**ՀԱՄԱՌՈՏԱԳԻՐ**

**ԹԵՄԱ՝ Թվային սխեմաների իրադարձային տրամաբանական մոդելավորման միջոցի մշակումը և հետազոտումը**

Կարապետյան Ռազմիկ Կարենի

Սույն մագիստրոսական ատենախոսության նպատակն է մշակել թվային ինտեգրալ սխեմաների իրադարձային տրամաբանական մոդելավորման միջոց, որը հնարավորություն կտա նախագծել և ուսումնասիրել թվային ինտեգրալ սխեմաները օգտագործելով օբյեկտ կողմնորոշված ծրագրավորման գաղափարները։  
Ատենախոսության ընթացքում հետազոտվել է մշակված ծրագրային միջոցի տրամաբանական մոդելավորման ժամանակի՝ սխեմայում գտնվող էլեմենտների քանակից կախվածությունը և առկա մոդելավորման միջոցների հետ համեմատումը։

Բովանդակություն

[ԳԼՈՒԽ 1 6](#_Toc8382493)

[ՆԵՐԱԾՈՒԹՅՈՒՆ 7](#_Toc8382494)

[ԳԼՈՒԽ 2 11](#_Toc8382495)

[ԳՐԱԿԱՆՈՒԹՅԱՆ ԱԿՆԱՐԿ 12](#_Toc8382496)

[2.1 Մոդելավորման ֆունկցիոնալ-տրամաբանական և սխեմատեխնիկական մակարդակները 12](#_Toc8382497)

[2.1.1 Թվային սխեմաների մոդելավորման մակարդակները 17](#_Toc8382498)

[2.2. Տրամաբանական մոդելավորման հայտնի մեթոդները 18](#_Toc8382499)

[2.2.1 Սինխրոն տրամաբանական մոդելավորում 20](#_Toc8382500)

[2.2.2 Ասինխրոն տրամաբանական մոդելավորում 22](#_Toc8382501)

[2.2.3 Ասինխրոն իրադարձային մոդելավորում 25](#_Toc8382502)

[2.2.4 Բազմարժեք մոդելավորման մեթոդը 27](#_Toc8382503)

[2.3. Իրադարձային տրամաբանական մոդելավորում 27](#_Toc8382504)

[2.3.1 Հիմնական գաղափարները 29](#_Toc8382505)

[2.3.2 Խափանումները տրամաբանական սխեմաներում 32](#_Toc8382506)

[2.3.3 Տրամաբանական սխեմաների սկզբնարժեքավորման խնդիրները 33](#_Toc8382507)

[ԳԼՈՒԽ 3 36](#_Toc8382508)

[Խնդրի դրվածքը 37](#_Toc8382509)

[ԳԼՈՒԽ 4 38](#_Toc8382510)

[Տեսական ա ռնչություններ 39](#_Toc8382511)

[4.1 Մշակված ծրագրային միջոցը 39](#_Toc8382512)

ԳԼՈՒԽ 1

**Ներածություն**

ՆԵՐԱԾՈՒԹՅՈՒՆ

Վաղ ժամանակներից սխեմայի նախագծի ստուգումը թվային թվային սխեմաների նախագծման գործընթացի կարեւորագույն մասն է կազմում։  
 Պատճառը պարզ է։ Գերարդյունավետ է ստուգել նախագծի ճշգրտությունը նախքան արտադրելը, քան վերանորոգել կամ վերակառուցել հազարավոր սխալ արտադրված սխեմաներ: Ոչ վաղ անցյալում ստուգումը կատարվել էր փաստացի նախատիպ կառուցելով արտաքին միացումներով փոխկապակցված բաղադրիչների միացումով: Այն ժամանակ նախատիպն էր օգտագործվում գնահատելու սխեմայի տրամաբանական ճշգրտությունը եւ ժամանակային բնութագրերը:Այս մեթոդը անհարմար դարձավ թվային սխեմաների չափերի պայթյունավտանգ աճով: Գերմեծ ինտեգրալ սխեմայի բաղադրիչների քանակը կարող է հասնել հարյուր միլիոնավոր տարրերի, միաժամանակ մեծացնելով սխեմայի բարդությունը:

Այն դարձել է շատ ծախսատար եւ ժամանակ սպառող կառուցելու համար նախատիպեր սխեմաների համար։ Այս գործոնները առաջ քաշեցին ավտոմատացված նախագծում անող գործիքներ ստեղծելու խնդիրը։ Դիզայնի ֆիզիկական նախատիպավորման համար կենսունակ փոխարինող գործիքը դարձավ մոդելավորման նիջոցները: Մոդելավորման միջոցը հնարավորություն է տալիս նախագծողին տեսնել, թե ինչպես կպահի նախագիծը իրականում, միաժամանակ հաստատելով դիզայնը հաճախորդի առաջադրած առանձնահատկությունների հետ: Այն թույլ է տալիս հայտնաբերել եւ չափել այն իրադարձությունները, որոնք կարող են շատ լինել դժվար է կամ անհնար է հայտնաբերել փաստացի համակարգում։ Մոդելավորման միջոցը նաեւ հնարավորություն է տալիս միացում դիզայներին նախագծային գործընթացում տարբեր գաղափարներ կիրառել փորձարկելու եւ օպտիմալացնելու համար դիզայնը։ Էլեկտրոնային սարքերի բարդությունը հասել է այնպիսի մակարդակի, որը նույնիսկ ոչ մի մոդելավորման միջոց չի կարող կարգավորել մոդելավորման բոլոր ասպեկտները։ Արդյունքում, տարբեր տեսակի մոդելավորման միջոցներ հայտնվեցին տարբեր ոլորտների խնդրիների լուծման համար: Մոդելավորման միջոցները դասակարգելու ամենատարածված ձեւը հիմնված է նրանց թվային համակարգի աբսրակցիայի մակարդակի վրա։ Հիմնականում կարելի է դիտարկել հետեվյալ հինգ տեսակները։

Վարքագծային մոդելավորման միջոցները գտնվում է ամենաբարձր մակարդակով: Այս մակարդակում համակարգը մոդելավորվում է կատարող ալգորիթմների առումով, եւ ընդգծում է ընդհանուր համակարգի կայունությունը:

Հաջորդ մակարդակում ֆունկցիոնալ մոդելավորման միջոցն է: Նաեւ կոչվում է ռեգիստր փոխանցման մակարդակ։ Այն օգտագործվում է տվյալների հոսքի եւ հսկողության ազդանշանների ներդաշնակեցման համար ֆունկցիոնալ բլոկների միջեւ, ինչպիսիք են ռեգիստրները, կոդավորիչները, ապակոդավորիչները, թվաբանական-տրամաբանությունը միավորները և այլն։

Հիերարխիայում հաջորդը տրամաբանական մոդելավորման միջոցն է, որը կոչվում է նաև տրամանական բանալիների մակարդակի մոդելավորման միջոց, որն արտացոլում է անջատման տարրերի կամ տրամաբանական փականների փոխկապակցումը համակարգում: Այստեղ ուշադրության կենտրոնում է սխեմայի տրամաբանական ճշգրտությունը ստուգելը։ Այս տեսակի մոդելավորման միջոցները կոչվում է դիզայնի ստուգման մոդելավորման միջոց:

Տրանզիստոր/էլեկտրական մակարդակի մոդելավորման միջոցը զբաղվում է տրամաբանական բանալին տրանզիստորների բաժանելով և նրա վարքագիծը ստուգելով։

Ցածր մակարդակի վրա է երկրաչափական մակարդակի մոդելավորման միջոցը, որը մոդելավորում է սխեման ֆիզիկական մարմինների առումով։

Մոդելավորումը ամենաբարձր մակարդակում պահանջում է ավելի մանրակրկիտ մշակում, հետեւաբար մոդելավորման մեծ արագություն։ Այնուամենայնիվ, տեղեկատվության կորուստը կարող է դժվարացնել հասկանալը սխեմայի վարքագծի։

Ժամանակակից ավտոմատացված նախագծման համակարգերի թվային սխեմաների վերլուծության ենթահամակարգի հիմքը կազմում են տրամաբանական մոդելավորման ծրագրերը: Տրամաբանական մոդելավորումը թույլ է տալիս նախագծվող սխեմայի մոդելի օգնությամբ նրա տրամաբանական-ժամանակային վարքագծի պատկերի ստացումը: Այդ պատկերի հիման վրա կարելի է ստուգել սխեմայի տրամաբանական կառուցվածքը մինչև սխեմայի իրականացումը, որոշել օգտագործվող էլեմենտային բազային ներկայացվող պահանջները, համեմատել թվային սխեմայի իրականացման լուծումների տարբերակները հուսալիության, արագագործության և այլ տեսանկյուններից:

Ի վերջո կարելի է ասել, որ տրամաբանական մոդելավորումը բաղկացած է երկու փուլից`

* թվային սխեմայի տրամաբանական մոդելի կառուցում
* տված մուտքային ազդանշանների համար սխեմայի տրամաբանական մոդելի ժամանակային դիագրամների հաշվարկում:

Գոյություն ունեցող մոդելավորման համակարգերը ստեղծված են եղել ավելի փոքր ինտեգրացման աստիճան ունեցող սխեմաների համար և չեն բավարարում ներկայիս պահանջներին՝ մոդելավորման արագության և անհրաժեշտ ճշտության տեսանկյունից, կամ վերը նշված պահանջների (արագագործություն, մոդելավորման ճշտություն) հետ, տրամաբանական մոդելավորման ժամանակ առաջ են գալիս նաև օժանդակ՝ այլ հնարավորությունների ավելացման խնդիրներ, որը բերում է նաև այդ պահանջների լուծմանը: Այդ խնդիրներից է թվային սխեմաների մեծ չափողականության հետևանքով նախագծման ընթացքում հնարավոր բաց թողումների առկայությունը: Այսինքն նախագծման ընթացքում հնարավոր է սխեմայում թողնվեն առանձին մասեր, որոնք չեն մասնակցում սխեմայում կատարվող փոխանջատումներին: Այդ մասերի սխեմայում թողնելը բերում է մասնավորապես տնտեսական կորուստների և մոդելավորման խնդրի բարդացմանը: Այս խնդիրը էլ ավելի կարևոր է դառնում, երբ էլեմենտների չափսերի փոքրացմանը զուգահեռ մոդելավորման մեջ հաշվի է առնվում կապի գծերի պարամետրերը, որը բերում է նախկին մեթոդների վերանայմանը, նոր մեթոդների մշակմանն ու իրագործմանը:

Տվյալ մագիստրոսական աշխատանքի շրջանակներում կուսումնասիրվեն տրամաբանական մոդելավորում անող առկա լուծումները, ու կմշակվի իրադարձային տրամաբանական մոդելավորում անող միջոց գրաֆիակակն ինտերֆեյսով։

ԳԼՈՒԽ 2

**Գրականության ակնարկ**

ԳՐԱԿԱՆՈՒԹՅԱՆ ԱԿՆԱՐԿ

2.1 Մոդելավորման ֆունկցիոնալ-տրամաբանական և սխեմատեխնիկական մակարդակները

Ֆունկցիոնալ-տրամաբանական նախագծումը հանդիսանում է հաշվողական համակարգերի նախագծման առանձին մակարդակ, որի ժամանակ, ինչպես վերը նշեցինք, մշակվում են սարքերի ֆունկցիոնալ և սկզբունքային սխեմաները (օր. պրոցեսորներ, հիշող սարքեր, և տարբեր տիպի այլ սարքեր):

Եթե հետևենք խնդրի լուծման ժամանակային հաջորդականությանը, ապա ֆունկցիոնալ-տրամաբանական նախագծումը հաջորդում է սխեմատեխնիկական նախագծման փուլին: Այդ պատճառով ֆունկցիոնալ-տրամաբանական նախագծման մուտքային տվյալներ են հանդիսանում սխեմատեխնիկական նախագծման արդյունքները. սարքերի համախումբը (կազմը) և նրանց ելքային պարամետրերի նկատմամբ եղած պահանջները, ալգորիթմների մասին ինֆորմացիան, տվյալ սարքերի ճարտարապետությունը և այլն:

Ֆունկցիոնալ-տրամաբանական նախագծման հիմնական խնդիրը ֆունկցիոնալ և սկզբունքային սխեմաների մշակման խնդիրն է: Նախագծման ժամանակ այդ սխեմաները պետք է արտացոլեն օգտագործվող միկրոսխեմաների կազմը, ինչպես նաև նրանց միջև կապերը:

Միկրոսխեմաների մեջ կարող են լինել ինչպես ցածր (օր. փականներ և տրիգերներ), այնպես էլ բարձր և միջին ինտեգրացման աստիճանի սխեմաներ (օր. հաշվիչներ, տարբեր ռեգիստրներ, կիսահաղորդչային հիշող սարքեր, տրամաբանական ծրագրավորվող մատրիցաներ և այլն ):       Մեծ ինտեգրալ սխեմաների մոդելավորման ժամանակ նախագծվող սխեմաների տարրեր են հանդիսանում ֆունկցիոնալ տարրերը ( փականները, որոնք իրականացնում են պարզ բուլյան ֆունկցիաներ, հիշող տարրերը, որոնցից են տարբեր տիպի տրիգերները, անալոգային սխեմաները, օր.` օպերացիոն ուժեղացուցիչները ):

Հաշվողական համակարգերի ֆունկցիոնալ և սկզբունքային սխեմաների նախագծման և մշակման գործընթացը իր մեջ ներառում է սինթեզի  և վերլուծության խնդիրների լուծումները.

 Սինթեզի խնդիրը , որպես կանոն, հնարավոր չէ լուծել ավտոմատ փաթեթով երկու պատճառով.

1)            ոչ բոլոր պրոցեդուրաներն են ձևայնացված ,

2)            մեծ թվով ձևայնացված պրոցեդուրաների համար հայտնաբերված չեն արդյունավետ ալգորիթմներ, որոնց վերջնական տեսքը մոտ կլինի գծային կախվածությամբ լուծման խնդիրներին:

Մեծ թվով սինթեզի ալգորիթմների համար մեքենայական ժամանակի ծախսերը աճում են ոչ գծային (օր. էքսպոնենցիալ) ձևով` սինթեզվող օբյեկտի բարդության աստիճանին զուգահեռ: Այդպիսի ալգորիթմները կարող են օգտագործվել միայն հարաբերականորեն պարզ սխեմաների հատվածների սինթեզի ժամանակ:

Նրանց օգտագործման համար անհրաժեշտ է սխեման նախապես ենթարկել տարանջատման: Ֆունկցիոնալ տրամաբանական նախագծման հիմնական խնդիրները, որոնք համեմատաբար ավելի ձևայնացված են, դրանք վերլուծության խնդիրներն են:

Վերլուծության հիմնական նպատակը սխեմայի առաջարկվող տարբերակի որակի գնահատումն է: Այդ գնահատականը կարող է լինել բազմամակարդակային:

Նախապես նպատակահարմար է ստուգել սխեմայի համապատասխանությունը տրված ֆունկցիաներին` առանց հաշվի առնելու ազդանշանների հապաղումը, տարրային բազայի սահմանափակումը և այլ տարբեր ապակայունացնող գործոններ: Այդպիսի ստուգումը կարող է կատարվել առանց մեքենայական ժամանակի զգալի կորուստների և թույլատրում է հայտնաբերել սինթեզի ժամանակ կառուցվածքում թույլ տրված սխալները: Հայտնաբերված սխալների վերացումից հետո կարելի է շարունակել վերլուծությունը` անցնելով սխեմայի ավելի կոնկրետ մոդելների օգտագործմանը, որոնք հաշվի են առնում տարրերի հապաղումները, տարբեր ապակայունացնող գործոններ և այլն:

Վերլուծության խնդրի այս փուլի ժամանակ հնարավորություն է ընձեռնվում հայտնաբերելու ազդանշանների կրիտիկական շեղումը, որն առաջանում է ասինխրոն սխեմաներում, ինչպես նաև խափանման այլ պատճառներ: Քանի որ վերլուծությունը կարող է կատարվել սխեմաների մի քանի համեմատվող տարբերակների համար, հնարավորություն է ընձեռնվում ընտրելու դրանցից լավագույնը ( օպտիմալ ), օր. արագագործության տեսանկյունից: Այդպիսով վերլուծության հիման վրա որոշվում է սխեմայի սինթեզի խնդիրը: Ֆունկցիոնալ տրամաբանական նախագծման ուրույն խնդիրը թեստերի սինթեզի խնդիրն է: Թեստերը անհրաժեշտ են սարքավորումների հսկման համար, դրանց պատրաստման եւ շահագործման ընթացքում: Թեստային հսկումը պետք է տա հավաստի եւ ամբողջական արդյունքներ: Այդ պատճառով արդյունավետ թեստերի ստեղծումը պատկանում է քոմփյութերային համակարգերի վրա լուծում պահանջող բավականին բարդ խնդիրների դասին:

Տրամաբանական և սխեմատեխնիկական մոդելավորման մակարդակները սկսեցին զարգանալ իրարից անկախ 50-ական թվակաների վերջերում: Այդ ժամանակահատվածում ստեղծվեցին մեծ քանակությամբ մոդելներ, մեթոդներ, ալգորիթմներ այդ թվում նաև ծրագրային փաթեթներ, որոնք այդ ժամանակ լուծեցին թվային սխեմաների նախագծողների  համար կոնկրետ խնդիրներ: Անգամ ընդունված է առ այսօր թվային համակարգերի  ստեղծված միջոցները բաժանել երկու սերնդի: Թվային սխեմաների մոդելավորման միջոցների զարգացման ընթացքի վերլուծությունը ցույց է տալիս, որ հիմնական շարժիչ գործոնն էր հակասությունը հաշվարկների արդյունքների ճշտության պահանջների և մեքենայական միջոցների անհրաժեշտ ծախսերի միջև: Զարգացման ամեն փուլում ստեղծվում էր այն իրադրությունը, երբ օգտագործվող թվային սխեմաների վերլուծության միջոցները միայն մասնակիորեն էին բավարարում թվային սարքավորումների նախագծողների պահանջներին: Այդ պատճառով արդեն գոյություն ունեցող մոդելավորման համակարգերի համար փնտրվում էին կատարելագործման նոր ճանապարհներ: Այդ իրավիճակի պատճառն հանդիսացան թվային սխեմաների պատրաստման տեխնոլոգիաների առաջ ընկնող տեմպերը մոդելավորման միջոցների նախագծման ոլորտի նկատմամբ: Տվյալ միտումը դարձավ ավելի նկատելի ինտեգրալ տեխնոլոգիային անցման հետևանքով և առավել սրվեց մեծ ինտեգրացիայի աստիճան ունեցող սխեմաների առաջ գալուն պես:

Միկրոսխեմաների պատրաստման ամեն մի նոր տեխնոլոգիայի անցումը թվային սխեմաների սխեմատեխնիկական և տրամաբանական մոդելավորման միջոցների նախագծողների համար բերում էր երկու կարևորագույն պրոբլեմների.

1.  Նախագծվող սարքերի էլեմենտների քանակը կտրուկ, հաճախ միանգամից մի քանի կարգով, աճում էր և տվյալ պահին ներկա վերլուծության միջոցները անկարող էին գտնվում մոդելավորել տվյալ քանակության էլեմենտներով սխեմա:

2.  Տեղի էր ունենում տարբեր ֆիզիկական երևույթների որակական վերագնահատում, որոնք տեղի էին ունենում թվային սխեմայում նրա   գործունեության ժամանակ: Երևույթները, որոնց կարելի էր անտեսել մինչև նոր տեխնոլոգիայի մշակումը, դառնում էին որոշիչ նոր տիպի սխեմաների գործունեության համար, իսկ գոյություն ունեցող վերլուծության համակարգերը նախատեսված չէին տվյալ հաշվարկների համար:

Վերլուծության տրամաբանական մակարդակի ժամանակ, հատկապես իրադարձային մոդելավորման մտքի առաջարկումից հետո, չափողականության պրոբլեմը ոչ այդքան սուր էր դրված, ինչպես սխեմատեխնիկական վերլուծության ժամանակ: Հիմնական պատճառը բուլյան հանրահաշվի վրա հիմնված թվային էլեմենտների պարզ տրամաբանական մոդելների օգտագործումն էր, որոնք հաշվարկի համար չէին պահանջում բարդ լուծումներ: Էլեկտրական վերլուծության համար միշտ առավել կարևոր պրոբլեմ էր հանդիսանում լուծվող դիֆերենցիալ հավասարումների համակարգի չափողականությունը: Հաշվարկների քանակի  ոչ գծային կախվածությունը սխեմայում էլեմենտների քանակից բերում է մեքենայական ժամանակի անթույլատրելի ծախսերի ժամանակակից ՄԻՍ-երի սխեմատեխնիկական մոդելավորման ժամանակ: Հաշվարկների թույլատրելի ժամանակի ընթացքում լավագույն սխեմատեխնիկական մոդելավորման ծրագրերի օգնությամբ կարելի է վերլուծել միայն սխեմաներ, որոնք պարունակում են ոչ ավել, քան 200-300 տրանզիստորներ: Եվ այդ ամենը տեղի է ունենում անգամ եթե այդ ծրագրերում կիրառվում են բոլոր առաջադեմ մտահաղացումները, որոնք առաջացել են սխեմատեխնիկական մոդելավորման զարգացման ընթացքում (դիֆերենցիալ հավասարումների համակարգերի, մակրոմոդելավորման, իրադարձային  և կառուցվածքային տարանջատման, ժամանակային և նիշային վերլուծության, զուգահեռ մոդելավորման  նոր, ավելի արագ թվային մեթոդներ):

Մոդելավորվող սխեմայի չափողականության նվազեցման նշված զարգացման ուղիներից առավել արդյունավետ հանդիսացավ խառը տրամաբանական-էլեկտրակական մոդելավորման մտահաղացումը: Այն հանգում է նրան, որ կախված օգտագործողի կողմից պահանջվող սխեմայում տեղի ունեցող պրոցեսների արտացոլման մանրակրկիտության աստիճանից, տարբեր դրվագները մանրամասնացվում են մինչև տարբեր մակարդակներ՝ էլեկտրական կամ տրամաբանական:

էլեկտրական մակարդակը ներկայացնում է դրվագներ, որոնք պահանջում են ազդանշանների ձևի առավել մանրակրկիտ հաշվարկներ, նկարագրվում է դիֆերենցիալ հավասարումների համակարգերի օգնությամբ և լուծվում է սխեմատեխնիկական վերլուծության հասարակ միջոցներով: Տրաբանական մակարդակը իր մեջ ներառում է սխեմաների դրվագներ, որոնցում պետք է հստակ մոդելավորել ժամանակային հարաբերակցությունները, իսկ ազդանշանի ձևը ունի երկրորդական նշանակություն: Տրաբանական մոդելավորումը տրվում է բուլյան հավասարումների համակարգով կամ էլ ռեգիստրային փոխանցումների տերմիններով և հաշվարկվում է տրամաբանական մոդելավորման մեթոդներով: Այդպիսով, մեկ ծրագրում համագործակցում են սկզբունքային տարբեր ալգորիթմներ՝  դիֆերենցիալ հավասարումների  համակարգերի լուծման թվային մեթոդներ, որոնք նկարագրում են էլեկտրական մակարդակը, և տրամաբանական հավասարումների լուծում տրաբանական մակարդակի համար: Տարբեր բնույթի ազդանշանների փոխադարձ վերաձևավորումները իրագործվում են մուտքի և ելքի մոդելների միջոցով: Ըստ էության, տեղի է ունենում փոխազդում մոդելավորման երկու մակարդակների (սխեմատեխնիկական և տրամաբանական) առավելությունների և թերությունների միջև: Տրաբանական մակարդակում իրադարձային տրամաբանական մոդելավորման օգտագործումը խառը վերլուծության ծրագրում ապահովում է մեքենայական միջոցների (ժամանակ և հիշողություն) էական կրճատում: Կրճատումը հասնում է մինչև 1-3 կարգի սխեմատեխնիկական վերլուծության համեմատ: Բայց դա նույնպես չի բավարարում ժամանակակից պահանջներին:

Գոյություն ունեցող տրամաբանական մոդելավորման միջոցները նախատեսված չեն ֆիզիկական երևույթների գնահատմանը (օրինակ` ազդանշանների պարազի­տային երևույթներ): Այդ պատճառով խառը տրամաբանական-էլեկտրական մոդելավորման ծրագրերի ստեղծման ժամանակ հիմնական ուշադրությունը ուղղված էր սխեմատեխնիկական մոդելների զարգացման վրա նրանց՝ տրամաբանականների հետ համատեղ գործունեության տեսանկյունից, հետևաբար այդ երևույթները այժմ հաշվի են առնվում:

Բացի այդ, տվյալ փաթեթներում հաճախ չեն նախատեսվում լրացուցիչ, մեծամասամբ կարևոր և օգտակար ֆունկցիաների կիրառում (օրինակ` սխեմաների նախորդ մոդելավորման արդյունքների մասին ինֆորմացիայի հավաքում):

Սակայն այդ խնդիրները լուծելու համար նախ և առաջ ծանոթանանք մոդելավորման հայտնի մեթոդներին:

2.1.1 Թվային սխեմաների մոդելավորման մակարդակները

            Մոդելավորման համակարգային մակարդակում մշակվում է հաշվողական սարքը, որը բաղկացած է մեծ բլոկերից, օրինակ՝ պրոցեսոր, օպերատիվ հիշողություն, տվյալների հաղորդման միջոցներ և այլն: Այդ մակարդակում, որտեղ մոդելավորման բնորոշ խնդիրներ են հանդիսանում առանձին բլոկերի ինֆորմացիոն համաձայնությունը և այլն, կիրառվում է զանգվածային սպասարկման տեսությունը:

            Ֆունկցիոնալ-տրամաբանական մակարդակում մոդելավորվում է թվային սարքը, որը մանրամասնեցվում է մինչև բազմակարգային էլեմենտները (ռեգիստրներ, հաշվիչներ, վերծանիչներ և այլն) կամ էլ ներկայացվում է առանձին փականներով և տրիգերներով (տրամաբանական ենթահամակարգ): Այդ դեպքում վերլուծության խնդիրներից են հանդիսանում միկրոծրագրի գործունեության տրամաբանության ստուգումը, առանձին էլեմենտների աշխատանքի ժամանակային համաձայնեցումը, սխեմայի սահմանային արագագործության որոշումը, սխեմայի գործունեության տարբեր տիպի խափանումների հայտնաբերումը և այլն: Տրամաբանական մոդելավորման  համակարգի կառուցման հիմքում ընկած է բուլյան հանրահաշիվը: Թվային սխեմաների առավել  հայտնի տրամաբանական մոդելավորման ծրագրերից են՝ HILO, ADM, VERILOG և այլն:

Վերլուծության սխեմատեխնիկական մակարդակում ստուգվում է սխեմայի փոքր դրվագների գործունեությունը, որոնք մանրամասնեցվում են մինչև առանձին էլեկտրոնային բաղադրիչների՝ տրանզիստորների, դիոդների, դիմադրությունների և այլն: Ստացվում են սխեմայի հանգույցներում լարումների և հոսանքների փոփոխման մանրամասն ձևերը, որոնց հիման վրա կարելի է լուծել այնպիսի խնդիրներ, ինչպիսիք են առանձին էլեմենտների էլեկտրական համաձայնեցումը, զգայունության վերլուծությունը սխեմայի այս կամ այն բաղադրիչի արժեքների փոփոխության նկատմամբ, հաճախականային բնութագրերի ստացումը և այլն:

            Բաղադրիչային մակարդակի վրա մոդելավորվում են կիսահաղորդչային կառուցվածքի առանձին դրվագներ (հաճախ՝ առանձին տրանզիստորը), որոնք ներկայացվում են որպես ֆիզիկա-տոպոլոգիական օբյեկտներ: Կիսահաղորդչային կառուցվածքների տեսության հիման վրա  հաշվարկվում են բաղադրիչների էլեկտրական պարամետրերը և նրանց մեջ կատարվող ֆիզիզկական պրոցեսները: Մոդելավորման բաղադրիչային մակարդակի մոդելների իրագործումը կատարվում է մասնակի ածանցյալներով դիֆերեցիալ հավասարումների համակարգերի միջոցով:

Նախագծողների մոտ հիմնական և առավել հայտնի թվային սխեմաների մոդելավորման մակարդակներից են համարվում տրամաբանականը և սխեմատեխնիկականը:

Այդ պատճառով ավելի մանրամասն անդրադառնանք այդ մակարդակներին:

2.2. Տրամաբանական մոդելավորման հայտնի մեթոդները

Տրամաբանական մոդելվորման ընթացքում կարող են լուծվել տարբեր խնդիրներ . սխեմաների աշխատանքի տրամաբանության ստուգում, անցողիկ  պրոցեսների վերլուծում, սխեմաների աշխատանքի հուսալիության որոշում էլեմենտների պարամետրերի ցրումից կախված, թեստերի գեներացիա և այլ: Դրված խնդրից կախված ընտրվում է մոդելավորման մեթոդը: Մեթոդները հիմնականորեն իրարից տարբերվում են  հետևյալով՝  սխեմայում ազդանշանի տարածման ժամանակի հաշվման ձևերով, ազդանշանի կոդավորման ձևերով, քոմփյութերում մոդելի կառուցման ձևերով, էլեմենտների մոդելավորման հերթականությամբ:

Սխեմայում ազդանշանի տարածման ժամանակի հաշվման ձևերից կախված մեթոդները լինում են սինխրոն (առանց սխեմայի էլեմենտների հապաղումները հաշվի առնման) և ասինխրոն (հապաղումները հաշվի առնելով): Ազդանշանի կոդավորման ձևերից կախված տարբերում են երկարժեք և բազմարժեք մոդելավորման մեթոդներ: Էլեմենտների մոդելավորման հերթականությունից կախված՝ միջանցիկ և իրադարձային: Մոդելավորման ծրագրի աշխատանքի կազմակերպման ձևերից կախված՝ կոմպիլյատիվ և ինտերպրետատիվ մեթոդներ:

Սինխրոն մոդել կոչվում է, այն մոդելը, որի ժամանակ հաշվի չեն առնվում էլեմենտների ֆիզիկական հատկությունները, օրինակ էլեմենտի հապաղման ժամանակը: Դա հաշվի է առնվում ասինխրոն մոդելների օգտագործման ժամանակ: Եթե սինխրոն մոդելում օգտագործվում են բուլյան փոփոխականներ, ապա մոդելը կոչվում է երկարժեք:

Երկարժեք մոդելները մեքենայական միջոցների տնտեսման տեսանկյունից ամենաշահավետն են, բայց այդ մոդելների օգնությամբ կարելի է լուծել խնդիրների նեղ շրջանակ, օրինակ ֆունկցիոնալ սխեմայի նախագծման ընթացքում կատարած կոպիտ սխալների հայտնաբերում:

   Անցողիկ պրոցեսների վերլուծության ընթացքում անհրաժեշտ է հաշվի առնել սխեմայում ազդանշանի տարածման վերջավոր արագությունը: Սխեմայի ցանկացած տրամաբանական էլեմենտի փոխանջատումը կատարվում է ոչ թե ակնթարթորեն, այլ որոշակի հապաղումից հետո նրա մուտքերի ազդանշանների փոփոխման պահի նկատմամբ: Հապաղումով է տարածվում նաև ազդանշանը սխեմայի կապի գծերով:

   Ազդանշանի հապաղումը տրամաբանական էլեմենտում կախված է տվյալ էլեմենտի ֆիզիկական իրագործման, սխեմայում միացման, մուտքային ազդանշանների համակցությունների, ջերմաստիճանի, պարամետրերի ցրումի ձևերից: Հետևաբար ազդանշանի հապաղումը ունի պատահական արժեք: Շղթաներով ազդանշանի տարածման տարբեր հապաղումները բերում են ազդանշանի ձևափոխությանը էլեմենտների փոխանջատման ընթացքում:

   Սխեմայի անցողիկ պրոցեսների վերլուծության նպատակները կարող են լինել տարբեր, բայց հիմնականում դա սխեմայի վարքագծի որոշումն է տրված մուտքային ազդանշանների դեպքում: Դա հատկապես կարևոր է ասինխրոն սխեմաների համար, որոնց վարքը կախված է ազդանշանների հապաղումներից:

   Ֆունկցիոնալ էլեմենտների և նյութերի, օգտագործվող կառուցվածքային հանգույցների պատրաստման համար, պարամետրերի ցրման պատճառով սխեմայի արձագանքը նույն մուտքային համակցության նկատմամբ կարող է լինել տարբեր: Սխեմաները պետք է նախագծվեն այնպես, որ պատրաստման պարամետրերի ցրումից անկախ սխեմաները նույն մուտքային ազդանշանների դեպքում միշտ անցնեն նույն վիճակներին: Եթե օգտագործվող մուտքային ազդեցության համար կա ոչ 0-ական հավանականություն, որ սխեման կանցնի տարբեր վիճակների, ապա սխեման նախագծված է սխալ:

   Կառուցվածքային հանգույցների սխեմաների համաձայնեցման ընթացքում պետք է հայտնի լինի կառուցվածքային հանգույցի ելքային ազդանշանների ժամանակային շեղումը: Այդպիսի արդյունքներ վերլուծության ընթացքում ստանալու համար անհրաժեշտ է ֆիքսել էլեմենտի փոխանջատման պահը:

   Էլեմենտի մուտքային ազդանշանի անհամաձայնեցման դեպքում կարող է առաջանալ կեղծ ազդանշան տրամաբանական էլեմենտի ելքին: Կեղծ ազդանշանների առաջացման հնարավորությունը կոչվում է խափանման ռիսկ: Եթե ազդանշանները սխեմայի ելքին երկու կից A և B մուտքային ազդեցությունների հավաքածուի տակ մնուն են նույնը, բայց անցողիկ պրոցեսի ընթացքում հնարավոր է հակառակ արժեքի կեղծ ազդանշանի առաջացումը, ապա այդպիսի դրությունը կոչվում է ստատիկ խափանման ռիսկ:  Այսինքն խափանման ռիսկը կեղծ ազդանշանների հայտնաբերման միջոց է: Խափանման ռիսկի որոշման համար ևս օգտագործում են տարբեր մոդելներ` բազմարժեք մոդելներ

   Դինամիկ խափանման ռիսկը ենթադրում է ելքում ազդանշանի արժեքի բազմակի փոփոխման հնարավորությունը A մուտքային հավաքածուից B հավաքածու փոխվելու դեպքում, երբ սխեմայի ելքային ազդանշանը փոխվում է հակառակին:

2.2.1 Սինխրոն տրամաբանական մոդելավորում

Սինխրոն մոդելավորման ընթացքում էլեմենտների մոդելները ներկայացվում են իրենց տրամաբանական ֆունկցիաներով առանց ազդանշանների հապաղումները հաշվի առնելու, իսկ ազդանշանները՝ 0 և 1 արժեքներով: Սինխրոն մոդելավորումը օգտագործվում է դիսկրետ սարքերի տրամաբանական գործառման ճշգրտության գնահատման համար առանց անցողիկ պրոցեսների հաշվի առնման: Սխեմայի աշխատանքի մոդելավորումը բերվում է տրամաբանական էլեմենտների ելքերի ազդանշանների արժեքների հաշվմանը ըստ տրված մուտքային ազդանշանների:

Մոդելավորումը կատարվում է մուտքերում ազդանշանի ամեն մի փոփոխման համար: Ենթադրվում է, որ անցողիկ պրոցեսները սխեմայում վերջանում են ինչ-որ մի Dt ժամանակի ընթացքում, որը փոքր է մի տակտի տևողությունից:

Սինխրոն մոդելավորման օգտագործումը առավելապես հարմար է կոմբինացիոն սխեմաների աշխատանքի վերլուծության համար հաստատված ռեժիմում: Այդ դեպքում մոդելավորման արդյունքը ճշգրտորեն համապատասխանում է իրական սխեմային:

Սխեմայի ամեն մի էլեմենտը նկարագրվում է Y = f (X1, X2, ..., Xn) տրամաբանական բանաձևով, որտեղ Y - էլեմենտի ելքային, X1, X2, ..., Xn - էլեմենտի մուտքային ազդանշաններն են: Արդյունքում լրիվ սխեման նկարագրվում է այդպիսի բանաձևերի համակարգով: Սինխրոն մոդելավորումը բերվում է նրանց հերթականորեն լուծմանը: Տրամաբանական բանաձևի լուծման տակ հասկանում ենք Y տրամաբանական արժեքի հաշվումը ըստ հայտնի X1, X2, ..., Xn տրամաբանական արժեքների: Սինխրոն մոդելավորման առանձնահատկությունն է տրամաբանական բանաձևերի լուծումը ըստ տրված հերթականությանը, որը համապատասխանում է ազդանշանի էլեմենտների միջով անցման հերթականությանը: Այդ հերթականությունը որոշելու համար սխեման նախնականորեն պետք է ռանգավորել, որ ամեն մի Y = f (X1, X2, ..., Xn) բանաձևի լուծման պահին բոլոր X1, X2, ..., Xn  արժեքները լինեն հայտնի: Ռանգավորման ժամանակ ընդունվում է, որ սխեմայի մուտքային ազդանշանները  մոդելավորման սկսման պահին հայտնի են և նրանց վերագրվում է r = 0 ռանգ:

Փոքր-ինչ ավելի բարդ է մոդելավորումը հետադարձ կապերով տրամաբանական սխեմաների դեպքում: Այդպիսի սխեմաների մոդելավորման սկզբունքը դիտարկելու համար հետադարձ կապի շղթայի մեջ հարմար է մտցնել հապաղման էլեմենտ: Իրական սխեմաներում այդ հապաղման էլեմենտը որպես ֆիզիկական էլեմենտ բացակայում է, իսկ հապաղումը տեղի է ունենում ամեն մի էլեմենտի գործարկման հապաղման պատճառով:

Հետադարձ կապերով տրամաբանական սխեմաների մոդելավորման ընթացքում սխեման ռանգավորվում ու մոդելավորվում է որպես հասարակ կոմբինացիոն սխեմա այն ենթադրությամբ, որ հապաղման էլեմենտի միացման տեղում  հետադարձ շղթան ժամանակավոր անջատվում է և էլեմենտի մուտքերին միացված հետադարձ կապի շղթաներին վերագրվում է 0 ռանգ: Այդ դեպքում նաև ընդունում ենք, որ հետադարձ կապի շղթաներով էլեմենտների մուտքերին տրվում են ազդանշաններ ՝ համապատասխանող նախկինում տրված 0 կամ 1 վիճակներին: Սխեման մոդելավորվում է և նոր հաշված հետադարձ կապի ազդանշանի արծեքները  տրվում են էլեմենտների մուտքերին ինչ որ Dt  ժամանակ անց, որը հավասար է հետադարձ կապի շղթայի հապաղման ժամանակին:

Սինխրոն մոդելավորման արդյունքն է ժամանակային դիագրամը, որը ներկայացվում է 0-երի և 1-երի հաջորդականությամբ: Մոդելավորման ամեն մի տակտի համար դիագրամում բերվում են մուտքային ազդեցությունների և սխեմայի էլեմենտների ելքերի ազդանշանների արժեքները: Նրանով վերլուծվում է սարքի աշխատանքի ճշգրտությունը: Սովորաբար սարքի մոդելավորումը կատարվում է ինչ-որ մի տեստային ազդեցությունների համախմբության համար, որոնց համար հայտնի է էտալոնային սխեմայի արձագանքը: Կոնկրետ դեպքում արդյունքները համատեղելով պահանջվող էտալոնային արձագանքի հետ, որոշում են սարքի աշխատանքի ճշգրտությունը:

Սինխրոն մոդելավորումը երկարժեք ազդանշանի ներկայացումով մոդելավորման հեշտագույն եղանակն է: Նրա կարևորագույն առավելությունը ՝ արագագործությունն է, սակայն սինխրոն մոդելավորունը չի թույլատրում սխեմայում վերլուծել անցողիկ պրոցեսները և բացահայտել սխալներ, որոնք առաջանում են սխեմայի էլեմենտներում աղդանշանների հապաղման պատճառով:

2.2.2 Ասինխրոն տրամաբանական մոդելավորում

Անցողիկ պրոցեսների վերլուծությունը տրամաբանական սխեմաներում կատարվում է մոդելավորման ասինխրոն մեթոդով, որտեղ հաշվի է առնվում ազդանշանի տարածման ժամանակը էլեմենտներում և սխեմայի միացնող շղթաներում:

Տրամաբանական էլեմենտի գործարկումը կատարվոմ է ինչ-որ հապաղումով ըստ մուտքային ազդանշանների, որը հաշվի է առնվում հապաղումով էլեմենտների մոդելներում: Ամեն մի էլեմենտը բնութագրվում է ինչ-որ մի միջին հապաղումով, որի արժեքը կարող է փոխվել կախված էլեմենտի աշխատանքի ռեժիմից, մուտքային ազդանշանների համակցությունից, ջերմաստիճանից և այլ:

Էլեմենտի մուտքային ազդանշանի ժամանակային ապահամաձայնեցումը կարող է բերել տրամաբանական էլեմենտի ելքին կեղծ ազդանշանի հայտնմանը:

Հիշողությունով տրամաբանական էլեմենտներում, այսինքն հետադարձ կապերով սխեմաներում, մուտքային ազդանշանների ազդեցության տակ մի քանի հիշողության էլեմենտեր միանգամից կարող են փոխել իրենց վիճակները: Մի քանի դեպքերում սխեմայի վերջնական վիճակը կախված է հիշողության էլեմենտների փոխանջատման հաջորդականությունից: Այդ դեպքերում սխեմայում կան հետադարձ կապերի ազդանշանների մրցակցություն: Եթե մուտքային ազդանշանի ազդեցության տակ սխեման մի վիճակից կարող է անցնել տարբեր վիճակների, կախված սխեմայի էլեմենտների հապաղումներից, ապա մրցակցությունները կոչվում են կրիտիկական:

Ասինխրոն մոդելավորումը թույլ է տալիս հայտնաբերել խափանման ռիսկերը և կրիտիկական մրցակցությունները:

Տրամաբանական էլեմենտի մոդելը ասինխրոն մոդելավորման համար ցուցաբերվում է ոչ իներցիոն տրամաբանական էլեմենտի, իրականացնող տվյալ ֆունկցիան, և հապաղման էլեմենտի հաջորդաբար միացման ձևով:

Տրամաբանական էլեմենտի ասինխրոն մոդելը կազմվում  է ելենելով նրանից, որ ամեն մի էլեմենտ հապաղում է ազդանշանը Dt ժամանակով: Անցողիկ պրոցեսը վերլուծելու համար անհրաժեշտ է սխեման դիտել որպես ասինխրոն նրա աշխատանքի մի տակտի ընթացքում: Այն նշանակում է, որ էլեմենտները փոխարկվում են ասինխրոն կերպով, որքան որ ազդանշանները փոխվում են նրանց մուտքերում: Սակայն անցողիկ պրոցեսի տևողությունը չպետք է գերազանցի սխեմայի աշխատանքի DT տակտի տևողությունից, որը պետք է պատիկ լինի Dt աբստրակտ հապաղմանը և գերազանցի սխեմայում ազդանշանի մաքսիմալ հապաղումից: Սխեմայի մուտքերին ազդանշանի փոխվելու պահին անցողիկ պրոցեսը ավարտվում է, որովհետև մուտքային ազդանշանների տևողությունը հավասար է սխեմայի աշխատանքի տակտին:

Սխեմայի վիճակի մոդելավորումը իրականացվում է ժամանակի դիսկրետ պահերին Dt աբստրակտ հապաղմանը հավասար քայլով: Մոդելավորման ամեն մի տակտի ընթացքում սխեմայում ազդանշանները  առաջ են մղվում Dt ժամանակով: Սխեմայի ամեն մի էլեմենտի համար մտցնվում են գրգռում և արձագանք հասկացողությունները: Ցանկացած ազդեցություն էլեմենտի մուտքին բերում է գրգռմանը, որը հասնում է էլեմենտի ելքին  Dt = k Dt ժամանակի ընթացքում: Էլեմենտը գտնվում է կայուն վիճակում, եթե նրա արձագանքը չի հակասում գրգռմանը: Եթե սխեմայի բոլոր էլեմենտները գտնվում են  կայուն վաճակում, ապա սխեման նաև գտնվում է կայուն վիճակում:

Մոդելավորման ընթացքում կրկնվում են տեսածրման և իրագործման ընթացակարգը: Տեսածրման ընթացքում բոլոր էլեմենտների մուտքերը դիտվում են  mDt ժամանակի պահերին, որտեղ m = 1, 2, ...: Տեսածրումը ընդհատվում է երբ գրգռումը  հասնում է էլեմենտի ելքին, այնուհետև կատարվում է իրագործման ընթացակարգը: Մոդելավորման ընթացքում որոշվում են ազդանշանների փոփոխությունները և սխեմայի վարքը անցողիկ պրոցեսի ընթացքում: Սխեմայի մոդելավորման արդյունքը ցուցաբերվում է էլեմենտների փոխարկման ժամանակային դիագրամների տեսքով, որոնց հիման վրա վերլուծվում է սխեմայի աշխատանքը:

Ասինխրոն մոդելավորումը պահանջում է մեքենայական ժամանակի մեծ ծախսեր, և այն իրականացնելու համար օգտագործվում է պատահարային սկզբունքը` մոդելավորվում են միայն այն տրամաբանական էլեմենտները, որոնց մուտքերը փոխել են իրենց վիճակը ներկա ժամանակի պահին:

Ասինխրոն պատահարային մոդելավորման ամեն մի տակտում դիտվում են այն էլեմենտները, որոնց գոնե մեկ մուտքը փոխում է իր արժեքը: Սկզբում արժեք է տրվում սխեմայի մուտքային շղթաներին: Առաջին տակտի ընթացքում մոդելավորվում են  A1 բազմության էլեմենտները, որոնց մուտքը միացվում է սխեմայի  գոնե մի մուտքային շղթայի: Այն էլեմենտների բազմությունը, որոնք արժեք են ընդունում մոտակա տակտում, նշանակենք B1, որտեղ B1 ծ A1: Այնուհետև արժեք է տրվում B1 բազմության էլեմենտների ելքային շղթաներին: Այդ երկրորդ տակտում առաջացնում է A2 բազմության այն էլեմենտների մոդելավորումը, որոնց մուտքերը պարունակում են B1 բազմության էլեմենտների գոնե մեկ ելքային շղթա: Այդպիսի էլեմենտների մոդելավորումից հետո ընտրվում են B2 ծ (A1 / B1) Լ A2 բազմության այնպիսի էլեմենտները, որոնք արժեք են ստանում մոտակա տակտում: B2 բազմությունը որոշում է A3 էլեմենտների բազմությունը մոդելավորման հաջորդ տակտի համար և այլն:

Երբ էլեմենտը անցնում է անորոշ վիճակի, երկուական մոդելավորման ժամանակ իտերացիոն պրոցեսը չի զուգամիտվում: Այդ պատճառով մոդելավորման ալգորիթմի մեջ պետք է մտցնել սահմանափակում իտերացիաների մաքսիմալ քանակի վրա, ինչին հասնելուց սխեմայոմ պետք է ֆիքսվի գեներացիա:

Սինխրոն մոդելավորման համեմատ ասինխրոն մոդելավորումը պահանջում է բավականին մեծ քանակի հաշվարկների կատարում: Ծրագրի ծավալը նույնպես մեծանում է հապաղման էլեմենտների մոդելավորման անհրաժեշտության պատճառով:

2.2.3 Ասինխրոն իրադարձային մոդելավորում

Դիսկրետ սարքերի աշխատանքի վերլուծությունը ցույց է տալիս, որ ցանկացած պահին ակտիվ վիճակում գտնվում են սխեմայի բոլոր էլեմենտներից միայն 1...2,5 %: Հետևաբար, մոդելավորման ժամանակի էական պակասացման կարելի է հասնել, եթե ամեն անգամ մոդելավորել միայն այն էլեմենտները, որոնց մուտքային ազդանշանները փոխվել են: Այդպիսի մեդելավորման մեթոդը կոչվում է իրադարձային: Պատահարը իրադարձային մոդելավորման համակարգում՝  դա  ինչ-որ մի էլեմենտի վիճակի և նրա հետ կապված շղթաների փոփոխումն է:

Ասինխրոն իրադարձային մոդելավորման ծրագրերում հիմնական դերն են խաղում երկու զանգվածներ՝  մոդելավորվող սխեմայի հանգույցների վիճակների զանգված և ապագա իրադարձությունների հերթը: Հանգույցների վիճակների զանգվածը պահում է մոդելավորվող սխեմայի բոլոր շղթաների ընթացիկ վիճակները 0 և 1 տրամաբանական տեսքով: Մոդելավորման ընթացքում ապագա իրադարձությունների հերթում գրանցվում են այն իրադարձությունները, որոնք պետք է ապագայում առաջանան մոդելավորվող սխեմայում: Ապագա իրադարձությունների հերթի ամեն մի էլեմենտ պարունակում է շղթայի համարը, որտեղ պետք է առաջանա վիճակի փոփոխում, և իրադարձության առաջացման ժամանակի պահը: Պատահարները ապագա իրադարձությունների հերթում գրված են ժամանակի աճման կարգով, և առաջին տեղում գրված է այն իրադարձությունը, որը կկատարվի բոլորից առաջ:

Ասինխրոն իրադարձային մոդելավորումը կատարվում է հետևյալ ձևով: Մոդելավորման սկզբում սահմանվում է սխեմայի սկղզբնական վիճակը՝ հանգույցների վիճակների արժեքները զանգված գրանցելով: Սխեմային տրվող թեստային մուտքային ազդեցությունները մտցնվում են ապագա իրադարձությունների հերթի մեջ ըստ նրանց առաջացման ժամանակի: Ապա սկսվում է մոդելավորումը, որը բաղկացած է հետևյալ գործողություններից:

1. Ապագա իրադարձությունների հերթից ընտրվում է առաջին էլեմենտը: Նրանում նշված ժամանակը մտցվում է մոդելավորման ժամանակի հաշվիչի մեջ, իսկ շղթաների վիճակների զանգվածի մեջ ըստ էլեմենտում նշված համարի գրանցվում է ապագա իրադարձությունների հերթում նշված, շղթայի նոր վիճակը հնի փոխարեն:

2. Գտնվում են տրամաբանական էլեմենտներ, որոնց համար տրված շղթան մուտքային է, և հաշվվում են այդ էլեմենտների ելքային ազդանշանների արժեքները (այսինքն որոշվում են շղթաների նոր վիճակները) և նրանց հապաղումները:

3. Ամեն մի շղթայի համար ազդանշանի արժեքը համեմատվում է շղթաների վիճակների զանգվածի մեջ գտնվող արժեքի հետ, և եթե նրանք չեն համընկնում, ապա կատարվում է շղթայի վիճակի փոփոխում, և իրադարձությունը գրանցվում է ապագա իրադարձութունների հերթի մեջ: Եթե արժեքները համընկնում են, ապա գրանցումը չի կատարվում:

Այնուհետև գործողությունները կրկնվում են սկսած 1-ից:

Մոդելավորման պրոցեսը վերջանում է, երբ ապագա իրադարձությունների հերթը վերջանում է կամ վերջանում է տրված մոդելավորման ժամանակը:

Ասինխրոն մոդելավորման արդյունքները ներկայացվում են ժամանակային դիագրամների տեսքով, որոնցով վերլուծվում է սարքավորումների աշխատանքը, օրինակ ՝ հայտնաբերվում են կրիտիկական մրցակցությունները, ստատիկ և դինամիկ խափանման ռիսկերը:

Խափանման ռիսկերը և կրիտիկական մրցումները ավելի արագ են հայտնաբերվում տրամաբանական մոդելավորման բազմարժեք մեթոդի օգնությամբ:

2.2.4 Բազմարժեք մոդելավորման մեթոդը

Բազմարժեք մոդելավորման մեթոդներից ամենատարածվածը եռարժեք մոդելավորումն է: Այդ դեպքում օգտագործվում են ազդանշանի երեք արժեքներ 0, 1, X, որտեղ X-ը դա անցումն է մի վիճակից մյուսը կամ անորոշ վիճակը:

Եռարժեք մոդելավորման առավելությունը՝ իրագործման պարզությունն է և բարձր արագագործությունը:

Եռարժեք մոդելավորման ժամանակ էլեմենտների հապաղումները ընդունվում են հավասար 0: Տրամաբանական էլեմենտի մոդելը ներկայացվում է ճշմարտացիության աղյուսակների ձևով, որտեղ ամեն մի մուտքային ազդանշանը կարող է ընդունել երեք արժեքներ:

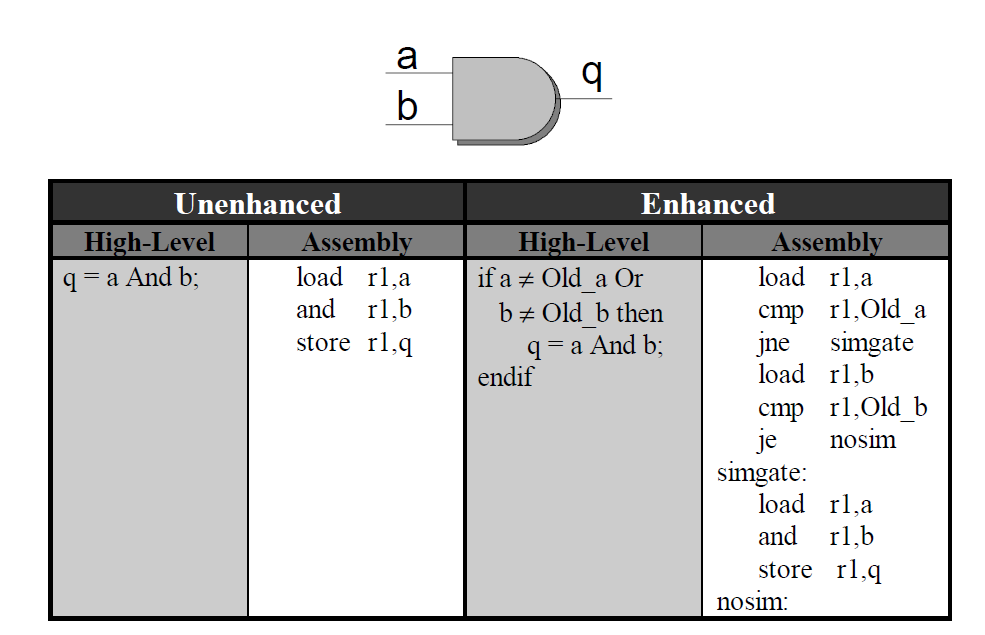
Հետադարձ կապերով սխեմաների եռարժեք մոդելավորման ժամանակ տրվում են հետադարձ կապերի ազդանշանների սկզբնական արժեքները (եթե նրանք անհայտ են, ապա ընդունվում են հավասար X արժեքին), հետադարձ կապերը խզվում են և իրագործվում է էլեմենտների ռանգավորումը: Այնուհետև հաշվվում են սխեմայի տրամաբանական էլեմենտների ելքերի արժեքները տրված մուտքային ազդեցության համար (անցողիկ կամ նոր ազդեցության) և որոշվում են հետադարձ կապերի ազդանշանների նոր արժեքները: Մոդելավորման պրոցեսը ամեն մի մուտքային ազդեցության համար իրականացվում է բազմակի անգամ մինչև կայուն վիճակի կամ գեներացիայի ռեժիմի հաստատումը:

Եթե եռարժեք մոդելավորման ժամանակ հայտնաբերվել է, որ սխեմայում հետադարձ կապի գոնե մի ազդանշան ընդունում է X արժեքը, ապա սխեմայում կան կրիտիկական մրցումներ:

Այսպիսով, բազմարժեք մոդելավորումը թույլ է տալիս հայտնաբերել հապաղման ռիսկի բոլոր հնարավոր տեղերը, նույնիսկ նրանց, որոնք իրական սխեմաներում չեն կարող պատահել:

2.3. Իրադարձային տրամաբանական մոդելավորում

Եթե տրամաբանկան փականի մուտքերը չեն փոխվում, կարիք չկա այն կրկին մոդելավորելու։ Այս գաղափարը առաջ է քաշում մոդելավորման արագությունը մեծացնելու նոր մեթոդ։ Այն է քչացնել մոդելավորվող էլեմենտների քանակը մուտքային վեկտորով մոդելավորելիս։ Ցավոք, հասարակ մոտեցումը չի բերի լուրջ կատարելագործման, քանի որ փականի մուտքի փոփոխության ստուգման ժամանակը գրեթե հավասար է փականը մոդելավորելու ժամանակին։ Նկար 1-ը ներկայացնում է տարբերությունը՝

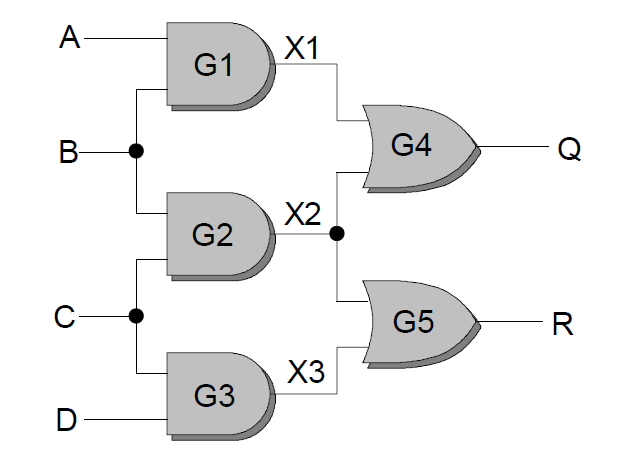


**Նկար 1**

Լավացրած կոդը աշխատեցնում է ամենաքիչը 6 հրաման, բայց կարող է հասնել նաև 9-ի, իսկ սովորականը՝ միշտ 3 հրաման է:

Եթե փականի մուտքեևը չեն փոխվում, ապա ելքը ևս չի փոխվի։ Նկար 1-ի պարագայում, եթե ոչ a-ն, ոչ b-ն չփոխվեն, q-ն չի փոխվի։ Եթե q-ն ինչ-որ ուրիշ սխեմայի մուտք է, ապա կարիք չկա այդ մուտքը ստուգել, այնքան ժամանակ, մինչև a-ն կամ b-ն չփոխվեն։

Գաղափարը ավելի լավ հասկանալու համար դիտարկենք նկար 2-ի սխեման՝

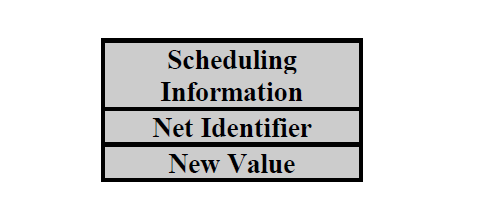


**Նկար 2**

Ենթադրենք սխեման մոդելավորվում է երկու տարբեր մուտքային վեկտորներով՝(0,0,0,0) և (0,0,0,1); Քանի որ A-ն, B-ն և C-ն չեն փոխվել, կարիք չկա մոդելավորելու G1,G2, և G3 փականները։ G4-ի մոդելավորելը կարելի է անտեսել, առանց ստուգելու x1 և x2-ը։ Ցավոք, կոդը այս ամենը իրականացնելու համար շատ բարդ կլինի գրեl, հատկապես B,C, և X1 ճյուղավորումների հատվածը։ Ոչ անհրաժեշտ փականների հեռացումը մոդելավորման պրոցեսից կարիք ունի հիմնովին այլ մոտեցման;

2.3.1 Հիմնական գաղափարները

Իրադարձային տրամաբանական մոդելավորումը նախատեսված է ավելորդ փականների մոդելավորման հեռացման համար, առանց անընդունելի քանակությամբ լրացուցիչ թեստավորման: Իրադարձության գաղափարը ամենահիմնականն է այս պրոցեսի ընթացքում։ Իրադարձությունը իրենից ներկայացնում է կապակցող մետաղական միացման արժեքի փոփոխությունը։ Ւրադարձությունը կարելի է ներկայացնել հետեվյալ տվյալների կառուցվածքի միջոցով, ինչպես ներկայացված է նկար 3-ում ՝



**Նկար 3**

Իրադարձային տրամաբանական մոդելավորման հիմնական ֆունկցաին իրադարձությունը հայտնաբերելն է և դրա հիման վրա համապատասխան փականների մոդելավորում կատարելը։

Եթե իրադարձություններ չկան, ուրեմն չկան փոփոխված միացումներ, ուստի նաև մոդելավորվող փականներ։

Իրադարձային տրամաբանական մոդելավորումը իրենից ներկայացնում է դինամիկ պլանամորման գործընթաց։ Այսինքն այն կարող է պլանավորել փականների մոդելավորում, առանց իմանալու նախնական հաջորդակարգի սխեմայի վերլուծման ընթացքում։ Սրա պատճառով մեկ կամ մի քանի դինամիկ հերթեր կարող են օգտագործվել իրագործման ընթացում։

Երկու անգամ է անհրաժեշտ լինում ստուգել միացումների արժեքների փոփոխությունները՝ երբ նոր մուտքային վեկտոր է տրվում և անմիջապես փականի մոդելավորումից հետո։

Հիմնական ալգորիթմը կարելի է նկարագրել հետեվյալ կերպ՝

* Սկզբնարժեքավորել հիմնական մուտքերը մուտքային վեկտորի արժեքներով
* Միացումների հերթի մեջ ավելացնել հիմնական մուտքերը
* Քանի դեռ միացումների հերթը դատարկ չէ
  + Պտտվել միացումների հերթի վրայով, հանել մի էլեմենտ հերթից
  + Փականների ցուցակից վերցնել հերթական փականը
  + Եթե միացման էլեմենտը հանդիսանում է մուտք հանված փականի համար, փականը ավելացնել փականների հերթի մեջ
  + Քանի դեռ փականների հերթը դատարկ չէ, պտտվել փականների հերթի վրայով, հերթով մոդելավորելով փականները
  + Մոդելավորվող փականի փոփոխված ելքը ավելացնել միացումների հերթի մեջ

Ծրագրավորման լեզուներին հարմար պսեվդո ներկայացումը հետեվյալն է՝

# setting primery\_inputs values from input\_vector

index = 0

for net in net\_list:

if net.name in prim\_ins:

net.old\_value = net.value

net.value = input\_vector[index]

index+=1

# putting changed nets to net\_queue

for net in net\_list:

if net.value != net.old\_value:

net\_queue.put(net)

# main process of simulation

while not net\_queue.empty():

#finding gates,whose inputs are the changed nets

while not net\_queue.empty():

temp\_net = net\_queue.get()

for gate in gate\_list:

if temp\_net in gate.input\_list:

if gate.in\_queue == False:

gate.in\_queue = True

gate\_queue.put(gate)

# simulation process

while not gate\_queue.empty():

temp\_gate = gate\_queue.get()

temp\_gate.in\_queue = False

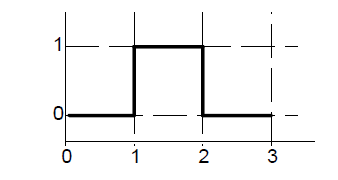
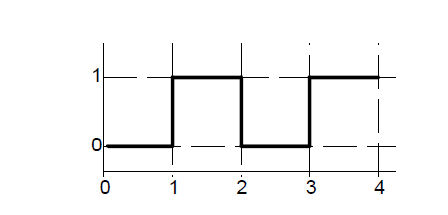
temp\_gate.simulate()

if temp\_gate.output.old\_value != temp\_gate.output.value:

net\_queue.put(temp\_gate.output)

2.3.2 Խափանումները տրամաբանական սխեմաներում

Եթե տրամաբանական սխեմայի մուտքը փոխվում է, անցանկանալի փոխանջատումներ կարող են առաջանալ սխեմայի ելքում։ Կան երկու տեսակի խափանումներ, ստատիկ խափանումներ, որոնք նկարագրված են Նկար 5-ում և դինամիկ խափանումներ, որոնք նկարագրված են Նկար 4-ում:



**Նկար 4 Նկար 5**

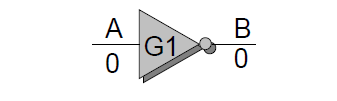
Ստատիկ խափանման ժամանակ միացման արժեքը փոփոխվում է ավելի քան մեկ անգամ, բայց ի վերջո վերադառնում է իր սկզբնական արժեքին։

Դինամիկ խափանման ժամանակ, արժեքը փոփոխվում է մեկից ավելի անգամ, ի վերջո, կայունանալով հակառակ արժեքին, որով սկսել էր։

Խափանումները կարող են առաջացնել խնդիրներ ասինխրոն հաջորդական սխեմաների կառուցման ժամանակ։ Քանի որ մակարդակավորված մոդելավորումը չի կարող բացահայտել խափանումները, իրադարձային մոդելավորման եղանակները նախընտրելի են սինքրոն սխեմաների մոդելավորման համար:

2.3.3 Տրամաբանական սխեմաների սկզբնարժեքավորման խնդիրները

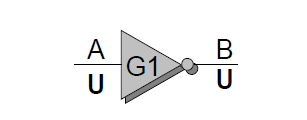
Նախքան մոդելավորելը սխեման ըստ մուտքային վեկտորի, անհրաժեշտ է բոլոր միացումները սկզբնարժեքավորել ինչ-որ արժեքներով։ Ենթադրենք բոլոր միացումները սկզբնարժեքավորվել են զրաներով։ Սա կարողե է բերել խնդիրների մոդելավորման ժամանակ, ինչպես օրինակ նկար 6-ում՝



**Նկար 6**

Եթե սխեման մոդելավորվի մուտքային 0 արժեքներով A միացման արժեքը չի փոխվի և իրադարձություն չի պլանավորվի, ուսի G1 փականը չի մոդելավորվի։ Սա էլ իր հերթին կբերի նրան, որ B միացման արժեքը երբեք չի փոխվի։ Վերջնական արդյունքում սխեմայի ելքային արժեքը սխալ կլինի նշված մուտքերի դեպքում։ Սկզբնարժեքավորելը սխեմայի միացումները բոլորը զրոներով բերում է սխեմայի անհամապատասխանության նրա համապատասխան տրամաբանության հետ։ Այս խնդիրը լուծելու կա երկու եղանակ։Առաջինը անցնել եռարժեք տրամաբանական մոդելի, որը իր մեջ ներառում է երրորդ “անհայտ” միացման վիճակը։ Ամենասկզբում բոլոր միացումները սկզբնարժեքավորվում են անհայտ արժեքներով: Անկախ նրանից, թե սկզբնական արժեքը 1-է, թե 0, այն տարբեր կլինի անհայտ արժեքից և իրադարձություն կգեներացվի, համապատասխան իրադարձությունը ավելացնելով իրադարձությունների հերթի մեջ։

Այն կբերի G1 փականի մոդելավորման և B միացման ճիշտ արժեքի ստացման։



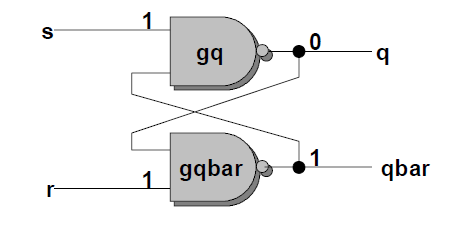
**Նկար 7**

Եռարժեք մոդելի թերությունը կայանում է նրանում, որ համապատասխան մոդելավորման կոդը շատ ավելի բարդ է դառնում, քան երկարժեք մոդելի դեպքում է։

Մյուս տարբերակը վերոնշյալ խնդիրը լուծելու համար հետեվյալն է։ Կարելի է մոդելավորման ամենասկզբում նշել բոլոր միացումները և ձևափոխել իրադարձություն գեներացնելու պայամանը։ Այս պարագայում իրադարձություն կստեղծվի, եթե միացման արժեքը փոխվել է, կամ միացումը նշվել է։ Երբ իրադարձությունը կավելացվի հերթի մեջ, միացման նիշը պետք է ջնջել։ Պլանավորման տեսանկյունից այս մեթեդը մեկին մեկ նույնն է, ինչ որ եռարժեք մեդելը։ Սակայն այն օգտագործում է ավելի պարզ երկարժեք մոդելը։

Եթե անհրաժեշտ է սխեման մոդելավորել սկզբնական անհայտ արժեքներով, ապա եռարժեք մոդելը պետք է օգտագործել, հակառակ դեպքում երկրորդ տարբերակը ավելի ավելի արդյունավետ է։

Իրադարձային տրամաբանական մոդելավորումը ունակ է մշակել նաև բոլոր տեսակի հաջորդական սխեմաները։



**Նկար 8**

Սինխրոն հաջորդական սխեմաները նաև կարող են մոդելավորել առանց հիմնական պրոցեսը ձևափոխելու։ Ավելին, հնարավոր է հաստատել սխեմայի սինխրոն վարքը այս մոդելավորման դեպքում, ինչը չէր կարելի անել մակարդակավորված մոդելավորման դեպքում։ Հաջորդական սխեմաների մոդելավորման ամենաբարդ հատվածը տատանումների հայտնաբերումն է։ Չնայած ուշադիր վերլուծությունը մոդելավորվող փականի հնարավորություն կտա մոդելավորման միջոցին հայտնաբերել տատանումը։ Լավ նախագծված սխեմայում ոչ մի փական չի կարող մոդելացվորվել 2 կամ 3 անգամից ավելին։ Սահմանափակում դնելով մոդելավորվող փականների քանակի վրա, տատանումները շատ հեշտ հնարավոր է լինում դուրս բերել, երբ արդեն ավելորդ փականներ են սկսում մոդելավորվել։ Քանի որ իրադարձային մոդելավորումը կարող է հայտնաբերել խափանումները, այն ավելի ճշգրիտ է ու էֆֆեկտիվ է համարվում, քան մակարդակավորված մոդելավորումը։ Ցավոք, դժվար է համեմատել այս երկու ալգորիթմների էֆֆեկտիվությունը, քանի որ իրադարձային մոդելավորումը շատ խիստ կախված է սխեման մոդելավորող մուտքային վեկտորից, այն դեպքում, որ մակարդակավորված մոդելավորումը կախված չէ մուտքային վեկտորից։

Իրադարձային մոդելավորումը գնահատելու համար օգտագործվում է ակտիվության մակարդակ հասկացողոությունը։ Այն միակ մոտքային վեկտորի համար հավասար է մոդելավորվող փականների քանակը բաժանած ընդհանուր փականների քանակին։ Մուտքային վեկտորների հավաքածուի համար այն հավասար է մոդելավորվող փականների քանակը բաժանած ընդհանուր փականների քանակը բազմապատկած մուտքային վեկտորների քանակի արտադրյալին։

Եթե ակտիվության մակարդակը մոտ է 100 տոկոսին, ապա մակարդակավորված մոդելավորումը զգալիորեն գերազանցում է իրադարձային մոդելավորմանը։ Եթե այն մոտ է զրոյին, ապա կարելի է ասել, որ իրադարձային մոդելավորումը զգալիորեն գերազանցում է մակարդակային մոդելավորմանը։

# **2.4 Qt գրադարանից վերցված մոդելներն ու տիպերը**

Qt-ն C++ լեզվի հիման վրա կառուցված անկախ համակարգչի պլատֆորմից և օպերացիոն համակարգից գրադարան է: Այն կառուցված է հետեվյալ մոտեցումով՝ բոլոր class-ները ժառանգում են մեկ QObject class-ից: Այն ծրագրավորողին լայն հնարավորություն է տալիս գրաֆիկական միջավայր ստեղծելու համար: Այսօր Qt-ն բավականին ընդգրկուն է դարձել և արդեն հնարավորություն ունի աշխատելու ոչ միայն գրաֆիկական տարբեր օբյեկտների հետ, այլ նաև տվյալների բազաներ, սերվերների, ինտերնետային տարբեր տեխնոլոգիաների հետ: Այն նաև ունի իր կոնտեյներները, որոնք բավականին նման են STL գրադարանի կոնտեյներներին:

Qt գրադարանը կառուցված է մի քանի կարևոր սկզբունքների վրա, նշենք դրանցից մի քանիսը:

1. **SIGNAL – SLOT**: Այս մեխանիզմը օգտագործվում է 2 տարբեր օբյեկտների միջև կապ ստեղծելու համար: Այս մեխանիզմի հիման վրա է կառուցված գրադարանի կմախքը: Այս մեխանիզմի հնարավորություն է տալիս մի օբյեկտին տալ ինֆորմացիա մյուս օբյեկտի մասին, որոնք իրար հետ կարող են կապ ընդհանրապես չունենալ: Այս մեխանիզմի այլընտրանքը callback մեխանիզմն է: Վերջինս ունի 2 մեծ թերություն. կանչողը չի կարող վստահ լինել, որ կանչվող callback ֆունկցիայի արգումենտների տիպերը ճիշտ կլինեն և այն ֆունկցիան, որը պետք է կանչի callback-ը պետք է գաղափար ունենա callback-ի մասին: Qt-ում մենք ունենք այլընտրանք callback-ին: signal-ը տեղի է ունենում (emit) երբ ինչ-որ իրադարձություն է տեղի ունենում: Qt օբյեկտներում կան ներդրված signal-ներ, սակայն ծրագրավորողը հեշտությամբ կարող է ավելացնել իր signal-ները՝ ժառանգման միջոցով: slot-ը ֆունկցիա է, որը կանչվում է երբ տեղի է ունենում համապատասխան signal-ը: Ինչպես signal-ները, այնպես էլ slot-երը ժառանգվում են: signal-slot մեխանիզմը անկախ է դրանց տիպից և նրանք class-ի member-ների պես կարող են լինել public, protected, private: slot-երը կարող են օգտագործվել, որպես class-ի սովորական member ֆունկցիաներ:   
    Signal-ի և դրան կապված slot-ի սահմանումները (signature) պետք է համընկնեն, այսինքն պետք է ունենան նույն քանակի, նույն տիպի և նույն հերթականությամբ արգումենտներ: Այն օբյեկտը, որից emit է արվել signal-ը երբեք չի իմանում, թե որ օբյեկտի որ slot-ն է հասցեատերը (reciver): Այս մեխանիզմը երաշխավորում է, որ slot-ը կկանչվի միայն այն դեպքում, երբ signal-ի արգումենտները ճիշտ են:   
    Մի signal-ը կարող է կապված լինել մի քանի slot-երի հետ և կարող է լինել նաև հակառակը: Եթե signal-ը մի քանի անգամ կապված է նույն slot-ի հետ, ապա երբ signal-ը emit արվի, ապա slot-ը կկանչվի ճիշտ այնքան անգամ, որքան անգամ այն կապված է signal-ի հետ:   
    Նշենք, որ signal-ի արգումենտների քանակը կարող է ավելի շատ լինել քան slot-ինը: Այս դեպքում փորձվում է կանչել slot-ը signal-ի առաջին արգումենտներով:   
    signal-slot մեխանիզմը աշխատում է հիմնվելով meta-object-ների վրա, որոնք ստեղծվում են compile-ի ժամանակ և մեծ դեր ունեն Qt գրադարանում:
2. **connect**: connect-ը QObject class-ի static անդամ-ֆունկցիաներից է: Այն ունի հետևյալ գրելաձևը.   
   [static] bool QObject::connect(const QObject\* sender, const char\* signal,   
    const QObject reciver, const char\* slot, Qt::ConnectionType type = Qt::AutoConnection);   
    Այս ֆունկցայի միջոցով կապ է հաստատվում sender-ի signal-ի և reciver-ի slot-ի միջև: connect-ը վերադարձնում է true երբ հաջողվում է կապել sender-ի signal-ը reciver-ի slot-ի հետ, այլապես վերադարձնում է false: Եթե signal-ը կապված է մի քանի slot-ների հետ, ապա երբ այդ signal-ը emit է արվում, ապա slot-ները կանչվում են այն հաջորդկանությամբ, որով connect էին արվել signal-ին: connect ֆունկցիան thread-safe է՝ այն կարող է կանչվել միաժամանակ տարբեր thread-երից և աշխատում է եթե նույնիսկ տարբեր thread-եր օգտագործում են նույն ինֆորմացիոն տվյալները:  
    Qt-ն նաև հնարավորություն է տալիս արդեն կապված signal-ը slot-ը առանձնացնել: Դա արվում է disconnectstatic ֆունկցիայի միջոցով, որը նույնպես thread-safe է և ունի հետևյալ գրելաձևը.   
     
   [static] bool QObject::disconnect(const QObject\* sender, const char\* signal,   
    const QObject reciver, const char\* slot);   
     
    Կարող ենք օգտագործել disconnect ֆունկցիայի հետևյալ տարբերակները.  
   disconnect(myObject, 0, 0, 0); - ստատիկ ֆունկցիա, որը myObject-ի բոլոր signal-ներին կապված բոլոր slot-ները առանձնացնում է:   
   myObject->disconnect() - ֆունկցիա, որը myObject-ի բոլոր signal-ներին կապված բոլոր slot-ները առանձնացնում է:   
   disconnect(myObject, SIGNAL(mySignal), 0, 0); - ստատիկ ֆունկցիա, որը myObject-ի mySignal signal-ին կապված բոլոր slot-ները առանձնացնում է:   
   myObject->disconnect(SIGNAL(mySignal())); - ֆունկցիա, որը myObject-ի mySignal signal-ին կապված բոլոր slot-ները առանձնացնում է:   
   disconnect(myObject, 0, myReciver, 0); ); - ստատիկ ֆունկցիա, որը առանձնացնում է myObject-ի բոլոր signal-ներին myReciver-ի բոլոր slot-ներից:   
   myObject->disconnect(myReciver); - ֆունկցիա, որը առանձնացնում է myObject-ի բոլոր signal-ներին myReciver-ի բոլոր slot-ներից:
3. **MetaObject** *–* *մեխանիզմ*: Այս մեխանիզմի միջոցով կարող ենք դինամիկ կերպով ինֆորմացիա ստանալ օբյեկտի մասին: MetaObjectմեխանիզմը հիմնված է հտևյալ 3 բաղադրիչների վրա.
   * QObject class-ը, լինելով ժառանգման հիերարխիյի ամենավերևում ապահովում է meta-object մեխանիզմի առկայությունը անհրաժեշտ դեպքերում,
   * Q\_OBJECT macro-ն, որը գրված է լինում class-ի private մասում meta-object մեխանիզմին ակտիվացնում է:
   * Meta-Object compiler-ը անհրաժեշտության դեպքում QObject class-ի ժառանգներին ձևափոխում է moc օբյեկտների, որոնցում մետաինֆորմացիան ներկայացվում է զուտ C++ լեզվով:

Շատ խնդիրների լուծման հիմքում ընկած է ին‎ֆորմացիայի մշակումը: Ին‎ֆորմացիայի մշակման հեշտացման համար ստեղծվում են ինֆորմացիոն համակարգերը (ԻՀ): Ավտոմատացված անվանում են այն ԻՀ-րը, որոնցում կիրառվում են տեխնիկական միջոցներ, մասնավոր դեպքում` ԷՀՄ: Գոյությունունեցող ԻՀ-րի մեծամասնությունը հանդիսանում է ավտոմատացված, այդ պատճառով նրանց կանվանենք պարզապես ԻՀ-եր: Լայն հասկացությամբ, ԻՀ-ի սահմանման տակ ընկած է ին‎ֆորմացիայի մշակման ցանկացած համակարգ: Ըստ կիրառման բնագավառի ԻՀ-րը կարելի է բաժանել համակարգերի, որոնք օգտագործվում են արտադրությունում, կրթությունում, առողջապահությունում, գիտությունում, ռազմական գործում, հասարակական ոլորտում, առևտրում և այլ բնագավառներում: Ըստ նպատակային ‎ֆունկցիայի ԻՀ-րը կարելի է պայմանականորեն բաժանել հետևյալ կարգերի. կառավարող, ինֆորմացիոնա-տեղեկատվական, որոշումներ ընդունող:

Ավանդական «թղթյա» տվյալների բազաներն ունեն մի շարք թերություններ, օրինակ՝

* պահանջվում են ֆիզիկական մեծ տարածքներ,
* անհրաժեշտ տվյալները գտնելու համար պահանջվում է բավականաչափ երկար ժամանակ,
* դարակները և կարգացանկերը կարգավորված վիճակում պահպանելու համար պետք է ծավալուն աշխատանք կատարել և մեծ ջանքեր գործադրել։

Տվյալների էլեկտրոնային բազաները թույլ են տալիս լուծել բոլոր այդպիսի խնդիրները։ Առավել լայն կիրառություն են գտել տվյալների ռելյացիոն բազաները։ Ռելյացիոն բազաներում ինֆորմացիան բաժանված է տրամաբանորեն կապակցված համեմատաբար փոքր և, դրա հետևանքով, ավելի հեշտ ղեկավարվող մասերի, որոնք կոչվում են աղյուսակներ։ Աղյուսակները իրենց կազմակերպման մակարդակի շնորհիվ պարզեցնում են բազայի ուղեկցումը և ապահովում դրա առավել օպտիմալ գործարկումը։

Հատուկ գրականությունում աղյուսակի տողը անվանում են գրանցում (record), իսկ սյունակը՝ դաշտ (field)

«Տվյալների բազա» տերմինը կարող է ավելի նեղ իմաստով օգտագործվել՝ նշելու կազմակերպված տվյալների բազայի կոնկրետ դրսևորումներին և կարող է վերաբերվել տվյալների բազայի տրամաբանական մոդելին, տվյալների բազայի ֆիզիկական մոդելին կամ այլ տվյալների բազայի ենթասահմանման։

Տվյալների բանկը (ԲՆկ) հանդիսանում է ԻՀ-ի տարատեսակ, որտեղ իրագործվում են մշակվող ին‎ֆորմացիայի կենտրոնացված պահպանման և կատարման ‎ֆունկցիաները, կազմակերպված որպես մեկ կամ մի քանի տվյալների բազա:

ՏԲկ–ը ընդհանուր դեպքում բաղկացած է հետևյալ տարրերից` տվյալների բազայից (բազաներից), տվյալների բազայի կառավարման համակարգից, տվյալների բառարանից, ադմինիստրատորից, հաշվողական համակարգից և սպասարկող անձնակազմից: Կրճատ դիտարկենք նշված տարրերը և նրանց հետ կապված որոշ կարևոր հասկացություններ:

Տվյալների Բազան (ՏԲ) իրենից ներկայացնում է հատուկ ձևով կազմակերպված տվյալների համախմբություն, որոնք պահվում են հաշվողական համակարգի հիշողությունում և արտապատկերում են օբյեկտների վիճակը և նրանց փոխկապակցվածությունը դիտարկվող առարկայական տիրույթում:

ՏԲ-ում պահվող տվյալների տրամաբանական կառուցվածքը անվանում են տվյալների ներկայացման մոդել: Տվյալների ներկայացման հիմնական մոդելներին են դասվում հետևյալները. հիեռարխիկ, ցանցային, ռելացիոն, պոստռելացիոն, բազմաչափ և օբյեկտա-կողմնորոշված:

Տվյալների բազաների կառավարման համակարգը (ՏԲԿՀ)` լեզվական և ծրագրային միջոցների համալիր է, նախատեսված մեծ քանակությամբ օգտագործողների կողմից ՏԲ-ի համատեղ օգտագործման, ինչպես նաև ստեղծման և վարման համար: Սովորաբար ՏԲԿՀ-րը տարբերում են ըստ օգտագործվող տվյալների մոդելի: Այսպես, ՏԲԿՀ-րը հիմնված ռելացիոն մոդելի օգտագործման վրա կոչվում են ռելացիոն ՏԲԿՀ-եր: Առաջին ՏԲԿՀ-րից են հետևյալ համակարգերը. IMS (IBM, 1968թ.), IDMS (Cullinet, 1971 թ.), ADABAS (Software AG, 1969 թ.): Ներկայիս ՏԲԿՀ-րի քանակը չափվում է հազարներով:

Ինֆորմացիոն Համակարգերի Ճարտարապետությունը**։** ԻՀ գործունեության է‎ֆեկտիվությունը մեծ մասամբ կախված է նրա ճարտարապետությունից: Ներկա ժամանակ առաջնային է համարվում կլիենտ-սերվեր ճարտարապետությունը: Բավականին տարածված տարբերակով նա ենթադրում է համակարգչային ցանցի և բաշխված ՏԲ-ի առկայությունը, որն ընդգրկում է կորպորատիվ ՏԲ (ԿՏԲ) և անհատական ՏԲ (ԱՏԲ): ԿՏԲ-ն տեղադրվում է սերվեր-համակարգչում, որը հանդիսանում է ԿՏԲ-ի կլիենտ:

Համակարգչային ցանցում որոշակի պաշարի սերվերկոչվում է համակարգիչը (ծրագիրը),որը կառավարում է պաշարը, իսկ կլիենտ` համակարգիչը (ծրագիրը), որը օգտագործում է այդ պաշարը: Որպես համակարգչային ցանցի պաշար կարող են հանդես գալ ՏԲ-րը, ‎ֆայլային համակարգերը, տպման ծառայությունները, փոստային ծառայությունները: Սերվեր տեսակը որոշվում է իր կառավորվող պաշարի տեսակով: Օրինակ, եթե կառավարվող պաշարը հանդիսանում է ՏԲ-ն, ապա համապատասխան սերվերըկոչվում է տվյալների բազայի սերվեր:

Կլիենտ-սերվեր ճարտարապետությամբ ԻՀ-ի կազմակերպման առավելություն է հանդիսանում կենտրոնացված պահման, սպասարկման, ընդհանուր կորպորատիվ ին‎ֆորմացիային համատեղ դիմման և անձնական ին‎ֆորմացիայի վրա օգտագործողների աշխատանքի հաջող զուգակցումը: Կլիենտ-սերվեր ճարտարապետությունը թույլատրում է իրագործման բազմազան տարբերակներ:

Պատմականորեն առաջինները հայտնվեցին ‎ֆայլ-սերվերի օգտագործմամբ բաշխված ԻՀ-րը: Այսպիսի ԻՀ-րում ըստ օգտագործողների հայցման ՏԲ-ի ‎ֆայլերը փոխանցվում են դեպի անձնական օգտագործման ԷՀՄ, որտեղ և տեղի է ունենում նրանց մշակումը: Ճարտարապետության այսպիսի տարբերակի թերությունն է մշակվող տվյալների փոխանցման բարձր ուժգնությունը: Ընդ որում, հաճախակի փոխանցվում են ավելցուկային տվյալները` անկախ նրանից, թե քանի գրանցում ՏԲ-ի անհրաժեշտ են օգտագործողին, ՏԲ-ի ‎ֆայլերը փոխանցվում են ամբողջությամբ:

Բազայում պահվող տվյալները ունեն որոշակի տրամաբանական կառուցվածք, այլ խոսքերով` նկարագրվում են ինչ-որ տվյալների ներկայացման մոդելով (տվյալների մոդելով), որն ապահովում է ՏԲԿՀ-ն: Դասական տվյալների մոդելների թվին պատկանում են հետևյալները`հիերարխիկ,ցանցային,ռելյացիոն:

Բացի այդ, վերջին տարիներին հայտնվեցին և սկսեցին գործնականում ներդրվել հետևյալ տվյալների մոդելները` պոստռելյացիոն,բազմաչափ,օբյեկտա-կողմնորոշված:

Մշակվում են նաև բազմազան համակարգեր, հիմնված այլ տվյալների մոդելների վրա, որոնք ընդլայնում են հայտնի մոդելները: Նրանց թվից կարելի է նշել օբյեկտա-ռելյացիոնը, դեդուկտիվա-օբյեկտա-կողմնորոշվածը, սեմանտիկը, կոնցեպտուալը (իմաստայինը) և կողմնորոշված մոդելները: Այս մոդելներից մի քանիսը ծառայում են ՏԲ-րի գիտելիքների բազաների և ծրագրավորման լեզուների ինտեգրացման համար:

Ռելացիոն մոդել**։** Տվյալների ռելացիոն մոդելը առաջարկվել էր IBM ընկերության աշխատակից` Էդգար Կոդդի կողմից և հիմնված է հարաբերություն (relation) հասկացության վրա:

Հարաբերությունը իրենից ներկայացնում է տարրերի բազմություն, որոնց անվանում են կորտեժներ: Հարաբերության պատկերավոր ներկայացման ձև է հանդիսանում մարդկային ընկալման համար սովորական` երկչափ աղյուսակը:

Աղյուսակն ունիտողեր (գրանցումներ) և սյուներ (սյունակներ): Աղյուսակի յուրաքանչյուր տողըունիմիևնույն կառուցվածք և բաղկացած է դաշտերից: Աղյուսակի տողերին համապատասխանում են կորտեժները, իսկսյուներին` հարաբերության ատրիբուտները:

Տվյալների մոդելավորման և ՏԲԿՀ-ների ստեղծման հարցերով զբաղվել են շատ հետազոտողներ, բայց այդ ուղղությամբ նշանակալի հետք է թողել Կոդդին: Նա մշակել է տվյալների ռելացիոն մոդելը, որի հիման վրա ստեղծվել է ժամանակակից ՏԲԿՀ- ների մեծ մասը:

Ռելացիոն տվյալների բազան կազմված է մի քանի աղյուսակներից: Աղյուսակներից յուրաքանչյուրը պարունակում է ին‎ֆորմացիա նույնտիպի առարկաների մասին: Աղյուսակի յուրաքանչյուր գրառում համապատասխանում է մեկ առարկային: Առարկաների միջև հարաբերությամբ որոշվում է հարաբերությունը աղյուսակների միջև: Աղյուսակի մեկ գրառմանը կարող է համապատասխանել մեկ այլ աղյուսակի մի քանի գրառում: Այդ դեպքում կստացվի մեկ-շատ հարաբերությունը: Աղյուսակների միջև կապը իրականացվում է առանցքային դաշտերի միջոցով:

Մեկ աղյուսակի օգնությամբ հարմար է նկարագրել տվյալների միջև պարզագույն կապերի տեսակը, իսկ ավելի ճիշտ` մեկ օբյեկտի (երևույթի, էության, համակարգի և այլնի) բաժանումը, որի մասին տեղեկությունները պահվում են աղյուսակում, ենթաօբյեկտների բազմության, որից յուրաքանչյուրին համապատասխանում է աղյուսակի տողը կամ գրանցումը:

Այս դեպքում ենթաօբյեկտներից յուրաքանչյուրը ունիմիևնույն կառուցվածք կամ հատկություններ, որոնք նկարագրվում են գրանցումների դաշտերի համապատասխան արժեքներով: Օրինակ, աղյուսակը կարող է պարունակել ուսուցանվողների խմբի մասին տեղեկություններ, որոնցից յուրաքանչյուրի մասին հայտնի են հետևյալ բնութագրերը` ազգանունը, անունը և հայրանունը, սեռը, տարիքը և կրթությունը: Քանի որ մեկ աղյուսակի շրջանակներում չի հաջվողում նկարագրել առարկայական տիրույթից տվյալների տրմաբանական ավելի բարդ կառուցվածքներ, ապա օգտագործում են աղյուսակների կապումը:

Ռելացիոն բազաներում տվյալների ‎ֆիզիկական տեղաբաշխումը արտաքին կրիչների վրա հեշտությամբ իրականացվում է սովորական ֆայլերի օգնությամբ:

Տվյալների ռելացիոն մոդելի առկայությունը` դա ԷՀՄ-ի վրա ‎ֆիզիկական իրականցման պարզությունն է, հասկանալիությունը և հարմարությունն է: Հենց օգտագործողի համար հեշտությունը և պարզությունը հանդես եկան նրանց լայն տարածման հիմնական պատճառը: Այս տիպի տվյալների մշակման է‎ֆեկտիվության հարցերն էլ դարձան տեխնիկապես լուծելի:

Ռելացիոն մոդելի հիմնական թերություններն են հանդիսանում հետևյալները. առանձին գրանցումների տարբերակման ստանդարտ մեթոդների բացակայությունը և հիերարխիկ ու ցանցային կապերի նկարագրման բարդությունը:

ԷՀՄ-երի համար արտասահմանյան ռելացիոն ՏԲԿՀ-րից հանդիսանում են հետևյալները` dBase III dBase IV (Ashton¬Tateընկերություն), DB2 (IBM), R:BASE (Microrim), FoxPro-ի առաջին տարբերակները և FoxBase (Fox Saftware), Paradox և dBASE for Windows (Borland), Foxpro -ի ավելի ուշ տարբերակները, Visual Foxpro և Access (Microsoft), Clarion (Clarion Software), Ingress (ASK Computer Systems) և Oracle (Oracle):

Նկատենք, որ ռելացիոն ՏԲԿՀ- ի վերջին տարբերակները ունեն օբյեկտա-կողմնորոշված համակարգերի որոշ հատկություններ: Այդպիսի համակարգի օրինակ կարելի է համարել Oracle 8.x արտադրանքները: Նախկին տարբերակների համակարգերը ներառյալ Oracle 7.x համարվում են «մաքուր» ռելացիոն:

Ռելացիոն տվյալների բազաների նախագծման ժամանակ անհրաժեշտ է լուծել տվյալների բազաների առավել արդյունավետ կառուցվածքի հարցը: Հիմնական նպատակները, որոնց հետամուտ է լինում բազան նախագծելիս, հետևյալներն են` ապահովել տվյալների արագ ստացումը աղյուսակներից տվյալների, առանց անհրաժեշտության կրկնման բացառում, որը կարող է մուտքի ժամանակ սխալի պատճառ հանդիսանալ, ապահովել տվյալների ամբողջականությունն այնպես, որ մեկ առարկայի փոփոխումըինքնուրույն անդրադառնա նրա հետ կապված առարկաների վրա:

Տվյալների ամբողջականությունը ապահովելու համար պետք է ծրագրային միջոցներով իրականացվի հետևյալ գործողությունները:

Աղյուսակից գրառման հեռացման ժամանակ, եթե այդ գրառման հետ կապված են այլ աղյուսակների գրառումներ, ապա նախ և առաջ պետք է հեռացնել այդ գրառումները և նոր միայն սկզբնական գրառումը:

Առանձնապես պետք է ուշադիր լինել կապող դաշտերում փոփոխություն կատարելիս:

Տվյալների բազայում ին‎ֆորմացիայի ավելցուկության նվազեցման գործընթացը անվանում են նորմալացում: Տվյալների բազայի նորմալացման տեսության մեջ մշակվել են բավականաչափ ձևայնացված մոտեցումներ` բարդ կառուցվածք ունեցող տվյալները մի քանի աղյուսակների տրոհելու համար:

Նորմալացման տեսությունը հիմնվում է աղյուսակների հինգ նորմալ ձևերի վրա: Այդ ձևերը նախատեսված են ին‎ֆորմացիայի ավելցուկության նվազեցման համար: Այդ պատճառով յուրաքանչյուր հաջորդ նորմալ ձևը պետք է պարունակի նախորդ նորմալ ձևի պահանջները:

Տվյալների բազաների կիրառական նախագծման ժամանակ չորրորդ և հինգերորդ նորմալ ձևերը, որպես կանոն, չեն օգտագործվում: ՏԲԿՀ-ներըլինում են 3 տիպի` ծառային, ցանցային, ռելացիոն:

Կախված առաջադրված խնդրի առանձնահատկություններից, կարելի է օգտագործել դրան համապատասխան ՏԲԿՀ: Այժմ առավել տարածում է գտել ռելացիոն տիպի ՏԲԿՀ-ները:

Ռելացիոն տիպի բազաների ստեղծման համար պետք է պահպանել որոշակի կանոններ` նախագծված բազաների լավագույն կառուցվածքը ապահովելու համար: Այդ կանոնները կոչվում են տվյալների բազաների նախագծման նորմալ ձևեր: Գոյությունունեն հինգ նորմալ ձևեր: Բայց ռելացիոն տիպի բազաների նախագծման ժամանակ բավական է պահպանել միայն առաջին երեք նորմալ ձևերը: Այդ նորմալ ձևերի էությունը կայանում է հետևյալում.

