացված մոդելները օգտագործվել են ԱՀ–ի կիրառական ծրագրերում մի քանի տասնյակ պատվիրատուների մոտ:

Կատարվել են փորձեր, որոնց ընթացքում հիշողության նմուշները թեստավորվել են տարբեր թեստավորման BP-ների (so, cd, rs, cs, dcb, drs, dcs) միջոցով (Տես՝ աղ. 1.2), կիրառելով fast-x և fast-y հասցեավորումները:

Նկար 1.25-ում ներկայացված է նմուշի խճողման ինֆորմացիայի օգտագործման փորձերից մեկի արդյունքը պատկերող գրաֆիկը: Փորձերի առաջին դեպքում «SE»



Նկար 1.25. Խճողման ինֆորմացիայի օգտագործման ազդեցությունը թեստավորման արդյունավետության վրա

(scrambling enabled) թեստավորման ալգորիթմը կազմված էր՝ հաշվի առնելով նմուշի խճողումների ինֆորմացիան, իսկ երկրորդ դեպքում՝ «SNE» (scrambling NOT enabled) առանց այդ ինֆորմացիայի: Նկար 1.25-ից երևում է, որ թեստավորման յուրաքանչյուր դեպքում խճողումով՝ «SE», *թեստերի արդյունավետությունը* հավասար է կամ գերազանցում է առանց խճողումով թեստերի արդյունավետությանը: Կատարված հետազոտությունների արդյունքում պարզ դարձավ, որ հիշողության նմուշի թեստավորման ընդհանուր արդյունավետությունը աճում է 30%-ով, երբ հիշողության թեստավորման ալգորիթմը իր աշխատանքի ընթացքում հաշվի է առնում հիշողության նմուշին բնորոշ՝ միայն այդ նմուշին յուրահատուկ կառուցվածքային խճողումների մասին ինֆորմացիան [16], [25]:

Եզրակացություններ

1. Նկարագրվել են ՀՍ-ի արտադրության օգտակար ելքի բարձրացման ձևերը, մասնավորապես, հիշողության թեստավորման արդյունավետության ազդեցությունը ՕԵ-ի վրա: Ներկայացվել են հիշողության սարքերի կառուցվածքային բաղադրամասերը, կառուցվածքային առանձնահատկություն-ները և դրանց կախվածությունը հիշողության բնութագրիչ պարամետրերից:

2. Հետազոտվել են հիշողության սարքերի խճողումների առաջացման պատճառները և դրանց ազդեցությունը հիշողության սարքերի թեստավորման վրա:

3. Մշակվել են. կառուցվածքային խճողումների դասակարգումը և կառուցվածքային մոդելի ներկայացման ձևերը: Իրականացվել է տարբեր տեխնոլոգիաների և տեսակների հիշողության կոմպիլյատորների կառուցվածքների հետազոտումը, և ստացված տվյալների հիման վրա գեներացվել են այդ կոմպիլյատորների կառուցվածքային մոդելները:

ԳԼՈՒԽ 2 – ՀԻՇՈՂՈՒԹՅԱՆ ԿՈՄՊԻԼՅԱՏՈՐԻ ԿԱՌՈՒՑՎԱԾՔԱՅԻՆ ՄՈԴԵԼԻ ՍՏՈՒԳՄԱՆ ԵՎ ՆՄՈՒՇԻ ՄՈԴԵԼԸ GDSII ՁԵՎԱՉԱՓԻՑ ԴՈՒՐՍԲԵՐՄԱՆ ԱՎՏՈՄԱՏԱՑՎԱԾ ՀԱՄԱԿԱՐԳԵՐԸ

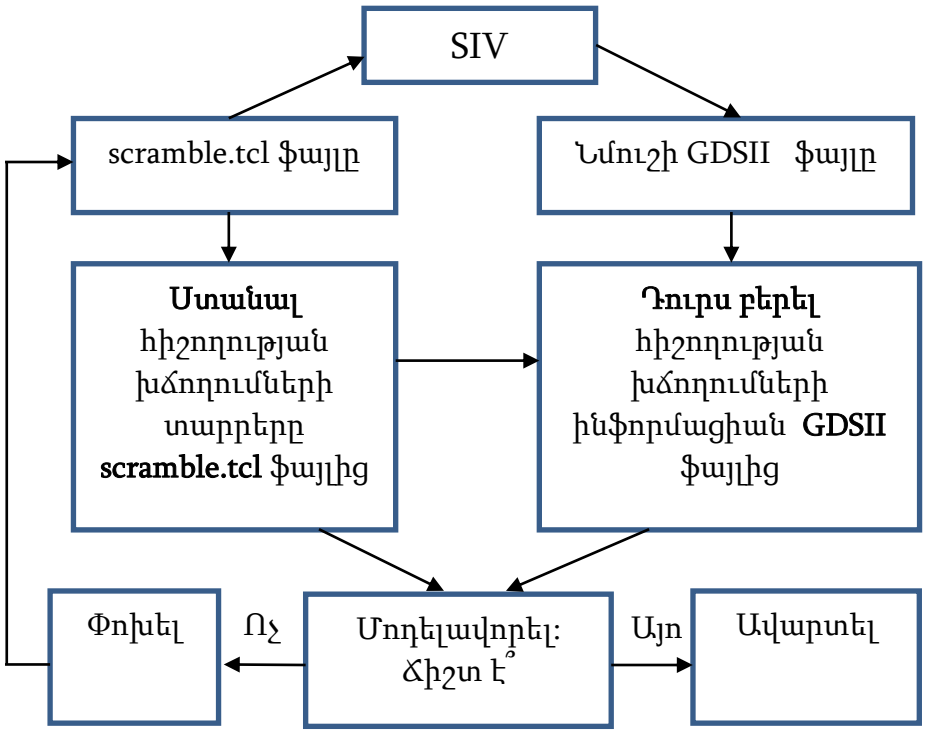
2.1 Հիշողության կառուցվածքային մոդելի ստուգման ավտոմատացված համակարգ

Հիշողության կոմպիլյատորի կառուցվածքային խճողումների TCL պրոցեդուրաները նկարագրվում և պահվում են կոմպիլյատորի համար անբաժանելի մաս կազմող scramble.tcl ֆայլում՝ մոդելում: scramble.tcl ֆայլը ստեղծվում է հիշողության կոմիպիլյատորը նախագծող ճարտարագետի կողմից, քանի որ՝ նախագծման փուլում, միայն նախագծի ճարտարագետն է հանդիսանում կոմպիլյատորի խճողումների մասին ինֆորմացիայի միակ ստույգ աղբյուրը: ՆՀՍ-երի ՂԱՀ-ում կառուցվածքային մոդելի գեներացման գործընթացը ներկայացվել է նկ. 1.22-ում:

Չնայած նրան, որ կառուցվածքային մոդելի հետ կապված հետազոտման և ներդրման աշխատանքների ընթացքում ստեղծվել է հիշողության կոմպիլյատորների համար կառուցվածքային մոդելի գեներացումը հեշտացնող MIG ավտոմատ ծրագրային գրաֆիկական գործիքը [17], կա մի անհաղթահարելի հանգամանք, որը արգելք էր հանդիսանում այդ մոդելի անմիջական կիրառմանը: Այդ արգելքը այն է, որ մոդելը ստեղծվում է հիշողությունը նախագծող ճարտարագետի կողմից, և այդ պարագայում անհնար է բացառել մարդկային գործոնի հետևանքով մոդելում սխալի առկայության հավանականությունը: Այս պատճառով, շատ կարևոր է դառնում մոդելի ստեղծմանը զուգահեռ նախագծել և ներդնել հիշողության կառուցվածքային մոդելի լիարժեք ստուգումը ապահովող ավտոմատ համակարգ [28], [29]: Ինչպես նշված է [27] և [29] – ում՝ խնդրի բարդության պատճառով, ներկայումս, անհնար է իրականացնել կառուցվածքային մոդելի ստուգման աշխատանքների լիովին ավտոմատացումը:

2.1.1 Հիշողության կառուցվածքային մոդելի ստուգման հիմունքները

Հետազոտությունների արդյունքում հայտնաբերվել, մշակվել և իրականացվել է հիշողության կոմպիլյատորի կառուցվածքային մոդելը ստուգող ավտոմատացված ծրագրային համակարգ [12], որը անվանվում է Հիշողության խճողումների



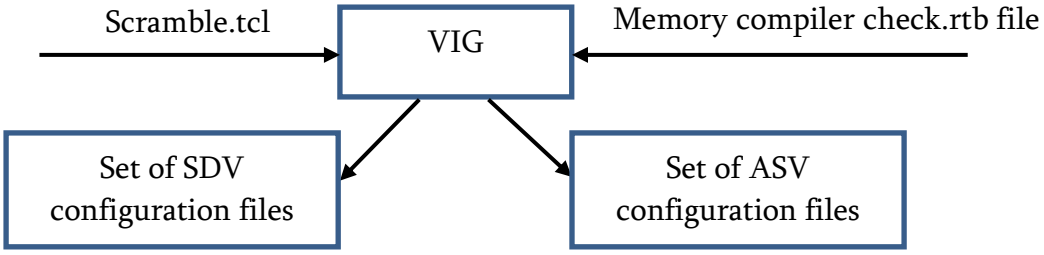
Նկար 2.1. Կառուցվածքային մոդելի ստուգման SIV ԱՀ-ի աշխատանքի նկարագրությունը

ինֆորմացիան ստուգող հոսք (անգլերեն՝ memory Structural Information Verification (SIV) flow): Հետագայում տեքստում կօգտագործենք SIV հապավումը:

Հիշողության կառուցվածքային մոդելի ստուգման հիմնական գաղափարը ներկայացված է նկար 2.1-ում: Նմուշի GDSII ձևաչափը պարունակում է նմուշի խճողումների մասին իրական տեղեկությունները և այդ պատճառով scramble.tcl ֆայլի իսկությունը ստուգվում է՝ համեմատելով նմուշի GDSII ֆայլից դուրս բերված համապատասխան խճողման ինֆորմացիայի հետ: Քանի որ scramble.tcl ֆայլը պարունակում է խճողումների ինֆորմացիա հիշողության կոմպիլյատորի ողջ տիրույթում, ապա հասկանալի է, որ նկար 2.1-ում ներկայացված քայլերը անհրաժեշտ է կատարել հնարավորինս շատ քանակով նմուշների համար և ինչքան այդ ստուգումների քանակը շատ լինի, ապա այնքան մեծ կլինի scramble.tcl ֆայլի ճշտությունը: Իրականում GDSII ֆայլից խճողումների մասին ինֆորմացիայի դուրս բերման գործընթացը բավական ժամանակատար է և պահանջում է էական հաշվողական ռեսուրսներ:

Ժամանակը և ռեսուրսները սահմանափակող հանգամանք և արգելք են հանդիսանում ստուգման համար անհրաժեշտ նմուշների կոնֆիգուրացիաների բազմությունը գեներացնող գործիքի համար: Ստուգող նմուշների քանակի լավարկումը կարևոր և բարդ խնդիրներից է, որը հաջողությամբ լուծվում է SIV ԱՀ-ի միջավայրում:

SIV ԱՀ-ի աշխատանքի ընթացքում առաջացած խնդիրների լուծումները ապահովելու նպատակով մշակվել է հիշողության GDSII ֆայլի ձևափոխման (այսպես կոչված ծրագրավորման) մեթոդը և ստեղծվել են այս մեթոդը իրականացնող լրացուցիչ ծրագրային գործիքներ:



Նկար 2.2. SIV ԱՀ-ի VIG գործիքով SDV և ASV նմուշների ստուգող կոնֆիգուրացիաների գեներացումը

Դիտարկենք GDSII ֆայլի ծրագրավորման մեթոդի ընդհանուր գաղափարը, *SIV ԱՀ-ի* ծրագրային գործիքների հակիրճ նկարագրությունները՝ նպատակը [17]:

- Verification Instance Generator (VIG) գործիքը (Տես՝ նկ. 2.2) որպես մուտք ընդունում է կառուցվածքային մոդելի նկարագրության scramble.tcl ֆայլը և հիշողության կոմպիլյատորի պարամետրերի գործող տիրույթը նկարագրող check.rtb ֆայլը: VIG –ի գործիքը գեներացնում է անհրաժեշտ թվով հիշողությունների նմուշների ստուգող կոնֆիգուրացիաների ֆայլերը (իրենց բնութագրիչ պարամետրերով), որոնց նվազագույն *քանակը* և ֆայլերում տրված *կոնֆիգուրացիաները* կբավարարեն տվյալ հիշողության կոմպիլյատորի կառուցվածքային խճողումների տարրերի լիարժեք ստուգման համար: Նվազագույն տերմինը վերաբերում է ինչպես հիշողության նմուշների քանակին, այնպես էլ նրանց չափսերին, ինչը շատ կարևոր է SIV հոսքի աշխատանքի տևողության (ժամանակի) նվազեցման գործընթացում [17]:
- Verification Pattern Generator (VPG) գործիքը որպես մուտք ընդունում է scramble.tcl ֆայլը և հիշողության նմուշի բնութագրիչ պարամետրերը: VPG -ին իր

աշխատանքի արդյունքում գեներացնում է նվազագույն թվով ստուգող հավաքածուներ (PAT, SVP), որոնց առկայությունը բավարար է տվյալ հիշողության նմուշի և նրա մոդելի համապատասխանությունը ստուգելու համար: Գեներացվող հավաքածուների թիվը կախված է ստուգող նմուշների կառուցվածքային խճողումների տեսակներից, ինչպես նաև հիշողության նմուշի բնութագրիչ պարամետրերից՝ ֆիզիկական չափսերից [17]:

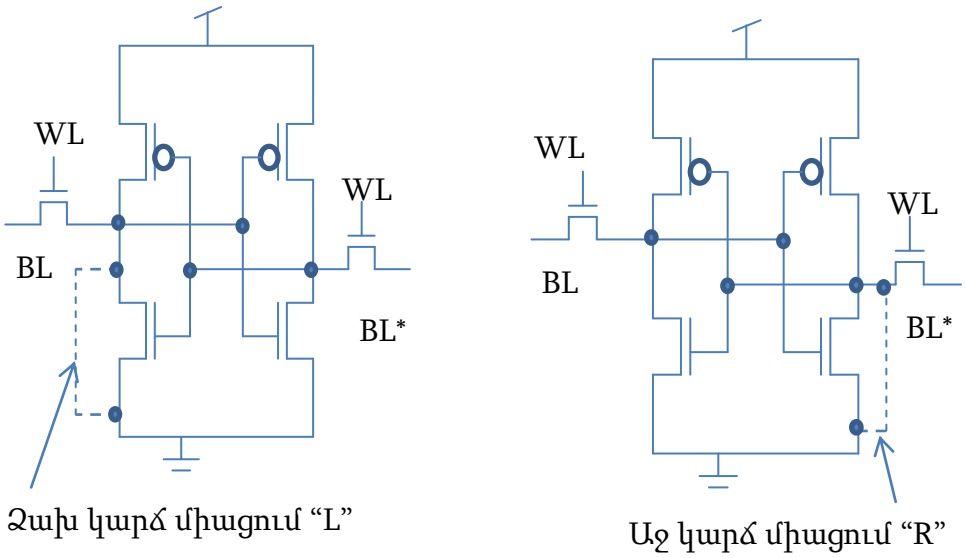
- Vector Output Generator (VOG) գործիքը որպես մուտք ընդունում է scramble.tcl ֆայլը և ստուգող նմուշների հավաքածուն, որոնց հիման վրա գեներացնում է հիշողության տրամաբանական հասցեներին համապատասխանող մոդելավորման արդյունքում *սպասվող* արժեքները: Հետագայում այս արժեքները համեմատվում են «ծրագրավորված» հիշողության նմուշի SPICE ֆայլի մոդելավորման միջոցով ստացված՝ընթերցված, արժեքների հետ [17]:
- Memory Compiler (MC) գործիքը որպես մուտք ընդունում է հիշողության նմուշի բնութագրիչ պարամետրերը և գեներացնում է համապատասխան հիշողության նմուշի GDSII ձևաչափով ֆայլը՝ օգտագործելով հիշողության կոմպիլյատորը: Այս գործիքը նախագծվում է հիշողության կոմպիլյատոր նախագծող ընկերությունների (Synopsys, Mentor Graphics, ARM և այլն) կողմից:
- Memory Programming Tool (MPT) գործիքը որպես մուտք ստանում է հիշողության նմուշի գրաֆիկական GDSII ներկայացումը և նմուշի ծրագրավորող ֆայլերը: MPT գործիքը, միաձուլելով նմուշի GDSII ֆայլը և ծրագրավորող բջիջները՝ ըստ ծրագրավորման ֆայլում տրված բջիջների արժեքների բաշխմանը, ստանում է հիշողության նմուշի «ծրագրավորված» GDSII ֆայլը, որի հիշողության զանգվածի ամեն մի բջիջ ծրագրավորված է նախապես, հստակորեն որոշված «0» կամ «1» արժեքով: Բացի այդ, MPT գործիքը, ստանալով համապատասխան հրաման, նմուշի GDSII ֆայլում հետազոտում է կառուցվածքային գոտիների բաշխումը և ավտոմատ ձևով գեներացնում է նմուշում գոտիների կառուցվածքային տվյալները (Տես՝ աղ. 1.11) պարունակող ֆայլը:Layout vs. SPICE (LVS) գործիքը որպես մուտք ստանալով հիշողության նմուշի «ծրագրավորված» GDSII ֆայլը, գեներացնում է նրա SPICE նկարագրությունը: Սխեմայի SPICE լեզվով նկարագրությունը թույլ է տալիս ավտոմատ կերպով մոդելավորել հիշողության

էլեկտրական սխեման՝ մուտքին տալով համապատասխան ազդանշանները, և ելքային ազդանշանները համեմատել սպասվող արժեքների հետ:

2.1.2 Ծրագրավորող բջիջները

Հիշողության նմուշի ծրագրավորումը իրականցվում է յուրաքանչյուր հիշողության կոմպիլյատորի համար անհատապես ստեղծվող «0» և «1» վիճակը ծրագրավորող բջիջների միջոցով:

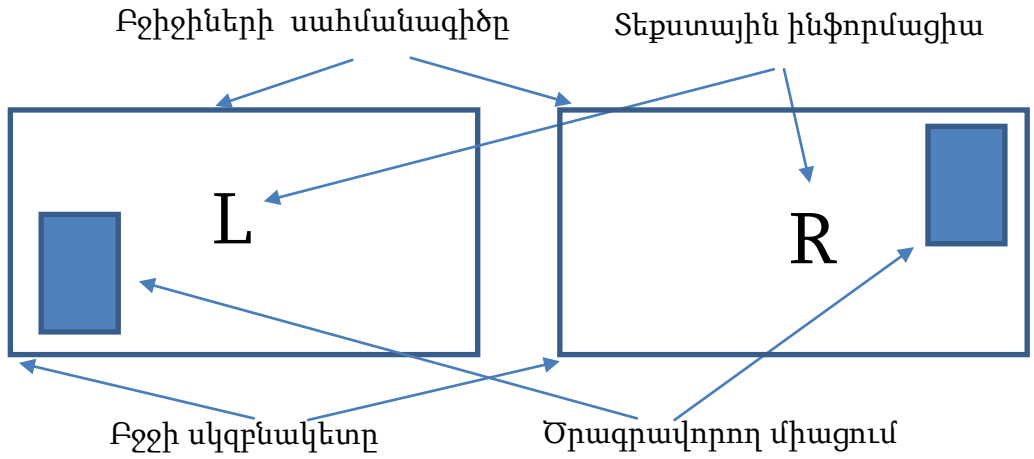
Նկար 2.3-ում ներկայացված են կարճ միացումների միջոցով ստացվող՝ «ծրագավորման» աջ և ձախ բջիջների գաղափարը ներկայացնող հիշողության բջիջների էլեկտրական սխեմաները: Ինչպես արդեն դիտարկել էինք, հիշողության բջջի



Նկար 2.3. SIV ԱՀ-ում հիշողության ծրագրավորող բջիջների էլեկտրական սխեման

էլեկտրական սխեման (Տես՝ նկ. 1.6 և 2.3) ունի լիովին սիմետրիկ կառուցվածք: Նմանապես, բջջի տոպոլոգիան նույնպես սիմետրիկ (նժարի նման հավասարակշռված) է բջջի տոպոլոգիայի կենտրոնի նկատմամբ: Այսպիսով, հիշողության բջիջը հարկադրաբար տրամաբանական «0» կամ «1» վիճակ գցելու համար անհրաժեշտ է տոպոլոգիայում խախտել այդ հավասարակշիռ վիճակը՝ վերացնել սիմետրիկությունը: MPT ծրագիրը տեղադրում է ծրագրավորող բջիջները հիշողության բջիջների վրա (մտցնելով բջջի տոպոլոգիայում ծրագրավորումը իրականացնող կարճ միացումը), որի արդյունքում համապատասխան շերտերի համակցման հետևանքով, հիշողության

զանգվածի բջիջների կառուցվածքի սիմետրիկությունը շեղվում է՝ ապահովելով հիշողության, այսպես կոչված, ծրագրավորումը: Ծրագրավորման բջիջները՝ ինչպես և հիշողության բջիջը, պատրաստվում են յուրաքանչյուր տեխնոլոգիայի համար անհատականորեն: Նկար 2.4 –ում պատկերված են «ծրագրավորումը» իրականացնող **«L»** և **«R»** բջիջների տոպոլոգիաների օրինակները:



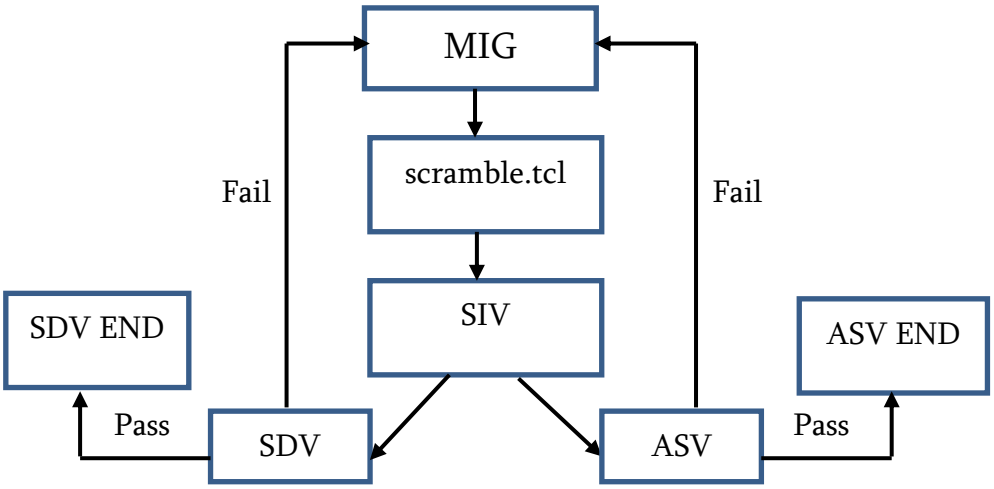
Նկար 2.4. Հիշողության ծրագրավորման L և R բջիջների տոպոլոգիայի օրինակը

2.1.3 SIV ավտոմատացված համակարգի աշխատանքի նկարագրությունը

Բացի նշված ծրագրերից, SIV հոսքում ծրագրավորված հիշողության նմուշի SPICE ձևաչափի էլեկտրական սխեման ստանալու համար օգտագործվում է Layout Vs SPICE (LVS) իրականացնող՝ տարբեր ընկերությունների կողմից մշակված ծրագրեր (օրինակ calibre, Hercules, ICV): Այնուհետև, ստացված SPICE ֆայլի մոդելավորման համար օգտագործվում է SPICE մոդելավորման գործիքը (HSPICE, HSIM, UltraSIM կամ XA), որն իր մուտքին ստանում է SPICE լեզվով նկարագրությունը, հիշողությունից ընթերցելու գործողության համար անհրաժեշտ մուտքային ազդանշանները և սպասվող արժեքների բազմությունը, գեներացնում է **սպասվող** և **ընթերցված** արժեքների համեմատությունների արդյունքը պարունակող ֆայլը:

Ինչպես բազմիցս նշել ենք, հիշողության կառուցվածքային մոդելը բաղկացած է երկու հիմնական մասերից՝ հիշողության գոտիների բաշխումը նկարագրող մասից, և հիշողության կառուցվածքային խճողումները նկարագրող պրոցեդուրաների մասից:

Կառուցվածքային մոդելի ստուգումը իրականացնող SIV հոսքը, նույնպես, բաղկացած է երկու անկախ՝ ա) հիշողության գոտիների բաշխումը ստուգող (անգլերեն Straps



Նկար 2.5. SIV ԱՀ-ում Հիշողության Կառուցվածքային մոդելի ստուգման բլոկ-սխեման

Distribution Verification (SDV)) և բ) հիշողության կառուցվածքային խճողումները ստուգող (անգլերեն Address Scramble Verification (ASV)) մասերից (Տես՝ նկ. 2.5):

2.1.4 Հիշողության գոտիների բաշխման ստուգումը (SDV)

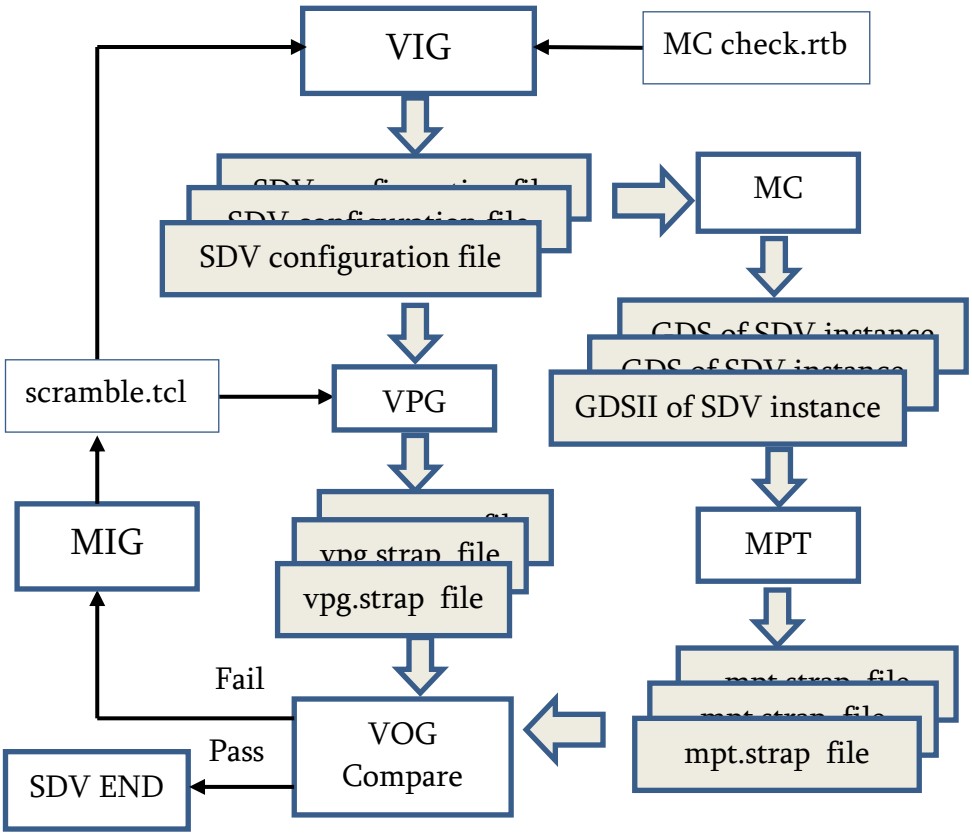
Ինչպես արդեն սահմանել էինք, հիշողության զանգվածը հիշողության նմուշի տոպոլոգիայի այն մասն է, որը կառուցված է հիշողության բջիջներով: Հիշողության բջիջները պարունակող հանգույցներն են՝

- հիշողության զանգվածը՝ մակերևույթը (memory area),
- պահեստային սյուները և տողերը (redundancy columns and rows),
- և հիշողության մակերևույթը հավասարակշռող, լրացուցիչ միավորները (dummy rows and dummy columns):

Նմուշի տոպոլոգիայի մնացած բոլոր մասերը անվանում են գոտիներ:

Կառուցվածքային մոդելի գոտիների բաշխման ստուգումը իրականցվում է SDV հոսքի միջոցով: Սկզբում, VIG ծրագիրը, վերլուծելով մոդելի scramble.tcl ֆայլում նկարագրված գոտիների բաշխումը և դրան համատեղ, հիշողության կոմպիլյատորի տիրույթի նկարագրությունը, գեներացնում է՝ անհրաժեշտ քանակությամբ ստուգող

նմուշների կոնֆիգուրացիաների ֆայլերը: Այդ կոնֆիգուրացիաների ֆայլերի քանակը և նմուշների տեսակները այնպես են ընտրված, որ ապահովում են կոմպիլյատորի ՆՀՍ-ի



Նկար 2.6. Հիշողության գոտիների ստուգման SDV ԱՀ-ի բլոկ-սխեման

գոտիների բաշխման լիարժեք ստուգումը:

Կոնֆիգուրացիոն ֆայլերի հիման վրա MC գործիքով գեներացվում են SVD ստուգման նմուշների GDSII ֆայլերը (Տես՝ նկ. 2.6): Հետագայում, այդ ֆայլերը մշակվում են MPT գործիքի միջոցով, որի արդյունքում գեներացվում են նմուշի գոտիների բաշխումը նկարագրող՝ տեքստային ձևաչափով և mpt.straps անվանումով, ֆայլեր:

Դրանից հետո, VPG գործիքը, իր հերթին, կոմպիլյատորի scramble.tcl ֆայլից, ստանում է գոտիների բաշխումը՝ յուրաքանչյուր SDV կոնֆիգուրացիայի համար (vpg.straps անվանումով): Այսպիսով, յուրաքանչյուր կոնֆիգուրացիայի համար ստացվում են գոտիների բաշխումը նկարագրող երկու ֆայլ՝ մեկը GDSII ձևաչափից, մյուսը հիշողության կառուցվածքային մոդելի (scramble.tcl) ֆայլից: Ստացված ֆայլերի համեմատության միջոցով (Տես՝ նկ. 2.6), SIV-ը ստեղծում է արդյունքները ամփոփող

ելքային ֆայլը և գեներացնում է ամփոփիչ հաղորդագրություն SDV ստուգման ընթացիկ վիճակի մասին: Նկար 2.7-ում տրված են MPT և VPG ֆայլերի օրինակները:

mpt.straps
H row #0 51.185
H row #6 0.8
H row #8 0.585
V col #0 1.34
V col #320 42.5
V col #640 6.425

ա.

vpg.straps
H row #0 51.185
H row #6:8 1.385
V col #0:4 1.34
V col #320 42.5
V col #640 6.425

բ.

