Анотація

В даному курсовому проекті потрібно вирішити задачу розміщення елементів, а саме – розміщення різногабаритних елементів. Для цього розроблено графічну оболонку, яка дозволяє користувачу завантажити файл, запустити обчислення вхідних даних та вивести отриманий результат у файл.

Програма забезпечує:

1) введення вхідних даних;
2) вирішення задачі розміщення;
3) вивід результатів у файл або вікна з помилкою, якщо некоректні введені дані.

Програмне забезпечення розроблено з використанням мови програмування Java та стандарту розробки інтерфейсу користувача Java FX.

Зміст

Вступ

З появою нової елементної бази (мікроконтролерів, програмних логічних інтегральних схем, трансп’ютерів) однією з основних задач проектування обчислювальних і керуючих систем стала задача розподілу елементів цих систем в монтажному просторі один відносно одного таким чином, щоб довжина ліній зв'язку була мінімальною.

Рішення зазначеної задачі полягає у визначенні такого порядку закріплення елементів в посадочних місцях монтажного простору, при якому найбільш зв’язані елементи розташовуються максимально близько один до одного.

Мінімізація довжини ліній зв'язку в результаті впливає на показники надійності схеми. Чим коротші провідники, тим менше шумів, менша затримка передачі сигналу, нагрів і споживання електроенергії.

З ростом інтеграції мікросхем процес їх проектування стає все більш трудомістким із-за великого числа (десятки і сотні тисяч) компонентів (транзисторів, резисторів, конденсаторів, генераторів частот), розташованих на кристалі мікросхеми, при їх проектуванні доводиться вирішувати завдання дуже великої розмірності.

1 Постановка задачі

Для набору, який задає користувач, потрібно вирішити задачу розміщення елементів. Результати вивести у вказаний файл.

2 Огляд методів вирішення

Так як різногабаритні елементи в переважній більшості мають прямокутний вигляд, тому абстрагуємося від елементів іншого формату. А також будемо вважати, що основна плата теж прямокутного форм-фактору.

В кожного елементу мають бути задані розміри, кількість контактів та їх розташування на самому елементі.

Розглянемо окремий елемент: при певному рівні наближення можна вважати його масивом зв’язаних один відносно одного фіксованих елементів. В такому разі можна використати поширений метод «Гілок і меж» з невеликою модифікацією для оперування масивами елементів.

3 Опис алгоритму

Алгоритм розміщення елементів:

1) Виробляються по черзі фіксація всіх елементів схеми в першому вільному посадковому місці монтажного простору і обчислюється сумарна довжина ліній зв'язку (L (G)) для кожного фіксованого елемента;

L (G) обчислюється як сума трьох складових:

- Сумарна довжина ліній зв'язку, між фіксованими елементами: L1 (G);

- Сумарна довжина ліній зв'язку, що пов'язують фіксовані та нефіксовані вершини L2 (G);

- Сумарна довжина L3 (G) ліній зв'язку, суміжних тільки нефіксованим елементам, яка визначається як сумарна довжина ребер стандартного графа,

побудованого на незафіксованих елементах, і неврахованих лініях зв'язку, що залишилися;

2) Вибираються варіанти фіксації елементів з мінімальною величиною L (G);

3) Проводиться повторення перерахованих вище кроків до тих пір, поки не будуть зафіксовані всі елементи.

Опис монтажного простору:

Лінії зв'язку між елементами на монтажному просторі можуть проходити тільки в ортогональних напрямках;

При вирішенні задачі розміщення вихідними даними є монтажний простір і схема з'єднань розміщуваних в просторі модулів.

Під монтажним простором розуміють метричний простір, в якому встановлюють вхідні в нього типові конструкції. Зазвичай такий
простір для типової конструкції має прямокутну форму з фіксованими позиціями установки елементів. В якості математичної моделі монтажного простору використовують неорієнтовний топологічний граф решітки. Для побудови графа необхідно:

- площину монтажного простору розбити на елементарні квадратні майданчики, сторони яких дорівнюють кроці прокладання зв’язків;
- кожному елементарному майданчику поставити у відповідність вершину графа;
- з'єднати між собою ребрами сусідні вершини в ортогональних (або інших, дозволених для прокладки з'єднань) напрямках.

