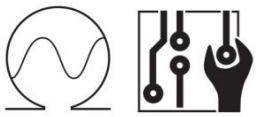
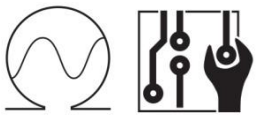


Technische Universität München
Fakultät für Elektro- und Informationstechnik
Lehrstuhl für Entwurfsautomatisierung

Area routing algorithm using flat spiral pattern delay lines to counter the speed-up effect present in routing using dense meander segment delay lines

Master Thesis

Jaroslaw Konrad Bejm



Technische Universität München
Fakultät für Elektro- und Informationstechnik
Lehrstuhl für Entwurfsautomatisierung

Area routing algorithm using flat spiral pattern delay lines to counter the speed-up effect present in routing using dense meander segment delay lines

Master Thesis

Jaroslaw Konrad Bejm

Supervisor :	Dr.-Ing. Tsun-Ming Tseng
Professor:	Prof. Dr.-Ing. Ulf Schlichtmann
Topic issue date :	08.05.2017
Submission date:	29.03.2018

Jaroslaw Konrad Bejm
Valpichlerstraße 55
80686 München

Contents

Illustrations

1. Introduction

With technological advancement, growth of scale and complexity of electronic systems the size of *printed circuit board*s (PCBs) increases while the dimensions of electronic components decrease. Hence, manual design, such as manual placement and routing of components, becomes in most cases infeasible and in other cases a time-consuming and error-prone task. One resulting major challenge, thus plaguing PCB design, are timing problems. In this category, matching delays on individual signal transmission wires of a bus is considered to be a mainstream one. Therefore, it will also be the main problem challenged by this thesis.

Luckily, many design automation methods already exist and steadily advance to tackle most problems concerning PCB design. When talking about signal delays on wires, the easiest and most intuitive way to make sure that two signals on two different wires arrive at their destination at the same time is to keep the wires the same length. There exist two main practices which are used when routing wires: escape routing and area routing. *Escape routing* is the practice of routing wires from their respective positions in a pin array inside a component to the boundary of mentioned component. *Area routing*, on the other hand, is the practice of routing wires in order to connect component boundaries and simultaneously avoid any obstacles, routing congestion and in case of routing buses also trying to maintain their planar topology (i.e. all wires in a bus are expected to be routed within specified length bounds). That is why area routing is often referred to as *length-matching routing*.

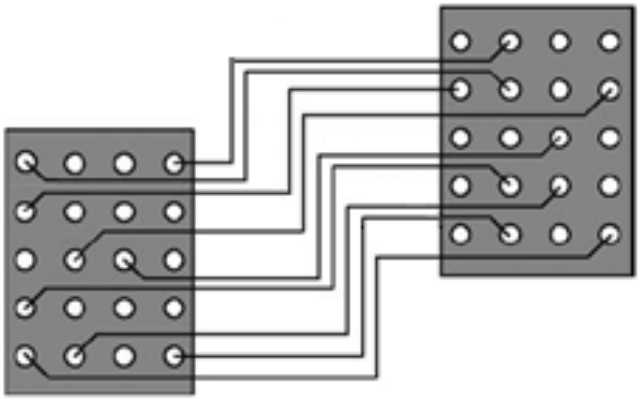


Figure 1.1. : Escape routing (inside the gray components) and area routing (between the two components) with no length-matching

More and more research is done on automated PCB routers which separately or in combination handle escape routing and area routing of wires with various predefined constraints. In [1]-[6] different PCB routers were introduced which use different or similar methods to tackle the problem of bus signals arriving with different delays on different wires. A commonly used and very inexpensive idea to solve that problem is to implement *delay line segments*. These are conveniently routed transmission wire segments with the sole purpose of introducing delays by increasing overall wire length. The part of the wire which makes up a delay line segment is the *delay line*. If two wires of a bus are of different lengths the signals travelling on them will practically always arrive at their destination at different times. By adding a delay line segment to the shorter wire, which increases its length to match the longer wire, the problem can be alleviated. This practice is called *delay-matching*.