Որտեղ
Գոտիների տեսակը - Vertical/Horizontal
Գոտիների դիրքը - row/column
Գոտիների չափսը - length/heght

Նկար 2.7. Գոտիների նկարագրությունը՝ ա. MPT և բ. VPG ֆայլերում

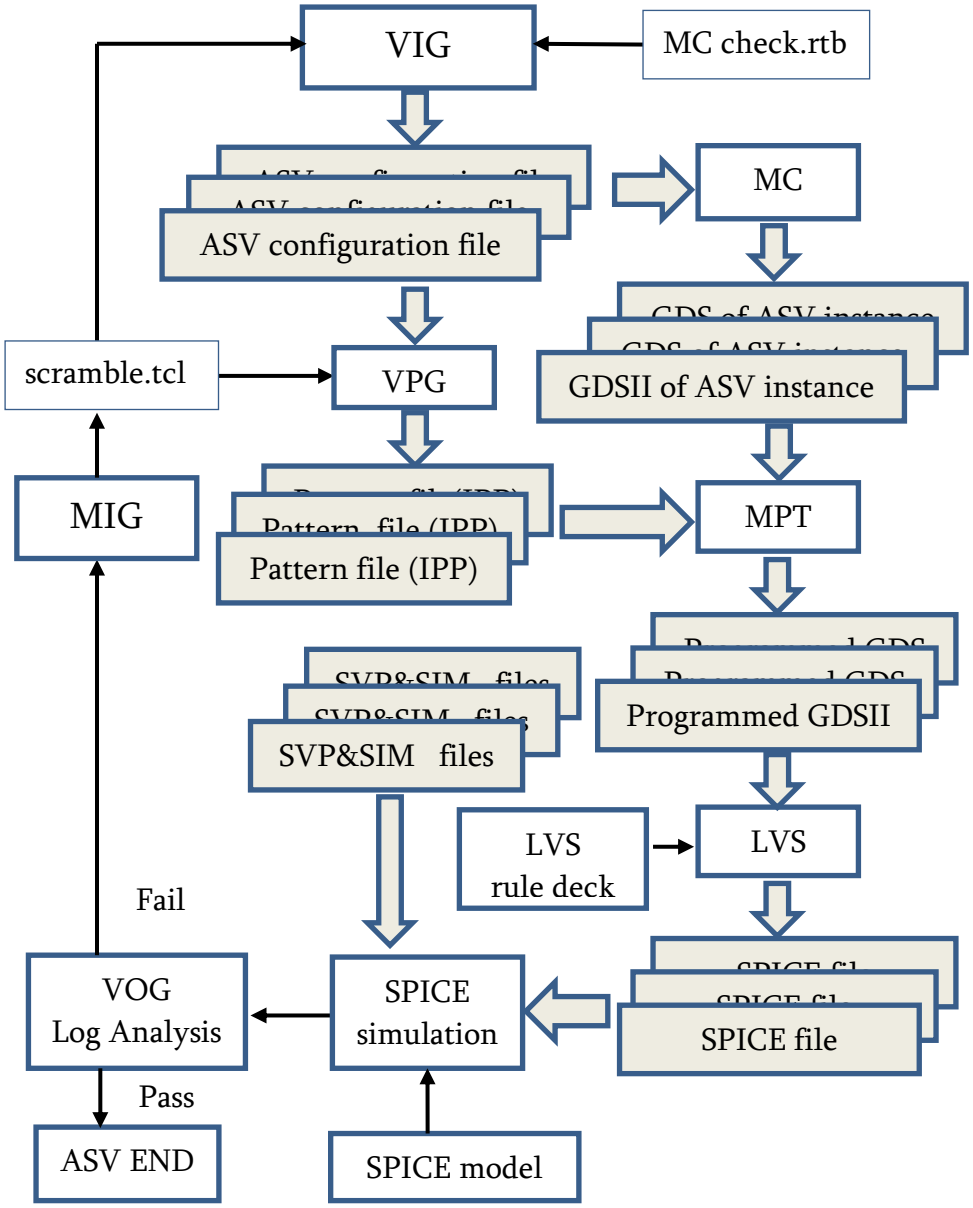
Եթե համեմատության արդյունքը բացասական է, ապա ամփոփող սխալների ֆայլում (Տես՝ նկ. 2.8) բերվում են սխալի պատճառները վերլուծելու համար անհրաժեշտ բոլոր տվյալները, որոնց հիման վրա և MIG ծրագրի օգնությամբ կատարվում են ուղղումներ

```
******************** m74_sdv ********************
{BK = 1} {CM = 4} {NB = 84} {NW = 48}
{N_rows = 12.0 } {N_rows_BK = 12.0 } {N_col = 336 }

vdda_enable = 0 pg_enable = 0
bist_enable = 1 redundancy_enable = 1 center_decode = 1
reconfig_reg = 0
Extracted from GDS|     Generated by scramble.tcl
------------------|-----------------------------
H row #0  41.76 |  H row #0 41.765  ...<< INVALID SIZE
H row #8  0.74  | H row #8 0.74
H row #12  2.81  |      H row #12 2.81
V col #0  1.49  |       V col #0:4 1.49
V col #120  22.5  |     V col #88 22.5 ...<< INVALID LOCATION
V col #172  66.49  |    V col #172:176 66.49
V col #228  22.5  |     V col #260 22.5 ...<< INVALID LOCATION
V col #344  1.49  |     V col #344 1.49
```

Նկար 2.8. ԱՀ-ի SDV սխալների ֆայլի օրինակը (հատված)

scramble.tcl ֆայլի գոտիների նկարագրման բաժնում: Ստուգումների այս հաջորդականությունը կրկնվում է այնքան ժամանակ (Տես՝ նկ. 2.6), մինչև որ վերջնական արդյունքում մոդելի գոտիների բաշխման նկարագրությունը լիովին համապատասխանի հիշողության կոմպիլյատորի GDSII ձևաչափին:



Նկար 2.9. Հիշողության կառուցվածքային խճողումների ստուգման ASV ԱՀ-ի բլոկ-սխեման

2.1.5 Հիշողության հասցեների խճողումների ստուգումը (ASV)

Նկար 2.9-ում ներկայացված է ASV-ի ստուգման բլոկ-սխեման: Ի սկզբանե նշենք, որ հիշողության հասցեների խճողումների ստուգման գործընթացը (ASV ստուգումը) ավելի ժամանակատար է, քան հիշողության գոտիների բաշխման ստուգումը (SDV), քանի որ այն իր մեջ ընդգրկում է երկու հաջորդաբար կատարվող և բավականին ժամանակատար գործողություններ՝ ա) նմուշի GDSII ձևաչափից նմուշի SPICE

ձևաչափի սխեմատիկայի ֆայլի դուրս բերումը (ինչպես նշել ենք, դա կոչվում է LVS գործողություն), և բ) LVS-ի աշխատանքի արդյունքում ստացված նմուշի SPICE ֆայլի մոդելավորումը: ASV-ի դեպքուն VGP գործիքը յուրաքանչյուր նմուշի համար գեներացնում է՝

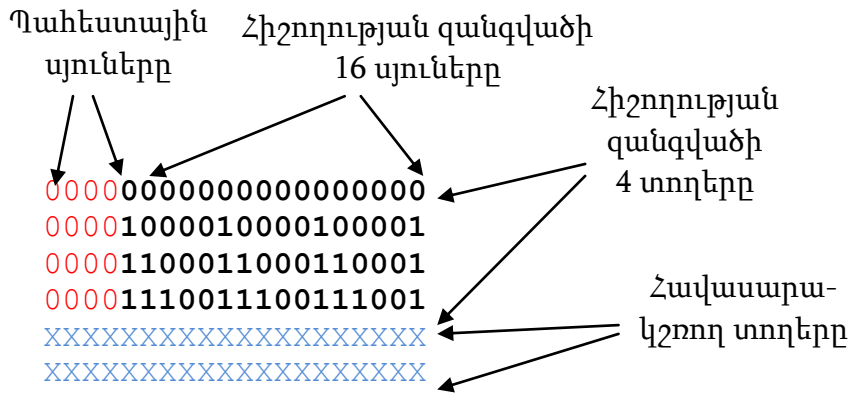
ա) ծրագրավորման օրինակների ֆայլերը (անգլերեն՝ Instance Programming Pattern (IPP)) (Տես՝ նկ. 2.10),

բ) SPICE մոդելավորման համար անհրաժեշտ՝ վեկտորային օրինակների մուտքային ֆայլերը (անգլերեն՝ SPICE vector pattern (SVP)), և

գ) SPICE մոդելավորող ծրագիրը աշխատեցնող՝ պարամետրերի ֆայլերը:

Պետք է ընդգծել, որ ASV ստուգման մոդելավորման ժամանակ ստացված SPICE ֆայլի միջոցով իրականացվում է բացարձակապես «կարդալ» գործողության մոդելավորումը:

IPP ծրագրավորման ֆայլը իրենից ներկայացնում է տեքստային ֆայլ, որը իր մեջ



Նկար 2.10. IPP ֆայլի օրինակը

ընդգրկում է տրամաբանական զրոների և մեկերի հավաքածուներ: Այդ հավաքածուները ունեն նմուշի զանգվածի ֆիզիկական կառուցվածքին համապատասխան քանակությամբ տողեր և սյուներ: IPP ծրագրավորման ֆայլերը մուտքային տեղեկություններ են տրամադրում MPT-ի գործիքի համար: Նկար 2.10 -ում պատկերված է IPP-ի օրինակը: Համաձայն այդ օրինակի՝ հիշողության նմուշը ունի 4 տող, 16 սյուն, 2 պահեստային սյուներ և 2 հավասարակշռող՝ պարապ տողեր: Հիշողության հիմնական զանգվածից դուրս գտնվող պահեստային և հավասարակշռող տողերին և սյուներին պատկանող բջիջների արժեքները պարտադիր ձևով ծրագրավորվում են տրամաբանական զրո արժեքով: Իսկ IPP ֆայլի առաջին տողի բջիջները ստանում են այնպիսի արժեքներ, որոնք կապահովեն *տրամաբանական* զրոյական արժեքների ծրագրավորումը նմուշի

ֆիզիկական առաջին տողում (Տես՝ նկ. 2.10)։ Այդ առաջին տողը հնարավորություն է տալիս բացահայտել հիշողության զանգվածի բիթային գծերի խճողումը (T, B)։ Ֆիզիկական երկրորդ տողից սկսած IPP-ը պարունակում է անկյունագծային ստուգող հավաքածու, որը նմուշի ֆիզիկական կենտրոնի առանցքի նկատմամբ և անկյունագծով նրանից վերև գտնվող բջիջներում ապահովում է տրամաբանական «1» արժեքներ, իսկ առանցքից ներքև անկյունագծով գտնվող բջիջներում՝ «0» արժեքներ (Տես՝ նկ. 2.11)։ Ինչպես տեսնում ենք, հիշողության հիմնական սյուներից և տողերից ոչ մեկը չի պարունակում ամբողջովին զրոյական արժեք, և տողերի (սյուների) արժեքները չեն կրկնվում, որը իր հերթին ապահովում է խճողումների վերծանման միանշանակությունը։ Այս ստուգող հավաքածուն ապահովում է խճողումների հետևյալ տարրերի ստուգումները. հիշողության հասցեավորման խճողման, ֆիզիկական և տրամաբանական տողերի և սյուների համապատասխանության, ինչպես նաև ստուգում է հավասարակշռող և պահեստային տողերի և սյուների դիրքերի բաշխման ճշտությունը։ Նշված խճողումների ցանկացած անհամապատասխան նկարագրումը

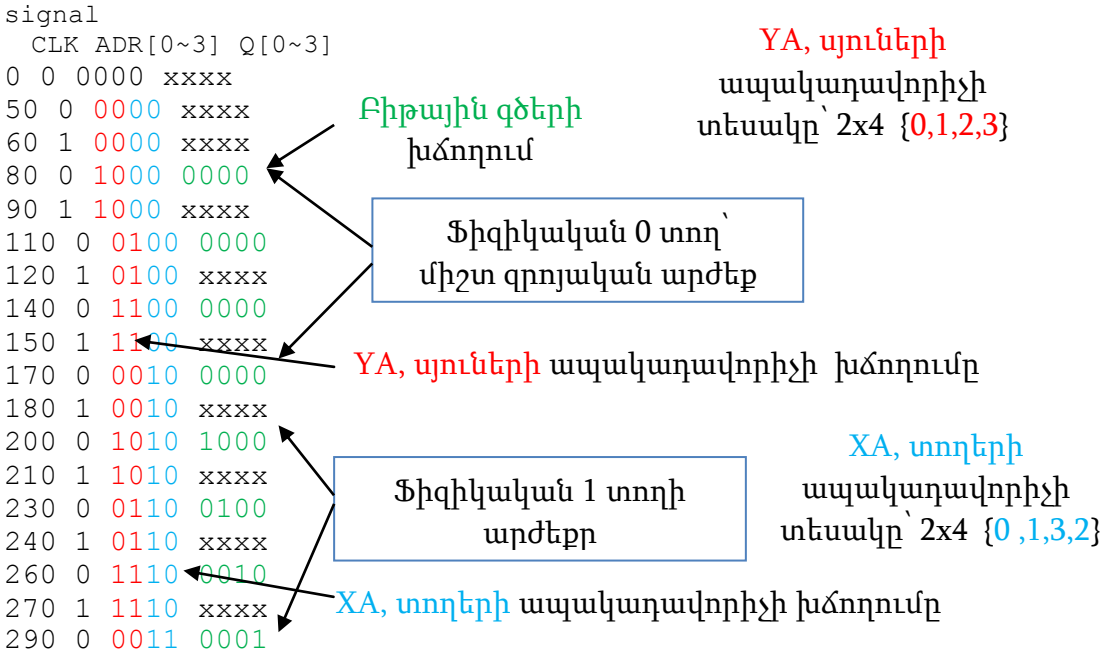
```
1 1 1 1 . . . 1 1
0 1 1 1 . . . 1 1
. . . . . . . . .
. . . . . . . . .
0 0 0 0 . . . 1 1
0 0 0 0 . . . 0 1
```

Նկար 2.11. ASV ԱՀ-ի հիմնական անկյունագծային ստուգող հավաքածուն

scramble.tcl ֆայլում բերում է մոդելավորման սխալ արդյունքի։ Կախված նուշի ֆիզիկական չափսերից՝ անհրաժեշտություն է առաջանում մեկ նմուշի համար կիրառել մեկից ավելի ծրագրավորման ֆայլեր։ Հաշվարկները ցույց տվեցին, որ մեկ նմուշի խճողումների լիովին ստուգման համար, վատագույն դեպքում, բավական է ունենալ երեք ծրագրավորող հավաքածու։ Հավելված 2-ում տրված է SIV հոսքի՝ իրական հիշողության նմուշի (NW=64, NB=8, CM=4, PR=16, PC=32) IPP-ի ֆայլի օրինակը։

Ինչպես երևում է ASV ստուգման բլոկ-սխեմայից (Տես՝ նկ. 2.9), MPT գործիքը ծրագրավորում է նմուշի GDSII ֆայլերը՝ այդ նմուշների IPP-ի ֆայլերին համաձայն։

Այնուհետև, անցկացնելով յուրաքանչյուր ծրագրավորված GDSII ֆայլը LVS ծրագրով ստանում ենք ծրագրավորված GDSII–ին համապատասխան SPICE ֆայլը, որը հետագայում և ենթարկվում է մոդելավորմանը: Մոդելավորումը կատարվում է համաձայն SVP ֆայլի, որը գեներացնում է VPG ծրագիրը, և որը իր մեջ ծածկագրված ձևով պարունակում է scramble.tcl ֆայլից ստացված խճողումների մասին ինֆորմացիան: SVP ֆայլում նկարագրվում են մոդելավորման ընթացքում փոփոխվող ազդանշանները և այդ ազդանշանների արժեքների փոփոխումը, որը կատարվում է մոդելավորման ժամանակային տիրույթում: Իսկ հիշողության սարքի մնացած ազդանշանները մնում են անփոփոխ և նկարագրվում են մոդելավորման պարամետրերի՝ SIM ֆայլում: ASV ստուգման ընթացքում կատարվում է միայն «կարդալ» (հիշողության սարքից) գործողության մոդելավորումը:



Նկար 2.12. ԱՀ-Ի SVP ֆայլի օրինակը և դաշտերի մեկնաբանումը

Ներկայացված SVP ֆայլի օրինակում (նկ. 2.12), փոփոխման են ենթարկվում CLK, ADR[0~3] և Q[0`3] ազդանշանները, ընդ որում հասցեների (ADR[0~3]) փոփոխման հերթականությունը կատարվում է սյուների և տողերի ապակոդավորիչների խճողմանը համաձայն, իսկ ելքային ազդանշանները (Q[0~3]) նկարագրված են բիթային գծերի, այդ գծերի և սյուների ոլորապտումների խճողումներին համաձայն: Լավագույն դեպքում, երբ scramble.tcl ճիշտ է նկարագրված, մոդելավորման արդյունքը սխալի չի բերում: Հակառակ դեպքում մենք ստանում ենք մոդելավորման համեմատության

սխալ/սխալներ, որի պատճառը համապատասխան խճողման սխալ նկարագրություն է scramble.tcl ֆայլում: Վերացնելով այդ սխալը scramble.tcl ֆայլում և նորից աշխատեցնելով ASV ստուգումը, մոդելավորման արդյունքում էլ սխալ չենք հայտնաբերի, ինչը նշանակում է, որ կոմպիլյատորի խճողումները ճիշտ են նկարագրված scramble.tcl ֆայլում՝ ASV ստուգումը ավարտված է:

SIV ԱՀ-ի և *ԱՀ-ի* հանգույցների աշխատանքի ամբողջական ղեկավարումը և կարգաբերումը կատարվում է ղեկավարող տեքստային ֆայլի միջոցով: Այն կառավարման վահանակի դեր է կատարում ողջ SIV ԱՀ-ի համար և իր մեջ ընդգրկում է յոթ բաժիններ (Տես՝ օրինակը Հավելված 3 -ում).

1. GENERAL_PARAMETERS – ԱՀ-ի, ASV, SDV հանգույցների, LVS և SPICE մոդելավորման գործիքների ղեկավարումը և կարգաբերումը ապահովող՝ *ընդհանուր* պարամետրերի բաժին,
2. CFG_PARAMETERS – նմուշների գեներացման համար անհրաժեշտ՝ *մասնակի* պարամետրերի բաժին,
3. SIMULATION_PARAMETERS – SPICE մոդելավորող գործիքի՝ *մասնակի կառավարման* պարամետրերի բաժին,
4. PINS_PARAMETERS – այս բաժինը պարունակում է «կարդալ» գործողության հիմնական ազդանշանների անունները: Ընդհանուր դեպքում SIV ԱՀ-ը ավտոմատ ձևով ստանում և ղեկավարում է նմուշից «կարդալ» գործողությունը իրականացնող ազդանշանները, բայց մասնակի (ոչ ստանդարտ կոմպիլյատորների) դեպքերում անհրաժեշտ է լինում նշել այդ ազդանշանների անունները:
5. MPT_PARAMETERS – MPT գործիքը կառավարող պարամետրերի բաժին,
6. HSIM_PARAMETERS -SPICE մոդելավորող ծրագրի՝ հարակից պարամետրերի բաժին,
7. TIE_OFFS – այս բաժինը նախատեսված է հատուկ ֆունկցիոնալ նշանակու-թյան ազդանշանների տրամաբանական արժեքները նկարագրելու համար:

2.2 Հիշողության նմուշի կառուցվածքային մոդելը GDSII ձևաչափից դուրս բերման ավտոմատացված համակարգ

Առաջին գլխում մենք նկարագրեցինք հիշողության խճողումների տեսակները և հիշողության կոմպիլյատորի կառուցվածքային մոդելը նկարագրող ֆայլը և այդ ֆայլի

ստուգման ծրագրային ԱՀ-ը: Խճողումները նկարագրող ֆայլը ստեղծվում է կոմպիլյատորը նախագծող ճարտարագետի կողմից կոմպիլյատորի ստեղծման հետ միաժամանակ: Ներկայումս հիշողության կառուցվածքային մոդելը պարտադիր մաս է հանդիսանում բոլոր տեսակի ներդրված ՆՀՍ-ի համար: Այն օգտագործվում է ՆՀՍ-ի հետ աշխատող բոլոր ավտոմատ ծրագրային միջոցների կողմից: Ներդրված ՀՍ-երի կառուցվածքային մոդելը գեներացվում է ՀՍ-ը նախագծողի կողմից (manual method) մի շարք ծրագրային գործիքների օգնությամբ՝ մասամբ ավտոմատ ձևով:

2.2.1 Մոդելի ավտոմատ ստացման ծրագրային համակարգի նկարագրությունը և ֆունկցիոնալ կառուցվածքը

Կառուցվածքային մոդելի նախագծողի կողմից գեներացված մեթոդի բացասական կողմերից են՝ 1) ճարտարագետից պահանջվում է ունենալ բարձր որակավորում և սպառիչ, լիարժեք ինֆորմացիա ՀՍ-ի կառուցվածքի մասին, 2) ստացված մոդելը պարտադիր կերպով պետք է անցնի ստուգման (վավերացման) փուլ, որը նույնպես հանգեցնում է լրացուցիչ ծախսերի: Այս գլխում նկարագրվող կառուցվածքային մոդելի ավտոմատ ստացման մեթոդը զուրկ է վերոնշյալ թերություններից, բայց պահանջում է ԱՀ-ի ծրագրային միջոցների՝ ԱՀ-ի միջավայրի ստեղծում: Իրականում կան պատվիրատուներ, որոնք ունեն հիշողության նմուշներ, բայց չունեն կիրառական ծրագրերում օգտագործվող խճողումները նկարագրող ֆայլը և միաժամանակ չունեն այն ստանալու համապատասխան ռեսուրսներ: Այդ դեպքում անհրաժեշտ է դառնում ունենալ մի գործիք, որը հնարավորություն կընձեռի ավտոմատ ձևով, դուրս բերել նմուշի կառուցվածքային մոդելը GDSII ձևաչափից: Այս գլխում տրվում է՝ հեղինակի մասնակցությամբ նախագծված և իրականացված, հիշողության նմուշի խճողումների ինֆորմացիան ավտոմատ դուրս բերելու (անգլերեն՝ Structural Information Extractor (SIE) flow) ծրագրային ԱՀ-ի նկարագրությունը [14]:

Իրականցման տեսակետից այս խնդրի դրվածքը ուղիղ հակադարձ է հիշողության կառուցվածքային մոդելի ստուգման խնդրին: Եթե SIV-ի ժամանակ ունենք նմուշի GDSII ձևաչափը, և այն նկարագրող մոդելը և խնդիր է դրվում մեկ-երկու՝ նախապես արդեն հայտնի, ծրագրավորող հավաքածուների միջոցով ստուգել մոդելի համապատասխանությունը այդ նմուշի գրաֆիկական ներկայացմանը, ապա SIE ԱՀ-ի

գաղափարն է՝ յուրաքանչյուր նմուշի դեպքում *ստանալ* ծրագրավորող օրինակների վերջնական հավաքածու, որոնց միջոցով

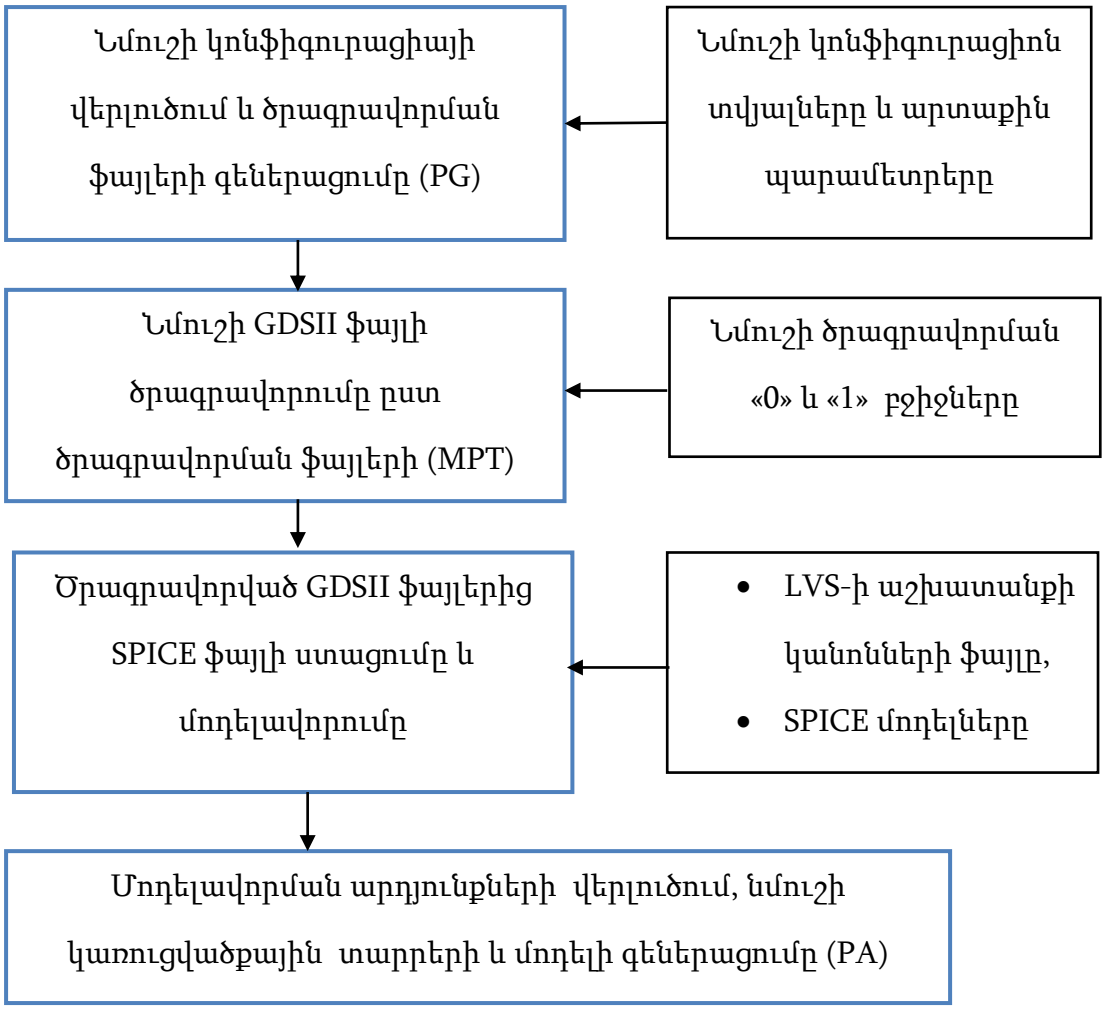
ա) ծրագրավորելով GDSII ֆայլը՝ ստանալ ծրագրավորված GDSII ֆայլեր,

բ) ծրագրավորված GDSII ֆայլերից գեներացնել SPICE ֆայլեր,

գ) մոդելավորել այդ SPICE ֆայլերը,

դ) SPICE մոդելավորման արդյունքները մշակելուց հետո ստանալ նմուշի խճողումների մասին տեղեկությունները և

ե) գեներացնել նմուշի կառուցվածքային մոդելը:



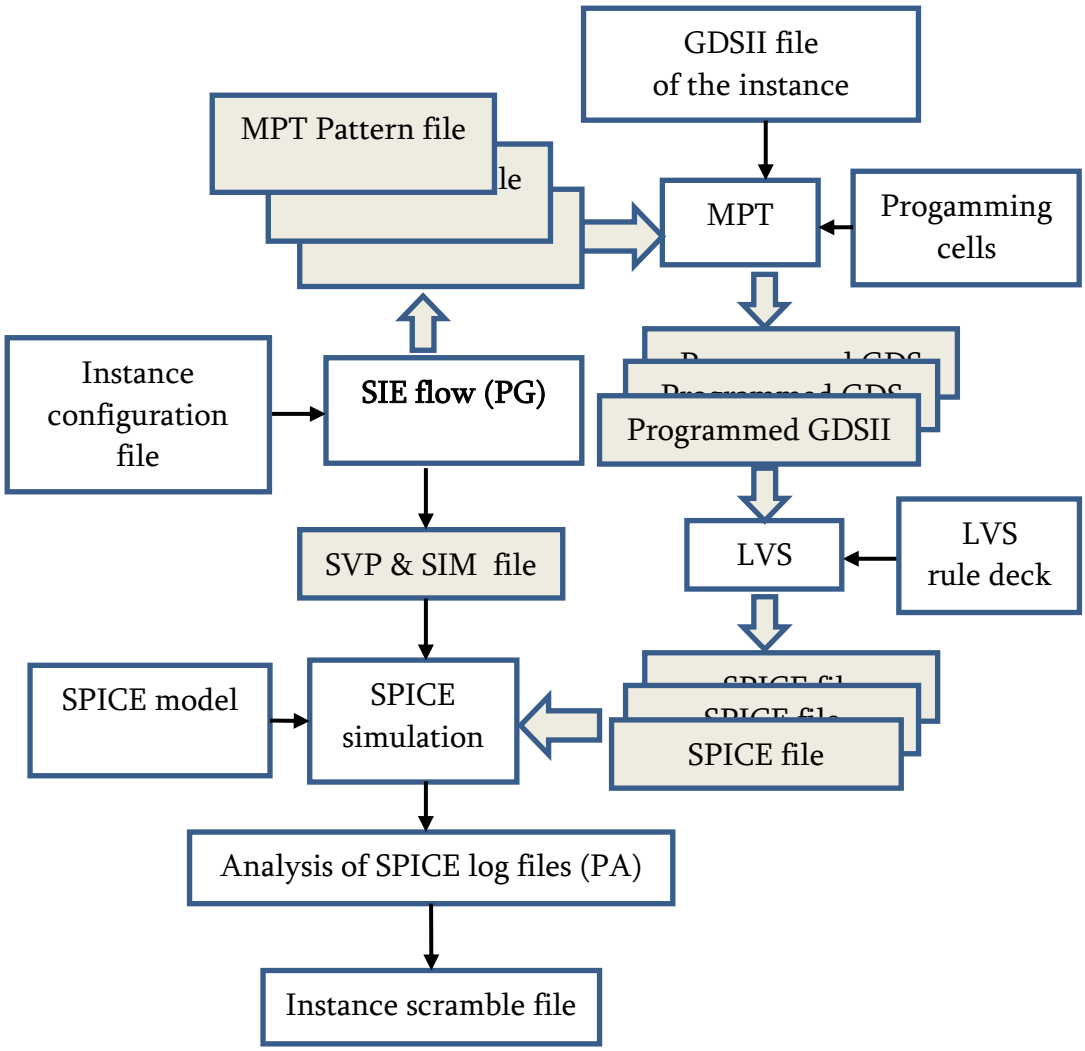
Նկար 2.13. Կառուցվածքային մոդելը դուրս բերող ծրագրային ԱՀ-ի ֆունկցիոնալ սխեման

Նկար 2.13-ում տրված է SIE ԱՀ-ի ֆունկցիոնալ բլոկ-սխեման: Ֆունկցիոնալ սխեմայից երևում է, որ SIE-ում օգտագործվում են SIV ԱՀ-ի ծրագրային գործիքներից մի

քանիսը՝ MPT, LVS և SPICE մոդելավորման գործիքները: SIE ԱՀ-ում, նշված ծրագրային գործիքներին, ավելացվել են ևս երկուսը՝ PG և PA պրոցեդուրաները:

- Pattern Generater (PG) – կատարում է նմուշի կոնֆիգուրացիայի վերլուծությունը և ծրագրավորման ֆայլերի գեներացումը
- Pattern Analyzer (PA) - կատարում է մոդելավորման արդյունքների վերլուծությունը և կառուցվածքային տարրերի դուրս բերումը (բացահայտումը):

SIE ԱՀ-ի աշխատանքային բլոկ-սխեման պատկերված է նկար 2.14-ում: Հարկ է նշել, որ հիշողության նմուշի կառուցվածքային մոդելը իրենից ներկայացնում է



Նկար 2.14. Կառուցվածքային խճողումների դուրս բերման SIE ԱՀ-ի աշխատանքի բլոկ-սխեման

տեքստային ֆայլ, որը պարունակում է նմուշի կառուցվածքային տարրերի՝ նմուշի խճողումների և գոտիների բաշխման նկարագրությունը: ՍՊԴՀ-երի կանոնավոր կառուցվածքը այդ սարքերի հիմնական հատկություններից է, որը մեծ խոչընդոտ է

հանդիսանում կառուցվածքային խճողումների ավտոմատ դուրսբերման համար: Այդ խոչընդոտը շրջանցելու նպատակին է ծառայում հիշողության բջիջների «ծրագրավորման», այսինքն՝ բջիջների անորոշ պարունակությունը հարկադրաբար, նախապես որոշված տրամաբանությամբ հաստատուն զրո կամ մեկ արժեքներից որևէ մեկի մեջ ձևափոխելու գաղափարը [14]: Այն հնարավորություն է տալիս, պայմանականորեն ասած, վերացնել հիշողության զանգվածի բջիջների անորոշ վիճակը և ստանալ հիշողության նմուշի խճողումների միանշանակորեն նույնականացման հնարավորություն: Ծրագրավորող բջիջների աշխատանքը և տոպոլոգիան լիովին համընկնում են SIV ԱՀ-ում օգտագործված և արդեն 2.1.2 գլխում ներկայացված (Տես՝ նկարներ 2.3 և 2.4) ծրագրավորման բջիջներին:

SIE ԱՀ-ի աշխատանքը ապահովելու համար նախագծվել են հետևյալ լրացուցիչ երկու գործիքները, որոնք կատարում են հետևյալ գործողությունները.

