ՀՏԴ 004.42

Ինֆորմատիկա

Գարեգին ՍԱՐԳՍՅԱՆ
«Սիմենս Ինդասթրի Սոֆթվեր», ֆ-մ. գ. թ.

Արթուր ՄԱԹԵՎՈՍՅԱՆ
ՀԱՊՀ – ի մագիստրանտ
«ՍԻՆՈՓՍԻՍ ԱՐՄԵՆԻԱ»
Էլ. Հասցե artur.matevosyan99@gmail.com

ՍԱՀՄԱՆԱՓԱԿ ՇԵՐՏԵՐԻ ԿԻՐԱՌՄԱՄԲ ԵՐԿԿՈՂՄԱՆԻ ՈՒՂԵԳԾՄԱՆ ԾՐԱԳՐԱՅԻՆ ՄԻՋՈՑԻ ՄՇԱԿՈՒՄԸ

Կատարվել է ծանուցում ինտեգրալ սխեմաների նախագծման փուլերի վերաբերյալ: Ուսումնասիրվել է անհրաժեշտ գրականություն գրաֆների վերաբերյալ, ինչպես նաև փնտրման և գրաֆի շրջանցման ալգորիթմներ առաջադրված խնդրի լուծման համար: Մշակվել և իրականացվել է ծրագրային միջոց, որը օգտագործողի կողմից մուտքագրված գրաֆի հիման վրա կատարում է ուղեգծում շերտերի վրա: Ծրագիրը ապահովում է գրաֆիկական ինտերֆեյս, ինչպես նաև ծրագրից օգտվելու համար նախատեսված գործիքներ, որոնք նպաստում են հարմարավետությանը: Աշխատանքն իրականացվել է C++ ծրագրավորման լեզվի միջոցով Qt միջավայրում՝ Windows և Linux օպերացիոն համակարգերի համար:

***Բանալի բառեր*՝** *գրաֆ, C++/QT, սխեմա, տրամաբանական տարրեր*

Было объявлено об этапах проектирования интегральных схем. Изучена необходимая литература по графам, а также алгоритмы поиска и обхода графов для решения поставленной задачи. Разработан и реализован программный инструмент, который на основе заданного пользователем графа выполняет трассировку по слоям. Программа предоставляет графический интерфейс, а также инструменты для использования программы, которые способствуют удобству. Работа

выполнена с использованием языка программирования C++ в среде Qt для операционных систем Windows и Linux.
Ключевые слова: *Граф , C++/QT, схема, логические элементы*

The stages of designing integrated circuits were announced. The necessary literature on graphs was studied, as well as algorithms for searching and traversing graphs to solve the problem. A software tool has been developed and implemented that, based on a graph specified by the user, performs tracing by layers. The program provides a graphical interface as well as tools for using the program that contribute to convenience. The work was done using the C++ programming language in the Qt environment for Windows and Linux operating systems.
Key words: *Graph, C++/QT, Schematic, Logic Elements*