So delay-matching and length-matching are closely related with one referring to signal timing while the other referring to wire length.

The most widespread delay line segment is a serpentine pattern. Such a pattern, called a *meander segment*, is shown in Figure 1.2. inside the dotted box. However, usually the delay line segment consists of multiple concatenated meander segments.

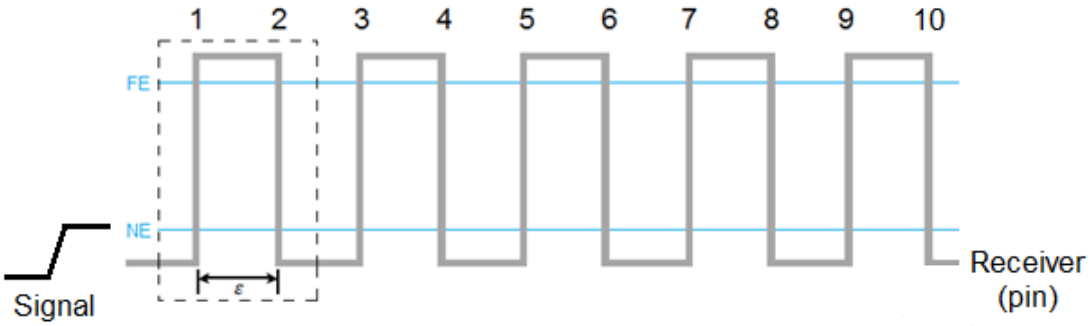


Figure 1.2. : Meander segment delay line

Although the meander segment delay line has gained wide acceptance due to its high routing density and relatively easy modeling, using it provides not exclusively advantages. One critical disadvantage is the speed-up effect which causes a signal passing through the meander segment to arrive prior to its expected arrival time. That means a situation may occur where wires are length-matched but due to the topology of the meander segment itself not necessarily delay-matched. The speed-up effect only becomes more and more severe with increasing wiring density and clock speeds. A more thorough explanation of the speed-up phenomenon will be presented later on.

In order to tackle this disadvantage of the meander segment delay line, other more complex topologies for delay lines were introduced and analyzed, for instance, the concentric delay line [8] or the flat spiral delay line [7]. The following thesis's focus will be on implementing a PCB router using the flat spiral delay line.

A flat spiral delay line has a topology which allows it to be printed on a single two dimensional circuit board layer and an example is shown in Figure 1.3.