- Pattern Generater գործիքը որպես մուտք ընդունում է նմուշի կոնֆիգուրացիայի ֆայլը, որի հիման վրա գեներացնում է նմուշի ֆիզիկական կառուցվածքին համապատասխանող ծրագրավորման ֆայլերը:
- Pattern Analyzer գործիքը որպես մուտք ստանում է SPICE մոդելավորման ընթացքում ստացված արդյունքների ֆայլերը, որոնց վերլուծելուց հետո ստանում է հիշողության նմուշում առկա խճողումները նկարագրող ֆայլը:

ՀԿՄ-ի ստանալու նպատակով նախագծված ծրագրային ԱՀ-ը կատարում է հետևյալ գործողությունների հաջորդականությունը (նկ. 2.14).

1. Նմուշի կոնֆիգուրացիայի վերլուծումը, որի արդյունքում գեներացվում են նմուշի ծրագրավորման ֆայլերը (IPP): Գեներացված IPP-երը ունեն տեքստային ձևաչափ և պարունակում են տեքստային տեղեկատվություն հիշողության մեջ ծրագրավորվող օրինակների (ԾՕ) (programmed pattern) մասին: Այդ օրինակների քանակը և տեսքը փոփոխական է և կախված է հիշողության նմուշի ֆիզիկական չափերից: Լավագույն տարբերակում այդ քանակը հավասար է երկուսի՝ զրոյական և անկյունագծային, իսկ վատագույն դեպքում կարող է հասնել մինչև ութի: Լավագույն դեպք է համարվում հիշողության նմուշի այն ֆիզիկական կառուցվածքը, երբ հիշողության բջիջներով զբաղեցված մակերևույթը քառակուսի է՝ ֆիզիկական տողերի և սյուների քանակները նույն են: Այս դեպքում հնարավոր է դառնում մեկ ծրագրավորող

«անկյունագծային» օրինակով ծածկել հիշողության ողջ մակերևույթը: Նկար 2.15-ում ներկայացված են զրոյական և անկյունագծային ԾՕ-ները 8 տող և 8 սյուն պարունակող հիշողության նմուշի համար: Զրոյական ԾՕ-ը պարտադիր օրինակ է հանդիսանում նմուշների բոլոր կառուցվածքների համար:

00000000	11111111
00000000	01111111
00000000	00111111
00000000	00011111
00000000	00001111
00000000	00000111
00000000	00000011
00000000	00000001

ա. զրոյական բ. անկյունագծային

Նկար 2.15. Ծրագրվորման ֆայլերի օրինակները

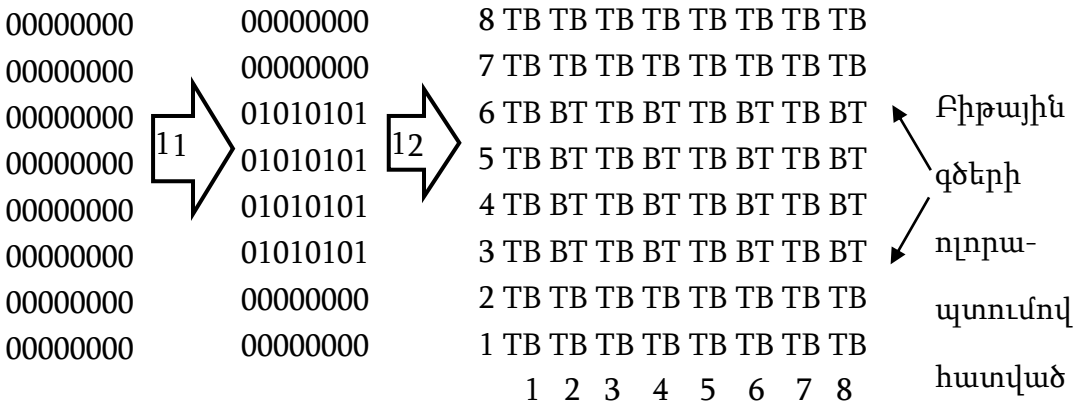
2. Օգտագործելով ստացված ԾՕ-ները՝ SIE ծրագրային ԱՀ-ի MPT գործիքի միջոցով, կատարում է նմուշի գրաֆիկական GDSII ֆայլի ծրագրավորում ԾՕ-րի համաձայն, որի արդյունքում ստացվում են ԾՕ-րի քանակությամբ ծրագրավորված GDSII ֆայլեր:

3. Օգտվելով SPICE ձևաչափի ֆայլեր ստանալու կանոններից՝ (LVS rule deck) LVS գործիքը, մշակելով ծրագրավորված GDSII ֆայլերը, ստանում է յուրաքանչյուր ծրագրավորված նմուշին համապատասխանող SPICE ֆայլերը: Հաջորդ քայլում կատարում են այդ ֆայլերի մոդելավորումը: Մոդելավորման ընթացքում իրականացվում է հիշողության նմուշից միայն «կարդալ» գործողությունը, որի արդյունքում ստացված և համապատասխան ձևով մշակված տեքստային ֆայլերը պարունակում են նմուշի մոդելը ստանալու համար անհրաժեշտ կառուցվածքային տարրերի, տիպերի մասին ինֆորմացիան:

4. SIE-ի ԱՀ-ը, վերլուծելով մոդելավորման արդյունքները և շարժվելով պարզից դեպի բարդը, քայլ առ քայլ դուրս է բերում կառուցվածքային մոդելի տարրերը: Այս փուլում SIE-ի MPT գործիքի օգնությամբ ավտոմատ ձևով հետազոտում է GDSII ֆայլը, ստանում նմուշի գոտիների բաշխումը և գեներացնում գոտիների նկարագրությունը կառուցվածքային մոդելում: Հետագայում գոտիների բաշխման մասին

տեղեկությունները օգտագործվում են ԱՀ-ում (մասնավորապես գոտիների դիրքերի բաշխումը) ստացված մոդելի վերջնական ինքնաստուգումը կատարելիս: SIE-ի երկրորդ քայլում ԱՀ-ը գեներացնում է զրոյական արժեքով ԾՕ (Տես՝ նկ. 2.15 ա.) և կատարում մշակումը՝ 2 և 3 քայլերը, այդ զրոյական ծրագրային օրինակի համար: Զրոյական ԾՕ-ը օգտագործվում է որպես առաջնահերթ ԾՕ, քանի որ այն հնարավորություն է տալիս միանշանակորեն բացահայտել նմուշի զանգվածում. ա) բիթային գծերի բաշխումը, բ) բիթային գծերի ոլորապտումների առկայությունը, և եթե այդ ոլորապտումները կան, ապա այդ ոլորապտումների դիրքը և տեսակը:

Մանրամասնենք բիթային գծերի բացահայտման գործընթացը: Ինչպես արդեն ներկայացրել ենք առաջին գլխում, հիշողության զանգվածում տոպոլոգիայում բջջի գծերի դասավորվածությունը կարող է ունենալ {T, B} կամ էլ {B, T} արժեք: Հիշողության զանգվածի սյուներում և տողերում երկու հարևան բջիջները համապատասխանաբար կարող են ստանալ բիթային գծերի հնարավոր չորս բաշխման տեսակներից որևէ մեկը՝ 1. {T B, T B}, 2. {B T, B T}, 3. {T B, B T}, և 4. {B T, T B}: ՀՍ-երի կանոնավոր ձևով կրկնվող կառուցվածքը ենթադրում է, որ տողերում և սյուներում բիթային գծերի բաշխումը բաղկացած կլինի նշված չորս տեսակների որևէ մեկի կրկնությունից: Այդ կրկնությունների քանակը համապատասխանաբար հավասար է նմուշում սյուների (#PC) և տողերի (#PR) քանակին: Ընդունենք, որ բիթերի {T, B} բաշխումը համապատասխանում է հիշողության բջիջում «0» արժեքին, ապա զրոյական ծրագրավորման օրինակը (նկ. 2.16 ա.) բջիջների ողջ զանգվածում



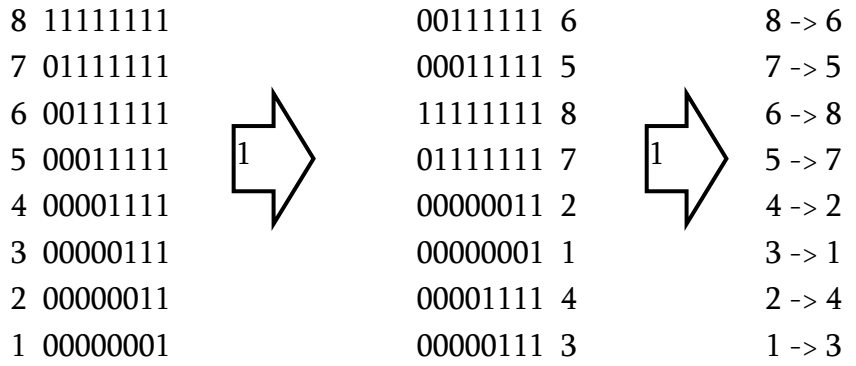
ա. զրոյական ԾՕ բ. մոդելավորման արդյունքր և վերծանումր

Նկար 2.16. Զրոյական Ծրագրավորման օրինակը

համապատասխանում է բիթային գծերի {T B} ֆիզիկական բաշխվածությանը:

Այդպիսով, եթե ծրագրավորելով հիշողության զանգվածը «0» արժեքներով հետագայում՝ մոդելավորման արդյունքում ստանանք, որ որևէ բջիջը ունի «1» արժեքը, ապա պարզ է, որ այդ բջջի բիթային գծերը ունեն {B, T} բաշխում, այլ ոչ թե {T, B}, ինչպես ենթադրում էինք ծրագրավորման սկզբում:

Նկար 2.16-ում տրված է զրոյական ԾՕ-ը և զրոյական ծրագրավորման օրինակի մոդելավորման արդյունքի բիթային գծերի վերծանումը 8 տող և 8 սյուն ունեցող հիշողության զանգվածի համար: Ինչպես երևում է նկարից, ԾՕ-ը առաջին քայլից հետո (GDSII-ի ծրագրավորում, SPICE ֆայլի դուրս բերում և SPICE ֆայլի մոդելավորում) պարզվում է, որ բիթային գծերի բաշխումը տարբերվում է «զրոյականից»: Երկրորդ քայլում, «0» և «1» արժեքների փոխարեն տեղադրելով համապատասխանաբար իրենց բիթային գծերի տեսակների արժեքները, կստանանք բիթային գծերի բաշխումը զանգվածում: Օրինակից երևում է, որ զանգվածում՝ զույգ համարներ ունեցող բիթային գծերում առկա է ոլորապտում ֆիզիկապես 2-ից դեպի 3-ը և 6-ից դեպի 7 տողերի անցման հատվածներում: Այսպիսով, բիթային գծերի խճողման մասին ստացված տեղեկությունը հետագայում համապատասխան ձևաչափով կներկայացվի նմուշի կառուցվածքային մոդելի ֆայլում: Զրոյական ԾՕ-ի մշակման արդյունքները հաշվի են առնվում այլ ծրագրավորման օրինակների գեներացման ընթացքում:



ա. անկյունագծային ԾՕ-ր

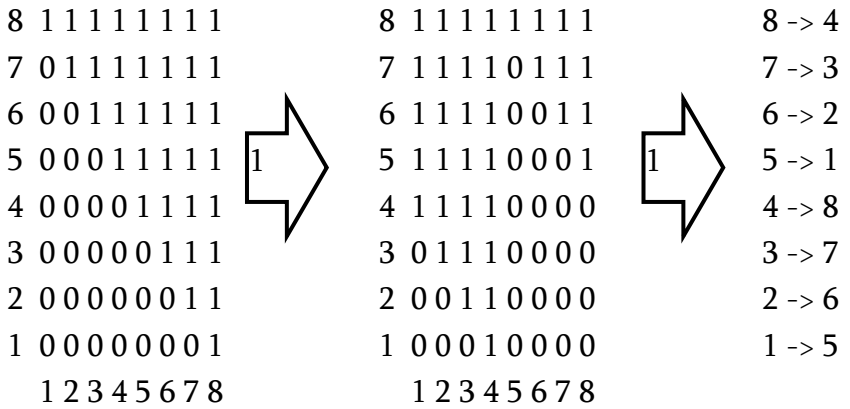
բ. մոդելավորման արդյունքը

գ. նմուշի տողերի խճողումը

Նկար 2.17. Անկյունագծային ԾՕ-ի կիրառման օրինակը՝ նմուշի տողերի խճողման դեպքում

Հաջորդ հիմնական ԾՕ է հանդիսանում անկյունագծային ԾՕ-ը: Նկար 2.17-ում տրվում է անկյունագծային ԾՕ-ի (նկ. 2.17 ա.) միջոցով, տողերի խճողման

բացահայտման օրինակ: Անկյունագծային ԾՕ-ի հատկությունն է, որ զրոների (մեկերի) քանակը յուրաքանչյուր տողի (սյան) մեջ յուրահատուկ է՝ չի կրկնվում ողջ զանգվածում: Այս հատկությունը հնարավորություն է ընձեռում միանշանակորեն բացահայտել նմուշի տողերի (սյուների) տրամաբանականից դեպի ֆիզիկականը անցման խճողումները: MPT-ի գործիքը կատարում է նմուշի GDSII ֆայլի ծրագրավորումը անկյունագծային օրինակով, որից հետո մոդելավորման արդյունքը տող առ տող մշակվում է PA պրոցեդուրայի միջոցով, արդյունքում ստացվում է նմուշի տողերի խճողման (Տես՝ նկ. 2.17 գ.) նկարագրությունը, որը հետագայում համապատասխան ձևաչափով կներկայացվի նմուշի կառուցվածքային մոդելի ֆայլում: Հաջորդ քայլում կատարվում է



ա. անկյունագծային ԾՕ-ը

բ. մոդելավորման արդյունքը

գ. նմուշի սյուների խճողումը

Նկար 2.18. Անկյունագծային ԾՕ-ի կիրառման օրինակը՝ նմուշի սյուների խճողման դեպքում

սյուների խճողման դուրս բերումը, նույն անկյունագծային ԾՕ-ի օգնությամբ: Ի տարբերություն տողերի խճողման բացահայտմանը՝ այս դեպքում, PA գործիքը մոդելավորման արդյունքի մշակումը կատարում է սյուների ուղղությամբ (ներքևից վերև) և արդյունքում ստանում է նմուշի սյուների խճողումը: Նկար 2.18-ում պատկերված է՝ անկյունագծային ԾՕ-ի միջոցով նմուշի սյուների խճողման դուրս բերման գործընթացի պարզաբանման օրինակը:

2.2.2 Կառուցվածքային մոդելի ավտոմատացված ձևով ստացման հետ կապված բարդությունները

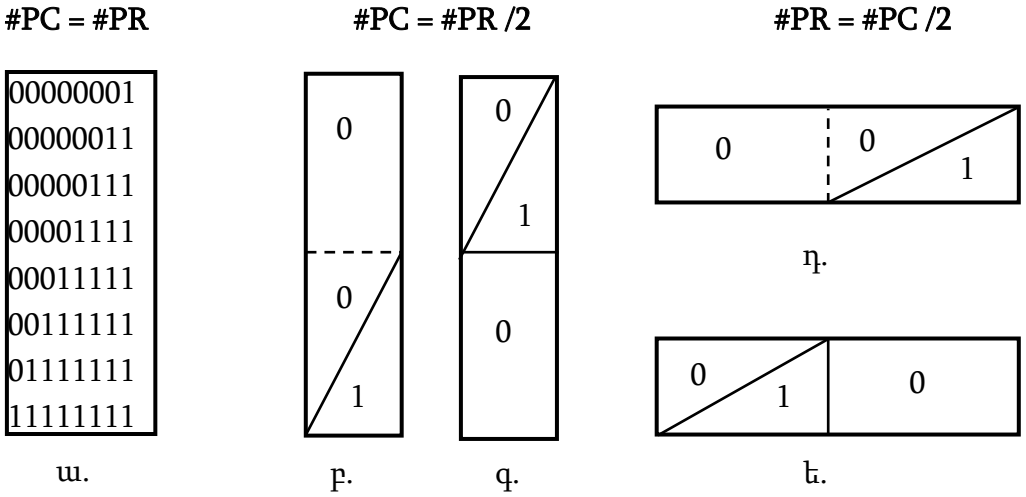
Հասկանելի է, որ հիշողության նմուշում միաժամանակ մի քանի տեսակի խճողումների առկայությունը էականորեն բարդացնում է PA գործիքի կողմից կատարվող մոդելավորման արդյունքների վերծանման՝ խճողումների կառուցվածքի բացահայտման, աշխատանքները: Այդ պատճառով, SIE ԱՀ-ում իրականցված ալգորիթմը հիմնված է հետևյալ պարզ հիմունքների վրա.

- Խճողումների դուրսբերման և բացահայտման ժամանակ կիրառել պարզից դեպի բարդ սկզբունքը՝ սկզբում ստանալով նմուշի հեշտ վերծանվող խճողման մասին տեղեկատվությունը, օգտագործել այն հաջորդ քայլում բացահայտվող խճողման վերծանման ժամանակ: Ինչպես արդեն նշվեց, առաջին քայլով PA գործիքը՝ օգտագործելով զրոյական ծրագրավորման օրինակը, ստանում է նմուշի բիթային գծերի խճողումը;
- Յուրաքանչյուր խճողումը բացահայտելու համար կիրառել առանձին՝ լրացուցիչ անհատական, ծրագրավորման օրինակ:

Պարզ է, որ այս դեպքում PA գործիքը մոդելավորման արդյունքները մշակում է հաջորդաբար, որը ԱՀ-ի աշխատանքի ժամանակի ծախսման տեսակետից օպտիմալ չէ: Բայց այդ հաջորդական մշակման շնորհիվ, հնարավոր է դառնում. ա) ապահովել մշակման արդյունքների միանշանակությունը, և, որ շատ կարևոր է, բ) բացառել PA աշխատանքի ընթացքում անորոշ իրավիճակների ի հայտ գալը, որը տեսականորեն հնարավոր է զուգահեռ մշակման դեպքում:

SIE ԱՀ-ում մեկ այլ կարևոր խնդիր է կիրառվող ԾՕ-ների քանակի լավարկումը: Զրոյական և անկյունագծային ԾՕ-ները (Տես՝ նկ. 2.17, նկ. 2.18) հիմնական ԾՕ-ներ են հանդիսանում այն հիշողության նմուշների դեպքում, որոնց հիշողության բջիջների զանգվածի երկրաչափական տեսքը քառակուսի է (այսինքն՝ զանգվածում տողերի (#PR) և սյուների (#PC) քանակները համընկնում են՝ #PC = #PR) կամ էլ մոտ է քառակուսիին: Այդ դեպքերում հնարավոր է դառնում կառուցել անկյունագծային ԾՕ, որը լիովին կընդգրկվի հիշողության զանգվածը պատկերող մակերևույթում (Տես՝ նկ. 2.19 ա.) և այդ դեպքում՝ PG գործիքով գեներացված ԾՕ-ների քանակը մոտ կլինի նվազագույնին:

Հակառակ դեպքերում, բացի հիմնական անկյունագծային ԾՕ-ից՝ գեներացվում են նաև լրացուցիչ ԾՕ-ներ: Օրինակ՝ երբ տողերի քանակը մեծ է սյուների քանակից #PR > #PC (Տես՝ նկ. 2.19 բ., գ.), կամ երբ սյուների քանակը մեծ է տողերի քանակից #PC > #PR (Տես՝ նկ. 2.19 դ., ե.):



ա. բ. գ. դ. ե.

Նկար 2.19. ԱՀ-ի անկյունագծային ԾՕ-ի կիրառման օրինակները

Իսկ երբ զանգվածի տողերի և սյուների հարաբերությունը մեծ է երեքից, ապա այդ դեպքերի համար նախատեսված է կիրառել ոչ անկյունագծային՝ լրացուցիչ, ծրագրավորող օրինակներ: Այդ ԾՕ-ների կիրառումը նախատեսված է միայն մեծ չափսեր ունեցող հիշողության նմուշների դեպքերում (այդ պատճառով այս դեպքի օրինակը ներկայացնելը դժվար է): Ոչ անկյունագծային ծրագրավորման օրինակները՝ ֆիզիկական մակարդակում, յուրաքանչյուր տողին (իսկ #PC > #PR դեպքում՝ սյունին) վերագրում են չկրկնվող կոդ, այսպես ասած, «համարակալում» են տողերը: Ոչ անկյունագծային ԾՕ-ները կիրառվում են միայն անկյունագծային ԾՕ-ներից հետո որպես լրացուցիչ ծրագրավորման օրինակներ: SIE-ի հոսքում բացի պարտադիր զրոյական և անկյունագծային ԾՕ-ներից, կարող են գեներացվել առավելագույնը մինչև վեց լրացուցիչ ԾՕ-ներ: ՀՍ-ի նմուշի լրացուցիչ ԾՕ-ի ֆայլի օրինակը ներկայացված է հավելված 4-ում:

Ինչպես և SIV-ի դեպքում՝ SIE ծրագրային ԱՀ-ի աշխատանքի կառավարումը կատարվում է ղեկավարման ֆայլի միջոցով: Հավելված 5 –ում ներկայացված է SIE ծրագրային ԱՀ-ի ղեկավարման ֆայլի օրինակը:

2.3 Փորձնական տվյալները

2.3.1 SIV ավտոմատացված համակարգի օգտագործման փորձնական տվյալները

SIV-ի ԱՀ-ը օգտագործվել է և, ներկայումս շարունակվում է օգտագործվել, որպես Synopsys ընկերության կողմից նախագծվող բոլոր տեսակի հիշողության կոմպիլյատորների կառուցվածքային տարրերի խճողումները նկարագրող՝ scramble.tcl, ֆայլերի ճշտությունը ստուգող հուսալի գործիք: SIV-ի ԱՀ-ը հաջողությամբ կիրառվում է մեկ և երկու պորտեր ունեցող, մեծ արագագործության և բարձր խտության, տարբեր չափսերի նմուշների գեներացումը ապահովող հիշողության կոմպիլյատորների համար (800-ից ավելի):

Աղյուսակ 2.1

SIV-ի հոսքի օգտագործման փորձնական տվյալները

ՍՊԴԿ կոմպիլյատոր	SDV կոնֆիգուրա-ցիաների քանակը	SDV GDSII ֆայլերի գեներացման ժամանակը (զուգահեռացման գործակից=16)	SDV-ի մշակման ընդհանուր ժամանակը
ts90nm 1p11HS 512K	84	մոտ 2 ժամ	մոտ 3 ժամ
ts90nm 1p11HD 512K	67	մոտ 2.2 ժամ	մոտ 2.4 ժամ
ts65nm 1p11HS 512K	165	մոտ 4.5 ժամ	մոտ 4.5 ժամ
ts65nm 2p22HD 512K	120	մոտ 3.5 ժամ	մոտ 4.2 ժամ
ts45nm 1p11 512K	263	մոտ 4.4 ժամ	մոտ 5.5 ժամ
ts32nm 1p11 1M	455	մոտ 5.2 ժամ	մոտ 6.3 ժամ
ts28nm 2p22 1M	630	մոտ 6.3 ժամ	մոտ 7 ժամ
ts22nm 1p11 1M	750	մոտ 7.1 ժամ	մոտ 8.2 ժամ
ts14nm 1p11 1M	810	մոտ 8.5 ժամ	մոտ 10 ժամ

Հարկ է նշել, որ ժամանակակից հիշողության կոմպիլյատորների SDV ստուգման կոնֆիգուրացիաների քանակը հասնում է մի քանի հարյուրների (Տես՝ աղ. 2.1), ուստի

շատ կարևոր է դառնում հնարավորինս ավտոմատացնել, արագացնել GDSII ֆայլերի գեներացումը և մշակումը: Այդ նպատակով SIV-ում իրականացված է նմուշների գեներացման աշխատանքների զուգահեռ կատարումը: Բացի այդ, իր աշխատանքի սկզբում նախապես համեմատվում և գեներացման հերթից դուրս են հանվում արդեն գեներացված կոնֆիգուրացիաները: Ցավոք, ներկայումս անհնար է լիովին ավտոմատացնել արդյունքների վերջնական մշակման աշխատանքները, այն կարելի է պարզեցնել, հեշտացնել [27-29]՝ առավելագույնս նվազեցնելով մեխանիկորեն մշակվող ինֆորմացիայի քանակը և տրամադրելով մշակմանը անհրաժեշտ ինֆորմացիան առավելագույնս ընթերցելի տեսքով: Աղյուսակ 2.1-ից երևում է, որ GDSII ֆայլերի գեներացման ժամանակի և այդ ֆայլերի քանակի աճման կախվածությունը գծային չէ: Դա բացատրվում է այն բանով, որ գեներացումը կատարող՝ ժամանակակից (32 նմ, 22 նմ, 14 նմ տեխնոլոգիաների համար օգտագործվող) համակարգչային ռեսուրսները ավելի հզոր են, քան նախկինում (90 նմ, 60 նմ տեխնոլոգիաների համար) օգտագործվող ռեսուրսները:

Աղյուսակ 2.2-ում տրված են հիշողության կոմպիլյատորների բնութագրերը և դրանց համապատասխան ASV ալգորիթմի աշխատանքի ժամանակային գնահատականները, որտեղ *«K» դա –կիլոբիթ է իսկ «M» - մեգաբիթ:*

Աղյուսակ 2.2

ASV ալգորիթմի աշխատանքի տևողությունը

ՍՊԴԿ կոմպիլյատոր	նմուշների չափսերի տիրույթը	ստուգվող նմուշների քանակը	ստուգող հավաքա-ծուների թիվը	ալգորիթմի աշխատանքի ժամանակ
ts90nm 1p11 HS 8M	256K – 8M	5	18	22 ժամ
ts90nm 1p11 HS 512K	512 – 512K	6	12	3.5 ժամ
ts90nm 2p22 HS 512K	512 – 512K	7	12	6.2 ժամ
ts90nm 1p11 HD 512K	512 – 512K	4	14	2.3 ժամ
ts90nm 2p22 HD 512K	512 – 512K	4	16	5.5 ժամ
ts90nm 2p11 HD 32K	64 – 32K	3	12	45 րոպե
ts90nm 2p11 HS 16K	64 – 16K	3	12	40 րոպե

Փորձնական տվյալները ցույց տվեցին, որ SPICE մոդելավորման արդյունքները կախված չեն մոդելավորման ընթացքում կատարվող հաշվարկների ճշտությունից, այսինքն՝ մոդելավորման ժամանակ բավական է օգտագործել ամենացածր ճշտությունը ապահովող մոդելավորման գործակիցը, դրանով զգալիորեն նվազեցնելով ASV ստուգումների տևողությունը:

SIV ԱՀ-ի ողջ ծրագրավորումը իրականացված է TCL ծրագրավորման լեզվով հեղինակի մասնակցությամբ: Իսկ VIG, VPG, MPT գործիքների ծրագրավորումը կատարված է C++ ծրագրավորման լեզվով [17]:

2.3.2 SIE ավտոմատացված համակարգի օգտագործման փորձնական տվյալները

Կառուցվածքային մոդելը ավտոմատ դուրս բերող ծրագրային հոսքը հաջողությամբ օգտագործվել է տարբեր տեխնոլոգիաների (45 նմ, 28 նմ) հիշողությունների կառուցվածքային մոդելները ավտոմատ ստանալու նպատակով: ԱՀ-ի աշխատանքի տևողությունը փոփոխական է՝ և՛ փոքր, և՛ պարզ կառուցվածք ունեցող նմուշների դեպքերում կարող է տևել մեկ, երկու ժամ, իսկ մեծ և բարդ կառուցվածք ունեցող նմուշների դեպքերում այն կարող է տևել մինչև մի քանի տասնյակ ժամեր: ԱՀ-ի հիմնական ժամանակի բաժին է կազմում SPICE ֆայլերի ստացումը, և ստացված SPICE ֆայլերի մոդելավորումը: SPICE ֆայլերի մոդելավորման ժամանակը ուղղակիորեն կախված է (ուղիղ համեմատական է) հիշողության նմուշի մեծությունից՝ հիշողության նմուշի զանգվածում բջիջների քանակից: Բացի այդ, SIE ԱՀ-ի աշխատանքի ժամանակը կախված է նմուշի երկրաչափական տեսքից, այսինքն ԾՕ-ի քանակից: Աղյուսակ 2.3-ում բերված են փորձնական արդյունքների տվյալները, որոնք ստացվել են տարբեր տեխնոլոգիաների և կոնֆիգուրացիաների հիշողության նմուշների մոդելի ավտոմատ դուրս բերման ընթացքում:

Աղյուսակ 2.3

SIE-ի հոսքի աշխատանքի տևողությունը

NW	NB	CM	PR	PC	NPP	ST (minutes)	TT (minutes)
2048	24	4	512	96	4	45	180
2048	32	8	256	256	2	60	120

1536	24	16	96	384	4	40	160
2048	192	4	512	768	3	65	195

Աղյուսակ 2.3 (շարունակություն)

SIE-ի հոսքի աշխատանքի տևողությունը

1536	16	8	192	128	3	40	120
192	256	4	48	1024	5	80	400
2048	12	4	512	48	5	40	200

- NW (Number of Words) – Բառերի քանակը հիշողությունում,
- NB (Number of Bits) – Հիշողության Մուտքերի/Ելքերի քանակը,
- CM (Column Mux) – Սյուների փոխանջատիչի երկարությունը,
- PR, PR=NW/CM - Ֆիզիկական տողերի քանակը հիշողությունում,
- PC, PC=NBxCM - Ֆիզիկական սյուների քանակը հիշողությունում,
- NPP (Number of Programming Patterns) – գեներացված Ծրագրավորման օրինակների քանակը,
- ST (Simulation Time in minutes) - նմուշի SPICE ֆայլի մոդելավորման ժամանակը րոպեներով,
- TT (Total Time in minutes) TT=NPPxST - Ընդհանուր ժամանակը րոպեներով:

Փորձերի ընթացքում նկատվեց, որ չնայած նրան, որ մոդելի դուրսբերման ժամանակը կախված է հիշողության պատրաստման տեխնոլոգիայից, ԱՀ-ի աշխատանքի տևողությունը հիմնականում կախված է նմուշի կոնֆիգուրացիայի պարամետրերից (NW, NB, CM, BK) և կատարվող մոդելավորման աշխատանքի տևողության ժամանակից:

Իհարկե, ԱՀ–ում աշխատանքի ընդհանուր ժամանակը գնահատելիս անհրաժեշտ է հաշվի առնել նախապատրաստական աշխատանքների ընթացքում ծախսված ժամանակը, որը կարող է կազմել 4-ից 8 ժամ: Շատ կարևոր է ընդգծել, որ ավտոմատ դուրսբերման միջավայրում ստացված կառուցվածքային մոդելը արդեն վավերացված է և պատրաստ է անմիջական օգտագործման:

SIV և SIE ԱՀ-ի ողջ ծրագրավորումը կատարված է TCL ծրագրավորման լեզվով: Ինչպես և SIV ԱՀ-ի դեպքում, SIE ԱՀ-ի կառավարումը, մուտքային պարամետրերի արժեքների փոխանցումը ծրագրերին, նույնպես, իրականացված է *կառավարող* տեքստային ֆայլի միջոցով:

Եզրակացություն

Այս գլխում ներկայացվել են մեր կողմից մշակված և իրականացված.