Նախագծվող բազան պետք է զերծ լինի տվյալների ավելցուկությունից: Բազաները չպետք է կրկնեն մեկը մյուսին: Պետք է պահվի այն ին‎ֆորմացիան, որիմիջոցով այնուհետև հնարավոր կլինի վերականգնել ցանկացած այլ ինֆորմացիա, որը կապված է դրա հետ:

Նախագծվող բազան պետք է զերծ լինի միևնույն ին‎ֆորմացիայի միարժեք կրկնությունից: Այսինքն` բազայում միարժեքորեն նույն գրառումներ չպետք է լինեն:

Պետք է պահպանել բազայի ամբողջականությունը: Սա նշանակում է բազաները պետք է նախագծել այնպես, որպեսզի հնարավոր լինի ստեղծել նրանց միջև միարժեք կապեր:

Վերը նշված նորմալ ձևերի պահպանման համար ցանկացած տիպի խնդիրներ լուծելիս անհրաժեշտություն է առաջանում ստեղծել տեղեկատուներ: Տեղեկատուներում պահվում է ին‎ֆորմացիայի այն մասը, որն օգտագործվում է հաճախ, բայց չի ենթարկվում փոփոխության: Այս ինֆորմացիան մեկ անգամ ներմուծելով, այնուհետև կարելի է այն օգտագործել անընդհատ փոփոխվող կամ ավելացվող ին‎ֆորմացիաների մեջ:

2.5 Ընդարձակ նշման լեզու (XML)

Այս լեզուն սահմանում է մի շարք կանոներ՝ փաստաթղթերը մարդու և համակարգչի համար ընթեռնելի լեզվով կոդավորելու համար։ Այս կանոնները սահմանված են XML 1.0-ում։

Սույն աշխատանքում օգտագործված բոլոր կարգավորման փաստաթղթերը սահմանված են ընդարձակ նշման լեզվով։ Ստորև բերված է XML լեզվի կարճ նկարագրությունը, ինչը բացատրում է կարգավորման փաստաթղթերի ձևաչափի ընտրությունը։

XML-ի նախագծման հիմնական նպատականերն են պարզությունըն ու ընթեռնելիությունը, ընդհանրությունը և համացանցում օգտագործման հեշտությունը։ XML-ը տեքստային ինֆորմացիայի ձևաչափ է՝ Յունիկոդի ուժեղ աջակցությամբ։ Չնայած, որ XML-ի դիզայնը հիմնականում կենտրոնանում է փաստաթղթերի վրա, այն շատ է օգտագործվում կամայական տվյալների կառույցների ներայացման համար, օրինակ՝ համացանցում, կամ ինչպես այն արվել է HiveLoader գործիքի համար։

Բազմաթիվ ծրագրերի ծրագրավորման ինտերֆեյսը զարգացվել են XML տեղեկատվության մշակմանը օգնելու համար։ 2009-ից սկսած փաստաթղթերի հարյուրավոր ձևաչափեր են ստեղծվել, օրինակ՝ RSS, Atom, SOAP և XHTML։ Հետագայում XML-ի վրա հիմնվածները դարձան հիմնականը բազմաթիվ գրասենյակա-արտադրողականային գործիքների համար, ներառելով Microsoft Office-ը (Office Open XML), OpenOffice-ը, LibreOffice-ը (OpenDocument) և Apple-ի iWork-ը։ XML լեզուն լայնորեն օգտագործվում է նաև կոմունիկացիոն պրոտոկոլների համար, ինչպիսին է, օրինակ, XMPP -ն։

XML փաստաթուղթը, ըստ սահմանման, նշանների շարք է։ Յունիկոդի գրեթե բոլոր նշանները կարող են օգտագործվել XML փաստաթղթի մեջ։

Նշում և բովանդակություն

Նիշերը, որոնք պարունակում է XML փաստաթուղթը, լինում են նշման և բովանդակային և կարող են տարբերակվել սովորական սինտակտային կանոններով ծրագրով։ Հիմնականում, շարքերը, որոնք ստեղծում են նշում, սկսվում են «<» նիշով և վերջանում «>» նիշով, կամ սկսվում «&» նիշով և վերջանում «;» նիշով։ Նիշերի այն շարքերը, որոնք նշման չեն՝ բովանդակային են։

Հայտանիշ (անգլ.՝ tag)

Նշումներըման կառույցները, որոնք սկսվում են «<» և վերջանում են «>» կոչվում են հայտանիշեր։ Հայտանիշները լինում են 3 տեսակ՝

• Սկսման հայտանիշեր։ Օրինակ՝ <section>

• Վերջացնելու հայտանիշեր։ Օրինակ՝ </section>

• Դատարկ-տարրային հայտանիշեր։ Օրինակ՝ <line-break />

Տարր (անգլ․՝ element)

Փաստաթղթի այն մասը, որը կա'մ սկսվում է սկսման հայտանիշով և վերջանում է վերջացնելու հայտանիշով, կա'մ էլ պարունակում է միայն դատարկ-տարրային հայտանիշեր, կոչվում է տարր։ Սկսման և վերջացնելու հայտանիշերի միջև գտնվող նիշերը, եթե առկա են, կոչվում են տարրերի բովանդակություն, և կարող են պարունակել նշում, ներառյալ այլ տարրեր, որոնք կոչվում են երեխա տարրեր։ Տարրի օրինակ է՝ <Greeting>Hello, world.</Greeting>, կամ՝ <line-break />։

Բնորոշիչ (անգլ․՝ attribute)

Նշման կառույցները, որոնք պարունակում են անուն/արժեք զույգ, որը ունի սկսման և դատարկ-տարրային հայտանիշ, կոչվում են բնորոշիչներ։ Ստորև ներկայացված օրինակում img տարրը ունի երկու բնորոշիչ՝ src և alt։

<img src="madonna.jpg" alt='Foligno Madonna, by Raphael'/>

Մեկ այլ օրինակ, որտեղ բնորոշիչի անունը line է, իսկ արժեքը 5։

<data line=”5”>Data of the line five</data>

XML-ի բնորոշիչը կարող է ունենալ մեկ արժեք և ամեն բնորոշիչը կրող է հայտնվել ամենաշատը մեկ անգամ ամեն տարրում։

XML-ի հայտարարումը

XML փաստաթղթերը կարող են սկսվել իրենց մասին որոշ տեղեկատվությամբ, ինչպես հետևալ օրինակում՝

<?xml version="1.0" encoding="UTF-8"?>

Մեկնաբանություններ (անգլ․՝ comments)

Մեկնաբանությունները կարող են հայտնվել ամեն տեղ փաստաթղթում՝ նշումից դուրս։ Մեկնաբանությունները չեն կարող հայտնվել XML հայտարարւթյունից առաջ։ Մեկնաբանությունները սկսվում են «<!—» և վերջանում «-->»։, ինչպես նաև կրկնակի «--» շարքը չի թույլատրվում մեկնաբանությունների ներսում։ Ամպերսանդը չունի ինչ-որ հատուկ նշանակություն մեկնաբանությունների համար։ Մեկնաբանության օրինակ՝

<!—Some comment can be here -->

ԳԼՈՒԽ 3

Խնդրի դրվածքը

Խնդրի դրվածքը

* Թվային ինտեգրալ սխեմաների իրադարձային տրամաբանական մոդելավորման միջոցի մշակումը
* Մշակված ծրագրային միջոցի տրամաբանական մոդելավորման ժամանակի՝ սխեմայում գտնվող էլեմենտների քանակից կախվածության հետազոտումը և առկա մոդելավորման միջոցների հետ համեմատումը

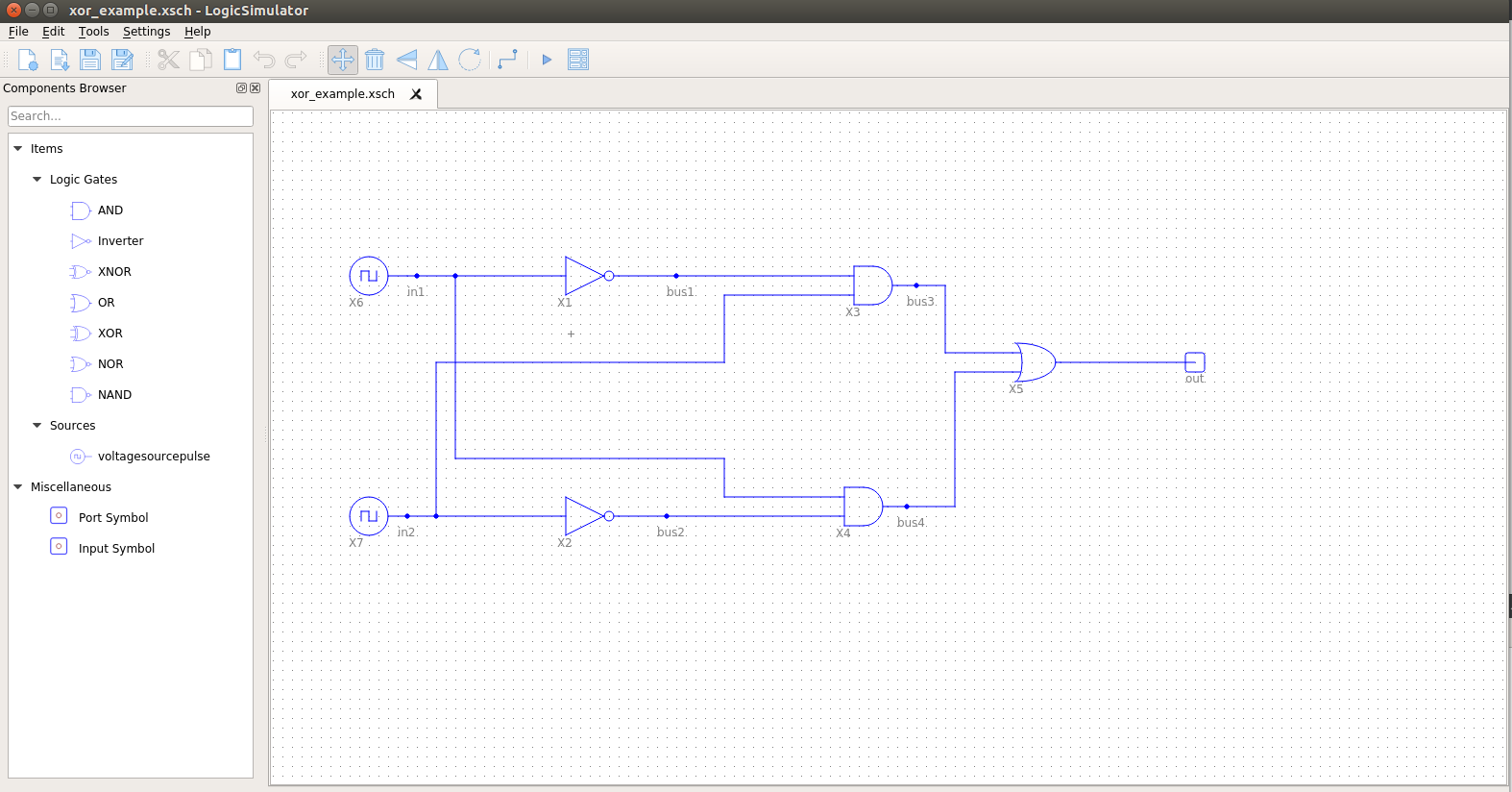
ԳԼՈՒԽ 4

Տեսական առնչություններ

Տեսական առնչություններ

4.1 Տրամաբանական սխեմայի նախագծում

Մշակված ծրագրային միջոցը իրականացվել է C++ ծրագրավորման լեզվի միջոցով, իսկ գրաֆիկական հատվածը QT գրադարանի օգնությամբ։



**Նկար 9**

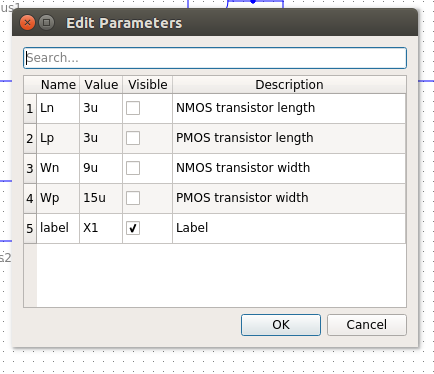
Ծրագրային միջոցը հնարավորություն է տալիս հավաքել տրամաբանական սխեման ծրագրային միջոցի գրաֆիկական խմբագրի հատվածում և կատարել իրադարձային տրամաբանական մոդելավորում։ Սխեմաների գրաֆիկական խմբագրի օգնությամբ կազմվում են սխեմաները տրամաբանական տարրերի գրաֆիկական պատկերների միջոցով: Պատուհանը բաժանվում է երկու մասի, որոնցից առաջինում խմբագրվում է թվային սխեման, իսկ երկրորդում պատկերվում է մոդելավորման արդյունքը ժամանակային դիագրամների տեսքով:

Ձախ հատվածում օգտագործողը հնարավորություն է ստանում ընտրելու տրամաբանական տարրերը, հավաքել սխեման տալով համապատասխան կապերը։

Օգտագործողը հնարավորություն ունի օգտագործել հետևյալ տրամաբանական էլեմենտները՝

* Տրամաբանական և (AND)
* Տրամաբանական կամ (OR)
* Բացառող կամ (XOR)
* Շրջիչ (Inverter)
* Տրամաբանական և-ոչ (NAND)
* Տրամաբանական կամ-ոչ (NOR)
* Բացառող կամ (XOR)

Տարրը կարելի է վերանվանել մկնիկի ձախ կոճակը երկու անգամ սեղմելով տարրի վրա ,սեղմելուց հետո բացվում է "Edit Parameters" պատուհանը որով կարելի է վերանվանել տարրը, փոփախել որոշ պարամետրեր։



**Նկար 10**

Ծրագրի համար տվյալների մուտք/ելքի համար մշակվել են հետեվյալ գործողությունները, որոնք երևում են նկ 11-ում՝

* Նոր ֆայլի ստեղծում (New)
* Առկա ֆայլի բացում (Open)

Բացվում է պատուհան, որը օգտագործողին հնարավորություն է տալիս ընտրել այն տեղը, որտեղ պահվելու է ծրագրի հիմնական մուտքային ֆայլը

* Վերջին օգտագործված ֆայլի բացում (Open Recent)
* Փակել բացված ֆայլը (Close)
* Պահպանել ֆայլը (Save)
* Պահպանել նոր տեղում (Save as)
* Պահպանել բոլորը (Save All)
* Փակել ծրագիրը (Quit)

«Open” հրամանի միջոցով օգտագործողը կարող է ծրագրի մուտքին տալ արդեն իսկ նախորոք մշակված սխեման, կարդալ և կատարել համապատասխան փոփոխություններ,

ուսումնասիել ժամանակային դիագրամները և կրկին պահպանել սխեման համակարգչի նախընտրելի տիրույթում։