Ներածություն

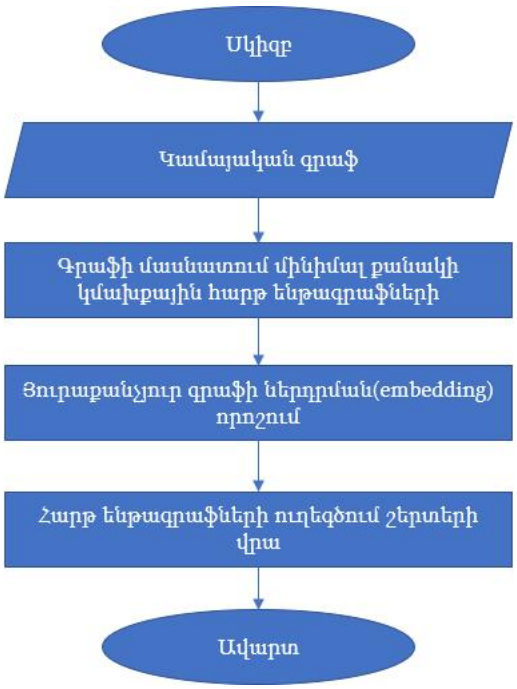
Ժամանակակից խելացի սարքավորումները, ինչպիսիք են՝ համակարգիչները, պլանշետները, բջջային հեռախոսները, հեռուստացույցները, մեքենաները իրենց ֆունկցիոնալությունը ապահովում են իրենց մեջ տեղակայված միկրոսխեմայի շնորհիվ: Միկրոսխեման ֆիզիկական տարրերից (տրանզիստորներից, դիմադրություններից, ունակություններից) կազմված և տրամաբանորեն իրար հետ կապված բաղադրիչների ամբողջությունն է, որը կատարում է որոշակի ֆունկցիա: Ժամանակակից միկրոսխեմաները բաժանվում են երկու տեսակի՝ թվային և անալոգային՝ կախված ազդանշանի տեսակից, որով սխեմայի տարրերը հաղորդակցվում են միմյանց հետ: Եվ կա սխեմաներ նախագծելու երկու եղանակ՝ ավտոմատ և ավտոմատացված: Ավտոմատացված նախագծման ժամանակ մարդը մասնակցում է նախագծման գործընթացին, և ամբողջ ընթացակարգը չի իրականացվում գործիքակազմով: Մարդը օգտվելով պատրաստի գործիքակազմից, ինքնուրույն նախագծում է սխեմայի առանձին բաղադրիչները, այնուհետև այդ բաղադրիչները օգտագործելով՝ նախագծում է տվյալ բաղադրիչներից ավելի բարդ ֆունկցիոնալություն ապահովող բլոկներ, և այդպես՝ մինչև վերջնական սխեմայի ստացումը: Իսկ ավտոմատ նախագծման ժամանակ ամբողջ ընթացակարգը կատարվում է ավտոմատ: Գործիքակազմը ստանալով՝ սխեմայի վարքային նկարագրությունը, որը գրվում է մարդկանց կողմից հիմնվելով պատվիրատուի կողմից՝ տրված պահանջների վրա, նախագծում է այն: Սխեմայի նախագծումը բաժանվում է մակարդակների: Ամենացածր մակարդակը տրանզիստորային, կամ սխեմատեխնիկական

մակարդակն է. որտեղ թվային սխեմաներում բազային էլեմենտներով՝ տրանզիստորներով, նախագծվում են տրամաբանական փականներ [1] (AND, OR, NAND, NOT, NOR, XOR)։ Վերջինը համակարգային (system) մակարդակն է, այս մակարդակում նախագծվում են պրոցեսորներ։

Խնդրի դրվածքը

Ալգորիթմի ընդհանուր նկարագրություն

Ծրագրային միջոցի ալգորիթմը բաղկացած է հիմնական երեք քայլերից, որոնց հաջորդական կատարման արդյունքում, հիմնվելով մուտքագրված գրաֆի վրա, կատարվում է ուղեգծում։



Նկ.1

Քանի որ մուտքագրված գրաֆը կարող է հարթ [2] չլինել, և այդպիսի գրաֆի վրա հիմնվելով հնարավոր չէ ուղեգծել մեկ հարթության վրա, այն պետք է մասնատել ենթագրաֆների և հետևելով խնդրի պահանջներին՝ այն պետք է մասնատել հնարավորինս քիչ ենթագրաֆների։ Բացի դրանից, այն պետք է մասնատել այնպես,որ յուրաքանչյուր ենթագրաֆը լինի կմախքային, քանի որ յուրաքանչյուր գագաթ իրենից ներկայացնում է տրամաբանական տարր, իսկ յուրաքանչյուր տարրֆիքսված է և ունի իր հստակ դիրքը հարթության վրա։