- SIV ավտոմատ ծրագրային համակարգի նկարագրությունը, որը հնարավորություն է տալիս կատարել հիշողության կոմպիլյատորի կառուցվածքային գոտիները և կառուցվածքային խճողումները նկարագրող ֆայլի ճշտության ստուգումը: Ներկայացված են ծրագրային ԱՀ-ի ֆունկցիոնալ, կառուցվածքային սխեմաները;

- SIE ավտոմատ ծրագրային համակարգի նկարագրությունը, որը հնարավորություն է տալիս ստանալ Ստատիկ պատահական դիմումով հիշողության (ՍՊԴՀ) նմուշի կառուցվածքային մոդելը այդ հիշողության նմուշի արտադրության մեջ օգտագործվող **GDSII** ձևաչափով ներկայացված ֆայլից: Ներկայացված են ծրագրային ԱՀ-ի ֆունկցիոնալ, կառուցվածքային սխեմաները, մոդելի դուրս բերման հետ կապված հիմնախնդիրները, բարդությունները և սահմանափակումները:

Բերված են կատարված փորձերի արդյունքների տվյալները:

ԳԼՈՒԽ 3 – ՀԻՇՈՂՈՒԹՅԱՆ ՆՄՈՒՇՆԵՐՈՒՄ ՖԻԶԻԿԱԿԱՆ ԵՎ ՎԱՐՔԱԳԾԱՅԻՆ ԹԵՐՈՒԹՅՈՒՆՆԵՐԻ ՆԵՐԱՐԿՄԱՆ ԵՎ ԱԼԳՈՐԻԹՄՆԵՐԻ ՍՏՈՒԳՄԱՆ ԱՎՏՈՄԱՏ ՀԱՄԱԿԱՐԳ

3.1 Անսարքությունների ներարկումը հիշողության նմուշի մեջ և դրանք հայտնաբերող ալգորիթմների ստեղծման խնդիրը

Ինչպես արդեն նշել ենք, հիշողությունների թեստավորումը կարևոր և բարդ խնդիր է մանավանդ ներդված հիշողությունների դեպքում: ՆՀՍ-ի արագ և լիարժեք թեստավորումը ապահովելու նպատակով ներդրված ՆՀՍ-ի հետ նույն միջավայրում ներդրվում են լրացուցիչ էլեկտրոնային սխեմաներ, պրոցեսորներ, որոնց նպատակն է. ա) օգտագործելով ներդրված ինքնաթեստավորման ալգորիթմները (անգլերեն Built in self-test)՝ կատարել ՆՀՍ-ի անմիջական ստուգումը, և բ) հնարավորության դեպքում, հիշողության աշխատունակության վերականգնումը (անգլերեն Built in redundancy analysis) [2], [17-21]: Այդ ներդրված պրոցեսորների կողմից իրականցվող ինքնաթեստավորումը (ՆԻԹ) ստուգման և վերահսկման հիմնական միջոց է հանդիսանում ներդրված հիշողության սարքերի համար: ՆԻԹ արդյունավետությունը բնութագրվում է երկու չափանիշներով. ա) օգտագործված թեստային ալգորիթմի ծածկույթը՝ ՆԻԹ ալգորիթմի միջոցով հնարավոր անսարքությունների հայտնաբերման ունակությունը, բ) այդ լրացուցիչ սխեմաների օգտագործման հետևանքով հիշողության սարքի տոպոլոգիայի ավելացված մակերևույթի տոկոսը: Այս երկու չափանիշները փոխկապակցված են, քանի որ շատ բարդ թեստավորման ալգորիթմի իրականացումը միկրոսխեմայում պահանջում է սարքի թիթեղի վրա ավելի շատ մակերևույթ և հակառակը, անհնար է իրականացնել լիարժեք ծածկույթ ունեցող ՆԻԹ ալգորիթմ թիթեղի սահմանափակ մակերևույթում: Էլեկտրոնային սխեմաների թիթեղի մակերևույթի մեծացումը անցանկալի է հատկապես հատուկ կիրառությունների դեպքերում, որոնցից են բժշկական, տիեզերական, բջջային և մի շարք այլ ոլորտների հատուկ սարքավորումները: Իհարկե, կան թեստավորման ալգորիթմներ, որոնք

հնարավորություն են տալիս ՆՀՍ-ում հայտնաբերել բոլոր տեսակի իրատեսական և տեսական սխալները: Այդ կարգի ալգորիթմները լինում են բարդ և ծավալուն և չեն կարող օգտագործվել ներդրված սարքերի դեպքում, քանի որ դրանց իրականացումը կպահանջի միկրոսխեմայի թիթեղի մակերևույթի մեծ տարածք, որը իր հերթին կբերի նոր անսարքությունների առաջացմանը և արտադրանքի վերջնական գնի անցանկալի թանկացմանը: Ավելցուկայնությունից խուսափելու համար անհրաժեշտ է օպտիմալացնել ՆԻԹ ալգորիթմը: Այս դեպքում մենք պետք է գնահատենք ՆԻԹ ալգորիթմը այդ հիշողության սարքին բնորոշ իրատեսական անսարքությունների մոդելների միջոցով և կատարենք օգտագործված թեստային ալգորիթմի օպտիմալացումը: Երբ մենք ասում ենք իրատեսական անսարքություններ, ապա մենք հասկանում ենք անսարքություններ, որոնք իրատեսական թերությունների արդյունք են հանդիսանում [30]: Այդ թերությունները բնորոշ են հատկապես այդ սարքին, ՆՀՍ-ի պատրաստման տոպոլոգիայի մասնավորապես այդ տեխնոլոգիային: Այս պնդումը շատ կարևոր է ՆԻԹ ալգորիթմի օպտիմալացման համար, քանի որ հետազոտությունները հաստատում են, որ տեսական անսարքությունների մեծ մասը իրատեսական չեն տվյալ ՆՀՍ-ի պատրաստման տեխնոլոգիայի համար [31]:

Բացի այդ, համաձայն գրականությանը, ըստ իրենց վարքագծի բնույթի և բջիջների մեջ հայտնվելու, դրսևորման ձևի, թերությունները լինում են և դասակարգվում են որպես «փափուկ» և «կոշտ»: «Կոշտ» (անգլերեն՝ “hard” defect) թերություն է անվավում այն թերությունը, որի վարքագիծը հաստատուն է և չի փոփոխվում ժամանակի ընթացքում նաև հիշողության սարքի սնուցման լարումը անջատել-միացնել գործողությունից հետո: «Փափուկ» են կոչվում (անգլերեն “soft” defect) այն թերությունները, որոնք հայտնվում և անհետանում են և՛ ՆՀՍ-ի աշխատանքի ընթացքում և՛ սարքի սնուցման լարումը անջատել-միացնելու գործողությունից հետո [32-35]: Ակնհայտ է, որ «փափուկ» թերությունների արդյունքում հայտնված անսարքությունները ժամանակավոր բնույթ են կրում և անհետանում են հիշողության բջջում այդ թերության դրդապատճառը վերանալու հետ միաժամանակ [36-40]: «Փափուկ» անսարքությունների հայտնաբերումը կատարվում է նույն ծրագրային միջոցների օգնությամբ, ինչ որ օգտագործվում են «Կոշտ» անսարքությունների համար: Մինչդեռ «Փափուկ» և «Կոշտ» անսարքությունների ազդեցության հետևանքով վնասված

ՆՀՍ-ի աշխատունակությունը վերականգնելու համար կատարվող մոտեցումները և գործողությունները տարբերվում են մեկը մյուսից [7]: Այս ատենախոսության սահմաններում, մենք դիտարկում ենք միայն «Կոշտ»՝ հաստատուն վարքագիծ ունեցող թերությունների հետևանքով առաջացած անսարքությունները:

Իրատեսական թերությունների մոդելները մշակելը և թերությունների գրադարանը ունենալու մեկ այլ առավելություն է այն, որ այն հնարավորություն է տալիս ստուգել թեստային ալգորիթմները իրական թերությունների գրադարանի սահմաններում: Ներկայումս թեստային ալգորիթմների և թերությունների մոդելների բարդության աճման պատճառով դրանց ստուգումը առանց հատուկ ծրագրային ավտոմատ միջոցների դառնում է անհնար [15]:

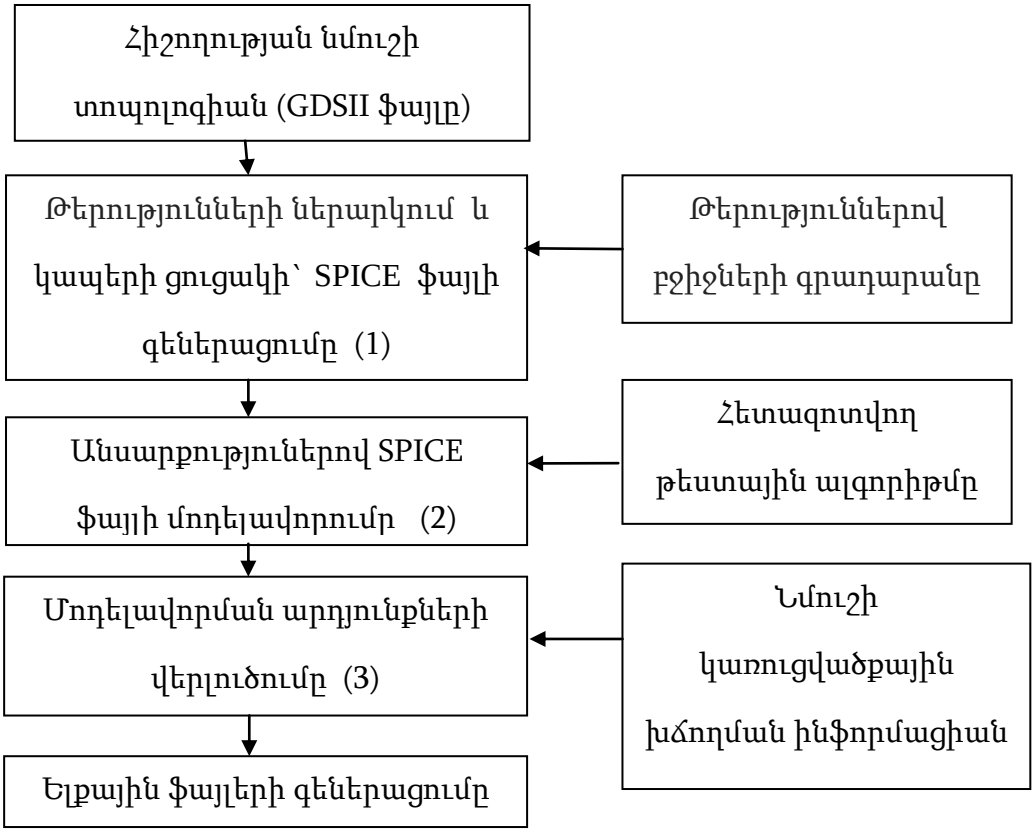
Այս գլխում ներկայացված է թերությունների ներդրման և հիշողությունը թեստավորող ալգորիթմը ստուգող (ԹՆևՀԹԱՍ) ծրագրային ԱՀ, որի նպատակն է կատարել թեստային ալգորիթմների աշխատանքի ստուգումը, գնահատել այդ ալգորիթմների արդյունավետությունը՝ օգտագործելով տարբեր տեսակի **իրատեսական** թերությունների մոդելներ, որոնք կարող են տեղադրվել հիշողության սարքի բոլոր հանգույցներում՝ ստեղծելով տարբեր վարքագիծ դրսևորվող անսարքություններ: Անհրաժեշտության դեպքում ԹՆևՀԹԱՍ ծրագրային ԱՀ-ը հնարավորություն է ընձեռում իրականացնել. ա) թեստային ալգորիթմների աշխատանքի ամբողջական արդյունավետության հաստատումը, բ) թեստային ալգորիթմների յուրաքանչյուր Մարշ ալգորիթմի տարրերի առանձին ստուգումը, գ) թեստային ալգորիթմների քայլ առ քայլ ստուգումը:

3.2 Անսարքությունների ներարկման ավտոմատացված ծրագրային համակարգի կառուցվածքը և աշխատանքը

ԹՆևՀԹԱՍ ծրագրային ԱՀ-ը բաղկացած է հետևյալ ֆունկցիոնալ մասերից.

1. *Թերություններ ներարկման հանգույց :* (Տես՝ նկ. 3.1) ԱՀ-ի այս ֆունկցիոնալ մասը ապահովում է հետազոտվող թերություններ պարունակող բջջի/բջիջների ներարկումը հիշողության նմուշի տոպոլոգիայի՝ GDSII ձևաչափի ֆայլի, մեջ: Թերությունների ներարկումը իրականացվում է ֆիզիկական մակարդակով՝ առանց հաշվի առնելու

հիշողության նմուշի կառուցվածքային խճողումները (scramble): Ներարկման արդյունքում ստացվում է տոպոլոգիայի մակարդակով թերությունը պարունակող՝



Նկար 3.1. ԹՆևՀԹԱՍ ծրագրային ԱՀ -ի ֆունկցիոնալ սխեման

«արատավոր» նմուշը: Այդ նմուշից գեներացվում է SPICE ձևաչափի ֆայլը՝ թերություն պարունակող SPICE ֆայլը:

2. *Մոդելավորման հանգույց:* Ծրագրային համակարգի այս հանգույցը գեներացնում է. ա) հետազոտվող թեստային ալգորիթմը՝ վերաձևափոխելով այն SPICE մոդելավորմանը համապատասխան ձևաչափի; բ) մշակում է մոդելավորման աշխատանքի ընթացքում օգտագործվող պարամետրերը; գ) յուրաքանչյուր դիտարկվող թերության համար գեներացնում են մոդելավորման համար անհրաժեշտ տվյալները: Երբ SPICE ֆայլերի նախապատրաստման աշխատանքը ավարտվում է, մոդելավորման հանգույցը սկսում է մոդելավորումը՝ օգտագործելով թերությունների դիմադրության տարբեր արժեքներ, և, անհրաժեշտության դեպքերում՝ անսարքությունների հայտնաբերման համար ստեղծելով անհրաժեշտ լրացուցիչ սթրեսային իրավիճակներ, կիրառելով մոտքային պարամետրերի (սնուցման լարում, աշխատանքային ջերմաստիճան, աշխատանքային հաճախականություն և այլն) տարբեր արժեքներ:

Մոդելավորման արդյունքում ստացված տվյալների մշակումը կատարվում է ծրագրային ԱՀ-ի հաջորդ հանգույցում:

3. Վերլուծման հանգույց: Ծրագրային համակարգի այս հանգույցում կատարվում է ստացված մոդելավորման արդյունքների մշակումը, կատարվում է արդյունքների վերջնական վերլուծումը, ելքային ֆայլերի ստեղծումը: ԱՀ-ի աշխատանքի այս հատվածում օգտագործվում է հիշողության նմուշի կառուցվածքային մոդելը:

3.3 Հիշողության նմուշի տոպոլոգիայի մեջ թերություններ պարունակող բջիջների ներարկման ձևերը

Ծրագրային ԱՀ-ում ֆիզիկական թերությունների ներարկումը նմուշի տոպոլոգիայի (GDSII) ֆայլի մեջ իրականցվում է երկու մեթոդով:

3.3.1 Հիշողության զանգվածի մակերևույթի ավտոմատացված ծրագրավորման մեթոդի նկարագրությունը

Թերություններ ներարկման այս եղանակը հնարավորություն է տալիս ավտոմատ ձևով ներարկել մեկ կամ ավելի հետազոտվող թերությունները՝ նմուշի հիշողության բջիջների մակերևույթի տարբեր դիրքերում: Հիամեմատած այն եղանակի հետ, երբ հետազոտվող թերությունը տեղադրվում է SPICE ձևաչափի ֆայլում անմիջապես հիշողության տարածքում թերություններ ներարկելու մոտեցումը ունի մի քանի էական առավելություններ [41]: Առաջին առավելությունը այն է որ թերությունները ներարկվում են հիշողության տարածքի նախապես որոշված հատվածում՝ որոշակի համարի ֆիզիկական տողում և սյունում: Այս որոշակիությունը հետագայում մշակման աշխատանքի ընթացքում հնարավորություն է ընձեռում վերահսկել հետազոտվող թեստային ալգորիթմի աշխատանքի յուրաքանչյուր քայլը, հետազոտման յուրաքանչյուր ժամանակային հատվածում և մակարդակում: Եկրորդ առավելություն է, որ ԱՀ -ում հաճախականության ժամանակացույցի կիրառումը հնարավորություն է տալիս. ա) հաշվարկել թեստային ալգորիթմի աշխատանքի տևողությունը՝ BIST-ի աշխատանքային հաճախականության մասին տվյալի առկայության դեպքում, բ)

կատարել թեստավորվող ալգորիթմի քայլ առ քայլ կարգաբերումը, որը էապես լավացնում է ալգորիթմի կարգաբերման աշխատանքի գործընթացը [15], [17]:

Հիշողության զանգվածի ծրագրավորումը իրականացվում է հիշողության ծրագրավորման MPT գործիքի միջոցով: Ինչպես արդեն նշել ենք, MPT գործիքը ներարկում է թերություններ պարունակող բջիջները բացառապես հիշողության զանգվածի մեջ՝ համաձայն իրենց ֆիզիկական դիրքերի, որոնք նկարագրվում են ծրագրավորման օրինակների ֆայլում:

Հարկ է նշել, որ MPT ծրագրավորող գործիքը ունի հետևյալ երկու էական սահմանափակումները. ա) MPT-ին ներարկում է թերություններ միայն հիշողության բջիջների զանգվածի տարածքի սահմաններում և չի կատարում ներարկման գործողություն այդ տարածքից դուրս՝ հիշողության սարքի մյուս (Մուտքային/Ելքային հանգույցներում բջիջներում, տողերի և սյուների ապակոդավորիչների հանգույցներում, զգայուն ուժեղացուցիչների բջիջներում և այլն) հանգույցներում, բ) ծրագրավորման ընթացքում MPT-ին կարողանում է ներարկել միայն երկու տեսակի՝ ծրագրավորող թերություններով բջիջներ, այսինքն միայն երկու տարբեր վարքագիծ ունեցող թերություններ կարող են միաժամանակ ներարկվել և հետազոտվել հիշողության բջիջների զանգվածում ԹՆևՀԹԱՍ ծրագրային համակարգի կողմից: Միաժամանակ այդ թերություններով բջիջների քանակը սահմանափակվում է միայն հիշողության սարքում պարունակվող հիշողության բջիջների գումարային քանակով: Ինչպես հետագայում փորձերը ցույց տվեցին, ԹՆևՀԹԱՍ ծրագրային համակարգի միջոցով ներարկվող թերությունների քանակը լիովին բավարար է թեստավորման ալգորիթմների լիարժեք ստուգման համար: Թեստերի հետազոտման աշխատանքների ընթացքում ստեղծվել են թերությունների բջիջների ամբողջական գրադարաններ, որոնք պարունակում են թերություններով բջիջներ տարբեր տեխնոլոգիական պրոցեսների համար: Թերությունների ամբողջական գրադարանների առկայությունը հնարավորություն է ընձեռում լիարժեք ստուգել թեստավորման ալգորիթմները [41-44]:

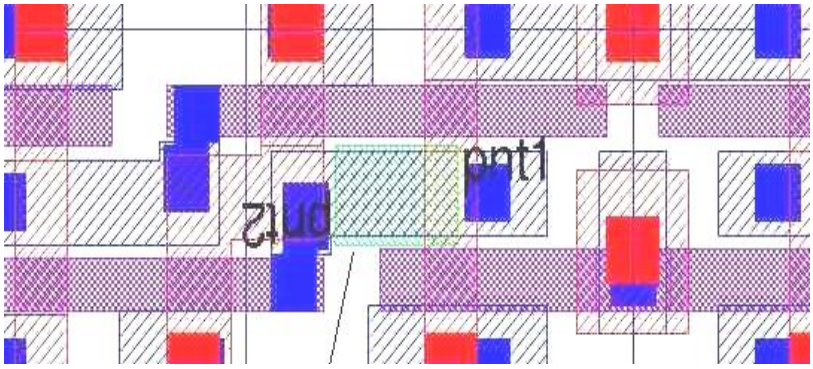
11111111	00000000	xxxxxx1x
11111111	xxxxxxxx	xxx1xxxx
11111111	00000000	xxxxxxxx
11111111	xxxxxxxx	xxxx10xx
11111111	00000000	xx0xxxxx
ա.	բ.	գ.

Նկար 3.2. ՕՕ ֆայլերի օրինակները

Նկար 3.2-ում ներկայացված են ծրագրավորման ֆայլերի օրինակները, որոնք ստեղծվել են 4 տող և 8 սյուն ունեցող հիշողության բջիջների զանգվածի համար: Նկարի 3.2 ա. օրինակում տրված է մի դեպք, երբ համաձայն այդ օրինակի, ամբողջ հիշողության զանգվածը ներարկված է «1» տեսակի թերություններով: Նույն ձևով՝ բ. օրինակին համաձայն, զանգվածում ներարկվելու են «0» տեսակի թերություններով: Իսկ գ. օրինակում ներկայացված է դեպք, երբ «0» տեսակի թերությունները ներարկվելու են 8-րդ տողի 3-րդ սյան մեջ և 7-րդ տողի 6-րդ սյան մեջ, իսկ «1» տեսակի թերությունները ներարկվելու են երեք բջիջներում՝ 7-րդ տողի 5-րդ սյան մեջ, 2-րդ տողի 4-րդ սյան մեջ և 1-ին տողի 7-րդ սյան մեջ: Ընդ որում, 7-րդ տողում «1» և «0» տեսակի թերությունները ներդրվում են հարևան բջիջներում: Թերությունների այդ տեսակ դիրքավորումը կարող է բերել յուրահատուկ վարքագծով անսարքությանը և լրացուցիչ ստուգման գործոն հանդիսանա թեստային ալգորիթմի համար: «1» և «0» տեսակի թերությունների օրինակներ են հանդիսանում առաջին գլխում նկարագրված, «ծրագրավորող» բջիջները (Տես՝ նկ. 1.25 և նկ. 1.26): Այս «ծրագրավորող» բջիջների կիրառումը ԹՆևՀԹԱՍ ծրագրային ԱՀ -ում համարժեք է «Stacked 0» և «Stacked 1» տեսակի անսարքությունների ներդրմանը հիշողության զանգվածի բջիջներում:

3.3.2 Թերությունների մեխանիկական ներարկման ձևը

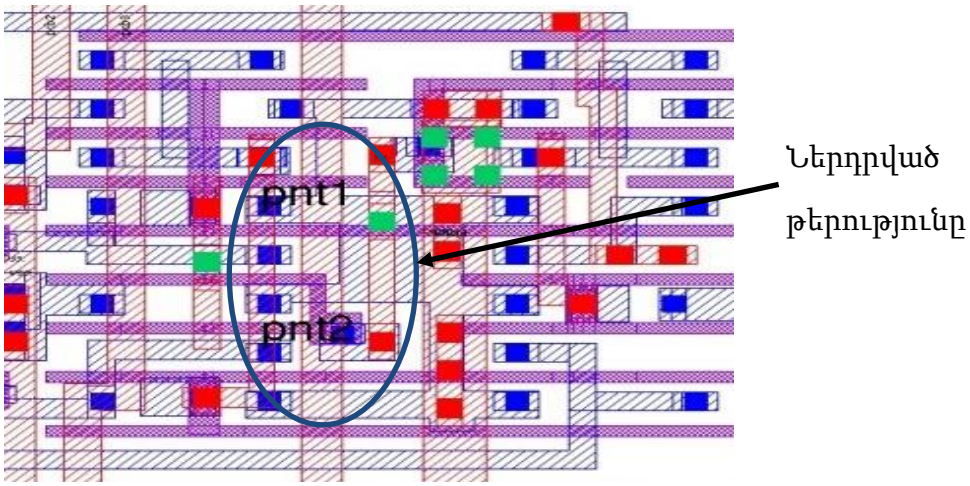
Նկար 3.3-ում ներկայացված է մեխանիկական (manual) ձևով ներարկված թերության օրինակը, որը ներարկվել է հիշողության բջիջում, որի արդյունքում բաց



Նկար 3.3. ՀՍ-ի բջջի տոպոպոլոգիայում բաց դիմադրողական թերության օրինակը

դիմադրությամբ անսարքություն է հայտնվում հիշողության զանգվածի բջիջում:

Հիշողության մակերևույթում թերությունների մեխանիկական ներարկման մեթոդը կիրառվում է այն դեպքերում, երբ ա) հետազոտվող թերությունը անհրաժեշտ է ներարկել հիշողության սարքի հիշողության բջիջների զանգվածի սահմաններից դուրս՝ հիշողության այլ հանգուցներում; բ) ներդրվող թերությունը ունի բարդ կառուցվածք, և այդ թերությունը ստանալու համար պետք է լինում բջիջի տոպոլոգիայում մեխանիկական ձևով կատարել բարդ ձևափոխություններ: Բնականաբար, օգտագործվող թեստային ալգորիթմը պարտավոր է հայտնաբերել այդ հանգույցներում գտնվող թերությունները ևս:



Նկար 3.4. Տողերի ապակոդավորման բջիջի տոպոլոգիայում ներարկված «դիմադրության բաց» թերություն

Նկար 3.4-ում տրված է թերության ներդրման օրինակ տողերի ապակոդավորման հանգույցում՝ ազդանշանի ժամանակային ուշացման վարքագծով անսարքություն ստանալու համար: Ներարկման արդյունքում «pnt1» և «pnt2» կետերի միջև ընկած դիմադրողական անսարքության մոդելը հնարավորություն է ընձեռում մշակել, ստուգել և հղկել այդ անսարքությունը հայտնաբերող թեստային ալգորիթմը:

3.4 Ավտոմատացված համակարգում թեստավորման ալգորիթմի նկարագրության ձևը

ԹՆևՀԹԱՍ ծրագրային ԱՀ-ի աշխատանքի հիմնական նպատակն է վավերացնել ՀՍ-ում օգտագործվող թեստային ալգորիթմի ներարկված անսարքությունը հայտնաբերելու ունակությունը: ՆՀՍ-երում կիրառվում են Մարշ տիպի թեստային

ալգորիթմներ: Թեստային ալգորիթմը մուտքային տվյալ է հանդիսանում մոդելավորման հանգույցի համար: Օգտագործվող Մարշ թեստային ալգորիթմը փոխանցվում է ծրագրային հոսքին առանձին ֆայլի միջոցով և բաղկացած է երեք պարտադիր տարրերից:

1. ***Ալգորիթմի Մարշ տարրերի հասցեավորման ձևերը***: Օգտագործվող հասցեավորման ձևերը հետևյալն են.

- **U** (անգլերեն - Upward addressing) – Ներքևից դեպի վերև հասցեավորում: Այս հասցեավորմամ ժամանակ թեստավորումը սկսվում է զրոյական հասցեից և աճելով շարունակվում հասնում է հիշողության նմուշի վերջին առավելագույն (maximal) հասցեին,
- **D** (անգլերեն - Downward addressing - reverse to U)՝ – Վերևից դեպի ներքև՝ հակառակ է U հասցեավորման ձևին:
- **P** (անգլերեն - Ping-pong addressing) – Պինգպոնգ հասցեավորում: Այս հասցեավորման ժամանակ թեստավորումը սկսվում է զրոյական հասցեից, այնուհետև, հաստատուն շեղումով, թռիչքաձև աճելով՝ հասցեների տիրույթի թույլատրելի սահմաններում, հասնում է հիշողության նմուշի վերջին՝ առավելագույն հասցեին:
- **Q** (Reverse Ping-pong addressing) - տեսակի հասցեավորումը հակադարձ է Պինգպոնգ տեսակի հասցեավորմանը:

2. ***Ալգորիթմի Մարշ տարրերում թույլատրելի գործողությունների տեսակները:*** Թեստային ալգորիթմում կարող են օգտագործվել հետևյալ գործողությունները.

- **w** – Գրել գործողությունը (անգլերեն - write operation),
- **r** – Կարդալ գործողությունը (անգլերեն - read operation),
- **n** – Հապաղում գործողությունը՝ գործողություն Չկա (անգլերեն - NO operation), հիշողության ժամանակաչափի ազդանշանը (CLK) չի կանգնեցվում կատարելով «**n**» քանակությամբ «դատարկ» գործողություն,
- **d** – Հապաղում գործողությունը (անգլերեն - delay operation): Հապաղում գործողությունը կառուցված է որոշակի N քանակի Հապաղում գործողություններից և իրականացված է որպես՝ d=n*N,

3. ***Մարշ ալգորիթմում գրել և կարդալ գործողությունների տվյալները:*** Թեստերում օգտագործվում են «a» և «b» տվյալների օրինակները (անգլերեն - Background Pattern

(BP)): Ընդ որում, «b» տեսակի տվյալի արժեքը միշտ հակադարձ է «a» տվյալի արժեքին: Ծրագրային հոսքում օգտագործվում են հետևյալ չորս տեսակի «a» տվյալները.