Таким чином, дві вершини графа з'єднані ребром, якщо між відповідними елементарними майданчиками можна провести з'єднання.

Відстань між вершинами графа при ортогональному з'єднанні в просторі можна визначити за формулою:

$$d_{ij} = \left| s_i - s_j \right| + \left| t_i - t_j \right| ,$$

Де (s_i, t_i), (s_j, t_j) - координати розташування вершин x_i, x_j в просторі.

Функцію відстаней для графа задають матрицею відстаней $D_r = \| d_{ij} \|_{n*n}$.

В якості математичної моделі схеми з'єднань розміщуваних елементів використовують неорієнтовний мультиграф $G(X, U)$, множині X, вершини якого ставляться у відповідність множині розміщуваних елементів, а ребра U вказують на наявність з'єднань між елементами. Таким чином, ребро мультиграфа схеми буде вказувати на наявність зв'язків між вершинами x_i, x_j відповідним елементам схеми.

Для ланцюгів автоматизованої обробки використовують матричні еквіваленти мультиграфа: матрицю суміжності і матрицю інцидентності. Для задачі розміщення зручно скористатися матрицею суміжності $A = \| a_{ij} \|_{n*n}$, елементи якої утворюються за правилом:

$$a_{ij} = \begin{cases} m, \text{якщо } x_i \text{ зв'язана } m \text{ ребрами з вершиною } x_j \\ 0, \text{якщо вершини } x_i \text{ и } x_j \text{ не зв'язані} \end{cases}$$

Завдання розміщення зводиться до відображення графа $G(X, U)$ схеми в сітку G_r таким чином, щоб вершини X графа G розміщувалися у вузлах сітки та величина сумарної довжини зв'язків

$$L(G) = \sum_i \sum_j a_{ij} * d_{ij} \to \min$$

була мінімальною для всіляких способів ототожнення вершин графа і вузлів сітки.

Розглянемо можливість реалізації методології гілок і меж для вирішення задачі лінійного розміщення графа в каналі.

Нехай заданий граф решітки, в якому число стовпців k дорівнює числу вершин графа, а число рядків m=1. Завдання полягає в тому, щоб відобразити вершини графа за критерієм мінімуму сумарної довжини ребер графа G, тобто знайти підстановку

$$t = \begin{pmatrix} 1, & 2, \ldots, n \\ x_{i1}, & x_{i2}, \ldots, x_{in} \end{pmatrix}$$

де $1,2,\ldots,n$ – номера позицій графа решітки, а x_i $(i=1,2,\ldots,n)$ – вершини графа.

Неважко бачити, що таких підстановок, тобто рішень задачі лінійного розміщення, існує $n!$.

Для ефективного вирішення задачі лінійного розміщення методом гілок і меж необхідно:

a) вибрати метод знаходження нижньої межі сумарної довжини ребер графа;

b) вибрати метод розгалуження дерева рішень, що дозволяє відсікати гілки, що містять завідомо непридатні рішення.

Для визначення нижньої межі сумарної довжини L (G) ребер графа G скористаємося поняттям стандартного графа $G\Delta$.

Нехай дано довільний граф, причому, і задана решітка з кроком, рівним 1, число вузлів якої більше або дорівнює. Граф G довільним чином відображений в решітку. Тоді ребра графа G матимуть вагу $1,2,\ldots,N$ (N - вага найбільш довгого ребра графа) залежно від відстані між вершинами, яким інцидентні зазначені ребра.

Неважко бачити, що якщо G_r має розміри $m \times k$, то $N=m+k-2$. Введемо поняття стандартного графа $G\Delta=(X\Delta,U\Delta)$ для графа, відображеного в решітку. Граф має стільки ж вершин і ребер, скільки граф G.

Граф $G\Delta$ утворюється наступним чином. Вершини послідовно відображаються в ті ж вузли решітки, що й вершини X. Потім заповнюються всілякі зв'язку в решітці, вага яких дорівнює одиниці, двом і т.д. до тих пір, поки загальне число ребер стане рівним r.