1. Այս քայլը նպատակաուղղված է առաջին քայլից ստացված

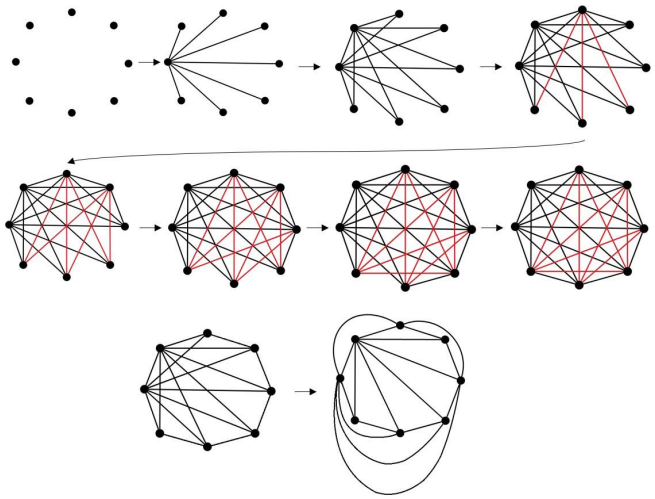
յուրաքանչյուր գրաֆի համար այսպես կոչված գրաֆի ներդրումը (embedding) ստանալուն: Այն ինֆորմացիա է, որը հնարաորություն է տալիս հարթ գրաֆի արտապատկերման գործնըթացը կազմակերպել: Այն կարելի է ներկայացնել ինչպես յուրաքանչյուր գագաթի համար նրան կից կողերի այնպիսի հաջորդականություն, որ այնպահպանելով հնարավոր է գրաֆը պատկերել հարթության վրա առանց կողերի հատումների:

2. Վերջին քայլով, ունենալով հարթ գրաֆներ և այդ գրաֆների embedding-ները յուրաքանչյուր գրաֆի համար կատարվում է ուղղեգծում չօգտագործված որևէ շերտի (հարթության) վրա:

Գրաֆը կմախքային հարթ ենթագրաֆների մասնատում

Ծրագրում գրաֆը կմախքային հարթ ենթագրաֆների մասնատման համար մշակվել է հետևյալ ալգորիթմը: Ալգորիթմ: Դիցուք G-ն տրված կամայական գրաֆ է: G գրաֆի գագաթների բազմությունով կառուցվում է H կմախքային ենթագրաֆը, $V(H) = V(G)$, իսկ $E(H) = \emptyset$: Որիցհետո հատ-հատ E(G) կողերի բազմությունից E(H) կողերի բազմություն են, ավելանում այնպիսի կողերը, որոնք չեն խախտում H գրաֆի հարթ լինելու պայմանը, արդյունքում ստանալով այնպիսի հարթ H գրաֆ, որ $\forall e,\ e \in E(G)$ և $e \notin E(H)$ կող H գրաֆ ավելացնելուց հետո կխախտի հարթ լինելու հատկությունը: Այնուհետև, վերոնշյալ գործողությունները կատարվում են $G' = (V(G), E(G) \setminus E(H))$ և H' գրաֆների համար և այդպես շարունակ այնքան ժամանակ քանի դեռ $E(G') \neq \emptyset$:

Նկ.2-ում բերված է ալգորիթմի աշխատանքի արդյունքում K_8 լրիվ գրաֆից առանձնացված հարթ կմախքային ենթագրաֆներից մեկը:



Նկ. 2

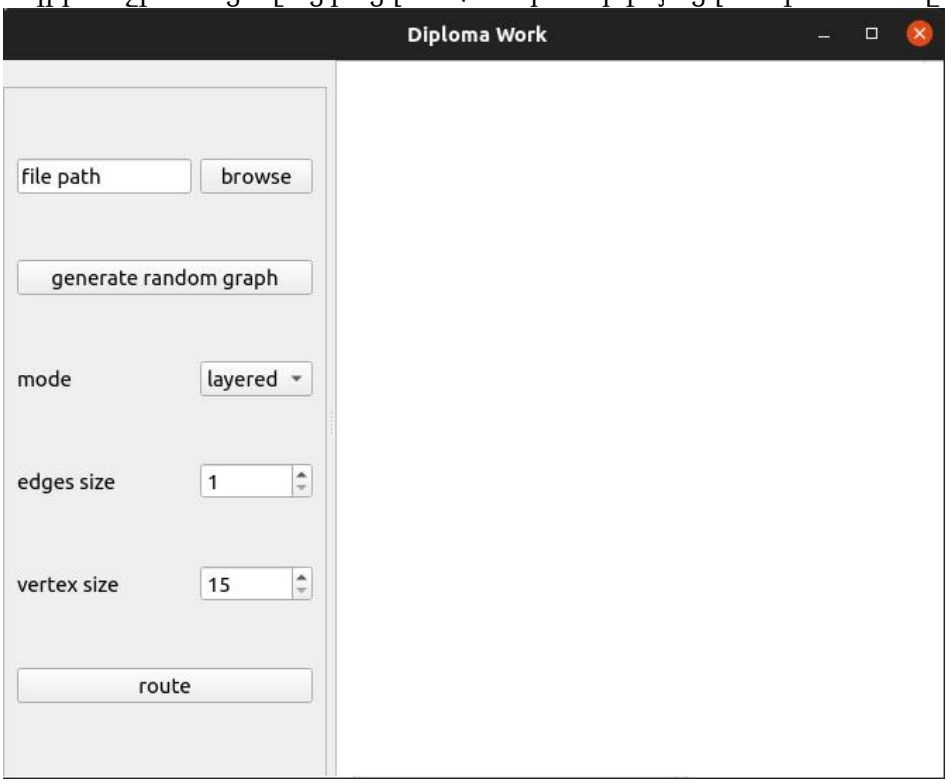
Ալգորիթմի բարդության գնահատումը

Քանի որ յուրաքանչյուր կող ավելացնելու համար պետք է ստուգել՝ գրաֆը հարթ է, թե ոչ, իսկ ստուգելու գործողությունը կատարվում է O(n) բարդությամբ, գրաֆից կմախքային հարթ ենթագրաֆ առանձնացնելու գործողությունը կլինի O(nm) բարդությամբ, որտեղ n-ն և m-ն համապատասխանաբար գրաֆի գագաթների և կողերի քանակներն են: Իսկ ամբողջ ալգորիթմի բարդությունը, օգտվելով այն փաստից, որ հարթ գրաֆների գագաթների քանակի և կողերի քանակի միջև գոյություն ունի գծային կապ, կարելի է պնդել, որ այն $O(n^2m)$ է:

Գրաֆիկական ինտերֆեյս

Գրաֆիկական ինտերֆեյսը իրագործված է Qt միջավայրում: Այն տրամադրում էպարզ և հարմարավետ ինտերֆեյս, որը նպաստում է ծրագրից օգտվելու հարմարավետությանը:

Ծրագիրն աշխատեցնելուց բացվում է ստորև ներկայացված պատուհանը՝



Նկ. 3

Ծրագրին մուտքային տվյալ տալու համար անհրաժեշտ է սեղմել **browse** կոճակը, որից հետո բացված ֆայլային համակարգից ընտրել համապատասխան ֆայլը կամ **browse** կոճակից աջ գտնվող հատվածում տալ ֆայլի հասցեն: Ֆայլում պետք է նկարագրված լինի գրաֆը հետևյալ կերպով՝ [գագաթի համարը] : [հարևան գագաթ], …: Օրինակ՝

