PCB Auto-routing

Computer Science Coursework

Edwin Shepherd

Contents

[Analysis 1](#_Toc530678550)

[The Problem 1](#_Toc530678551)

[What is routing 1](#_Toc530678552)

[Different types of routing 2](#_Toc530678553)

[Users, Limitations and Requirements 3](#_Toc530678554)

[Interview 4](#_Toc530678555)

[Perquisites and missing requirements 4](#_Toc530678556)

[Objectives 4](#_Toc530678557)

[Prototyping 5](#_Toc530678558)

[Objectives 5](#_Toc530678559)

[Algorithm Research 5](#_Toc530678560)

[What I found out 5](#_Toc530678561)

[Bibliography 6](#_Toc530678562)

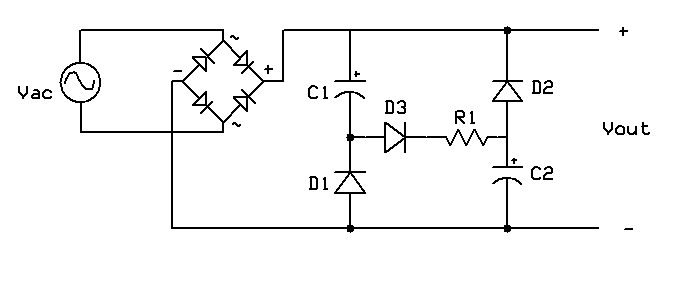
# Analysis

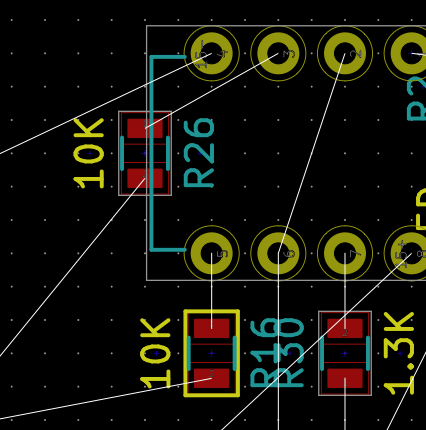
## The Problem

### What is routing

Electronics engineers develop circuits which are often printed on to a circuit board (PCB), keeping wiring tidy. However routing the wires is a very tedious task. Additionally the wiring must conform to rules or else malfunctions in the circuit may occur. For example, high current tracks must be of a set width, certain components cannot have tracks under them. This is takes more time and is difficult to ensure 100% accuracy.

To solve this problem PCB Routing Software is common place in PCB design tools. Its purpose is to automatically route all of the components and to perform “Design Rule Checking” DRC. PCB software usually has two modes one where the circuit is designed:

[](https://www.google.co.uk/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjSqtfQ-NPdAhVIWBoKHcJ_DAsQjRx6BAgBEAU&url=https://en.wikipedia.org/wiki/File:Valley-fill_circuit_schematic_1.png&psig=AOvVaw1u6-1MKZyxtGEkY9-Jxwfn&ust=1537888868345021) Figure 1 a schematic for a basic diode rectifier coupled with a valley-fill circuit - Image credit: WikiMedia available: <https://goo.gl/NJdaqG> , accessed 16:28 24/09/2018

[](https://www.google.co.uk/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwj-0Oz0rqveAhUOThoKHfDXAwgQjRx6BAgBEAU&url=https://electronics.stackexchange.com/questions/200093/what-does-yellow-border-around-component-in-kicad-pcbnew-mean&psig=AOvVaw12652bQlXOE6Df1Zygvx7C&ust=1540892724447123)Then another mode where the “nets”[[1]](#footnote-1) highlighted, these are the connections between components (in figure 2 they are blue) and then can be routed by the user (or by software). Finally it’s converted into a file that can be printed.

In this project I will solely focus on the challenges that are presented by the routing.

Figure 2 - Nets are shown in white straight lines. This screen shot is taken from KiCAD's PCBnew software. Image credit; Scanny avalible <https://goo.gl/muyghy> date accessed 29/10/2018