- Բոլորը զրո (0000),
- Բոլորը մեկ (1111),
- Շախմատաձև (0101) և
- Հակադարձ Շախմատաձև (1010):

Թեստային ալգորիթմի նկարագրությունը գտնվում է «անուն».alg ֆայլում և պարունակում է լրացուցիչ ևս երեք պարամետրեր: Դրանք են. ա. ԹԱ-ի սկզբնական

```
a = 0101 b = !a

Q w a n b r a
P w b r b n a
U w b d9 b r b
D r b d16 a r b w a
D r a

stop adress 0x9
inst_max_adr 0xf
start_adr 0x2
```

Նկար 3.5. ԱՀ-ի Թեստային ալգորիթմ ֆայլի օրինակը

հասցեն, բ. ԹԱ-ի «կանգառ» հասցեն և բ. հիշողության նմուշի հասցեների տիրույթի առավելագույն հասցեի արժեքը: Այս պարամետրների արժեքները ալգորիթմի ֆայլում ներկայացվում են տասնվեցական ձևաչափով: Այս պարամետրերի արժեքները նույնպես մշակվում են ԱՀ-ում: Տվյալների տասնվեցական ձևաչափով ներկայացման ընտրությունը դրսևորվել է պատմականորեն և պայմանավորված է այն բանով, որ հիշողությունը նախագծող ճարտարագետները, հասցեավորումը նկարագրելիս մեծ մասամբ օգտագործում են տասնվեցական ձևաչափը: Նկար 3.5-ում ներկայացված է թեստային ալգորիթմ ֆայլի օրինակը: *Կանգառ հասցեն* օգտագործվում է ԹԱ-ի աշխատանքը հարկադրաբար կանգնեցնելու նպատակով՝ այն կատարվում է, երբ ԹԱ-ի աշխատանքային հասցեն հավասարվում է նշված Կանգառ հասցեի արժեքին: *Հիշողության նմուշի առավելագույն հասցե պարամետրը* (instance_max_addr) օգտագործվում է այն դեպքերում, երբ թեստային ալգորիթմի կողմից հաշվարկվող

առավելագույն հասցեի արժեքը կարող է գերազանցել ՀՍ-ի իրական՝ ֆիզիկական, հասցեի արժեքը. max_addr > instance_ max_addr :

Հաշվարկային Առավելագույն հասցեի հաշվարկման բանաձևն է.

$$max\_addr=2^{(\#ADRi)}$$, որտեղ

(#ADRi) – Հիշողության նմուշի տրամաբանական հասցեների ազդանշանների քանակն է,

max_addr – Հիշողության նմուշի հաշվարկվող առավելագույն հասցեն է,

instance_max_addr - Հիշողության նմուշի ֆիզիկական առավելագույն հասցեն է:

ԱՀ-ում օգտագործված թեստային ալգորիթմի նկարագրման ձևաչափը լիովին ապահովում է Մարշ տեսակի հանրահայտ թեստային ալգորիթմների մեծամասնության նկարագրման հնարավորությունը:

ԹՆևՀԹԱՍ ծրագրային ԱՀ-ը կատարում է թեստային ալգորիթմի ձևաչափի ձևափոխումը՝ SPICE մոդելավորող ծրագրային գործիքին հասկանալի, վեկտորային ձևաչափի (SVP): SVP ֆայլը պարունակում է տեղեկություններ կատարվող գործողությունների, և դրանց համապատասխան տվյալների մասին՝ ներկայացված թվային կոդերի տեսքով: ԱՀ-ի միջոցով՝ ավտոմատացված ձևով, ստեղծվում է SPICE մոդելավորման աշխատանքը կառավարող SIM ֆայլը: Կառավարման ֆայլի մեջ ընդգրկված են մոդելավորման մուտքային տվյալները, ազդանշանների սկզբնական վիճակը ապահովող արժեքները, աշխատանքային լարումների, ջերմաստիճանի, աշխատանքային հաճախականության արժեքները, մոդելավորման համար անհրաժեշտ գրադարանների ճանապարհները և մի շարք այլ կառավարող տվյալներ:

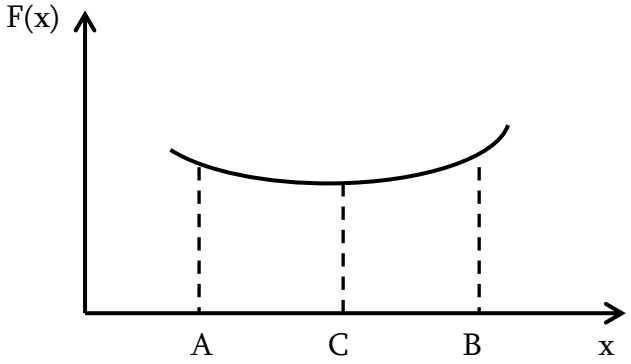
Հավելված 6-ում տրված է ԹՆևՀԹԱՍ ծրագրային ԱՀ-ի կառավարման ֆայլի օրինակը:

3.5 Ավտոմատացված համակարգի մոդելավորման արդյունքների վերլուծումը և թերության դիմադրության ճշգրտման ալգորիթմը

Թերությունները GDSII –ում ներարկումից և թերություններ պարունակող SPICE ֆայլի ստանալուց հետո ԱՀ-ի հաջորդ քայլն է հանդիսանում գտնել ստացված անսարքության դիմադրության այն արժեքը, որի դեպքում հետազոտվող ԹՕԱ-ը

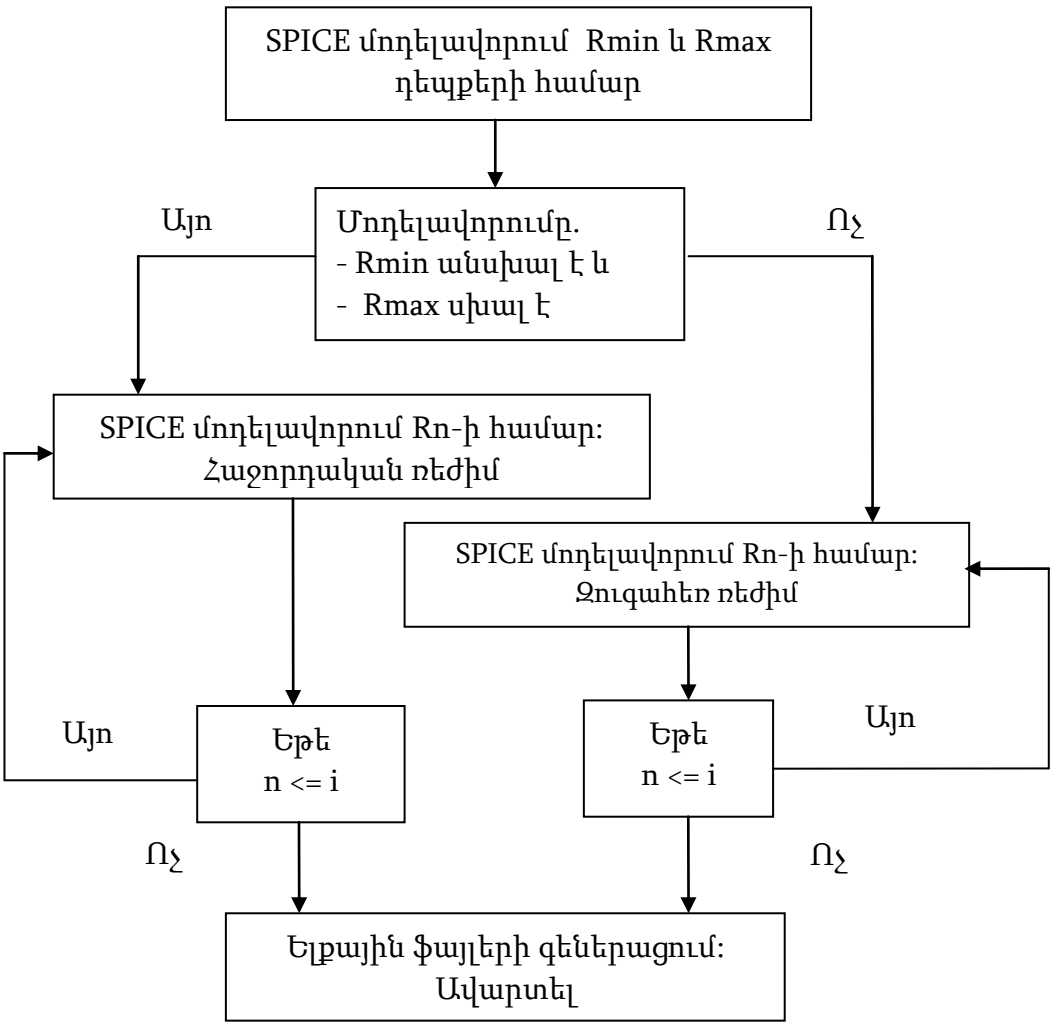
ալգորիթմը կհայտնաբերի ներարկված անսարքությունը՝ ալգորիթմի «զգալիությունը»: Դիմադրության հաշվարկման այս խնդիրը դասվում է մեկչափանի օպտիմալացման մեթոդի խնդիրների դասին: Այս մեթոդի շարքին են պատկանում. ոսկե կտրվածքի, Ֆիբոնաչիի թվերի մեթոդը, պոլինոմինալ մոտարկման մեթոդը, դիխոտոմիկ բաժանման մեթոդը և նրանց մի շարք մոդիֆիկացիաները [28], [29]: Այս աշխատանքի սահմաններում մենք օգտագործում ենք դիխոտոմիկ բաժանման մեթոդի ձևափոխված՝ մեր խնդրի մասնավոր դեպքին հարմարեցված տարբերակը:

Դիխոտոմիկ բաժանման մեթոդի աշխատանքի էությունը հետևյալն է [29]. ենթադրենք ունենք հատված [A,B], որի վրա կա մեկ նվազագույն (մինիմում) (ընդհանուր դեպքում կենտ քանակի մինիմումներ): Համաձայն դիխոտոմիկ բաժանման մեթոդի (Տես՝ նկ. 3.6)՝ հատվածը բաժանվում է երկու կեսի և C կետից «q» թույլատրելի շեղումի մեծությամբ հաշվարկվում է նպատակային ֆունկցիայի արժեքը F(C+q) և F(C-q):



Նկար 3.6. Դիխոտոմիկ բաժանման մեթոդը

Եթե պարզվի, որ F(C+q)>F(C-q), ապա մինիմումը գտնվում է [A,C] հատվածում: Եթե F(C+q)<F(C-q), ապա մինիմումը գտնվում է [C,B] հատվածում: Իսկ եթե F(C+q)=F(C-q), ապա մինիմումը գտնվում է [C-q,C+q] հատվածում: Այսպիսով, հաջորդ քայլում [A,B] հատվածի փոխարեն պետք է հետազոտել նեղացված համապատասխան հատվածը [A,C], [C,B] կամ [C-q,C+q]: Քայլերը կրկնվում են այնքան, մինչև որ հատվածի երկարությունը դառնա փոքր թույլատրելի շեղումի՝ «q»-ի, արժեքից: Այսպիսով, հնարավոր քայլերի քանակը N է, դա ոչ ավել, քան log((B-A)/q) թվին մոտ ամբողջ թիվ է: Այս մեթոդի թերությունը այն է, որ նպատակային ֆունկցիայի արժեքը պետք է հաշվարկել երկու անգամ ամեն մի քայլի ժամանակ:



Նկար 3.7. Մոդելավորման Դիմադրության համապատասխանեցման ալգորիթմը

ԹՆևՀԹԱՍ ծրագրային ԱՀ-ում նպատակային ֆունկցիան ստացված SPICE ֆայլի մոդելավորումն է, որը, հիմնականում, ժամանակատար է և պահանջում է հաշվողական մեծ ռեսուրսներ, ուստի մենք չենք կարող աշխատեցնել SPICE ֆայլի մոդելավորումը մեծ քանակով դիմադրության արժեքների տիրույթի համար: Նկար 3.7-ում տրված է ԹՆևՀԹԱՍ ծրագրային ԱՀ-ում իրականցված անսարքության դիմադրության մշակման ալգորիթմի բլոկ-սխեման: Այս խնդրի լավարկման նպատակով մշակվել է և օգտագործվում է դիմադրության համապատասխանեցման ալգորիթմը (ԴՀԱ): ԴՀԱ-ի հաջորդական մշակման ճյուղում օգտագործվում է դիխոստոմիկ բաժանման մեթոդի հիմունքներով կառուցված ալգորիթմը: ԴՀԱ-ը իրականացված է ԹՆևՀԹԱՍ ծրագրային հոսքի Վերլուծման հանգույցում: ԱՀ-ի կառավարման ֆայլում տրվում է հետազոտվող անսարքության դիմադրության նվազագույն Rmin (մինիմալ) և առավելագույն Rmax

(մաքսիմալ) հաշվարկային արժեքները և նախատեսված իտերացիաների քանակը (Տես՝ նկ.3.8): «Իտերացիաների քանակը» պարամետրով ճարտարագետը սահմանում է քայլերի քանակը, որոնց ընթացքում ԱՀ-ը պետք է կատարի դիմադրության հետ կապված ԹԱ-ի հետազոտությունները: Այս պարամետրը համարժեք է Դիխոտոմիկ բաժանման մեթոդի N պարամետրին: Վերլուծման սկզբում կատարվում է մոդելավորում դիմադրության մինիմալ և մաքսիմալ արժեքների համար: Դրանից հետո, կախված այդ մոդելավորման արդյունքներից, վերլուծումը շարունակվում է *հաջորդական* կամ էլ *զուգահեռ* եղանակներով՝ ռեժիմներում:

```
resistor_min 93750
resistor_max 95000
iteration_number 10
```

Նկար 3.8. Դիմադրության պարամետրերի նկարագրությունը կառավարման ֆայլում՝ օրինակ

Զուգահեռ ռեժիմ՝ դա նախնական մոդելավորման արդյունքների ստուգման եղանակ է: ԱՀ-ը անցնում է այս ռեժիմին երբ մոդելավորման արդյունքները Rmin և Rmax դիմադրությունների համար տվել են նույնատիպ՝ միաժամանակ «սխալ» կամ «անսխալ» արդյունքը: Զուգահեռ ռեժիմի դեպքերն են.

1. Մոդելավորումը ավարտվել է սխալով անսարքության դիմադրության Rmin արժեքի համար: Դա նշանակում է որ մինիմալ դիմադրության արժեքի կետը սխալ է սահմանված:
2. Մոդելավորումը ավարտվել է առանց անսարքությունը հայտնաբերելու և Rmin և Rmax արժեքների համար միաժամանակ:

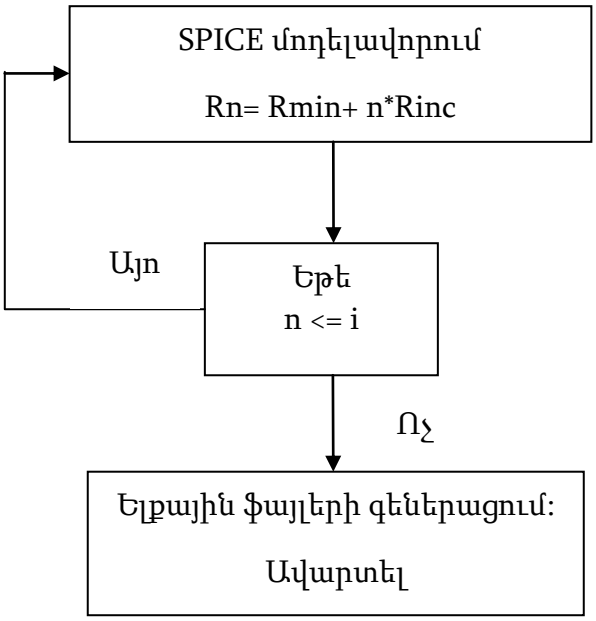
Նշված այս երկու դեպքում, ԱՀ -ը անցնում է հետազոտման – Զուգահեռ ռեժիմ: Այս դեպքում անսարքության մոդելավորման կետերի քանակը և այդ կետերին համապատասխան անսարքությունների դիմադրությունների արժեքները հաշվարկվում են հետևյալ ձևով.

- Մոդելավորման կետերի քանակը հավասար է իտերացիաների քանակին՝ «i» -ին:
- Յուրաքանչյուր կետում մոդելավորման դիմադրության արժեքը՝ Rn-ը հավասար է.

$$Rn = Rmin + n * Rinc ,$$ որտեղ

- n ∈ {1…i} ,
- Rinc - դա գումարվող դիմադրության արժեք է, որը հաստատուն փոփոխական է և հաշվարկվում է հետևյալ բանաձևով.

Rinc= (Rmax-Rmin)/i

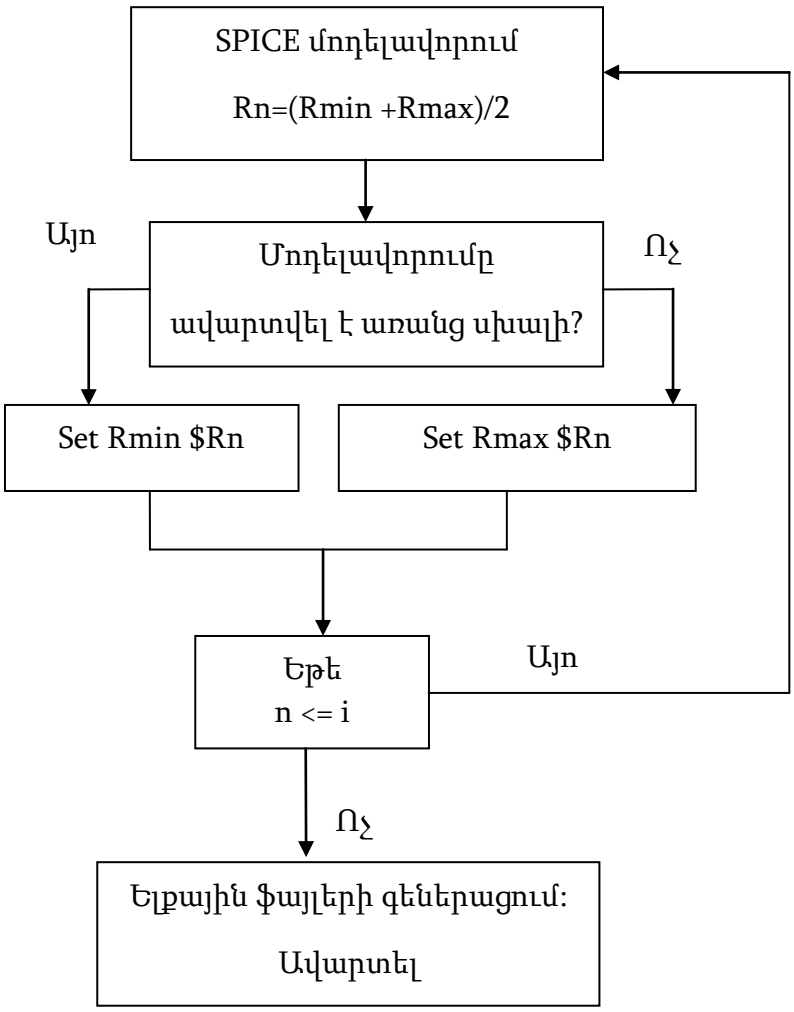


Նկար 3.9. Դիմադրության համապատասխանեցման ալգորիթմը - Զուգահեռ ռեժիմի աշխատանքի բլոկ-սխեման

Զուգահեռ ռեժիմի բլոկ-սխեման ներկայացված է Նկար 3.9-ում: Զուգահեռ ռեժիմի նպատակն է կատարել մոդելավորման արդյունքների լրացուցիչ՝ վերահսկող ստուգում: Այդ ստուգումը կատարվում է պարզելու համար կա արդյո՞ք Rmin-ից մինչև Rmax ընկած տիրույթում դիմադրության որևէ արժեք, որի դեպքում մոդելավորման արդյունքում կստանաք «սխալ» արդյունք: Այսինքն, ԹԾԱ-ը կհայտնաբերի այդ մասնակի դիմադրության արժեքին բնորոշ անսարքությունը:

Հաջորդական ռեժիմը օգտագործում է Դիխոտոմիկ բաժանման մեթոդի ձևափոխված տեսակը Rn-ի որոշակի արժեքի համար: Նկար 3.10-ում ներկայացված է Հաջորդական ռեժիմի աշխատանքը նկարագրող բլոկ-սխեման: Հաջորդական ռեժիմում հետազոտվող թերության դիմադրության արժեքի հաշվարկը կատարվում է քայլ առ քայլ և կախված է նախորդ մոդելավորման արդյունքից:

Rn=(Rmin+Rmax)/2



Նկար 3.10. Դիմադրության համապատասխանեցման ալգորիթմը

Նկար 3.10-ից երևում է, որ յուրաքանչյուր մոդելավորման քայլից հետո Rmin և Rmax արժեքները փոփոխվում են, և այդ փոփոխությունը ունի նվազելու կամ էլ աճելու որոշակի ուղղվածություն՝ կախված նախորդ քայլում ստացված մոդելավորման արդյունքից:

ԹՆևՀԹԱՍ ծրագրային ԱՀ-ի վերլուծման հանգույցը մշակում է մոդելավորման արդյունքները: Մոդելավորման արդյունքում գեներացված ֆայլերում գտնվում է լիարժեք տեղեկություն մոդելավորման փուլերի մասին, որոնք, իրենց հերթին, համապատասխանում են թեստավորվող Մարշ ալգորիթմի քայլերին: Այդ ֆայլերը պարունակում են տեղեկություններ ալգորիթմի քայլերի ժամանակային կետերի բաշխման մասին, թեստային ալգորիթմի յուրաքանչյուր Մարշ էլեմենտի համարը, գործողության տիպը, օգտագործվող տվյալների օրինակները, և, հիշողությունում

կատարվող գրել և «կարդալ» գործողությունների հասցեները: Տեղեկությունների այս խումբը հեշտացնում է SPICE մոդելավորման արդյունքների վերլուծումը:

Ներկայացված թերության դիմադրության հաշվարկները բերված են «բաց դիմադրության թերությունների» տեսակների համար, բայց բոլոր ներկայացված պնդումները ճիշտ են նաև, այսպես կոչված, «կարճ միացումով դիմադրողականությամբ» թերությունների համար ևս: Տարբերությունը միայն նրանում է, որ այդ դեպքում Rmin և Rmax ունեն հակառակ իմաստը և որպես հետևանք հաշվարկների ուղղությունը հակառակ է՝ Rmax-ից դեպի Rmin:

3.6 Ավտոմատացված համակարգի մոդելավորման ելքային ֆայլերի ձևաչափի ձևափոխումը

Ինչպես արդեն բազմիցս նշել ենք, որ ներկայումս անհնար է մոդելավորման

```
End of transient, CPU time used: 2.05 sec (2.13 sec wall)
Memory usage Physical: 434 MB, Virtual: 509 MB
DOUT Error: t=306 ns     expect=1 state=0 node=qb2
        file=ts90ngkt2p22stdcs512sa16x4cm4_3_0.svp
DOUT Error: t=426 ns     expect=1 state=0 node=qb2
        file=ts90ngkt2p22stdcs512sa16x4cm4_3_0.svp
DOUT Error: t=456 ns     expect=1 state=0 node=qb2
....
        file=ts90ngkt2p22stdcs512sa16x4cm4_3_0.svp
DOUT Error: t=516 ns     expect=1 state=0 node=qb1
        file=ts90ngkt2p22stdcs512sa16x4cm4_3_0.svp
DOUT Error: t=546 ns     expect=1 state=0 node=qb1
        file=ts90ngkt2p22stdcs512sa16x4cm4_3_0.svp
DOUT Error: t=576 ns     expect=1 state=0 node=qb1
        file=ts90ngkt2p22stdcs512sa16x4cm4_3_0.svp
Simulation Statistics
Comparison Errors          : 21
Accepted timesteps         : 4679
Repeated timesteps         : 157
Minimum  timesteps          : 1161
Maximum  timesteps          : 270
MOS evaluations            : 1336281
```

Նկար 3.11. HSIM մոդելավորման ծրագրի ելքային ֆայլի օրինակը

արդյունքների վերլուծման լիովին ավտոմատացումը այդ խնդրի բարդության պատճառով [4], [29]: Նույնիսկ այս պարագայում, մենք չենք կարող լիովին հրաժարվել մոդելավորման ելքային ֆայլերի մշակման, գոնե մասնակի, ավտոմատացումը ապահովելու խնդրից: Հարկ է նշել, որ վերլուծման ընթացքում գեներացված ֆայլերի կառուցվածքը, ծավալը, ստացված արդյունքների ներկայացման տեսքը (Տես՝ նկ. 3.10) դառնում է չափազանց կարևոր մոդելավորման արդյունքների վերլուծման ժամանակը նվազեցնելու համար: Մոդելավորման ելքային ֆայլերի մշակման ժամանակը անմիջապես կախված է այն բանից, թե որքանով են այդ ֆայլերի պարունակությունը ճարտարագետի համար ընթերցելի և մատչելի մշակման համար: Ինչպես երևում է նկար 3.11-ից, տվյալները ներկայացված են մարդու համար դժվար ընթերցելի, խճճված ձևաչափով, և այս ելքային ֆայլի մշակումը միանշանակ ճարտարագետից կպահանջի. առաջինը, բավականին շատ ժամանակ, և երկրորդը, որը այս դեպքում շատ կարևոր է, բարձր որակավորում և փորձ:

```
Comparison Errors      : 40
DOUT Error: t=70 ns
q_number: 01234567
expect:   +1+1+1+1
state:    +0+0+0+0

DOUT Error: t=88 ns
q_number: 01234567
expect:   +1+1+1+1
state:    +0+0+0+0
...
DOUT Error: t=214 ns
q_number: 01234567
expect:   +++101+1
state:    +++010+0

The number of the error times is 10
```

Նկար 3.12. Մոդելավորման արդյունքի ֆայլի ձևափոխման օրինակը

Մոդելավորման արդյունքների մշակումը հեշտացնելու, դյուրին դարձնելու նպատակով ԱՀ-ը գեներացնում է լրացուցիչ երկու ելքային ֆայլեր ևս: Այդ երկու նոր ֆայլերը իրենցից ներկայացնում են մոդելավորման ելքային ֆայլերի վերամշակման և միաձուլման արդյունքները: Նրանք կառուցված են, այնպես, որ ինֆորմատիվ են ամեն մեկը առանձին և, միաժամանակ, լրացնում են մեկմեկու, օգնում են մոդելավորման արդյունքների կարգաբերմանը և սխալների հայտնաբերմանը, արագ մշակմանը:

Այդ ֆայլերից առաջինը (Նկար 3.12) պարունակում է ձևափոխված հետևյալ տվյալները.

1.ստացված սխալների քայլերի ժամանակային կետերը «t», 2. ելքային տվյալները՝ սպասված (expect) և փաստացի ստացված (state) բառային ձևաչափով տվյալների արժեքները, և 3. ստացված բիթերի

արժեքների դիրքերը (q_number) բառի մեջ, որտեղ սպասված և ստացված տվյալներում գումարել «+» նշանը նշանակում է, որ սպասված և ստացված տվյալները այդ դիրքում համընկնում են: Ֆայլը պարունակում է լրացուցիչ տեղեկություն հայտնաբերված սխալների ընդհանուր քանակի (Comparison Errors : 40) և նրանց համապատասխան ժամանակային կետերի քանակի (The number of the error times is 10) մասին:

```
CLK WEA MEA MEB ADRA[0~3] ADRB[0~3] DA[0~7] Q[0~7] signals
00 0 0 0 0 0000 0000 00000000 xxxxxxxx
50 0 0 0 0 0000 0000 00000000 xxxxxxxx 00: *** setup ***
54 0 1 1 0 0000 0000 01010101 xxxxxxxx
59 1 1 1 0 0000 0000 01010101 xxxxxxxx |59: me 1, mo 1 (write)
63 0 0 0 1 0000 0000 01010101 xxxxxxxx
68 1 0 0 1 0000 0000 01010101 xxxxxxxx |68: me 1, mo 2 (read)
72 0 1 1 0 1000 1000 01010101 01010101 |72: +0+0+0+0 DOUT  Error: qb7 qb5 qb4
77 1 1 1 0 1000 1000 01010101 xxxxxxxx |77: me 1, mo 1 (write)
212 1 0 0 1 0001 0001 01010101 xxxxxxxx |212: me 1, mo 2 (read)
216 0 1 1 0 1001 1001 01010101 01010101 |216: +++010+0 DOUT Error: qb7 qb5 qb3
221 1 1 1 0 1001 1001 01010101 xxxxxxxx |221: me 1, mo 1 (write)
225 0 0 0 1 1001 1001 01010101 xxxxxxxx
230 1 0 0 1 1001 1001 01010101 xxxxxxxx |230: me 1, mo 2 (read)
***End of march_el(1): U w a r a  stop_adr is 0x9***
```

Նկար 3.13. Ձևափոխված մոդելավորման ելքային երկրորդ ֆայլի օրինակը

2. գեներացված լրացուցիչ ելքային ֆայլը (Նկար 3.13) պարունակում է տեղեկություններ. ա) թեստային ալգորիթմի բաղադրիչների (Տես՝ բանաձև (6)-(9))՝ Մարշ տարրի համարը - «me 1», ընթացիկ կատարվող գործողությունը - «mo 1 (write)», բ) տվյալների օրինակների և ալգորիթմի քայլերի ժամանակային կետերի արժեքները, և գ) գրել, կարդալ - գործողությունների հասցեները և ղեկավարող ազդանշանների արժեքները: Բոլոր ա, բ, և գ կետերում նշված տվյալները, յուրաքանչյուրը առանձին և միաժամանակ իրար հետ, շատ կարևոր են ԱՀ-ում օգտագործվող թեստային ալգորիթմի կարգաբերման գործընթացի համար: Այս երկրորդ ֆայլը ձևավորվում է մոդելավորման ծրագրի մուտքային և ելքային ֆայլերի համաձուլման արդյունքում: Իհարկե, այդ երկու ֆայլերում կա տվյալների կրկնություն և այդ կրկնության նպատակն է՝ դարձնել դյուրին. ա) մոդելավորման արդյունքում ստացված սխալների վերլուծումը և բ) թեստային ալգորիթմի աշխատանքի կարգաբերումը և ձևափոխումը: Բացի այդ, այս դեպքում պետք չէ միաժամանակ մի քանի ֆայլեր դիտարկել՝ տվյալների արժեքները

ստանալու համար, որը իր հերթին նույնպես հեշտացնում է արդյունքների վերլուծումը: Նկար 3.13 ֆայլի ձևաչափը ավելի «լայն» է, հիմնված է մոդելավորման վեկտորային ֆայլի վրա և մոդելավորման արդյունքների մասին պարունակում է ավելի մանրամասն նկարագրություն: Մոդելավորման արդյունքում ստացված ելքային ֆայլի (նկ. 3.11) և ԹՆևՀԹԱՍ ԱՀ-ի կողմից գեներացվող ֆայլերի (նկ. 3.12 և 3.13) պարզ համեմատությունը ցույց է տալիս վերջիններիս առավելությունը՝ մոդելավորող ծրագրի ելքային ֆայլի հետ համեմատ այդ ֆայլերը ավելի ընթերցելի են: Հետազոտվող թեստային ալգորիթմը ձևափոխվում և մոդելավորվում է այնքան ժամանակ, մինչև որ հետազոտվող թերության հետևանքով ստացված անսարքությունը հստակորեն կհայտնաբերվի մշակված ալգորիթմի միջոցով:

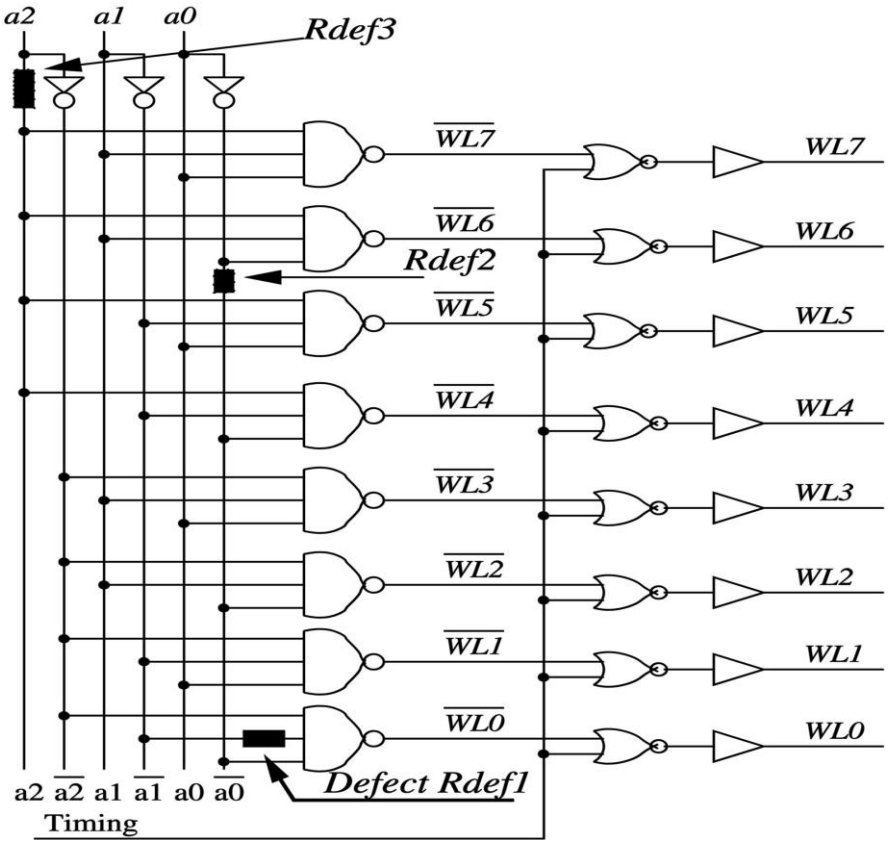
3.7 Փորձնական արդյունքները

ԹՆևՀԹԱՍ ծրագրային ԱՀ -ը օգտագործել է և օգտագործվում է ՍՊԴՀ սարքերում տարբեր տեսակի թերություններ ներարկելու և ստացված անսարքությունների հետազոտման աշխատանքներում: Այդ աշխատանքների փորձնական արդյունքները տպագրվել են մի շարք հոդվածներում: ԹՆևՀԹԱՍ ԱՀ-ի միջոցով կատարվել են SRAM հիշողությունների հետ կապված հետևյալ խնդիրների հետազոտումները.