1 : 2, 3

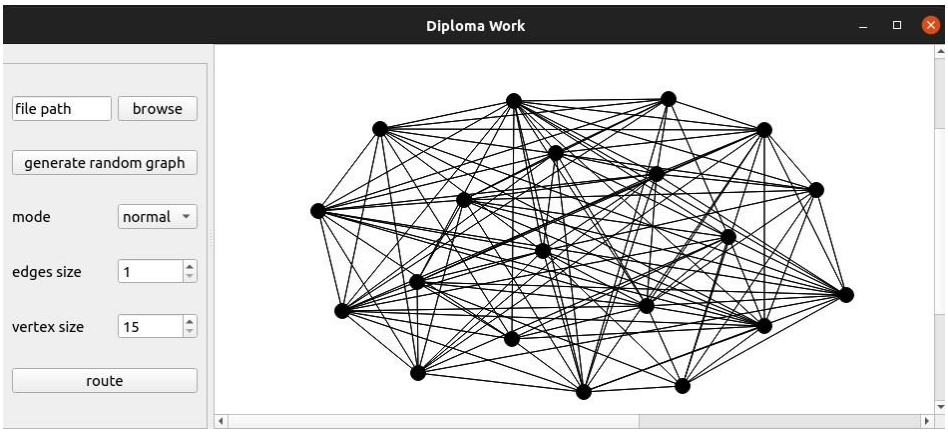
2 : 1, 3

3 : 1, 2

Եթե տրվի անհամապատասխան ֆայլ, կամ ֆայլում լինի որևէ սխալ մուտքագրված տվյալը, այն չեղյալ կհամարվի:

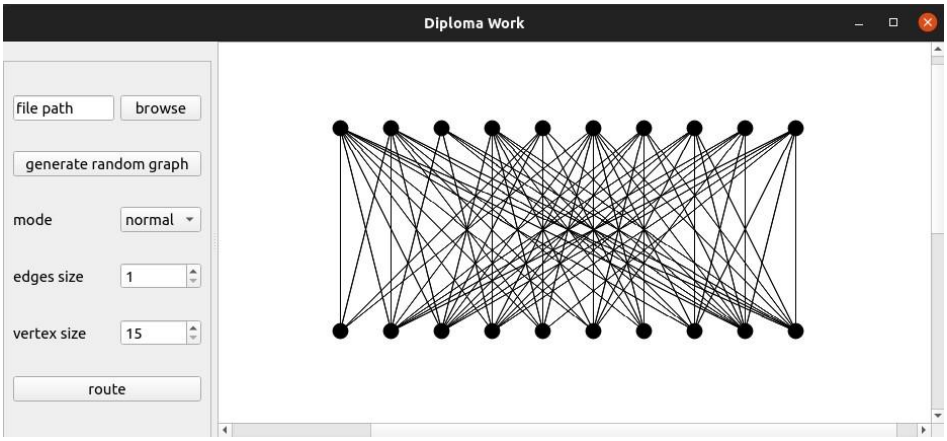
Ծրագրին բացի ֆայլի միջոցով մուտքային տվյալ տալուց նաև հնարավորություն է տրված գեներացնել պատահական տվյալ (գրաֆ): Պարզապես պետք է սեղմել **generaterandom graph** կոճակը, հետո բացված պատուհանում նշել գագաթների քանակը, որից հետո կգեներացվի նշված քանակի գագաթներով և պատահանականորեն ընտրված գագաթներից կազմված կողերով գրաֆ:

Գրաֆ գեներացնելուց կամ ճիշտ մուտքային տվյալներ տալուց հետո այն կարտածվի գրաֆիկական միջավարյում: Եթե գրաֆը երկկողմանի չէ, գագաթները դասավորվում են պատահական կերպով(նկ. 9), իսկ եթե գրաֆը երկկողմանի է, ապագագաթները



Նկ.4

տրոհվում են երկու ենթաբազմությունների այնպես, որ ենթաբազմությունների գագաթները իրար հարևան չլինեն և դասավորվեն առանձին տողերում, ինչպես պատկերված է նկ. 5 ում:

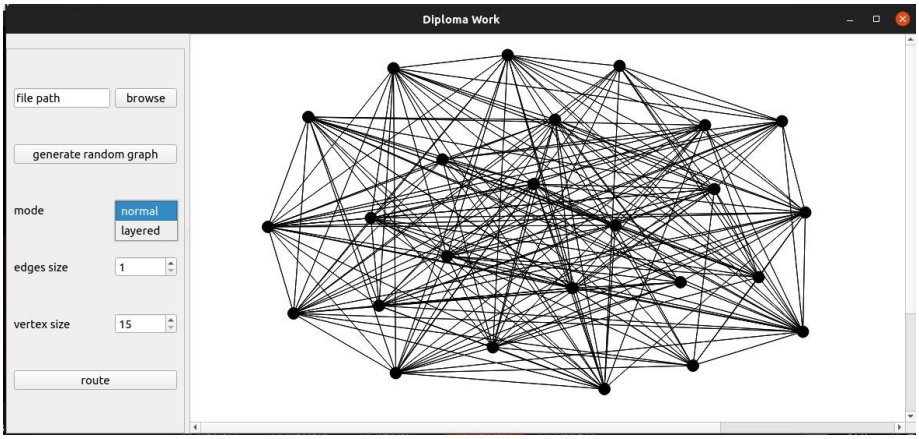


Նկ.5

Ցանկության դեպքում մկնիկի ցուցիչով հնարավոր է պատկերվող գրաֆի գագաթներըղիրքափոխել։

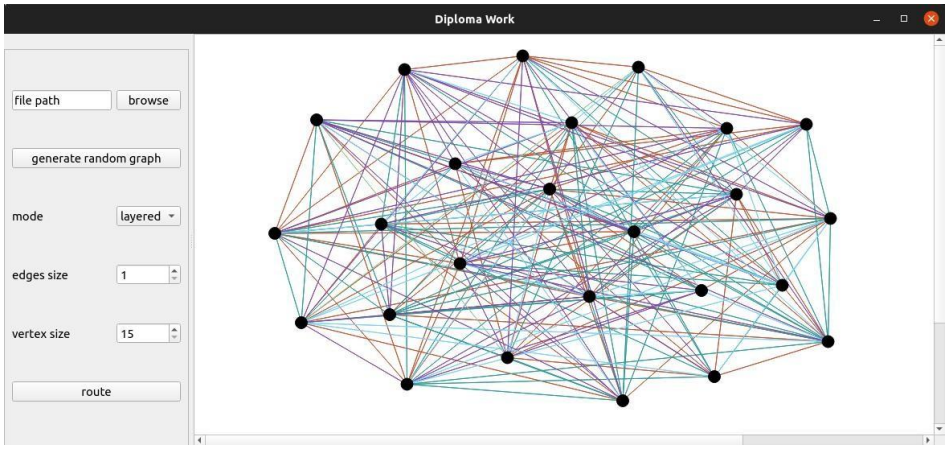
Հնարավորություն է տրված նաև փոփոխել պատկերվող գրաֆի կողերի հաստությունը նգագաթների չափերը, համապատասխանաբար, edge size-ում և vertex size-ում տալով դրանց թվային արժեքներ։

Հնարավոր է նաև փոփոխել գրաֆի արտապատկերման ձևը mode-ից ընտրելով normal, կամ layered ռեժիմը։ [3-4] Normal mode – ի ժամանակ անկախ գրաֆը հարթ է թե ոչ, պատկերում գրաֆը ամբողջությամբ ներկվում է միագույն (սև)։