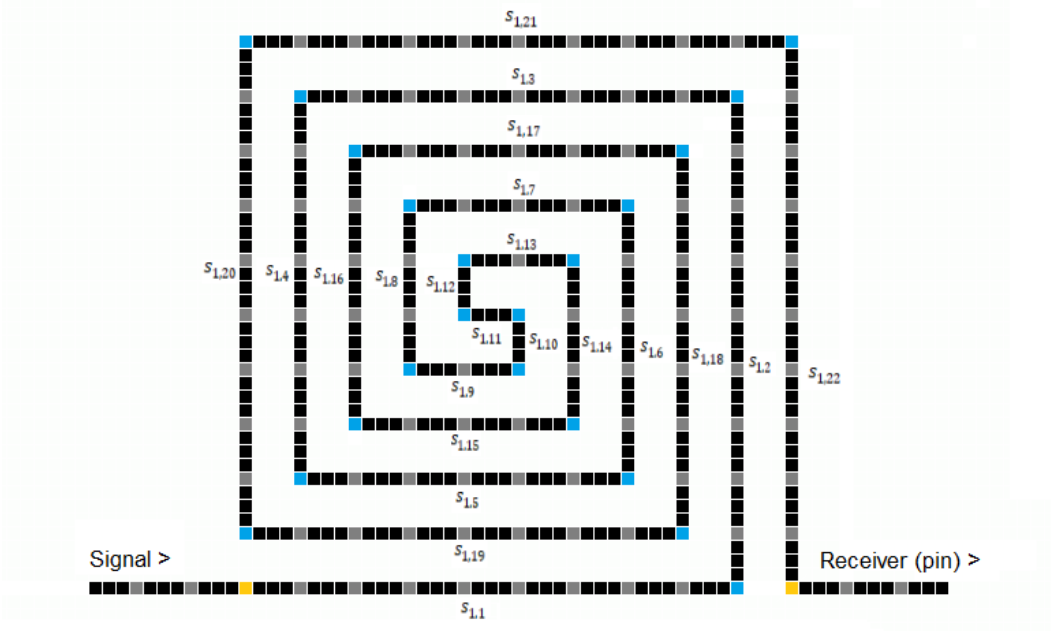


Figure 1.3. : Flat spiral delay line (22 segments)

The thesis consists of 7 chapters. In the next, second, chapter some important definitions necessary for the understanding of this thesis as well as the mathematical models for a meander segment delay line and a flat spiral delay line will be introduced. In chapter 3 the speed-up effect will be looked at in more detail and thus the motivation for choosing the flat spiral delay line over the meander segment delay line will be discussed. Then, in chapter 4 constraints for the PCB router described in this thesis will be sought out by means of a problem formulation. In chapter 5 the problem will be solved by use of a newly developed algorithm and in chapter 6 the challenges of implementing the algorithm in software will be addressed. Finally, in chapter 7 possible improvements to the algorithm and the program which might be feasible but go beyond the scope of this thesis as well as possibilities for future research which appeared upon the writing of this thesis will be presented.

2. Definitions and mathematical models

2.1. Definitions (ε, wire segments, free space)

In order to be able to describe a meander segment delay line and a flat spiral delay line mathematically, first a basic unit which will allow for a comparison between the two is required. This unit shall be denoted by ε and be dependent on the manufacturing capabilities of modern PCB makers, namely, wiring pitch (i.e. the etch distance when machine wiring is done) and the wire width. This will allow all following equations to be independent of manufacturing technologies and also provide a result which has practical use (in contrast to models where wire widths are assumed to be 0).