1 ՆՀՍ-ի հասցեների ապակոդավորման հանգույցում ակտիվացման և ապաակտիվացման թերությունների ներարկումը, մոդելավորումը և այն հայտնաբերող համապատասխան թեստային ալգորիթմի հետազոտումը [8], [13] :
2 Պատահական վարքագծով, այսպես կոչված, «հեռագրական» աղմուկի (անգլերեն՝ «Telegraph» noise) թերության մոդելավորումը հիշողության բջիջներում և համապատասխան անսարքությունը հայտնաբերող թեստային ալգորիթմի մշակումը [13]:
3 Տեխնոլոգիական պրոցեսների տատանումների հետևանքով առաջացած թերությունների (ՏՊՏՀԱԹ) հետազոտումը TCMS 45 նմ, 28 նմ, 16 նմ և 14 նմ տեխնոլոգիաներով պատրաստված SRAM հիշողությունների նմուշների համար: Այդ անսարքությունները հայտնաբերող Մարշ տեսակի թեստային ալգորիթմների մշակումը և ստուգումը [9]:

4 Ժամանակակից FinFET տեխնոլոգիաներում իրատեսական թերությունների հետազոտումը և դրանց հայտնաբերման համար՝ Մարշ տեսակի թեստային ալգորիթմների մշակումը [46], [47]:

Դիտարկենք այդ կարևոր փորձնական արդյունքներից մի քանիսը:

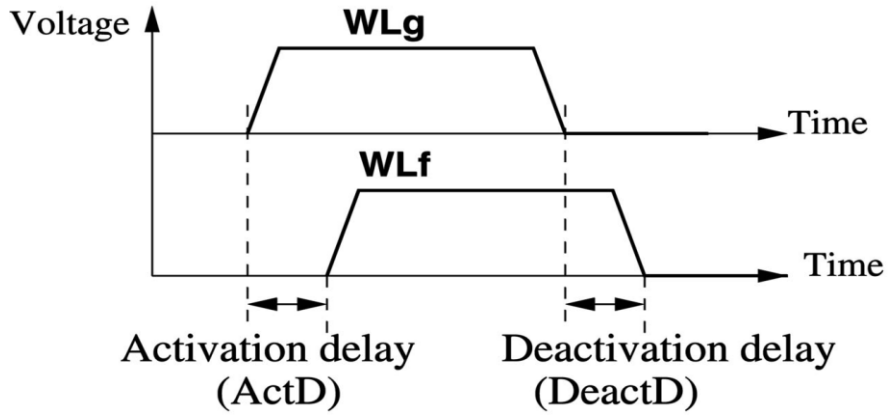


Նկար 3.14. Հասցեների ապակոդավորման սխեմայում «դիմադրության բաց» թերության մոդելավորման էլեկտրական սխեման

3.7.1 ՀՍ-ի հասցեների ապակոդավորման հանգույցում ակտիվացման և ապաակտիվացման թերությունների հետազոտումը և թեստային ալգորիթմի վավերացումը

Հասցեյավորման թերության (Rdef1) ազդեցությունը պարզաբանող էլեկտրական սխեման ներկայացված է Նկար 3.14-ում: Հետազոտությունը կատարվել է 45 նմ տեխնոլոգիայով պատրաստված ՍՊԴՀ սարքի նմուշում հասցեների ապակոդավորման հանգույցի համար: Այս թերության հետևանքով առաջացած անսարքության վարքագիծը

մանրամասնորեն ներկայացված է [48]-ում: Rdef1 անսարքության հետևանքով ապակոդավորման սխեմայում առաջանում է WL0 ազդանշանի, անթույլատրելի շեղում, որը ՆՀՍ-ի սխալ աշխատանքի պատճառ է հանդիսանում: Նկար 3.15–ում տրված է ՀՍ-ի հասցեների ապակոդավորման սխեմայում, «դիմադրության բաց» թերության



Նկար 3.15. Ակտիվացման և ապաակտիվացման ժամանակային հապաղումը

հետևանքով առաջացած Ակտիվացման և ապաակտիվացման ժամանակային փոփոխության վարքագիծը նկարագրող գրաֆիկը: Նկարից երևում է, որ WLf ազդանշանի շեղումը WLg-ի նկատմամբ բերում է երկու ժամանակային շեղման հատվածների առաջացմանը՝ ա) WLf ազդանշանի ակտիվացման ուշացման հատվածը (ActD) և բ) WLf ազդանշանի, ապա-ակտիվացման ուշացման հատվածը (DeactD), որը ևս բացասական ազդեցություն ունի ՀՍ-ի նորմալ աշխատանքի վրա, քան ActD-ը:

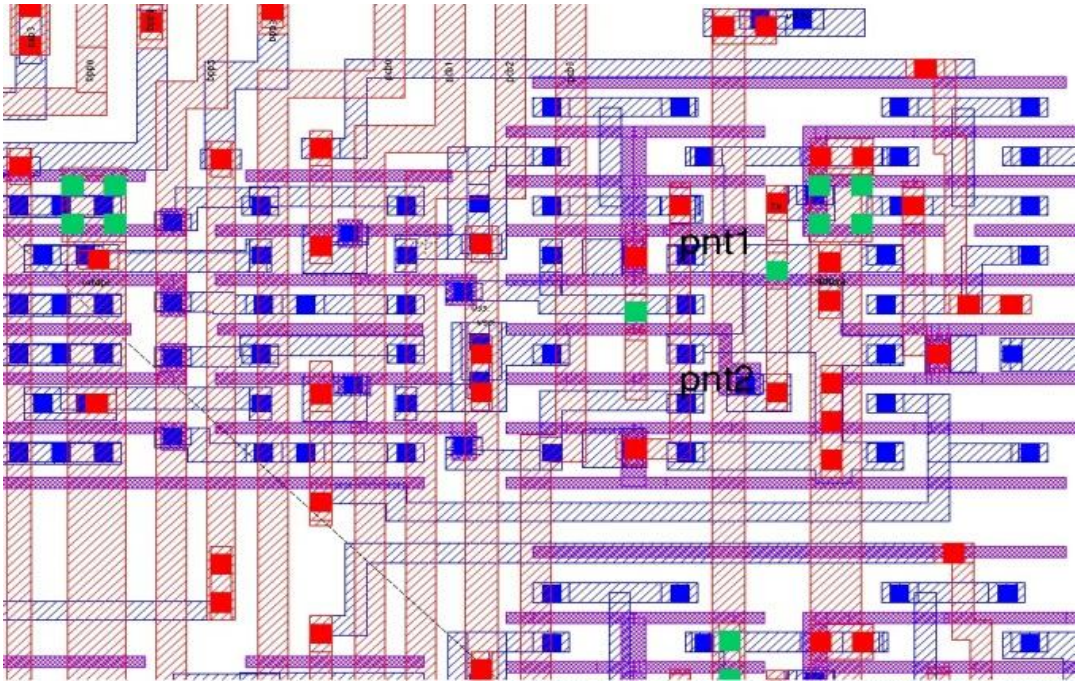
Անսարքության վարքագիծը հետազոտելու համար անհրաժեշտ էր հիշողության նմուշի GDSII ձևաչափի հասցեների ապակոդավորման հանգույցում ներդնել բաց դիմադրողական տեսակի թերություն, որի արդյունքում ՀՍ-ը կստանար Ակտիվացման և ապաակտիվացման համապատասխան վարքագծով անսարքություն: Այնուհետև, պետք է հետազոտել գործող թեստային ալգորիթմի կողմից այդ անսարքությունը հայտնաբերելու ունակությունը:

Աշխատանքի ընթացքում հետազոտվել է հասցեավորման ապակոդավորման սխեմայում Ակտիվացման և ապաակտիվացման անսարքությունը՝

ա) պարզվել է, որ գործող թեստ ալգորիթմը չի հայտնաբերում այդ անսարքությունը,

բ) մշակվել է նոր թեստավորման գործողություն և հիշողության հասցեավորման նոր ձև, և արդյունքում,

գ) ստեղծվել է Ակտիվացման և ապաակտիվացման անսարքությունը հստակորեն հայտնաբերող Մարշ տեսակի ալգորիթմ:



Նկար 3.16. Հասցեների ապակոդավորման բջիջի տոպալոգիայում ներարկված «դիմադրության բաց» թերությունը

Նկար 3.16-ում տրված է հիշողության սարքի հասցեավորման ապակոդավորման բջջի տոպոլոգիան, որի հանգույցում՝ {pnt1 pnt2} կետերի միջև մեխանիկական ձևով ներդրվել է դիմադրության «բաց» թերությունը և որի օգնությամբ մոդելավորվել է անսարքությունը: Այդ թերությունը պարունակող հիշողության նմուշի GDSII ֆայլից, ԹՆևՀԹԱՍ ԱՀ-ի միջավայրում ստացվել է համապատասխան անսարքությամբ նմուշի SPICE ֆայլը, որը պարունակում է անսարքության վարքագիծը վերարտադրող դիմադրությունը՝ տեղադրված «pnt1» և «pnt2» կետերի միջև (Տես՝ նկ.3.16): Հետագայում ԹՆևՀԹԱՍ ԱՀ-ի օգնությամբ կատարվել են այդ դիմադրության փոփոխումը արժեքների մեծ տիրույթում և ստացված SPICE ֆայլերի մոդելավորումները:

3.7.2 Տեխնոլոգիական պրոցեսների տատանումների հետևանքով առաջացած թերությունների հետազոտումը 45 նմ, 28 նմ, 16 նմ և 14 նմ տեխնոլոգիաների ՄՊԴՀ հիշողության նմուշների համար

Մեկ այլ հանրաճանաչ թերությունների դասին են պատկանում Տեխնոլոգիական պրոցեսների տատանումների հետևանքով առաջացած թերությունները (ՏՊՏՀԱԹ) (Process Varietion Defect) [49-56], որոնք նույնպես հետազոտվել են ԹՆևՀԹԱՍ ծրագրային ԱՀ-ի միջավայրում: ՏՊՏՀԱԹ-ի հավանականությունը աճել է ժամանակակից տեխնոլոգիաների (45 նմ, 28 նմ, 22 նմ և ցածր) համար, քանի որ այս տեխնոլոգիաներում սկսում են իրենց դրսևորել այն գործոնները, որոնք անտեսվում էին նախորդ՝ համեմատաբար քիչ խտությամբ 65 նմ, 90 նմ և բարձր տեխնոլոգիաներում [52-68]: Հետազոտման աշխատանքները կատարվել են 45 նմ և ցածր տեխնոլոգիաներով գեներացված հիշողության նմուշների համար: Փորձերը կատարվել են երկու տեսակի թերությունների համար. ա) տրանզիստորների փոխանջատման *լարման* տատանումների և բ) տրանզիստորների չափերի տատանումների համար: Հետազոտությունները հաստատեցին, որ մեծ խտությամբ տեխնոլոգիայով պատրաստ-ված հիշողության բջիջները ավելի խոցելի են ՏՊՏՀԱԹ-ի կողմից, քան բարձր՝ 45 նմ-ից ավելին, տեխնոլոգիաներով պատրաստած բջիջները:

Հոդվածներ [52], [18]-ում և [15] գրքում ներկայացված են տեխնոլոգիաների տատանումների թերությունների ֆունկցիոնալ մոդելները (ԹՖՄ) (անգլերեն functional fault models (FFMs)), որոնք կառուցված են տեխնոլոգիաների տատանումների հիման վրա: Համաձայն այդ ԹՖՄ-ի՝ թերությունները դրսևորելու համար անհրաժեշտ է ՀՍ-ում ստեղծել է մի շարք, այսպես կոչված, լրացուցիչ սթրեսային իրավիճակներ՝ լարում, ջերմաստիճան, հաճախականություն՝ (PVT) պարամետրերի միջոցով դրանց տալով ամենավատ՝ սահմանային արժեքներ: Հետազոտվել են մի շարք, հայտնի, «Մարշ» տեսակի թեստավորման ալգորիթմներ [16], [69-80], որոնց արդյունքում հանգել ենք հետևյալ եզրակացության.

ա) որպեսզի այդ թեստային ալգորիթմները հայտնաբերեն ՏՊՏՀԱԹ թերությունները, անհրաժեշտ է դրանք ձևափոխել՝ ավելացնելով գործողությունների հատուկ հաջորդականություն,

բ) թեստային ալգորիթմը պետք է բազմիցս, հաջորդաբար աշխատեցնել, որպեսզի այն հայտնաբերի ՏՊՏՀԱԹ-ը:

Ընդ որում, այդ պարբերաբար՝ ցիկլիկ, աշխատանքի ընթացքում պետք են հաջորդաբար կիրառվեն տարբեր պարամետրեր «վատագույն» դեպքերի արժեքներ:

ՏՊՏՀԱԹ-ի կողմից առաջացված անսարքությունների հիմնական տեսակներն են.

- «Կարդալ» գործողության սխալը՝ փոխում է հիշողության բջջի արժեքը «կարդալ» գործողության ընթացքում;
- «Գրել» գործողության սխալ՝ հիշողության բջջի սխալը առաջանում է «գրել» գործողության ընթացքում;
- Դիմելու ժամանակային սխալ՝ երկարացնում է բջիջին դիմելու ժամանակը՝ խախտելով հապաղման թույլատրելի պահանջները;
- Պահպանման սխալ՝ փոխում է բջջում պահպանվող տվյալը ՀՍ-ի սնուցման թույլատրելի ցածր լարման ժամանակ (անգլերեն standby mode):

Կատարված փորձերի արդյունքները բացահայտեցին, որ հաճախականության փոփոխումը իր իրական՝ թույլատրելի, տիրույթում չի բերում սխալի դրսևորմանը: Այդ պատճառով կատարված բոլոր փորձերի մոդելավորման համար, մենք հաճախականությունը թողել ենք անփոփոխ՝ ընտրելով թույլատրելի 500 ՄՀց արժեքը: Աղյուսակ 3.1-ում բերված են պարամետրերի «ծայրային» արժեքները, որոնք 28 նմ վեց տրանզիստորային հիշողության բջիջների հետ կատարված փորձերի ընթացքում բացահայտում են ՏՊՏՀԱԹ-ը:

Աղյուսակ 3.1

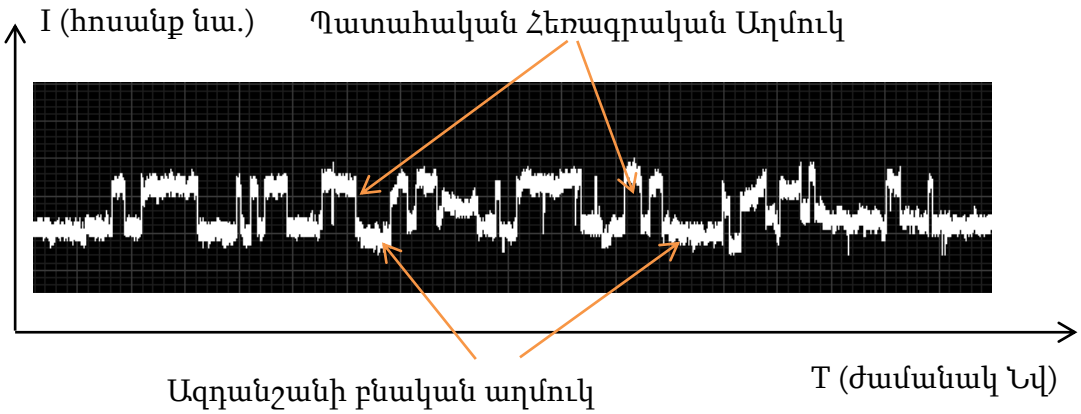
ՏՊՏՀԱԹ-րի սինթեզման սթրեսային պարամետրերի՝ լարման և ջերմության արժեքները

Պարամետր	Նվազագույնը	Աշխատանքայինը	Առավելագույնը
Սնուցման Լարումը(V)	0.765	0.85	0.935
Աշխատանքային Ջերմաստիճանը (°C)	-40	25	125

ՏՊՏՀԱԹ-ի համար կատարված հետազոտությունների արդյունքները մանրամասնորեն ներկայացված են [8], [9], [46], [48] և [57] հոդվածներում:

3.7.3 Պատահական «Հեռագրական» (Telegraph) աղմուկի թերության մոդելավորումը հիշողության սարքերում

Մեկ այլ թերություն ևս՝ Պատահական «Հեռագրական» աղմուկ (ՊՀԱ), անգլերեն Random Telegraph Noise (RTN) մանրամասնորեն հետազոտվել է ԹՆևՀԹԱՍ ծրագրային հոսքի միջավայրում: Հետաքրքրությունը այս թերության նկատմամբ աճել է (ինչպես և Տեխնոլոգիական պրոցեսների տատանումների հետևանքով առաջացած թերությունների դեպքում) ցածր տեխնոլոգիաների (45-ից մինչև 20 նմ) հիշողության սարքերի շահագործմանը զուգահեռ [83-94]: ՊՀԱ-ի պատճառ է հանդիսանում՝ տրանզիստորների ղեկավարման ելքի՝ Gate-ի, մեկուսիչ օքսիդային շերտում, անկառավարելի, փոքր արժեք ունեցող «անհատական» հոսանքի հայտնվելը: Այդ հոսանքի աղբյուրն են հանդիսանում միայնակ, ակտիվ, դրական կրիչները: ՊՀԱ-ի հոսանքի տևողությունը, մեծությունը և հաճախականությունը լիովին պատահական բնույթ են կրում [87]: Հասկանալի է, որ արտադրության ընթացքում անհնար է ապահովել օքսիդային շերտերի կատարյալ կառուցվածքը շերտի ողջ ծավալում:



Նկար 3.17. Պատահական Հեռագրական Աղմուկ հոսանքի բաշխման օրինակը

Այդ պատճառով պատրաստման ժամանակ օքսիդական շերտում առաջանում են թերություններ, որոնք և անհատականորեն ավելացնում են ՊՀԱ հոսանքի մեծությունը (Տես՝ նկ. 3.17): Այդ հոսանքը, գումարվելով ազդանշանին կամ ել հակառակ դեպքում հանվելով ազդանշանի արժեքից, ժամանակակից տեխնոլոգիաների համար կարող է

դառնալ «Փափուկ»՝ վերականգվող տեսակի անսարքության պատճառ: Այս տեսակի թերությունների ֆիզիկական մոդելավորումը GDSII ֆայլում բարդ խնդիր է [85], [87]: Օգտվելով այն բանից, որ ԹՆևՀԹԱՍ ծրագրային ԱՀ-ը հնարավորություն է տալիս ազդանշանների մոդելի ստեղծումը SPICE-ի միջավայրում, մեզ հաջողվեց իրականացնել ՊՀԱ-ի մոդելի ստեղծումը և մոդելավորումը:

Մեր կողմից կատարված հետազոտությունները ցույց տվեցին, որ ՊՀԱ-ի հետևանքով առաջացած անսարքությունների վարքագիծը համապատասխանում է Տեխնոլոգիական պրոցեսների տատանումների հետևանքով ձևավորված անսարքությունների վարքագծին: Միակ տարբերությունը այն է, որ ՊՀԱ-ի անսարքությունները պատահական բնույթ են կրում և անհետանում են ժամանակի ընթացքում, իսկ Տեխնոլոգիական պրոցեսների տատանումների հետևանքով առաջացած անսարքությունները մշտապես (կայուն ձևով) գտնվում են ՀՍ-ում և ժամանակի ընթացքում չեն անհետանում: ՊՀԱ-ի մեկ այլ հատկությունից է, որ այդ անսարքությունների քանակը ավելանում է ՀՍ-ի աշխատանքային ցածր լարման և բարձր ջերմաստիճանի պարագայում:

Հետազոտությունների արդյունքում ստեղծվել և ստուգվել է ՊՀԱ-ի անսարքությունները հայտնաբերող Մարշ տեսակի ալգորիթմը (Տես՝ նկ.3.18) և պարզվել են այդ ալգորիթմը աշխատեցնելու պայմանները, որոնց կիրառման ժամանակ ՊՀԱ-ի անսարքությունները հայտնաբերելու հավանականությունը առավելագույն է: Այս արդյունքները մանրամասնորեն ներկայացված են արտոնագրում [13]:

⇑(W0); ⇑(W1); Delay; ⇑(R1); Delay; ⇑(R1); ⇑(W0); Delay; ⇑(R0); Delay; ⇑(R0).

Նկար 3.18. March RTNF (7N) ալգորիթմը

ՊՀԱ-ի թերությունների լիարժեք թեստավորման պայմաններն են.

1. Աշխատեցնել թեստավորման ալգորիթմը՝ օգտագործելով սնուցման լարման նվազագույն թույլատրելի արժեքը,
2. Աշխատեցնել թեստավորման ալգորիթմը՝ օգտագործելով ջերմաստիճանի թույլատրելի բարձր արժեքը,

3. Կրկնել թեստավորումը բազմակի անգամ:

Ալգորիթմում օգտագործվող հապաղման տևողությունը (Delay) և ալգորիթմի հայտնաբերող հաջորդականության կրկնության քանակը կախված է ՀՍ-ի պատրաստման տեխնոլոգիայի տեսակից և հատկություններից: Դրանք որոշվում են օգտագործողի կողմից՝ յուրաքանչյուր ՀՍ-ի մասնակի դեպքի համար անհատականորեն [58], [72]:

3.74. Ժամանակակից FinFET տեխնոլոգիաներում իրատեսական թերությունների հետազոտումը և Մարշ տեսակի թեստային ալգորիթմի մշակումը

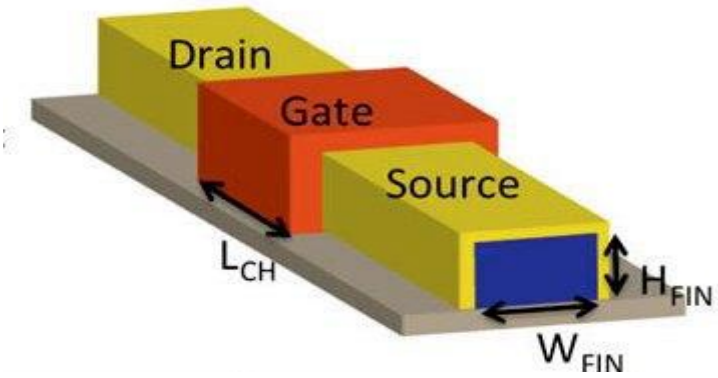
Ժամանակակից տեխնոլոգիաներում MOSFET տրանզիստորների չափերի շարունակական փոքրացումը բերել է.

ա) տրանզիստորներում հոսանքի արտահոսքի (անգլերեն leakage current) էապես մեծացմանը, և

բ) այսպես կոչված «կարճ հոսանքատարի» էֆեկտի (անգլերեն short channel effect), որոնց արդյունքում այլևս անհնար էր դարձել MOSFET տեսակի տրանզիստորների հետագա փոքրացումը:

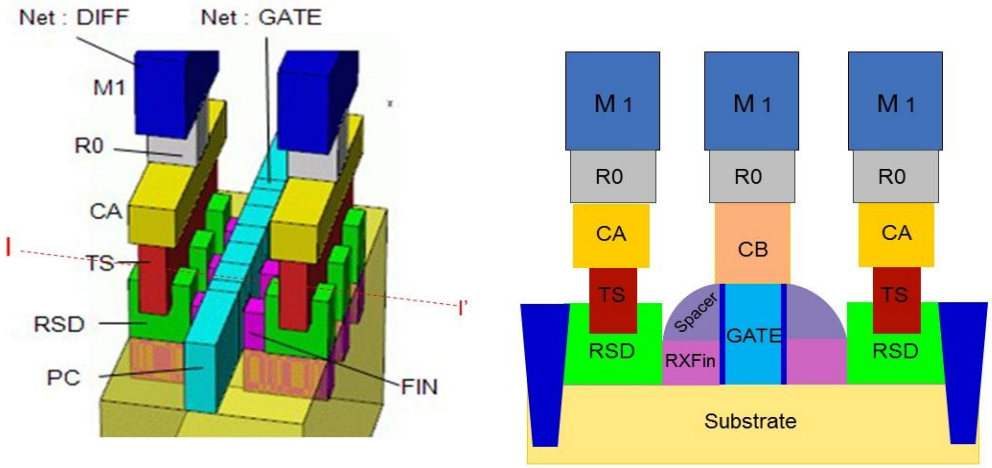
Նշված սահմանափակումները հանգեցրել են MOSFET տիպի տրանզիստորների օգտագործման տեխնոլոգիական սահմանին: Այս պատճառով, սկսած 22նմ-ից փոքր տեխնոլոգիաներում, ճարտարագետները անցել են տրանզիստորի մեկ այլ տեսակի՝ FinFET (Fin Field Effect Transistor) տեխնոլոգիայի, օգտագործմանը: Եթե 22 նմ տեխնոլոգիայում նախագծերը կատարված են հիմնականում MOSFET տրան-զիստորների հիման վրա և հանդիպում էին փորձնական FinFET նախագծեր, ապա 16 նմ և ցածր տեխնոլոգիաների նախագծերում օգտագործվում են բացառապես FinFET տիպի տրանզիստորներ: FinFET կառուցվածքով տրանզիստորի մասին առաջին նշումները բերվել են 1999 թվականին [95] աշխատանքում: Այդ աշխատանքում ներկայացված է FinFET տեխնոլոգիայով պատրաստված, երկմուտքանի կառուցվածք ունեցող տրանզիստորը՝ որպես հնարավոր փոխարինող գործող պլանար տրանզիստորին: Հետագայում FinFET կառուցվածք ունեցող տրանզիստորը, որպես ոչ պլանար կառուցվածք, ներկայացվել է մի շարք աշխատություններում [96-103]:

Ներկայումս FinFET տրանզիստորի լայնածավալ օգտագործումը, այդ թվում և հիշողության սարքերում, ի հայտ բերեց այդ տեխնոլոգիային բնորոշ թերությունների բացահայտման և հետազոտման անհրաժեշտության խնդիրը: Այդ հետազոտությունների նպատակն է FinFET տեխնոլոգիային բնորոշ անսարքությունները հայտնաբերող թեստային ալգորիթմի մշակումը:



Նկար 3.19. 14 նմ FinFET տրանզիստորի եռաչափ, սխեմատիկ կառուցվածքը

Նկար 3.19-ում բերված է FinFET տրանզիստորի կառուցվածքի եռաչափ ներկայացումը: Իսկ Նկար 3.20 ա.-ում ներկայացված է FinFET տրանզիստորի կառուցվածքի եռաչափ սխեման և տրանզիստորի կտրվածքը I-I’ առանցքով (Նկար 3.19 բ.) [1]: FinFET տրանզիստորի հոսանքատարը կառուցված է բարակ՝ թիթեղի հիմքին

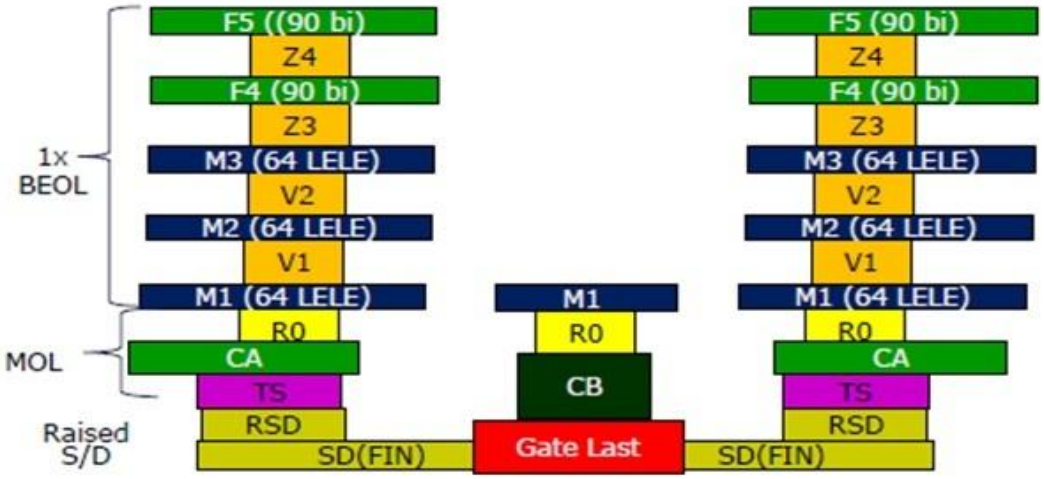


ա. բ.