### Different types of routing

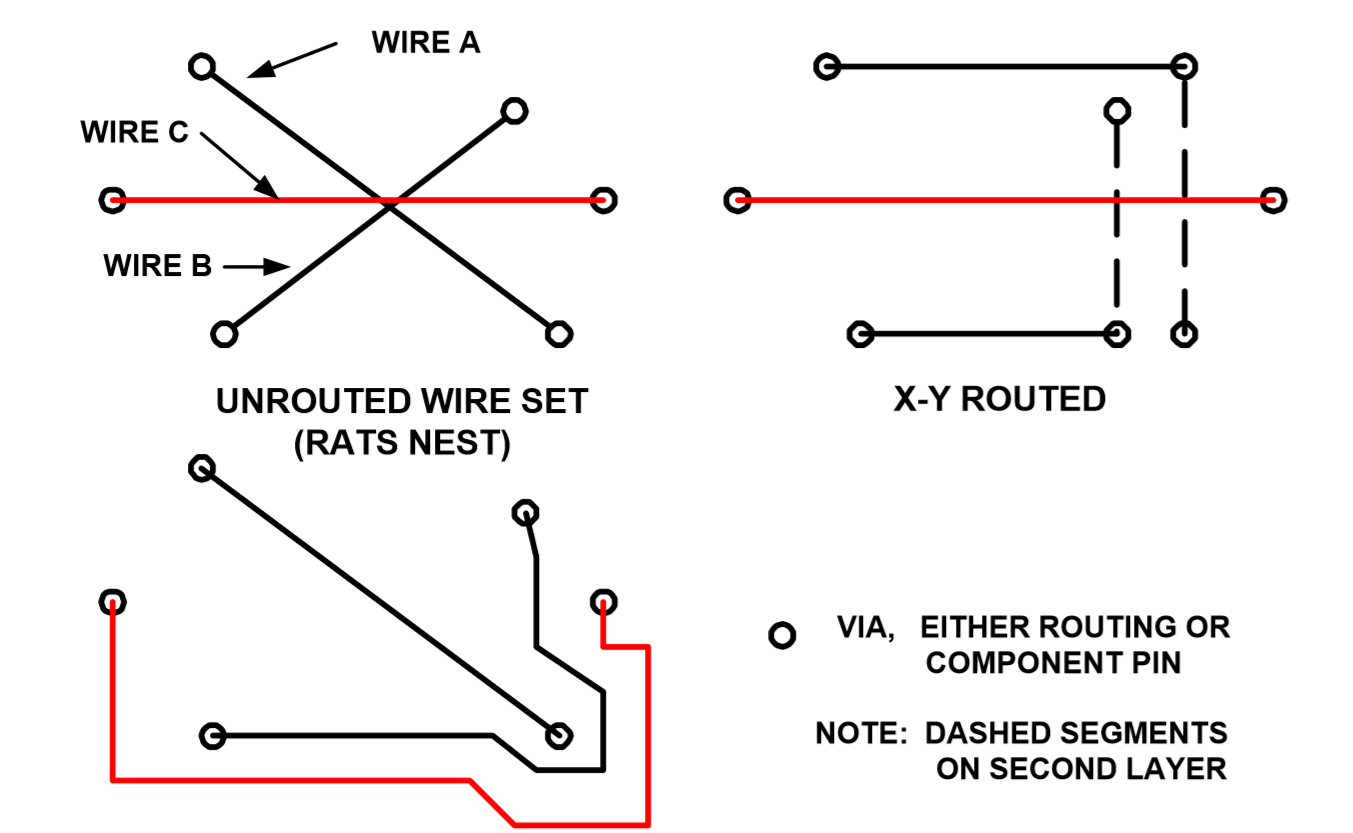
There are two main types of routing X-Y routing and Maze routing (LEE W. RITCHEY, 1999). 

Figure 3 - A diagram to show the different methods of routing. Image Credit LEE W. RITCHEY, 1999 Accessed: 21/11/2018 Available <http://www.speedingedge.com/PDF-Files/pcbrouters.pdf>

Initially, I will focus on maze routing as it the preferred method of routing by industry, due to the fact that X-Y routing is almost always impossible with less than 4 layers due to the component footprints disallowing straight lines to be drawn. However as the project progresses adding support for X-Y routing or even creating a hybrid router would have its uses.

### Users, Limitations and Requirements

Auto-routing is a tool to make electronic design easier for engineers to design a board. The user for the system, who I have reached out to, Stuart Jessop, should be able to use the final version to carry out the following tasks:

* Import a Net either by manually entering the coordinates of the pads or by parsing a file
* Set design constraints as detailed in the “interview”
* Route the board:
  + This will involve finding a route from two points
  + Moving on to the next two points
  + If routing impossible ripping up and retrying
* Export the routed file in a format that can be printed (SVG[[2]](#footnote-2))

Limitations:

* The software must run on a school computer. I have to demonstrate this to my user at school and I don’t have my own laptop so I either need to compile to a “.exe” or use a language with a run-time environment on the schools machines.
* My own coding abilities, I should do this project in a language that I am very familiar with such as node.js or python
  + Need to perform I/O in those languages which could difficult
* Time – This project needs to be done in the time allocated in the time allocated for the project in my computer science course.