Основна ідея знаходження нижньої оцінки сумарної довжини ребер довільного графа полягає в наступному. Спочатку підраховуємо число вершин і ребер графа, який розташований в решітці. Далі в решітці будується стандартний граф, що має таке ж число вершин і ребер, як і граф G. Побудова ведеться шляхом послідовного нашарування в решітку спочатку всіх ребер, вага яких дорівнює

одиниці. Якщо число одиничних ребер графа дорівнює або більше кількості ребер графа G, то процес побудови закінчується.

В іншому випадку послідовно додаємо ребра з вагою рівним двом, трьом і т.д. до тих пір, поки загальне число ребер графа стане рівним числу ребер графа G. Підрахувавши сумарну довжину ребер графа, отримаємо нижню оцінку мінімальної сумарної довжини для графа G і для будь-яких інших графів з таким же числом вершин і ребер таким же чином відображених в задану ґрати.

Оскільки йдеться про лінійне розміщення, розглянемо граф, в якому число стовпців дорівнює числу вершин розміщується графа схеми з'єднань, а число рядків дорівнює одиниці. Для отримання дерева рішень будемо фіксувати деякі вершини в певних позиціях решітки. Якщо, наприклад, зафіксувати вершину на будь-якому місці решітки, то в цьому випадку отримуємо підмножина, що містить рішень. При фіксації двох, трьох і т.д. вершин отримаємо відповідно, $(n-2)!$ і $(n-3)!$ рішень. Неважко бачити, що, використовуючи такий метод розгалуження, отримаємо підмножину, що містить одне рішення.

Нехай вершини $x_i, x_{i2}, \ldots, x_{iq}$ фіксовані відповідно в позиціях графа решітки. Нефіксовані вершини графа G можуть розташовуватися у вільних вузлах графа решітки довільним чином. При цьому сумарна довжина складається з трьох частин:

$$L(G) = L_1(G) + L_2(G) + L_3(G)$$

Зауважимо, що фіксація вершин при розміщенні графа може лише збільшити або залишити без зміни сумарну довжину ребер графа в порівнянні з нижньою оцінкою, отриманої за стандартним графом.

На третьому етапі проводиться стиснення отриманого розподілу, шляхом спроби перенесення вершин з двох процесорів на один.

На четвертому етапі проводиться моделювання роботи обчислювальної системи організованої у вигляді вектора.

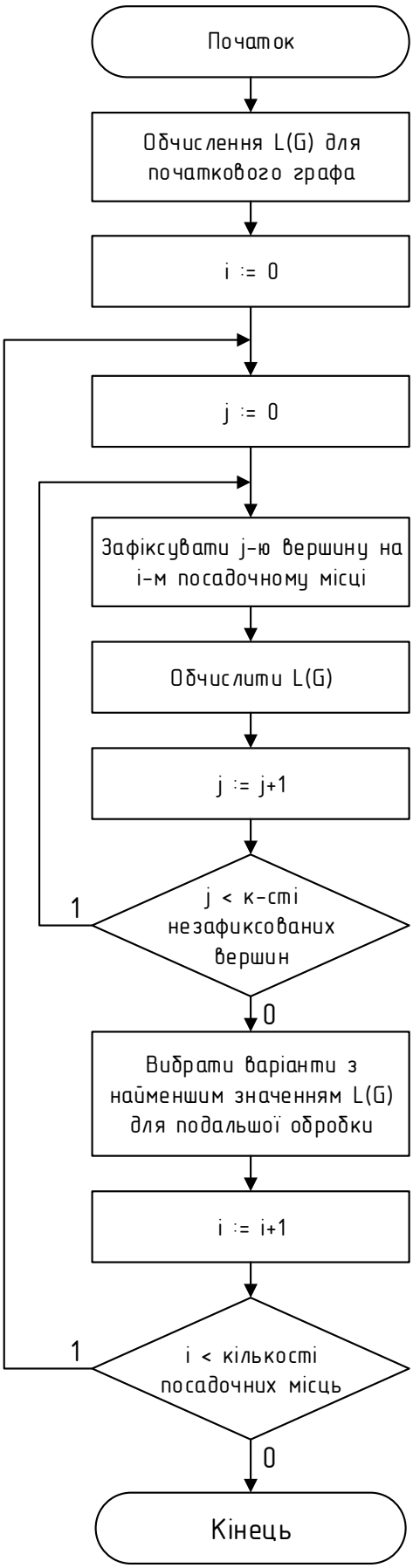

Рис 1.1. Блок-схема алгоритма

4 Опис реалізації

Програмна реалізація виконана на мові високого рівня Java в середовищі програмування Intellij IDEA.