Նկար 3.20. 14 նմ FinFET տեխնոլոգիայում տրանզիստորի տոպոլոգիայի կառուցվածքը և օգտագործվող շերտերը

ուղղահայաց, սիլիցիումի շերտով, որը անգլերեն անվանում են Fin: Այդ հոսանքատարը շրջապատված է ղեկավարող թիթեղով (Gate): Այս կառուցվածքը ապահովում է հոսանքատարի հստակ և հուսալի կառավարումը՝ տրանզիստորում արտահոսքի հոսանքի նվազեցումը և «կարճ հոսանքատարի» էֆեկտի վերացումը: Նկար 3.21-ում բերված է FinFET տեխնոլոգիային բնորոշ շերտերը և նրանց դասավորվածությունը թիթեղում: Հետազոտությունները ցույց տվեցին՝

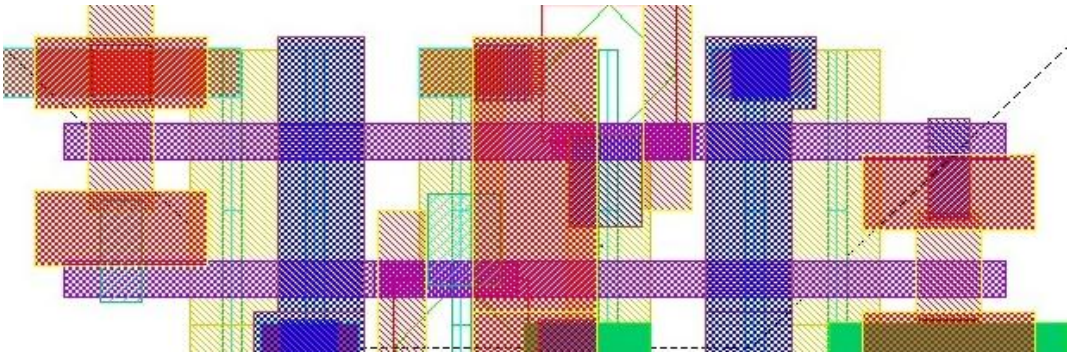
1. FinFET տոպոլոգիայում առաջացող թերությունները որոշվում են միայն FinFET տեխնոլոգիային բնորոշ շերտերով,
2. իսկ մնացած շերտերում (MOSFET տեխնոլոգիայի հետ համընկնող շերտերում՝ մետաղական M1-M3 և F4, F5) թերությունների հետևանքով առաջացած անսարքությունների վարքագիծը և տեսակները լիովին համընկնում են MOSFET տեխնոլոգիայի հետ:



Նկար 3.21. 14 նմ FinFET տեխնոլոգիայում օգտագործվող շերտերը

Այդ պատճառով մեր հետազոտումները սահմանափակվում են միայն FinFET-ին բնորոշ շերտերում դրսևորվող թերություններով՝ դրանով էապես պակասեցնելով հետազոտվող փորձերի քանակը և մշակմանը ենթակա արդյունքների ծավալը: Նկար 3.20-ում տրված է FinFET տեխնոլոգիայում օգտագործվող բոլոր շերտերը և միայն FinFET-ին բնորոշ շերտերը: Ինչպես տեսնում ենք նկարից, մետաղական M1 շերտից սկսած բոլոր շերտերը համապատասխանաբար համընկնում են MOSFET տեխնոլոգիայի շերտերին: Տարբերություն են կազմում FinFET տրանզիստորը կառուցող,

ձևավորող շերտերը: Թվարկենք, ներքևից վերև, այդ շերտերը SD(FIN), RSD, TS, CA, CB և R0: SD(FIN) շերտում ձևավորվում է տրանզիստորի հոսանքատարը, որը մետաղական առաջին M1 շերտին միանալու համար օգտագործվում են RSD և CA շերտերը, իսկ TS և R0 շերտերը դրանք համապատասխան շերտերի միացումները ապահովող՝ անցքերի շերտերն են: CB շերտը R0 անցքային անցումով միացնում է տրանզիստորի ղեկավարող թիթեղը (Gate) անմիջապես մետաղական առաջին՝ M1 շերտին: RSD, TS, CA, CB և R0 շերտերը պատրաստվում են տարբեր հաղորդիչ նյութերից ապահովելով լավագույն համակցումը (անգլերեն՝ adhesion) շերտերի միջև: Այս շերտերում առաջացած թերություններով են ձևավորվում/որոշվում FinFET տեխնոլոգիային յուրահատուկ անսարքությունները [100], [101]:



Նկար 3.22. 14 նմ FinFET հիշողության բջջի տոպոլոգիայի օրինակը

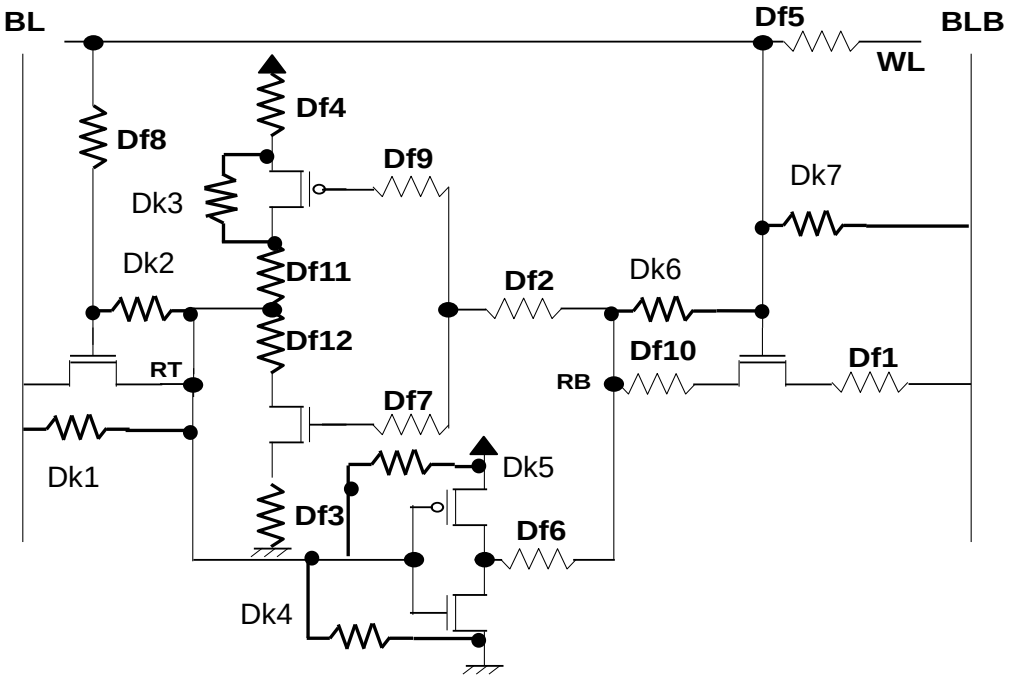
Նկար 3.22-ում ներկայացված է FinFET կառուցվածքի տոպոլոգիայի իրականացումը 14 նմ տեխնոլոգիական պրոցեսի համար:

ԹՆևՀԹԱՍ ծրագրային ԱՀ-ի միջոցով կատարվել են FinFET տեխնոլոգիայում պատրաստված հիշողության սարքերի տարբեր նմուշների հիշողության բջիջների մանրակրկիտ հետազոտությունները հետևյալ անսարքությունների համար.

ա) հոսանքատարի (Fin) «բաց միացման» թերություններ՝ տրանզիստորի Fin –ում «լրիվ բաց» և դիմադրությամբ «բաց միացման» թերություններ,

բ) տրանզիստորի ղեկավարող ելքի (Gate) «բաց միացման» թերություններ՝ տրանզիստորի Gate-ում լրիվ և դիմադրությամբ «բաց միացման» թերություններ,

գ) հոսանքատարի «կարճ միացման» թերություններ՝ տրանզիստորի Մուտքի (Source)



Նկար 3.23. Հիշողության բջջում հետազոտված «բաց» և «կարճ» միացման թերությունների բաշխումը

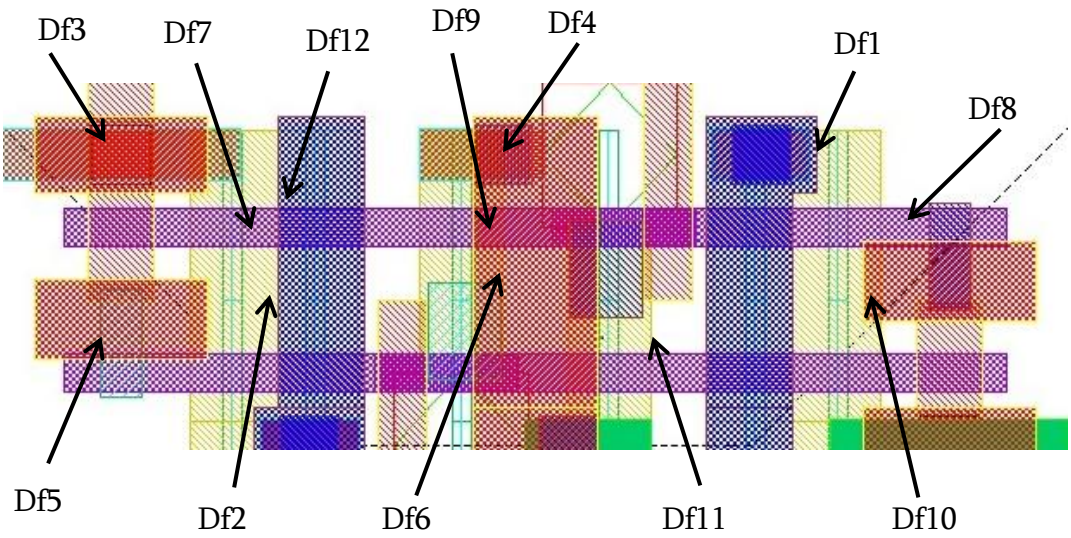
և Ելքի (Drain) լրիվ և դիմադրությամբ «կարճ միացման» թերություններ,

դ) հոսանքատարի և ղեկավարող ելքի միջև «կարճ միացման»՝ Gate Fin լրիվ և դիմադրությամբ «կարճ միացման» թերություններ,

ե) տեխնոլոգիայի տատանման հետևանքով առաջացած թերություններ FinFET-ին յուրահատուկ շերտերում:

Նկար 3.23-ում տրված է հետազոտման աշխատանքների ընթացքում դիտարկված «բաց միացման»՝ ա) և բ), և «կարճ միացման» գ) և դ) հիշողության բջջում թերությունների տեղաբաշխման էլեկտրական սխեման: Նկարում ներկայացված Df1-Df12 և Dk1-Dk7 թերությունները հետազոտվում են առանձին-առանձին՝ հետազոտման ընթացքում որոշելով թերությանը համապատասխան անսարքության տեսակը և բնութագրերը: Պետք է ընդգծել, որ «կարճ միացման» թերությունները դրսևորվում են բացառապես հիշողության բջջի տրանզիստորներում: Հաշվի առնելով այն փաստը, որ հիշողության բջջի սխեմատիկան և տոպոլոգիան ունեն սիմետրիկ կառուցվածք, «կարճ միացման»

թերությունները հետազոտվել են բջջի տրանզիստորների միայն սիմետրիկ մեկ մասի համար՝ դրանով երկու անգամ կրճատելով հետազոտությունների քանակը: Նկար 3.24-ում, որպես օրինակ, ցույց է տրված Df1-Df12 թերությունների ֆիզիկական տեղաբաշխման սխեման՝ 14 նմ FinFET տեխնոլոգիայի հիշողության բջջի տոպոլոգիայում: Նշված Df1-Df12 թերությունների ֆիզիկական բաշխումը տոպոլոգիայում անհատական է յուրաքանչյուր տեխնոլոգիայի և բջջի կառուցվածքի համար:



Նկար 3.24. Df1-Df12 թերությունների ֆիզիկական բաշխման սխեման 14 նմ FinFET հիշողության բջջի տոպոլոգիայում

Հետազոտությունները կատարվել են տարբեր արտադրողների մի շարք MOSFET և FinFET տեխնոլոգիաների հիշողության նմուշների համար: Համեմատության նպատակով նույն տեսակի թերությունները զուգահեռ ներարկվել են, 14 նմ, 16 նմ FinFET, և MOSFET 28 նմ, 45 նմ հիշողության նմուշներում: Կատարվել են բազմակի մոդելավորումներ ստացված անսարքությունների համար: Հետազոտությունների արդյունքները ցույց են տալիս [46].

1. FinFET հիշողության նմուշները ավելի կայուն են դինամիկ անսարքությունների նկատմամբ քան MOSFET նմուշները: Դինամիկ անսարքությունները դիտարկվել են, երբ հատուկ տեսակի թերություններ ներարկվել են FinFET հիշողության նմուշների մեջ: Նույն ժամանակ, նույնատիպ թերությունները ներարկվել են MOSFET նմուշների մեջ, ի պատասխան մենք ստանում ենք ստատիկ վարքագծով անսարքություններ

2. FinFET հիշողության նմուշները ավելի կայուն են տեխնոլոգիական շեղումների հետևանքով առաջացած թերությունների նկատմամբ: FinFET հիշողության բջիջների տոպոլոգիայի կառուցվածքի յուրահատկություններից է նաև այն, որ տրանզիստորների չափերի մինչև 50% տոկոս խախտումը չի հանգեցնում որևէ անսարքության, մինչդեռ 20%-ից 40% տոկոս չափերի շեղումը MOSFET տեսակի հիշողության բջջի տոպոլոգիայում հանգեցնում է տարբեր տեսակի անսարքությունների [57]:
3. Ստատիկ մեկ-բջջային և ստատիկ միացումային թերությունները բնորոշ են երկուսին էլ՝ FinFET և MOSFET տեսակի հիշողության բջիջներին: Ներարկված մի շարք թերություններ հանգեցրել են նույն վարքագծով անսարքությունների միաժամանակ այդ երկու տեխնոլոգիաներում:

Մի շարք հետաքրքիր և ՆՀՍ-ի թեստավորման համար շատ կարևոր արդյունքներ ևս հայտնագործվել են այդ փորձերի ընթացքում.

- Ներարկված թերությունները՝ հիշողության բջջի սնուցմանը միացված տրանզիստորում, մեծամասնությամբ բերում են նմանատիպ անսարքությունների FinFET և MOSFET տեսակի տեխնոլոգիաներում,
- dDRDF տեսակի անսարքությունը դիտարկվում է հիշողության բջջի սնուցմանը միացված տրանզիստորում, իսկ TF*IRF տեսակի անսարքությունը դիտարկվում է հիշողության բջիջը բիթային գծերին միացնող տրանզիստորների թերությունների ժամանակ: Այս երկու տեսակի անսարքությունները բացակայում են MOSFET տեխնոլոգիաներում: Այս արդյունքը համընկնում է [102], [103] հոդվածներում բերված արդյունքներին, ըստ որի dDRDF անսարքությունները, որոնք ներկա են 130 նմ, 90 նմ տեխնոլոգիաներում, բացակայում են 28 նմ և 45 նմ հիշողության սարքերում,
- Տրանզիստորի Ղեկավարման ելքի «բաց» թերությունները FinFET հիշողության բնորոշ թերություններ չեն,
- PVT (հաճախականությունը, սնուցման լարումը, ջերմաստիճանը) վիճակները շատ կարևոր դեր են խաղում թերությունները հայտնաբերելու գործընթացում, քանի որ PVT-ի գործող տիրույթում դիմադրության թերությունները չեն հայտնաբերվում թեստային ալգորիթմների կողմից: Միայն կիրառելով PVT-ի

համար ծայրային արժեքներ ինչպիսիք են՝ բարձր հաճախականությունը, սնուցման ցածր լարումը, բարձր ջերմաստիճանը, հաջողվում է ապահովել թերության հայտնաբերումը կիրառվող թեստային ալգորիթմի կողմից:

Եզրակացություն

1. Այս գլխում ներկայացված է ԹՆևՀԹԱՍ ծրագրային ԱՀ-ը, որի մեջ իրականացված է թեստային թերությունների վերլուծության վրա հիմնված մեթոդը: Այդ մեթոդը օգտագործվում է կիսահաղորդչային սարքերի ներկառուցված մոդուլներում, ինչպիսիք են՝ ներդրված հիշողությունները, թեստային ալգորիթմների թերությունը հայտնաբերման ունակությունը գնահատելու համար [53], [56]:
2. ԹՆևՀԹԱՍ ԱՀ -ը ապահովում է հետևյալ խնդիրների իրականացումը՝
   - Ներարկել տարբեր տեսակի թերություններ ՆՀՍ-ի նմուշի բոլոր ֆունկցիոնալ հանգույցների բջիջների տոպոլոգիայում: Այդ թերությունները կարող են լինել տարբեր տեսակի՝ դիմադրությամբ «բաց» և «կարճ» միացումները, տեխնոլոգիական պրոցեսների շեղումների հետևանքով առաջացած թերությունները և այլն:
   - Վերլուծել ներարկված իրատեսական թերությունների ազդեցությունը հիշողության ֆունկցիոնալության վրա: Գտնել անսարքության դիմադրության շեմային արժեքը, որից սկսած հիշողությունը սկսում է չաշխատել: Դասակարգել ներարկված թերության հետևանքով առաջացած անսարքությունները:
   - Անսարքության դիմադրության լայն տիրույթում ընկած արժեքների համար մոդելավորել հիշողության նմուշի GDSII ֆայլից ստացված SPICE ձևաչափով էլեկտրական սխեման:
   - Ստուգել և կարգաբերել հիշողության թեստավորման ալգորիթմը:
3. Ներկայացված են ԹՆևՀԹԱՍ ծրագրային ԱՀ-ի կիրառման արդյունքները ժամանակակից MOSFET և FinFET տեխնոլոգիաների հիշողության սարքերի համար: Այդ արդյունքները ցույց են տալիս նախագծված ԱՀ-ի կիրառական բարձր արդյունավետությունը հիշողության սարքերի հետազոտման ասպարեզում:

Ամփոփում

1. Հետազոտվել են ՍՊԴՀ տեսակի հիշողության կառուցվածքային տարրերը և ստեղծվել է ՍՊԴՀ տեսակի հիշողության կառուցվածքային պրիմիտիվների դասակարգումը: Մշակվել է կառուցվածքային տարրերի ներկայացման ձևերը, տարրերի ներկայացման լեզուն (Տես՝ արտոնագիր [11]): Մշակվել է ՍՊԴՀ հիշողության կառուցվածքային մոդելը:

2. Մշակվել է ալգորիթմ, որը թույլ է տալիս ստուգել հիշողության կոմպիլյատորի կառուցվածքային մոդելը՝ այն համեմատելով հիշողության գրաֆիկական ներկայացման ֆայլերի՝ GDSII, կառուցվածքների հետ (Տես՝ արտոնագիր [12]): Այս ալգորիթմի հիման վրա իրականացվել է հիշողության կոմպիլյատորի կառուցվածքային մոդելը ստուգող ավտոմատացված համակարգ:

3. Մշակվել է ալգորիթմ, որը թույլ է տալիս նմուշի գրաֆիկական ներկայացման GDSII ֆայլից ավտոմատ ձևով ստանալ հիշողության նմուշի կառուցվածքային մոդելը (Տես՝ արտոնագիր [14]): Այս ալգորիթմի հիման վրա իրականացվել է հիշողության նմուշի կառուցվածքային մոդելը գեներացնող ավտոմատացված համակարգ:

4. Մշակվել և իրականացվել է ծրագրային ավտոմատացված համակարգ, որը թույլ է տալիս հիշողության սարքերում ներդնել տարբեր տեսակի թերություններ և մշակել թերությունը հայտնաբերող, Մարշ տեսակի արդյունավետ թեստային ալգորիթմը [8]:

5. Մշակվել է ԹՆևՀԹԱՍ ավտոմատացված համակարգը: ԹՆևՀԹԱՍ ավտոմատացված համակարգը կիրառվել է ժամանակակից MOSFET և FinFET տեխնոլոգիաների ներդրված հիշողության սարքերի համար: Կիրառման արդյունքները ցույց են տալիս նախագծված ԱՀ-ի կիրառական բարձր արդյունավետությունը հիշողության սարքերի հետազոտման ասպարեզում:

Գրականության ցանկ

1 ITRS 2001, Կիսահաղորդչային տեխնոլոգիաների միջազգային ուղեցույց 2001, http://public.itrs.net/

2 Sargsyan V.K. MEMORY OPTIMIZED REPAIR ORGANIZATION // National Research University of Electronic Technology, MIET, Zelenograd, Russia, Sworld - 2013, 26 December - pp. 245-250.

3 Zorian Y. DTRM for FinFET Zorian tours // ITC, 2014. http://www.itctestweek.org/

4 Zorian Y., Shoukourian S. Embedded-Memory Test and Repair: Infrastructure IP for SOC Yield // IEEE Design & Test of Computers, Vol. 20, Issue 3, 2003 - pp. 58-66.

5 System Marginality Validation of DDR3|DDR4 Memory and Serial I/O / Adam Ley, Alan Sguigna, Al Crouch // ASSET book 2014 -P. 60.

6 Graphic Data System (GDS) and GDSII / Cadence Design Systems, Standard 1978, - P53.

7 Zorian Y., Vardanian V., Aleksanyan K., Amirkhanyan K. Impact of Soft Error Challenge on SoC Design // 11th IEEE Int. On-Line Testing Symposium, July 2005 - pp. 63-68.

8 Amirkhanyan K., Defect injection and a memory test algorithms verification flow// 8th Computer Science and Information Technologies CSIT 2011, Yerevan, 2011 - pp. 283-286.

9 Amirkhanyan K., Davtyan A., Harutyunyan G., Melkumyan T., Shoukourian S., Vardanian V., Zorian Y. Application of Defect Injection Flow for Fault Validation in Memories // 10 th IEEE East-West Design & Test Symposium EWDTS. 2012, - pp 1-4.

10 Ամիրխանյան Կ. Հիշողության նմուշների կառուցվածքային մոդելի ավտոմատ դուրս բերումը գրաֆիկական GDSII ձևաչափի ֆայլից // ՀԳԱԱ և ՀՊՃՀ «ՏԵՂԵԿԱԳԻՐ», ՏԳ սերիա, 2014 N4, հատոր 67 - էջ. 458-466.

11 Memory Modeling Using an Intermediate Level Structural Description/ Zorian Y., Vardanian V., Aleksanyan K., Amirkhanyan K., Shoukourian // US Patent 7786840, August 3, 2010, -P. 17.

12 Various methods and apparatuses for memory modeling using a structural primitive verification for memory compilers / Amirkhanyan Karen, Aleksanyan Karen,

Karapetyan Sergey, Shubat Alexander, Shoukourian Samvel, Vardanian Valery, Zorian Yervant // US patent 8112730 , Feb.7, 2012 -P. 22.

13 DETECTING RANDOM TELEGRAPH NOISE INDUCED FAILURES IN AN ELECTRONIC MEMORY / Amirkhanyan Karen (Yerevan, AM), Grigoryan Hayk (Yerevan, AM), Harutyunyan Gurgen (Abovyan, AM), Melkumyan Tatevik (Yerevan, AM), Shoukourian Samvel (Yerevan, AM), Shubat Alex (Los Altos Hills, CA, US), Vardanian Valery (Yerevan, AM), Zorian Yervant (Santa Clara, CA, US)// US patent 8850277 B2, Sep.30, 2014, -P. 14.

14 Generation of Memory Structural Model Based on Memory Layout / Amirkhanyan Karen, Darbinyan Karen, Davtyan Arman, Harutyunyan Gurgen, Shoukourian Samvel, Vardanian Valery, Zorian Yervant // Patent application number: 20130346056, Publication date: 2013-12-26, -P. 43, http://www.faqs.org/patents/app/20130346056

15 Advanced Test Methods for SRAMs. Effective Solution for Dynamic Fault Detection in Nanoscaled Technologies / A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel // Springer 2010, -P. 171.

16 TESTING SEMICONDUCTOR MEMORIES Theory & Practice / A.J. van de Goor // ComTex Published, 1999, -P. 512.

17 ԾՐԱԳՐԱՅԻՆ ՄԻՋՈՑՆԵՐԻ ՍՏԵՂԾՈՒՄ ՀԻՇՈՂՈՒԹՅԱՆ ՍԱՐՔԵՐԻ ԱՐՏԱԴՐՈՒԹՅԱՆ ՕԳՏԱԿԱՐ ԵԼՔԻ ՄԵԾԱՑՄԱՆ ՀԱՄԱՐ / Ալեքսանյան Կ. // Ատենախոսություն, 2006, - P 120.

18 Introduction to CMOS VLSI Design / David Harris // Harvey Mudd College, Spring, 2004, -P. 615.

19 High Prformence Memeory Testing. Design principles, fault modeling and Self-Test / R. Dean Adams // Kluwer Academic Publisher, 2003, -P. 247.

20 Zorian Y., Skoukourian S. Embedded-Memory Test and Repair: Infrastructure IP for SOC Yield // IEEE Design & Test of Computers, Vol. 20, Issue 3, May 2003, - pp. 58-66.

21 Shoukourian S., Vardanian V., Zorian Y. An approach for evaluation of redundancy analysis algorithms // Records of IEEE Int. Workshop on Memory Technology, Design and Testing, MTDT'01, San Jose, USA 2001, - pp. 51-55.

22 Shoukourian S., Vardanian V., Zorian Y. A methodology for design and evaluation of redundancy allocation algorithms // Proc IEEE VLSI Test Symposium, Napa Valley, USA, 2004, - pp. 249-255.

23 Shoukourian S., Vardanian V., Zorian Y. SoC yield optimization via an embedded memory test and repair infrastructure // IEEE Design & Test of Computers, vol.21, May-June, 2004, - pp. 200-207.

24 Azimane M., Majhi A., Gronthoud G., Lousberg M., Einchenberger S., Lloris Ruiz A. New Algorithm for Dynamic Faults Detection in RAMs // 23rd IEEE VLSI Test Symposium, May 2005, - pp 177-182.

25 Hamdioui S., A.J. van de Goor, Rodgers M. March SS: a test for all static simple faults // Records of IEEE Int. Workshop MTDT, 2002, - pp. 95-100.

26 A.J. van de Goor, Ivo Schanstra Address and Data Scrambling: Causes and Impact on Memory Tests // Delta 2002, - pp 10-19.

27 Practical Programming in TCL and TK / Brent B. Welch, Ken Jones // PRENTICE HALL, NJ USA, 4-th edition, 2004, - P. 1122.

28 Dilillo L., Girard P., Pravossoudovitch S., Virazel A. Data Retention Fault in SRAM Memories: Analysis and Detection Procedures // Proceedings of the 23rd IEEE VLSI Test Symposium (VTS'05), 2005,- pp 183-188.

29 Operation Reserch an Introduction / Hamdy A. Taha // Third Edition 1982, Collier MacMillan Publishers, London, -P. 455.

30 Основы автоматизированного проектирования / Наренков И.П.// 2-ое издание, МГУ 2002, 336 стр.

31 Principles of testing electronic systems / Mourad S., Zorian Y. // John Wiley & Sons 2000, -P. 217.

32 Rei-Fu Huang, Yung-Fa Chou and Cheng-Wen Wu. Defect oriented fault analysis for SRAM // Asian test symposium, ATS 2003, - pp. 256 - 261.

33 Dhillon Y., Diril U., Chatterjee A. Soft-Error Tolerance Analysis and Optimization of Nanometer Circuits // Design, Automation and Test in Europe, Proceedings 2005, - pp. 288 – 293.

34 Baumann R., Technology scaling trends and accelerated testing for soft errors in commercial silicon devices // On-Line Testing Symposium, IOLTS 2003. 9th IEEE, 2003,- pp. 4-8.

35 Mitra S., Ming Zhang, Kee Sup Kim Robust System Design with Built-In Soft-Error Resilience // Computer, Volume 38, Issue 2, Feb. 2005,- pp. 43 – 52.

36 Yamamoto S., Kokuryou K., Okada Y., Komori J., Murakami E. Neutron-induced soft error in logic devices using quasi-monoenergetic neutron beam // Reliability Physics Symposium Proceedings, 42nd Annual. IEEE International, 2004, - pp. 305 – 309.

37 Shim B., Shanbhag N.R., Lee S. Energy-efficient soft error-tolerant digital signal processing // Signals, Systems and Computers, 2003, - pp. 1493 – 1497.

38 Maheshwari A., Koren I., Burleson W. Accurate estimation of soft error rate (SER) in VLSI circuits // Defect and Fault Tolerance in VLSI Systems, 2004,- pp. 377 – 385.

39 Cannon E., Reinhardt D., Gordon M., Makowenskyj P. SRAM SER in 90, 130 and 180 nm bulk and SOI technologies // Reliability Physics Symposium Proceedings, 2004, - pp. 300-304.

40 Krishnaswamy S., Viamontes G., Markov I., Hayes J. Accurate Reliability Evaluation and Enhancement via Probabilistic Transfer Matrices // Design, Automation and Test in Europe, 2005, - pp. 282-287.

41 Bolzani L., Rebaudengo M., Reorda M., Vargas F., Violante M. Hybrid soft error detection by means of infrastructure IP cores [SoC implementation] // IEEE On-Line Testing Symposium, 2004, - pp. 79-84.

42 Azimane M., Majhi A. and et. al, New Agorithm for Dynamic faults detection in RAMs", VTS05, 2005, - pp. 445 – 452.

43 Dilillo L.and et. al. Data retention faults in SRAM memories: Analysis and Detection Procedures // VTS, 2005, pp. 183-188.

44 Dilillo L. and et. al. Resistive-Open Defect injection in SRAM core-cell: Analysis and comparision between 013um and 90nm technologies // DAC05, 2005, - pp. 37-43.

45 Dilillo L.and Al-Hashimi B. M. March CRF: an Efficient test for Complex Read Faults in SRAM memories // DDECS, 2007, - pp. 305 – 312.

46 Dubois T., Azimane M., Larsson E. Test quality analysis and improvement for an embedded asynchronous FIFO // DATA07, 2007, - pp. 256 – 363.

47 Harutyunyan G., Tshagharyan G., Vardanian V., Zorian Y. Fault Modeling and Test Algorithm Creation Strategy for FinFET-Based Memories // IEEE VLSI Test Symposium (VTS), USA, 2014, - pp. 49-54.

48 Handioui S., Ad J. van de Goor Open and Delay faults in CMOS RAM addres Decoders // IEEE Transactions on Computers, Vol. 55, No. 12, December, 2006,- pp. 1630-1639.

49 Harutyunyan G., Tshagharyan G., Zorian Y. Test and Repair Methodology for FinFET-Based Memories // IEEE transactions on device and materials reliability, VOL. 15, NO. 1, MARCH 2015, - pp. 6-9.

50 Chen Q., Mahmoodi H., Bhunia S., Roy K. Efficient Testing of SRAM With Optimized March Sequences and a Novel DFT Technique for Emerging Failures Due to Process Variations // IEEE Transactions on Very Large Scale Integration Systems, Vol. 13, Issue 11, November 2005, - pp. 1286-1295.

51 Vatajelu E., Panagopoulos G., Roy K., Figueras J. Parametric Failure Analysis of Embedded SRAMs using Fast & Accurate Dynamic Analysis // European Test Symposium, 2010, - pp. 69-74.

52 Nassif S. Modeling and Analysis of Manufacturing Variations// IEEE Conference on Custom Integrated Circuits, 2001, -pp. 223–228.

53 Mukhopadhyay S., Mahmoodi H., Roy K. Modeling of Failure Probability and Statistical Design of SRAM Array for Yield Enhancement in Nanoscaled CMOS // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 24, No. 12, December 2005, - pp. 1859-1880.

54 Agarwal K., Nassif S. The Impact of Random Device Variation on SRAM cell stability in Sub-90-nm CMOS Technologies // IEEE Transactions on Very Large Scale Integration Systems, Vol. 16, Issue 1, January 2008, - pp. 86-97.

55 Calhoun B., Khanna S., Mann R., Wang J. Sub-threshold Circuit Design with Shrinking CMOS Devices // IEEE International Symposium on Circuits and Systems, 2009, - pp. 2541-2544.