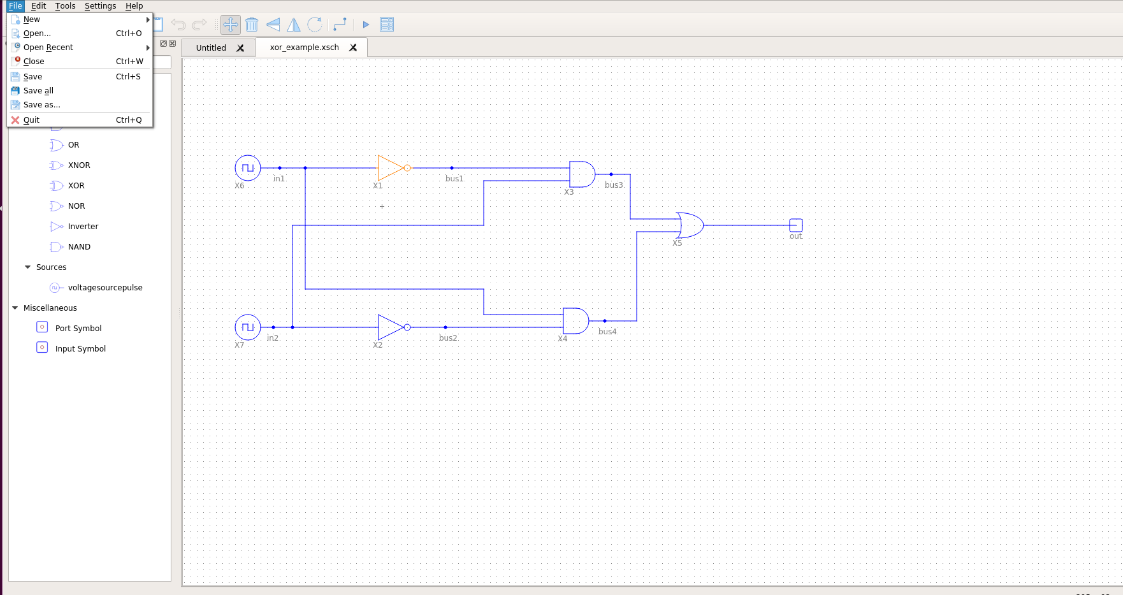
Օգտագործողը կարող է նաև զուգահեռաբար մշակել, հետազոտել և ուսումնասիրել

մի քանի սխեմա։ Այսինքն միանգամից հնարավոր է ունենալ մի քանի ակտիվ պատուհան,

մի քանի ակտիվ սխեմա համապատասխան հետազոտություններով։

“Close all” հրամանը հնարավորություն է տալիս պահպանել բոլոր ակտիվ բաց պատուհանները, միաժամանակ պահպանելով փոփոխությունները։

"Save" հրամանը ընտրելիս կատարված փոփոխությունները գրանցվում են տվյալ ֆայլի մեջ: Եթե այդ ֆայլը չի գտնվում ընթացիկ գրադարանում, ապա օգտագործողն ինքնուրույն պետք է նշի նրա ճանապարհը "File Open" պատուհանի միջոցով: Եթե ֆայլը կարդալիս, տարրերի ցուցակում նշված տարրերի գրաֆիկական պատկերների ֆայլերը չեն գտնվում նշված գրադարաններում, ապա օգտագործողը ինքնուրույն պետք է նշի նրանց ճանապարհները նույն ձևով:



**Նկար 11**

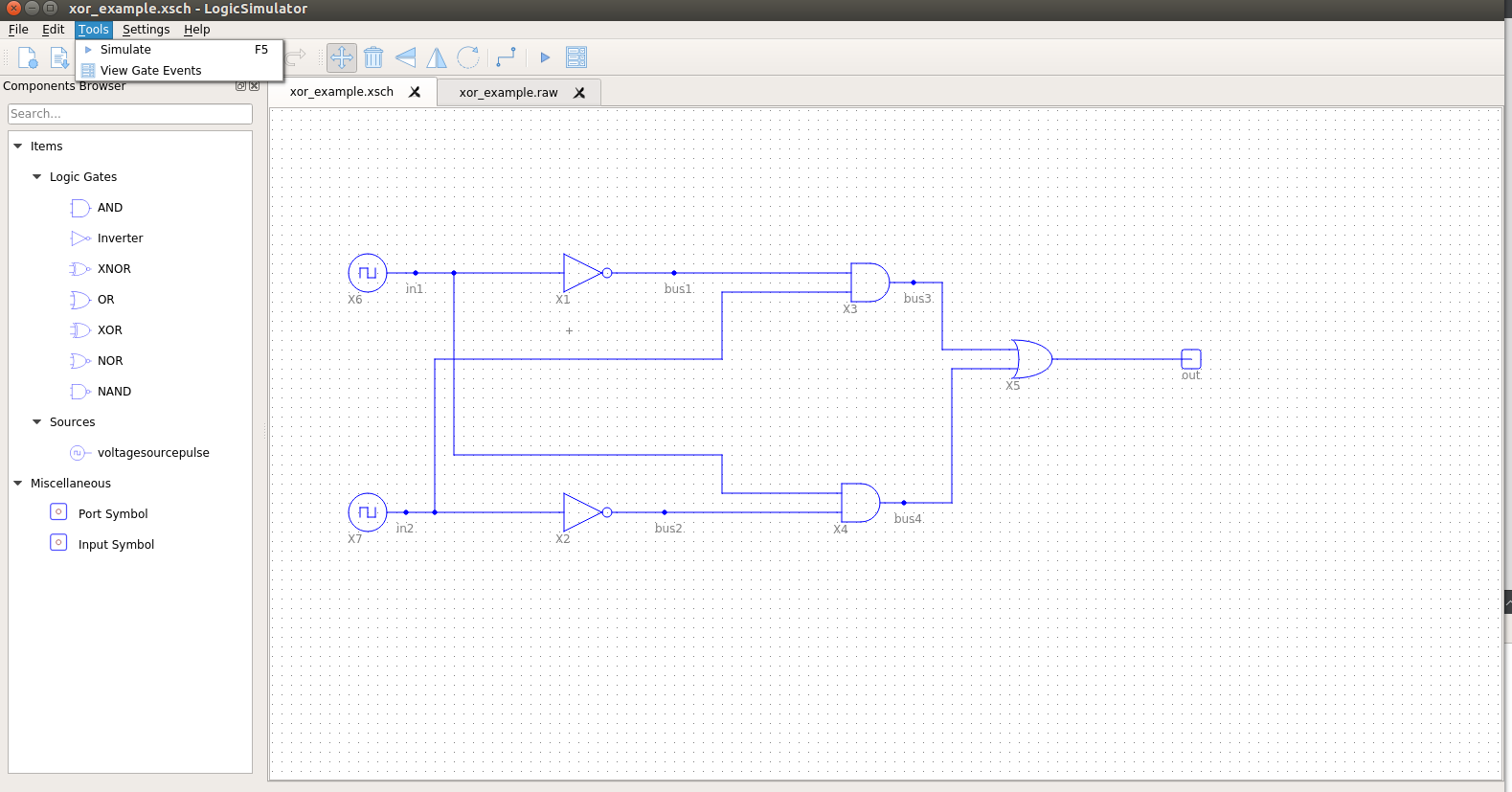
Սխեմայի հետ կատարվող հիմնական հետազոտություններն են՝

* Սխեմայի մոդելավորում
* Տրամաբանական փականների փոխանջատման բնութագծերի ցուցադրում

Սխեմայի հիմնական մուտքերին լարում հատկացնելու համար նախագծվել են լարման աղբյուր հանդիսացող voltage-source pulse տարրը։

Միացումներին անվանակոչելու համար նախագծվել է port symbol տարրը, որը հնարավորություն է տալիս վերագրել անուններ համապատասխան միացումներին։

Այդ իսկ վերագրված անունների շնորհիվ միացումները տարբերակվում են ժամանակային դիագրամների պատուհանում։



**Նկար 12**

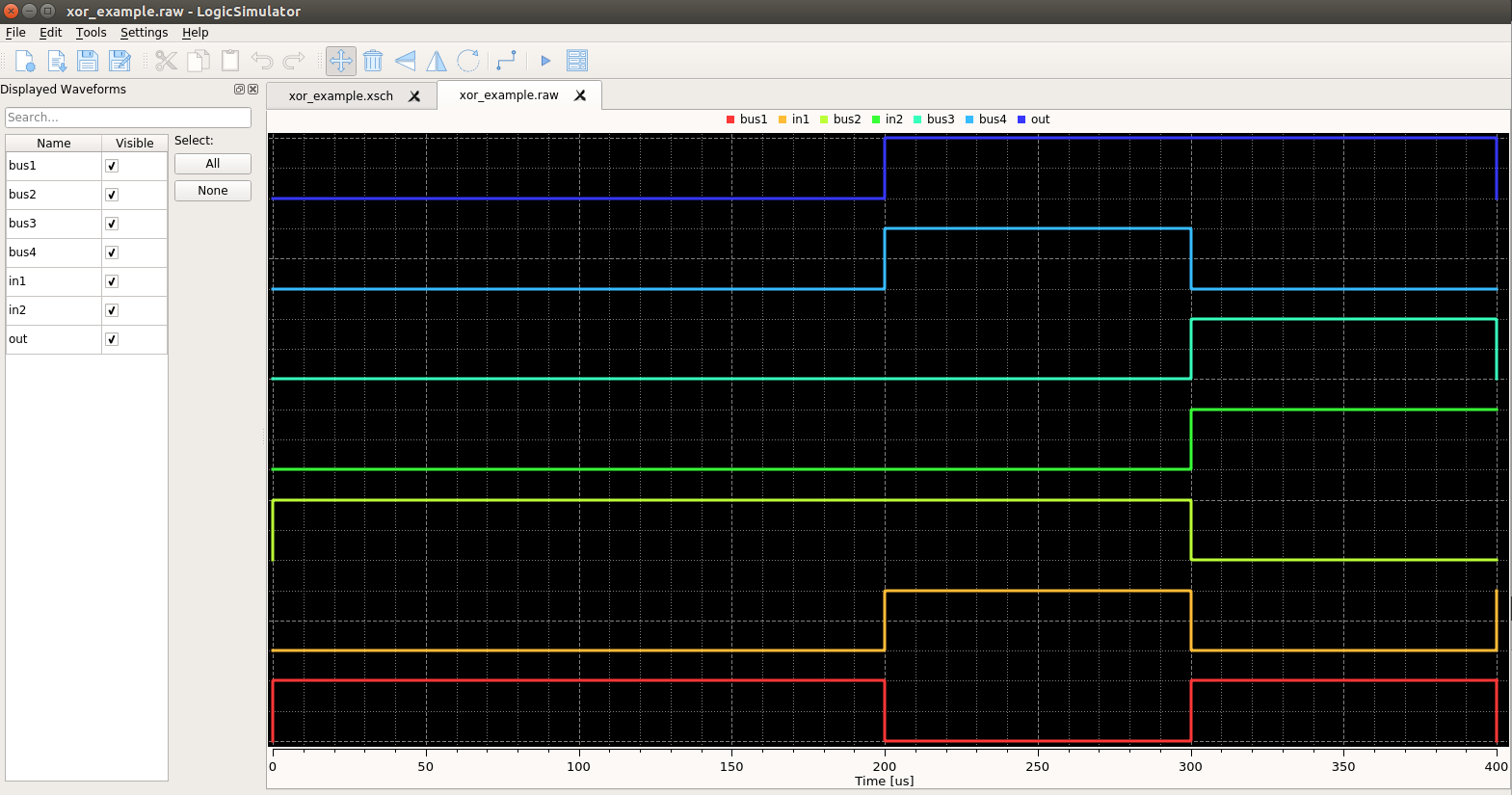
Երբ լրիվ սխեման արդեն կազմված է և բոլոր միացումները արդեն հաստատված են, պետք է տալ մուտքային ազդանշանները սխեմայի մուտքերին, որը կարելի է իրականացնել “voltage pulse” տարրերի միջոցով: Ազդանշանները մուտքերին տալուց հետո պետք է սեղմել "Simulate" կոճակը, և եթե կան տրամաբանական կամ շարահյուսական սխալներ, ապա դուրս է բերվում համապատասխան հաղորդագրություն։

Մուտքային ազդանշանները տալուց հետո, օգտագործողը կարող է նշել այն լարերը, որոնց ժամանակային դիագրամները պետք է դիտվեն օգտագործողի կողմից: Լարերը ընտրվում են գլխավոր մենյուի “connection” տարրերի միջոցով։

Տվյալ սխեման կարելի է հիշել գլխավոր մենյուի "File" բաժանմունքից "Save" կամ "Save As..." հրամանով։

4.2 Մոդելավորման արդյունքները ժամանակային դիագրամների տեսքով

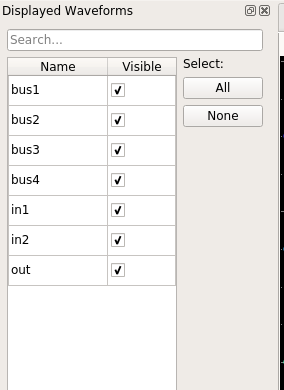
Հետևյալ տեսքը ունի ժամանակային դիագրամների պատուհանը՝



**Նկար 13**

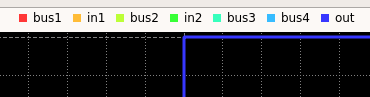
Ծրագրային միջոցը հնարավորություն է տալիս տեսնել բոլոր միացումների վրա լարման արժեքները մոդելավորման ընթացքում, հատուկ ժամանակային դիագրամների բաժնում։

Օգտագործողին հնարավորություն է ընձեռվում ընտրել թե կոնկրետ որ միացումների ժամանակային դիագրամներն է ցանկանում տեսնել, համապատասխանաբար ընտրելով կամ հեռացնելով միացումները ցուցակից՝



**Նկար 14**

Ժամանականին դիագրամների պատուհանում օգտագործողը կարող է տեսնել թե որ միացմանը որ գույնն է համապատասխանում, անհրաժեշտության դեպքում նաև կարող է փոխել գունային համապատասխանությունները՝



**Նկար 15**

ԳԼՈՒԽ 5

Փորձարարական տեխնիկա

Փորձարարական տեխնիկա

Թվաբանական սխեմաների իրադարձային տրամաբանական մոդելավորման ծրագրային միջոցի մշակման և հետազոտման համար օգտագործվել են հետեևյալ ապարատային և ծրագրային միջոցները՝

* Intel (R) Core(TM)i7-3520M CPU @ 2.90GHz(4 CPUs) ~2.9GHz, 4Gb RAM, 250 GB համակարգիչ
* C++ 11, Qt 4.8 ծրագրավորման միջոցների համար նախատեսված թարգմանիչները
* Microsoft Windows 10 Enterprise, Ubuntu 18.04 Linux օպերացիոն համակարգերը
* VCS ծրագրային միջոցը
* ModelSim ծրագրային միջոցը

ԳԼՈՒԽ 6

**Փորձնական հետազոտություններ**

ՓՈՐՁՆԱԿԱՆ ՀԵՏԱԶՈՏՈՒԹՅՈՒՆՆԵՐ

Հետազոտվել է մշակված ծրագրային միջոցի սխեմայի տրամաբանական մոդելավորման ժամանակի՝ սխեմայում գտնվող էլեմենտների քանակից կախվածությունը՝

Ինչպես երևում է ստացված դիագրամից, սխեմայում էլեմենտների աճին զուգահեռ, մոդելավորման ժամանակը աճում է կարելի է ասել գծայնորեն։

ԳԼՈՒԽ 7

**Եզրակացություն**

Եզրակացություն

1. Աշխատանքում հետազոտվել են այժմ գոյություն ունեցող ծրագրային միջոցները, որոնք կատարում են իրադարձային տրամաբանական մոդելավորում:
2. Հետազոտվել են այժմ գոյություն ունեցեղ ալգորիթմերը, որոնք օգտագործվում իրադարձային տրամաբանական մոդելավորում իրականացնելու համար:
3. Հետազոտվել է ծրագրային միջոցի մոդելավորման ժամանակի կախումը սխեմայի էլեմենտների քանակից:
4. Հետազոտման արդյունքները համեմատվել են VCS (Synopsys) և ModelSim(Mentor Graphics) ծրագրային միջոցների հետ։

Այսպիսով՝ կատարվել են խնդրի դրվածքում առաջադրված պահանջները:

Գրականության ցանկ

1. Mark Summerfield Creating Great Software with C++/Qt 4 // Prentice Hall (2013) – P 553.
2. Stroustroup B. The C++ Programming Language 4th edition// Addison-Wesley Professional (2012) – P1368.
3. Allen M. Data structures and algorithm analysis in C++ // Addison-Wesley Professional (2006) – P676.
4. Jasmin Blanchette C++ GUI Programming with Qt 4// Addison – Wesley Professional (2006) - P 556.
5. Booch G. Object Oriented Analysis and Design with Applications // Addison-Wesley Professional (2007) – P543
6. Melikian V. Logic simulation of digital circuits exposed to radiation // Facta universitatis,  
   (1999) -Vol.12, No.l. -P.1-16.