Комбінаційна схема включає в себе сутності трьох класів: елементи, зв’язки між ними, входи/виходи.

Кожен елемент містить в собі тип, ідентифікатор, кількість вхідних зв’язків, інформацію про дані на входах та на виході. Зв’язок містить дані про елементи, які він поєднує та графічне відображення. Зв’язок не існує без елементів. Входи та виходи містять ідентифікатори.

Користувач створює набір вхідних параметрів, використовуючи доступні елементи та входи-виходи. Це створює структуру, що буде будуватися.

Перед виконанням парсер зчитує дані з файлу. Якщо знаходить помилки – видає повідомлення з характером помилки і описом, якщо такий передбачено.

Далі за алгоритмом Гілок і меж розмішуються задані елементи. А результат пропонується зберегти у файл, який задає користувач.

5 Опис структури даних

Клас Main.java запускає програму на виконання. Користувач завантажує вхідні дані у заданому форматі (JSON), показаному нижче:

```
{
        id:1,
        name:”IPK200N”,
        x_size: 4,
        y_size:3,
        pins_top:[
```

```
    {
        pin_id:1,
        x_large: 0,
        y_large: 0,
        x_small:0,
        y_small:2,
        {
            link_to_id:2,
            link_to_pin_id:4
        }
    }
],
pins_bottom:[
    {
        pin_id:2,
        x_large: 3,
        y_large: 2,
        x_small:0,
        y_small:1,
        {
            link_to_id:2,
            link_to_pin_id:2
        }
    }
],
pins_left:[
],
pins_right:[
    {
```

```
                pin_id:3,
                x_large: 3,
                y_large: 2,
                x_small:0,
                y_small:2,
                {
                        link_to_id:2,
                        link_to_pin_id:1
                }
        },
        {
                pin_id:4,
                x_large: 3,
                y_large: 1,
                x_small:0,
                y_small:2,
                {
                        link_to_id:3,
                        link_to_pin_id:3
                }
        }
    ]
}
```

Де використовуються такі параметри:

id – ідентифікатор елементу;

name – назва елементу;

x_size – розмір по осі x;

y_size – розмір по осі y;

pins_top – верхні контакти;

pins_bottom – нижні контакти;

pins_left – ліві контакти;

pins_right – праві контакти;

pin_id – ідентифікатор контакту;

x_large – позиція контакту по осі X у великій сітці;

y_large – позиція контакту по осі У у великій сітці;

x_small – позиція контакту по осі X у малій сітці;

y_small – позиція контакту по осі У у малій сітці;

link_to_id – вказує на зв’язок з елементом;

link_to_pin_id – вказує на зв’язок контакту елемента.

При виводі результатів видається така сама структура опису, але додаються додаткові властивості позиціонування елементу:

pos_x – положення на осі X;

pos_y – положення на осі У;

angle – кут повороту елементу за часовою стрілкою.