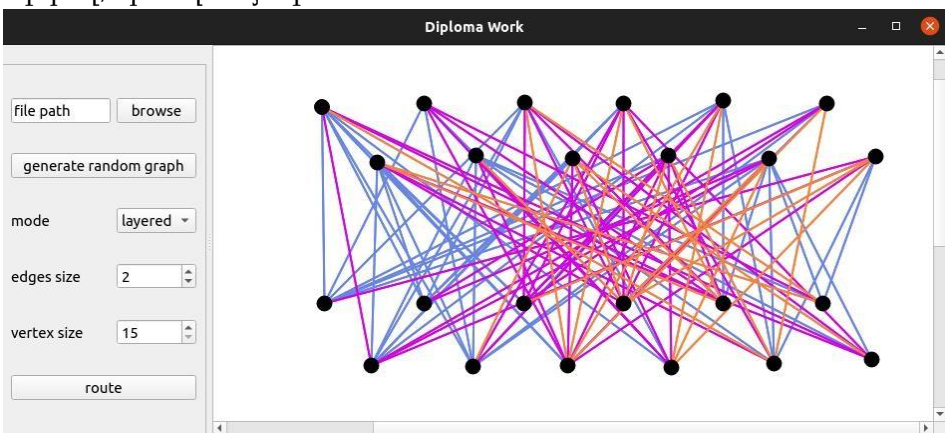
Նկ.6

Layered mode – ի ժամանակ կողերը ներկվում են համապատասխանաբար շերտերի (հարթությունների), որտեղ միևնույն գույնի կողերով կազմված գրաֆները հարթ են (գույները յուրաքանչյուր շերտի համար ընտրվում են պատահականորեն)։ Այլ կերպ ասած, գույներով տարբերակվում են գրաֆից մասնատված կմախքային մաքսիմալ հարթ ենթագրաֆները։



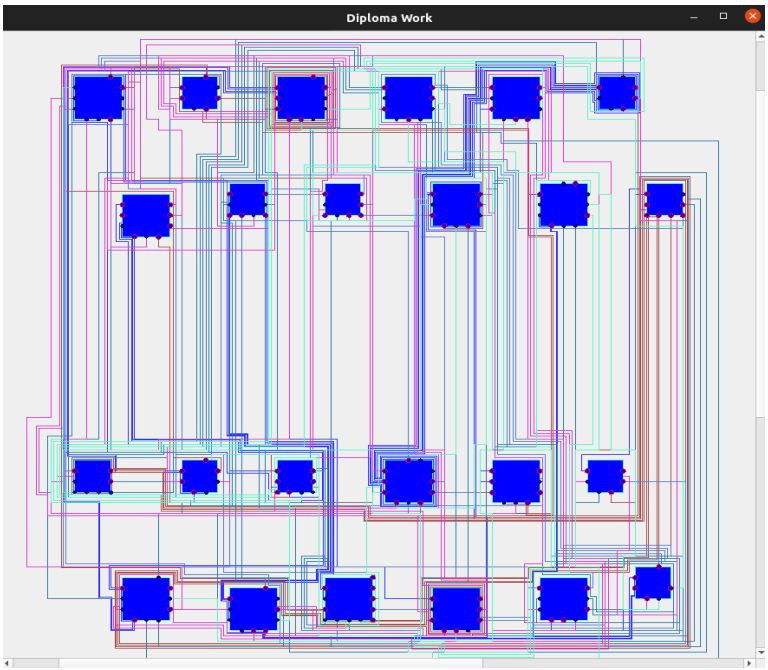
Նկ.7

Եվ վերջապես route կոճակը սեղմելուց հետո սկսվում է ուղեգծման գործընթացը։ Այն կատարվելուց հետո բացվում է նոր պատուհան սպասված պատկերով, որտեղ նույնպես



Նկ.8

յուրաքանչյուր շերտ առանձնացված է գույներով (գույները ուղեգծված պատկերում կարող է չհամընկնել layerd mode-ում պատկերված գրաֆի շերտերի գույների հետ)։ Տրամաբանական տարրերը պատկերվում են կապույտ գույնի քառակուսիների տեսքով, իսկ նրանց դիրքերը որոշվում է համապատասխանաբար գրաֆիկական միջավայրում պատկերված գրաֆի գագաթների դիրքերով (Նկ. 8)։



Նկ.9

Գրականություն

1. Պ.Ա.Պետրոսյան, Վ.Վ.Մկրտչյան, Ռ.Ռ.Քամալյան, Գրաֆների տեսություն։
2. Programming: Principles and Practice Using C++: Bjarne Stroustrup.
3. The Left-Right Planarity Test: Department of Computer & Information Science, Universityof Konstanz: Ulrik Brandes.
4. Planarity Testing And Maximal Planar Subgraph: Ibrahim Gurgil.

Հոդվածը տպագրության է երաշխավորել խմբագրական խորհրդի անդամ, ֆ.մ.գ.թ. Գ.Հ.Սահակյանը։