56 Mukhopadhyay S., Chen Q., Roy K. Memories in Scaled technologies: A Review of Process Induced Failures, Test methodologies, and Fault Tolerance // IEEE Design and Diagnosis of Electronic Circuits and Systems, 2007, - pp. 1-6.

57 Advanced Test Methods for SRAMs: Effective Solutions for Dynamic Fault Detection in Nanoscaled Technologies / Bosio A., Dilillo L., Girard P., Pravossoudovitch S., Virazel A. // Springer, 2010, -P. 512.

58 Harutyunyan G., Shoukourian S., Vardanian V., Zorian Y. Impact of Process Variations on Read Failures in SRAMs // IEEE East-West Design and Test Symposium, 2013, - pp. 15-18.

59 Singh J., Mathew J. A nano-CMOS process variation induced read failure tolerant SRAM cell // IEEE Circuits and Systems, 2008. ISCAS 2008, - pp. 3334-3337.

60 Yu S., Zhao Y., Du G., Kang J., Han R., Liu X. Impact of stochastic mismatch on FinFETs SRAM cell induced by process variation // EDSSC, IEEE, Electron Devices and Solid-State Circuits, 2008, - pp. 1-4.

61 Yadav N., S. Dutt, Pattnaik M., Sharma G. Double-gate FinFET process variation aware 10T SRAM cell topology design and analysis // Circuit Theory and Design (ECCTD), European Conferenc,e 2013, - pp. 64-68.

62 Yeknami A., Hansson M., Mesgarzadeh B., Alvandpour A. A low voltage and process variation tolerant SRAM cell in 90-nm CMOS // VLSI Design Automation and Test (VLSI-DAT), 2010, - pp. 78-81.

63 Moradi F., Wisland D., Berg Y., Aunet S., Tuan V. Process variations in sub-threshold SRAM cells in 65nm CMOS // Microelectronics International Conference (ICM), 2010, - pp. 371-374.

64 Qikai C., Mahmoodi H., Bhunia S., Roy K. Efficient testing of SRAM with optimized march sequences and a novel DFT technique for emerging failures due to process variations // IEEE Very Large Scale Integration (VLSI) Systems, 2005, - pp. 1286-1295.

65 Vatajelu E., Figueras J. Statistical analysis of 6T SRAM data retention voltage under process variation // Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2011, - pp. 365-370.

66 Jungseob L., Davoodi A. Comparison of Dual-Vt Configurations of SRAM Cell Considering Process-Induced Vt Variations // Circuits and Systems, 2007. ISCAS IEEE International Symposium, 2007, - pp. 3018-3021.

67 Chung-Kuan T., Marek-Sadowska M. Analysis of process variation's effect on SRAM's read stability // Quality Electronic Design, ISQED '06. 7th International Symposium, 2006, - pp. 610-615.

68 Qikai C., Mahmoodi H., Bhunia S., Roy K. Modeling and testing of SRAM for new failure mechanisms due to process variations in nanoscale CMOS // 23rd IEEE, VLSI Test Symposium, 2005, - pp. 292-297.

69 Kansal S., Lanuzza M., Corsonello P. Impact of Random Process Variations on Different 65nm SRAM Cell Topologies // Emerging Trends in Engineering and Technology (ICETET), 3rd International Conference, 2010, - pp. 703-706.

70 Gong F., Yu H., Shi Y., Kim D., Ren J., He L. QuickYield: An Efficient Global-Search Based Parametric Yield Estimation with Performance Constraints // Design Automation Conference, 2010, - pp. 392-397.

71 Dilillo L., Girard P., Pravossoudovitch S., Virazel A. Dynamic Read estructive Fault in Embedded-SRAMs: Analysis and March Test Solution // IEEE European Test Symposium, 2004, - pp. 140–145.

72 Harutyunyan G., Shoukourian S., Vardanian V., Zorian Y. Extending Fault Periodicity Table for Testing Faults in Memories under 20nm // IEEE East-West Design and Test Symposium (EWDTS), Ukraine, 2014, - pp. 12-15.

73 Champac V. et. al. Testing of stuck-open faults in nanometer technologies // IEEE Des. Test Comput., vol. 29, no. 4, 2012, - pp. 80–91.

74 Hamdioui S., J. van de Goor A., and Rodgers M. Linked faults in random access memories: Concept, fault models, test algorithms, industrial results // IEEE Trans. Comput.-Aided Des. Integr. Circuits Systems, vol. 23, no. 5, 2004, - pp. 737–756.

75 Benso A., Bosio A., Di Carlo S., Di Natale G., and Prinetto P. Automatic march test generation for static and dynamic faults in SRAMs // in Proc.IEEE Eur. Test Symp, 2005, - pp. 122–127.

76 Harutyunyan G., Shoukourian S., Vardanian V., and Zorian Y. A new method for March test algorithm generation and its application for fault detection in RAMs //

IEEE Trans. Comput.-Aided Des. Integr. CircuitsSyst., vol. 31, no. 6, 2012, - pp. 941–949.

77 Klaus M. and J. van de Goor A. Tests for Resistive and Capacitive Defects in Address Decoders // Proc. Asian Test Symp., 2001, - pp. 31-36.

78 Baosheng W., Josh Y., Yuejian W., Ivanov A. A retention-aware test power model for embedded SRAM // Design Automation Conference, Asia and South Pacific, 2005, - pp. 1180-1183.

79 Prates W., Bolzani L., Harutyunyan G., Davtyan A., Vargas F., Zorian Y. Integrating embedded test infrastructure in SRAM cores to detect aging // IEEE On-Line Testing Symposium (IOLTS), 2013, - pp. 25-30.

80 Al-Harbi S., Gupta S. Generating complete and optimal march tests for linked faults in memories // VLSI Test Symposium, 2003, - pp. 254-261.

81 Dilillo L., Girard P., Pravossoudovitch S., Virazel A. Efficient March Test Procedure for Dynamic Read Destructive Fault Detection in SRAM Memories // Journal of Electronic Testing: Theory and Applications, Vol. 21, No. 5, Oct. 2005, - pp. 551–561.

82 Dekker R., Beenker F., Thijssen L. Fault Modeling and Test Algorithm Development for Static Random Access Memories // In Proc. IEEE International Test Conference, September 1988, - pages 343–352.

83 Kolhatkar J.S., Vandamme L.K.J., Salm C., Wallinga H. Separation of Random Telegraph Signals from 1/f Noise in MOSFETs under Constant and Switched Bias Conditions // European Solid-State Device Research, ESSDERC '03. 33rd Conference, 2003, - pp.549-552.

84 Tong Boon Tang, Alan F. Murray Integrating RTS Noise into Circuit Analysis // IEEE ISCAS Circuit and Systems, 2009, - pp. 585-588.

85 Leyris C., Pilorget S., Marin M., Minondo M., Jaouen H. Random Telegraph Signal Noise SPICE Modeling for Circuit Simulators // European Solid-State Device Research, 37rd Conference, 2007, - pp. 187-190.

86 Oon Toh S., Tsukamoto Y., Guo Z., Jones L. et. al. Impact of Random Telegraph Signals on Vmin in 45nm SRAM // IEEE Electron Devices Meeting (IEDM), 2009, - pp. 1-4.

87 Press W. H., Van Der Ziel A., Horn M. et. al. Chapter 9 - 1/f Noise and Random Telegraph Signals // 2005, -P.10. http://www.nii.ac.jp/qis/first-quantum/forStudents/lecture/pdf/noise/chapter9.pdf

88 Tomasik J., Bronskowski C. and Krautschneider W. A Model for Switched Biasing MOSFET 1/f Noise Reduction // IEEE Journal, Volume: 42,_Issue: 3, 2007, - pp. 540-550.

89 Tian H. and El Gaman A. Analysis of 1/f noise in switched MOSFET circuits // IEEE Trans. Circuits Syst. II, vol. 48, Feb. 2001, - pp. 151-157.

90 Leyris C., Martinez F., Valenza M., Hoffmann A., Vildeuil J.C., Roy F. Impact of Random Telegraph Signal in CMOS Image Sensors for Low-Light Levels // IEEE Solid-State Circuits Conference, 2006. ESSCIRC 2006. Proceedings of the 32nd European ESSCIRC 2006, - pp. 376-379.

91 Leyris C., Hoffmann A., Valenza M., Vildeuil J-C and Roy F. Trap competition inducing R.T.S noise in saturation range in NMOSFETs // Fluctuation and Noise Processing, Proceedings of the SPIE, Volume 5844, 2005, - pp. 41-51.

92 Martin-Martinez J., Diaz J., Rodriguez R., Nafria M., Aymerich X. New Weighted Time Lag Method for the Analysis of Random Telegraph Signals // Electron Device Letters, IEEE Volume: 35, 2014, - pp. 479-481.

93 Kolhatkar J., Vandamme L., Salm C., Wallinga H. Separation of random telegraph signals from 1/f noise in MOSFETs under constant and switched bias conditions // European Solid-State Device Research, 2003, - pp. 549-552.

94 Miller D., Poocharoen P., Forbes L. Subthreshold Leakage Due to 1/F Noise and Rts(Random Telegraph Signals) // Microelectronics and Electron Devices, IEEE Workshop, WMED, 2007, - pp. 23-24.

95 Huang X., Lee W., Kuo C., et. al, Sub 50-nm FinFET:PMOS // International Electron Devices Meeting Technical Digest, 1999, - pp. 67-70.

96 King T.-J. FinFETs for Nanoscale CMOS Digital Integrated Circuits // IEEE-ACM International Conference on Computer- Aided Design, 2005, - pp. 207-210.

97 Bansal A., Mukhopadhyay S., Roy K. Device-Optimization Technique for Robust and Low-Power FinFET SRAM Design in Nanoscale Era // IEEE Transactions on Electron Devices, Vol. 54, No. 6, 2007, - pp. 1409-1419.

98 Jurczak M., Collaert N., Veloso A., Hoffmann T., Biesemans S. Review of FinFET Technology // IEEE International SOI Conference, 2009, - pp. 1-4.

99 Cheng H.-W., Li Y. 16-nm Multigate and Multifin MOSFET Device and SRAM Circuits // International Symposium on Next-Generation Electronics, 2010, - pp. 32-35.

100 Liu Y., Xu Q. On Modeling Faults in FinFET Logic Circuits // IEEE International Test Conference, 2012, - pp. 1-9.

101 Lin C.-W., Chao M. C.-T., Hsu C.-C. Investigation of GateOxide Short in FinFETs and The Test Methods for FinFET SRAMs // VLSI Test Symposium, 2013, - pp. 1-6.

102 Carlson A., Guo Z., Balasubramanian S., Zlatanovici R., Liu T.-J. K., Nikolić B. SRAM Read/Write Margin Enhancements Using FinFETs // IEEE Transaction on Very Large Scale Integation. Systems, Vol. 18, No. 6, 2010, - pp. 887-900.

103 Gu J.J., Liu Y.Q., Wu Y.Q., Colby R., Gordon R.G., Ye P.D. First Experimental Demonstration of Gate-All-Around III-V MOSFETs by Top-Down Approach // IEDM Tech. Dig., 2011, - pp. 769-772.
http://arxiv.org/ftp/arxiv/papers/1112/1112.3573.pdf

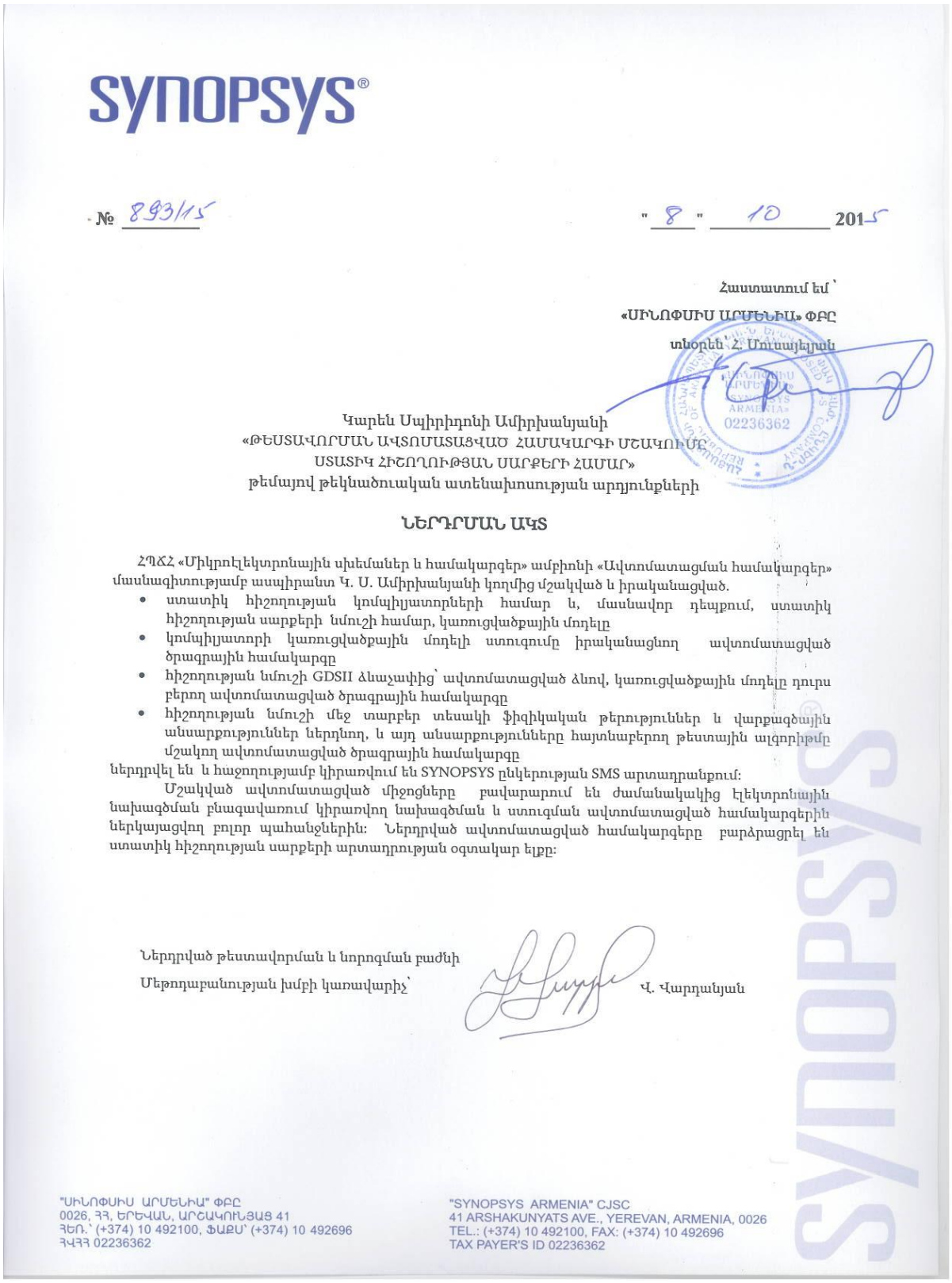
Հիմնական սահմանումներ և նշանակումներ

- Արտադրության օգտակար ելք (yield) – աշխատունակ և արտադրված սխեմաների հարաբերություն
- Թերություն (defect) – միկրոսխեմայի տոպոլոգիայում տեխնոլոգիայի խախտման արդյունքում առաջացող ֆիզիկական արատ
- Անսարքություն (fault) – թերության ազդեցությունը սխեմայի աշխատանքի վրա
- Ավելցուկային էլեմենտ – հիշողության սարքում ավելցուկային տող կամ սյուն
- Հիշողության բջջի (բառի) տրամաբանական հասցե – հիշողության բջջի (բառի) հերթական համարը
- Հիշողության բջջի ֆիզիկական հասցե (դիրք) – բջջի ֆիզիկական դիրքը, տողի և սյան համարներով, հիշողության բջիջների երկչափ զանգվածում
- Հասցեների լարանցումային խումբ (address bus) – մուտք հիշողության սխեմայի համար, որը պարունակում է դիմվող բջջի (բառի) տրամաբանական հասցեն
- B1 - Բիթային կառուցվածք ունեցող հիշողության սխեմա – հիշողության սխեմա, որի ամեն մի հասցեին համապատասխանում է մեկ հիշողության բջիջ
- Bi -Բառային կառուցվածք ունեցող հիշողության սխեմա – հիշողության սխեմա, որի ամեն մի հասցեին համապատասխանում են սահմանված թվով հիշողության բջիջներ, որոնց անվանում ենք հիշողության բառ
- ՆԱԹՀ - ներդրված ավտոմատ թեստավորման համակարգեր
- ՍՊԴՀ – Ստատիկ պատահական դիմումով հիշողություն (անգլերեն Static Random Access Memory – SRAM)
- SRAM – Static Random Access Memory
- GDSII - Graphic Database System II
- PVT - Process/Voltage/Temprature
- MOSFET - metal–oxide–semiconductor field-effect transistor
- FinFET - Fin Field Effect Transistor
- SIV – Structural Information Verification flow
- SIE - Structural Information Extraction flow

- ՂԱՀ – ղեկավարման ավտոմատացված համակարգ
- ԱՀ – ավտոմատացված համակարգ
- ԹՆևՀԹԱՍ – Թերությունների ներդրման և հիշողության թեստավորման ալգորիթմը ստուգման ավտոմատացված համակարգ
- Բիթ – հիշողության բջիջ
- ԱՕԵ – արտադրության օգտակար ելք
- ՀԲՎ – համակարգ բյուրեղի վրա
- *NW* – հիշողության սխեմայի բառերի քանակը
- *NB* – հիշողության սխեմայի մեկ բառի բիթերի քանակը
- *CM* – մեկ ուժեղացուցիչ միացվող սյուների թիվը (սյուների ապակոդավորման մուլտիպլեքսոր)
- *BK* – հիշողության ամբողջական բանկերի թիվը
- *PR* – հիշողության բջիջների երկչափ զանգվածի ֆիզիկական տողերի թիվը
- *PC* – հիշողության բջիջների երկչափ զանգվածի ֆիզիկական սյուների թիվը
- *N – Well* – բացասական կրիչներով հարստացված կիսահաղորդչային խոռոչ
- *P – Well* – դրական կրիչներով հարստացված կիսահաղորդչային խոռոչ
- ՄԱՓ – մեկանգամյա փոփոխությունները
- ՓՍՄ – «Փափուկ» սխալներ մակարդակ
- Հիշողության թեստավորում – անսարքությունների հայտնաբերում
- BIST (Built-in self-test) - հիշողության անսարքությունների հայտնաբերման ալգորիթմ
- BIRA (Built-in redundancy analysis) - հիշողության նորոգման ալգորիթմ

ՀԱՎԵԼՎԱԾ 1

Ներդրման ակտ



SYNOPSYS®

№ 893/15 "8" 10 2015

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

Կարեն Սպիրիդոնի Ամիրխանյանի
«ԹԵՍՏԱՎՈՐՄԱՆ ԱՎՏՈՄԱՏԱՑՎԱԾ ՀԱՄԱԿԱՐԳԻ ՄՇԱԿՈՒՄԸ
ՍՏԱՏԻԿ ՀԻՇՈՂՈՒԹՅԱՆ ՍԱՐՔԵՐԻ ՀԱՄԱՐ»
թեմայով թեկնածուական ատենախոսության արդյունքների

ՆԵՐԴՐՄԱՆ ԱԿՏ

ՀՊՃՀ «Միկրոէլեկտրոնային սխեմաներ և համակարգեր» ամբիոնի «Ավտոմատացման համակարգեր» մասնագիտությամբ ասպիրանտ Կ. Ս. Ամիրխանյանի կողմից մշակված և իրականացված.

- ստատիկ հիշողության կոմպիլյատորների համար և, մասնավոր դեպքում, ստատիկ հիշողության սարքերի նմուշի համար, կառուցվածքային մոդելը
- կոմպիլյատորի կառուցվածքային մոդելի ստուգումը իրականացնող ավտոմատացված ծրագրային համակարգը
- հիշողության նմուշի GDSII ձևաչափից՝ ավտոմատացված ձևով, կառուցվածքային մոդելը դուրս բերող ավտոմատացված ծրագրային համակարգը
- հիշողության նմուշի մեջ տարբեր տեսակի ֆիզիկական թերություններ և վարքագծային անսարքություններ ներդնող, և այդ անսարքությունները հայտնաբերող թեստային ալգորիթմը մշակող ավտոմատացված ծրագրային համակարգը

ներդրվել են և հաջողությամբ կիրառվում են SYNOPSYS ընկերության SMS արտադրանքում:

Մշակված ավտոմատացված միջոցները թավարարում են ժամանակակից էլեկտրոնային նախագծման բնագավառում կիրառվող նախագծման և ստուգման ավտոմատացված համակարգերին ներկայացվող բոլոր պահանջներին: Ներդրված ավտոմատացված համակարգերը բարձրացրել են ստատիկ հիշողության սարքերի արտադրության օգտակար ելքը:

Ներդրված թեստավորման և նորոգման բաժնի
Մեթոդաբանության խմբի կառավարիչ՝ Վ. Վարդանյան

"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

SYNOPSYS®

ՀԱՎԵԼՎԱԾ 2

SIV ավտոմատացված համակարգի ՝ իրական հիշողության նմուշի (NW=64, NB=8, CM=4, PR=16, PC=32) IPP-ի ֆայլի օրինակը

11111111111111110000000000000000
01111111111111110100000000000000
00111111111111110110000000000000
00011111111111110111000000000000
00001111111111110111100000000000
00000111111111110111110000000000
00000011111111110111111000000000
00000001111111110111111100000000
00000000111111110111111110000000
00000000011111110111111111000000
00000000001111110111111111100000
00000000000111110111111111110000
00000000000011110111111111111000
00000000000001110111111111111100
00000000000000110111111111111110
00000000000000010111111111111111
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

ՀԱՎԵԼՎԱԾ 3

SIV ավտոմատացված համակարգի ղեկավարման ֆայլի օրինակը

GENERAL_PARAMETERS

techlib techlib/foundry/tsmc/40nm/cln40lp

lvs_rule_deck = layout_ver/hercules/LVSA/ rel_1.3C_1 P10M_7X2Z_ALRDL_Jan312013/ tsmc_M10_7X2Z_ALRDL.lvsa

run_fast = 0

run_asv = 1

run_sdv = 0

skip_vig = 1

parallel = 16

extraction_tool = hercules

spice_simulator = hsim

#sdv_instances = m01_sdv.cfg

#asv_instances = m01_asv.cfg

#spice_version = 2011.09-SP1

#lvs_version = 2013.09-SP1

CFG_PARAMETERS # start custom configuration parameters

Freq = 250

generate_verilog_netlist = yes

generate_spice = no

generate_gds = yes

generate_plef = no

pipe_line_read = 0

select_ring = 0

#bist_enable = yes

#awt_enable = yes

instance_orientation = R0

viewselect = "gds" "verilog"

pvt_enable = 1

SIMULATION_PARAMETERS # start VPG parameters

hsim parameters

vih = 1.0v

vil = 0.0v

voh = 0.7v

vol = 0.3v

tunit = 300ns

slope = 50ps

strobe_start_offset = 4ns

strobe_end_offset = 5ns

sim_start_offset = 50

tsetup = 10

tch = 20

PINS_PARAMETERS # start ADDR, CLK, Q and RST pin assignments

#ADDR = ADRB

CLK = CLKB

#Q = QB

#RST = RST

MPT_PARAMETERS # start mpt parameters

input_gds = siv/mb41a7s_ram8t_x1_l.gds siv/mb41a7s_ram8t_x1_r.gds

pt_cells = mb41a7s_ram8t_x1_l mb41a7s_ram8t_x1_r

pt_cell = mb41a7s_ram8t_x1

dx = 1550

dy = 380

HSIM_PARAMETERS # start hsim parameters

.param HSIMSPEED=3

.param HSIMPOSTL=0

.param HSIMVDD=1

.param hsimdetailbsim4=1

.print tran v(*) level=1

.inc ' full_gds_release/ts40npk42p22sadsl512sa/models/TTQB.sp

#.prot

#.unprot

.param vh=1.0

.param vl=0

TIE_OFFS # start describing the memory pin tie-offs (the default tie-off is with VSS)

TEST1A 1.0

TEST1B 1.0

VDDP 1.0

ՀԱՎԵԼՎԱԾ 4

ՀՍ-ի նմուշի (NW=334, NB=11, CM=4, PR=84, PC=44) ՕՕ-ի ֆայլի օրինակը

```
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000001
00000000000000000000000000000000000000000010
00000000000000000000000000000000000000000011
00000000000000000000000000000000000000000100
00000000000000000000000000000000000000000101
00000000000000000000000000000000000000000110
00000000000000000000000000000000000000000111
00000000000000000000000000000000000000001000
00000000000000000000000000000000000000001001
00000000000000000000000000000000000000001010
00000000000000000000000000000000000000001011
00000000000000000000000000000000000000001100
00000000000000000000000000000000000000001101
00000000000000000000000000000000000000001110
00000000000000000000000000000000000000001111
00000000000000000000000000000000000000010000
00000000000000000000000000000000000000010001
00000000000000000000000000000000000000010010
00000000000000000000000000000000000000010011
00000000000000000000000000000000000000010100
00000000000000000000000000000000000000010101
00000000000000000000000000000000000000010110
00000000000000000000000000000000000000010111
00000000000000000000000000000000000000011000
00000000000000000000000000000000000000011001
00000000000000000000000000000000000000011010
00000000000000000000000000000000000000011011
00000000000000000000000000000000000000011100
00000000000000000000000000000000000000011101
00000000000000000000000000000000000000011110
00000000000000000000000000000000000000011111
00000000000000000000000000000000000000100000
00000000000000000000000000000000000000100001
00000000000000000000000000000000000000100010
00000000000000000000000000000000000000100011
00000000000000000000000000000000000000100100
```

00000000000000000000000000000000000000100101
00000000000000000000000000000000000000100110
00000000000000000000000000000000000000100111
00000000000000000000000000000000000000101000
00000000000000000000000000000000000000101001
00000000000000000000000000000000000000101010
00000000000000000000000000000000000000101011
00000000000000000000000000000000000000101100
00000000000000000000000000000000000000101101
00000000000000000000000000000000000000101110
00000000000000000000000000000000000000101111
00000000000000000000000000000000000000110000
00000000000000000000000000000000000000110001
00000000000000000000000000000000000000110010
00000000000000000000000000000000000000110011
00000000000000000000000000000000000000110100
00000000000000000000000000000000000000110101
00000000000000000000000000000000000000110110
00000000000000000000000000000000000000110111
00000000000000000000000000000000000000111000
00000000000000000000000000000000000000111001
00000000000000000000000000000000000000111010
00000000000000000000000000000000000000111011
00000000000000000000000000000000000000111100
00000000000000000000000000000000000000111101
00000000000000000000000000000000000000111110
00000000000000000000000000000000000000111111
00000000000000000000000000000000000001000000
00000000000000000000000000000000000001000001
00000000000000000000000000000000000001000010
00000000000000000000000000000000000001000011
00000000000000000000000000000000000001000100
00000000000000000000000000000000000001000101
00000000000000000000000000000000000001000110
00000000000000000000000000000000000001000111
00000000000000000000000000000000000001001000
00000000000000000000000000000000000001001001
00000000000000000000000000000000000001001010
00000000000000000000000000000000000001001011
00000000000000000000000000000000000001001100
00000000000000000000000000000000000001001101
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000

ՀԱՎԵԼՎԱԾ 5

SIE ավտոմատացված համակարգի ղեկավարման ֆայլի օրինակը

GENERAL_PARAMETERS

lvs_rule_deck =
/remote/proj_alg002a/pipe/techlib/foundry/smic/65nm/65ll/layout_ver/hercules/rel_1.5_nov262010/smic65ll_M10_1.5.lvsa

run_ase = 1

run_sde = 0

extraction_tool = hercules

spice_simulator = hsim

manual discription of the compiler_name if don`t specifaed in CFG file and

should be comented if it is pesent in the CFG file

compiler_name = sm65ncky1p11sadsl512sa

bankType = Reg

#bankType = Inv

#bankType = MirrShSA

#bankType = MirrCnSA

RdummyType = 2dt

RdummyType 2rb3rt

CdummyType = 1cl3cc4cc2cr

RredundType = 0

CredundType = 0

SIMULATION_PARAMETERS # start VPG parameters

hsim parameters

#check_window_before = 2ns

#check_window_after = 1ns

#check_window_unit = 1

check_window 2ns 1ns 1

vih = 1.2v

vil = 0.0v

voh = 0.84v

vol = 0.3v

tunit = 1ns

slope = 50ps

old names of the param. CLKLow CLKHigh

#strobe_start_offset = 4ns

#strobe_end_offset = 5ns

sim_start_offset = 50ns

#tsetup = 10

#tch = 20

CLKLow = 5ns

CLKHigh = 4ns

PINS_PARAMETERS # start ADDR, CLK, Q and RST pin assignments

#ADDR = ADRB

#CLK = CLKB

MPT_PARAMETERS # start mpt parameters

input_gds= /remote/proj_alg002a/TCL_test/ScrInfoExtraction/ProgCells/md82a02_ram6t_l.gds /remote/proj_alg002a/TCL_test/ScrInfoExtraction/ProgCells/md82a02_ram6t_r.gds

pt_cells = md82a02_ram6t_l md82a02_ram6t_r

#pt_cells = md82a02_ram6t_r md82a02_ram6t_l

pt_cell = me47a02_ram6t

#BITCELL_NAME : md82a02_ram6t

#pt_cell = md82a02_ram6t

dx = 1050

dy = 500

SPICE_PARAMETERS # start hsim parameters

.param HSIMSPEED=5

.param HSIMVDD=1

.param hsimdetailbsim4=1

*.print tran v(*) level=1

.lprint tran 0.3 0.84 v(*)

.protect

.lib /remote/proj_alg002a/pipe/techlib/foundry/smic/65nm/65ll/spice/logic/rel_TD_L065-SP-2001v8R_1.5_jan282011/smic65ll_logic_1.2V_1.8V_2.5V_1.5_jan282011.spice TT

.lib /remote/proj_alg002a/pipe/techlib/foundry/smic/65nm/65ll/spice/logic/rel_TD_L065-SP-2001v8R_1.5_jan282011/smic65ll_logic_1.2V_1.8V_2.5V_1.5_jan282011.spice DIO_TT

.lib /remote/proj_alg002a/pipe/techlib/foundry/smic/65nm/65ll/spice/bitcell/rel_4R_1.1_jul092010/smic65ll_0525um_1.2v_hvtsram_OR_1.1_dec152010.spice TT

.unprot

.param vh=1.2

.param vl=0

TIE_OFFS # start describing the memory pin tie-offs (the default tie-off is with VSS)

*MEB 1.2

*MEENABLE 1.2

*PIPEMEB 1.2

ՀԱՎԵԼՎԱԾ 6

DI ավտոմատացված համակարգի ղեկավարման ֆայլի օրինակը

resistor_min 1

#resistor_max 95000

resistor_max 0

iteration_number 0

defect_injection_mode = mpt

#defect_injection_mode = manual

#defect_injection_mode = spice

lvs_rule_deck=

/remote/proj_alg013/lvs/rel_1.0_1a_1p10M_7X2R_ALRDL_oct312011/tsmc_M10_7X2R_ALRDL_1.0_1A.lvsa

extraction_tool = hercules