6 Приклад

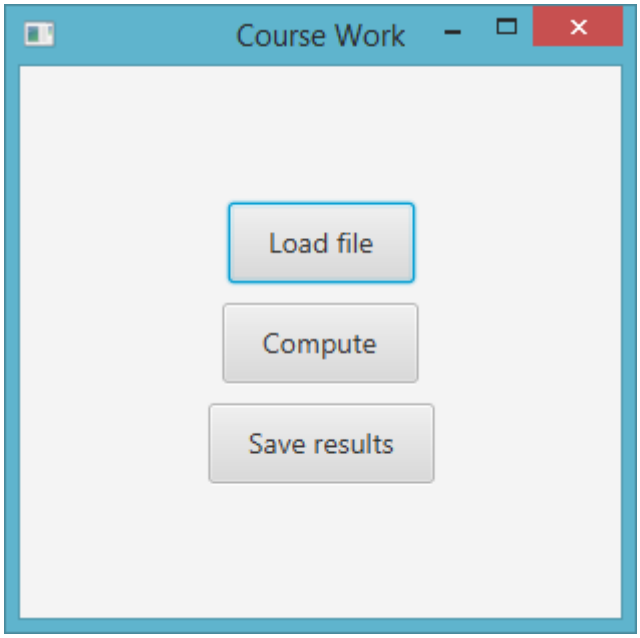

Рис. 6.1 Головне вікно програми

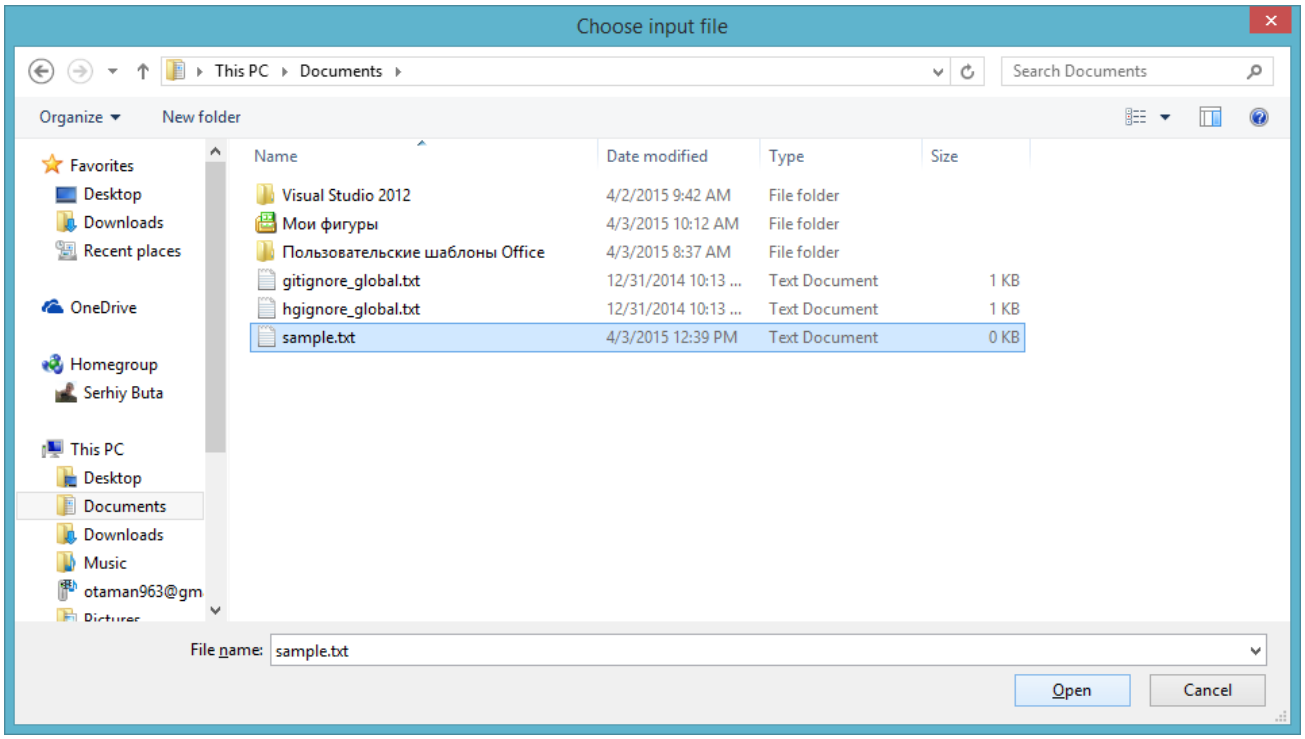

Рис. 6.2 Вибір файлу з вхідними елементами

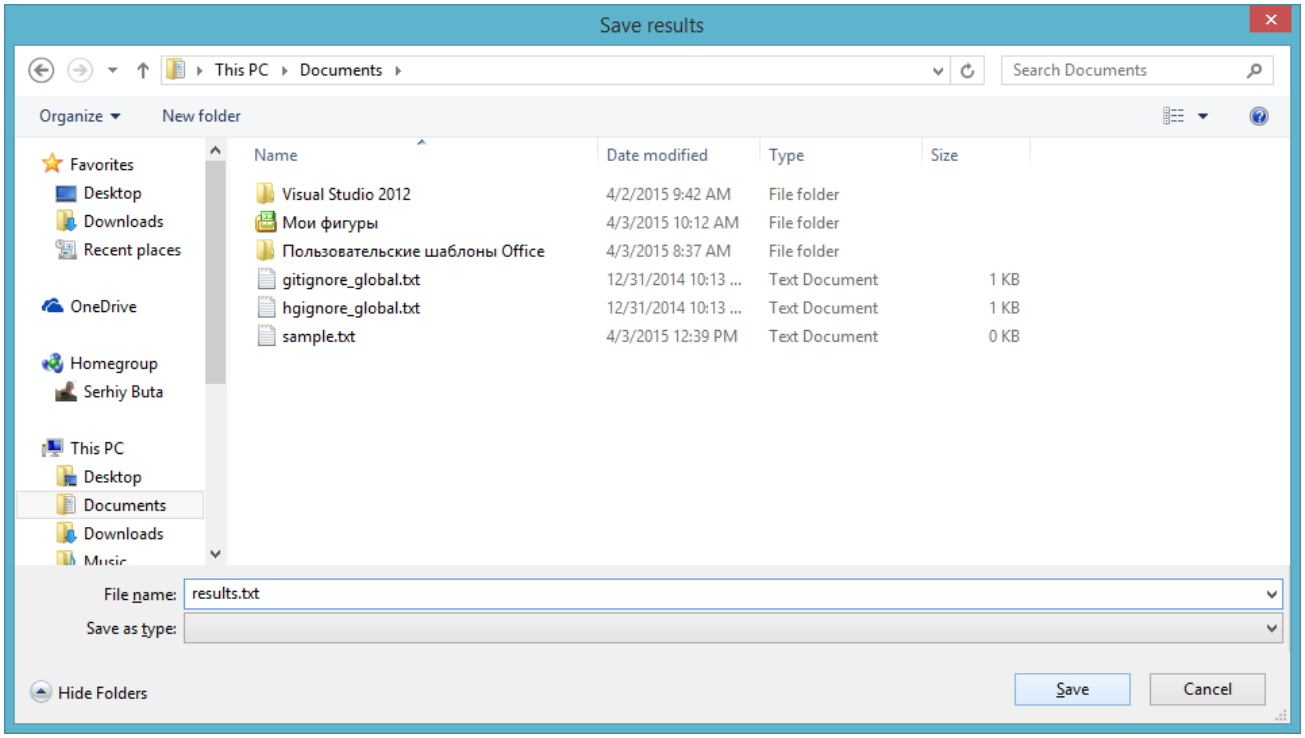

Рис. 6.3 Збереження результату у файл

Після збереження можна використовувати даний формат файлу в спеціалізованих програмах моделювання схем.

7 Висновок

В даному курсовому проекті було досліджено та вирішено задачу розміщення різногабаритних елементів на схемі. Для цього була розроблена програма, в якій реалізовано удосконалений алгоритм Гілок і меж.

Програма забезпечує:

1) введення та редагування вхідних даних;
2) автоматичне розміщення різногабаритних елементів на схемі;
3) запис результатів у файл.

Формати вводу-виводу зручні для розуміння людиною, а також легкі в процесі зчитування іншими спеціалізованими програмами.

8 Список літератури

1. Конспект лекций по курсу «Автоматизация проектирования компьютерных систем»
2. Курейчик В.М. Математическое обеспечение конструкторского и технологического проектирования с применением САПР. – М.: Радио и связь, 1990. - 352 с.
3. Kang S., Lebeltvici Y: CMOS digital integrated circuits - Analysis and design, Boston, McGrow-Hill, 1999
4. A.Miczo: Digital logic testing and simulation - John Wiley&Sons, 2003.- 673p
5. Яблонский С.В: Введение в дискретную математику - М.:Наука, 1979