Research

### Interview

After speaking with my user there are a range of areas that my project could support and they were ranked according to value. The list below is ranked.

1. Different track widths
2. Keep out areas
3. Thermal sinks[[3]](#footnote-3)
4. Isometric routing[[4]](#footnote-4)
5. Integration with EDA (Electronic Design Application) software.

### Perquisites and missing requirements

After my interview with my user I investigated other useful features which auto-routers have which are perquisites to being able to implement the features requested. The definition for what auto-routing is on Wikipedia is “*the routing step adds wires needed to properly connect the placed components while obeying all*[*design rules*](https://en.wikipedia.org/wiki/Design_rules)” (Wikimedia Foundation, 2018). This definition clearly points out a few areas which my project would need to support before the requirements suggested by my user:

1. Being able to find a route between two points
2. The route such does not interfere with other routes
3. Support for X-Y routing.
4. Ability to export files

Other features which auto routers sell themselves on is the ability to route across multiple layers (Altium LLC, 2018).

### Objectives

The requirements listed above are quite vague and need to be made more precise.

Being able to find a route between two points– The route should not waste board space, this means that it is likely to be the shortest possible route and so is “efficient”.

The route such does not interfere with other routes– Interfere here means cause a short or a break in another track.

The other requirements have been reworded, ordered and can be seen bellow:

1. Finding an efficient routes between points
2. Program can export result in SVG format
3. Ensuring that the routes do not violate any the DRC requirements listed in order bellow:
   1. Shorts and breaks do not occur
   2. No tracks cross “keep out area’s”
   3. Areas of board which are specified as thermal sinks meet the minimum size requirements
   4. Track widths are within the tolerances
   5. Routes set to be the same length are correct within tolerance
   6. Integration with EDA (Electronic Design Application) software, KiCAD[[5]](#footnote-5)
4. Support for X-Y routing

## Prototyping

### Objectives

From the prototyping period I needed answers to the following questions:

* What language should I use?
  + If it’s not Delphi how do I get it to work on the school PC’s
  + How do I compile/run code
* How hard is this problem:
  + What solution and algorithms can I implement?
  + Is it of the right difficultly level for my coursework?
* What should a basic outline of the classes for my project look like?

### Algorithm Research

Before I could begin prototyping I needed to do some research into algorithms for how this is done.

### What I found out

#### Language

I do not want to learn a new language for my project and of the languages which I know I feel most confident in python or JS, graphical output is hard in python and so I chose to use JS. The schools computers have node.js installed which means that I can write my code in a text editor and execute it using PowerShell.

#### Algorithms

My problem is effectively how to solve a maze and so all maze solving algorithms apply here. There are two main types of routing algorithms which apply “breadth-first” and “depth-first” (Zhou, 2000). Breadth-first algorithms are slower, require more resources but often find better routes as they will explore all possible routes over just prioritising one route. Depth-first algorithms attempt a single and likely path which is much faster (particularly for long tracks[[6]](#footnote-6)). For prototyping I wanted to show that it would be possible and so I chose a breadth first approach, however for my final project I may want to use a hybrid.

For my prototyping I chose to implement the “Lee-algorithm”.

# Bibliography

Altium LLC. (2018). *BEST AUTOROUTER*. Retrieved November 18, 2018, from Altium: https://www.altium.com/solution/best-autorouter

LEE W. RITCHEY, S. E. (1999). *PCB ROUTERS AND ROUTING METHODS.* Retrieved from Speeding Edge: http://www.speedingedge.com/PDF-Files/pcbrouters.pdf

Wikimedia Foundation. (2018, October 22). *Routing (electronic design automation)*. Retrieved November 18, 2018, from Wikipedia: https://en.wikipedia.org/wiki/Routing\_(electronic\_design\_automation)

Zhou, P. H. (2000). *EECS 357 Introduction to VLSI CAD.* Retrieved from eecs.northwestern.edu: http://users.eecs.northwestern.edu/~haizhou/357/lec6.pdf

1. A *Net* is a collection of wires that connects all of the points or pins in a single circuit [↑](#footnote-ref-1)
2. I have chosen to export in SVG form as it is easy to describe using a programing language which will make the export stage easier. [↑](#footnote-ref-2)
3. A Thermal Sink is an area of the PCB which is entirely copper and is used to sink heat from high current devices [↑](#footnote-ref-3)
4. Isometric routing is where multiple routes are routed to be approximately the same length to reduce skew in parallel data transition. [↑](#footnote-ref-4)
5. KiCAD was chosen as the target platform as it is open source and so there is better documentation on the file structure. [↑](#footnote-ref-5)
6. Most breadth first algorithms have a complexity similar to [↑](#footnote-ref-6)