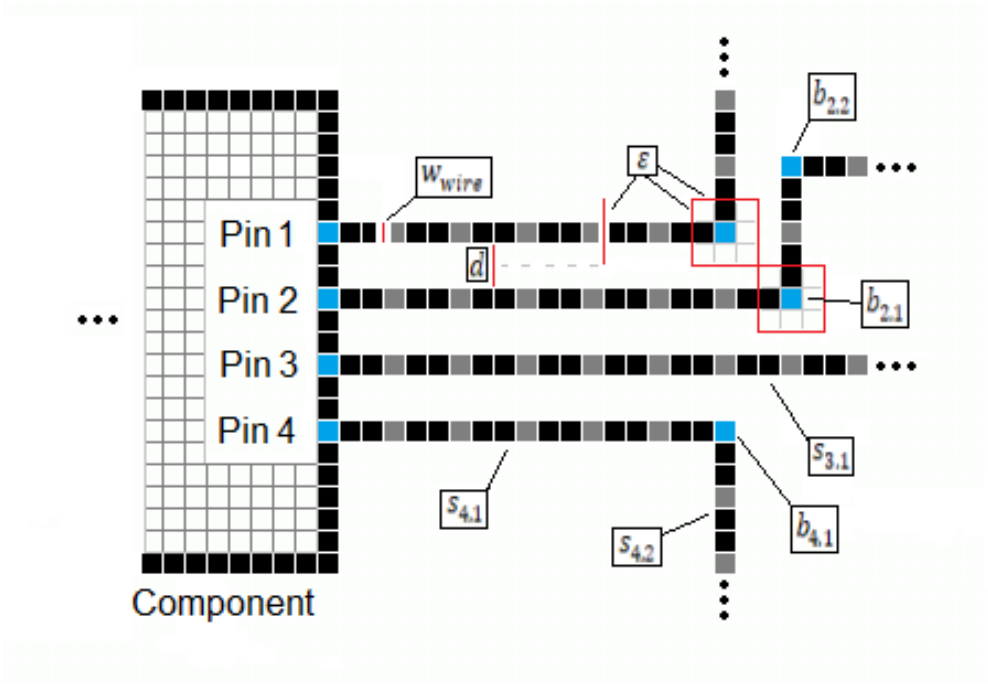


Figure 2.1. : Wiring model definitions visualization

In Figure 2.1. four wires leaving four neighboring pins of a component are shown. The wire widths are all equal and equal to w_{wire}. The wire pitch, so the minimum distance between two neighboring parallel wires, is d and depending on the manufacturing process: $d \geq w_{wire}$ (in Figure 2.1. $d = 2w_{wire}$). Hence,

$$\begin{aligned} \varepsilon &= \tfrac{1}{2}d + w_{wire} + \tfrac{1}{2}d \\ \varepsilon &= w_{wire} + d \geq 2w_{wire} \end{aligned} \quad (1)$$

Furthermore, it is also assumed the minimum length a wire can have is equal to ε and so it shall become a unit of wire length i.e. the length of any wire is a multiple of ε (in Figure 2.1. $\varepsilon = 3w_{wire}$). Although, in truth during a PCB manufacturing process the minimum length a wire might have is w_{wire} (i.e. a dot of wiring material $w_{wire} \times w_{wire}$), changing it to ε for the purpose of this thesis and the designed PCB router will not have a negative impact on the accuracy of the results. That is so because ultimately every wire has specified lower and upper bounds for its length requirement and their range is never smaller than ε. Additionally, choosing ε as a unit of wire length allows for easier modeling of a wire bending. The routing becomes a problem of taking steps in a gridded 2D space.

Every wire shall be named after the pin number it originates from i.e. wire $i = 1$ connects pin 1 of component 1 to a pin of component 2. A wire can be further subdivided into segments $s_{i,j}$, where i is the wire index and j is the consecutive index of the segments of the ith wire. Every segment can be defined by its length $|s_{i,j}|$, which is a multiple of ε, as well as a starting and an end point which are either a pin or a bending point $b_{i,k}$, where i is the wire index and k is the consecutive index of the bending points of the ith wire. Every pin and bending point are defined by their coordinates. A bending point is the point where a wire

changes direction by 90°. Henceforth, every pin and bending point, which are not part of delay line segments, will be colored blue in all figures.

Delay line segments which are used to increase the overall length of a wire can be inscribed inside a rectangular *free space* of width W and height H. A free space is therefore the space above or below a wire segment $s_{i,j}$ which is free of any other wires passing through. This means a free space can have a maximum length equal to the length of the segment and a maximum height which is only constrained by the segment's opposite routing area boundary and whether or not a wire is passing in-between. Free spaces, especially of neighboring wire segments which are perpendicular to one another, can overlap but once anyone of them is filled with a delay line pattern all others have to be adjusted. Additionally, if a segment owns a big enough free space, it can share it with one or more parallel neighboring wire segments, creating a wire group. However, in general for this case the condition will apply where all the delay lines have to stay parallel throughout the entire delay line pattern (i.e. distance between wires kept constant). Finally, if it is not possible for any not yet routed wires to access a free space that space becomes *wasted space*. It appears when wires are not routed in a dense fashion as close to each other as possible.

A segment for which a free space exists shall be called an *available (wire) segment*, while a segment for which no free space exists (e.g. wire 2 segment $s_{2,1}$ in Figure 2.1) shall be called *unavailable (wire) segment*. In general every segment which starts or ends at a pin will be an unavailable segment, otherwise filling the free space would hinder routing to and from neighboring pins. However, sometimes this does not apply (i.e. pin at the corner of a component) or can be avoided thanks to smart choices. Also after choosing a free space, where a delay line should be inscribed, the segment which owns this free space shall be

called *original (wire) section* and its length will be used to determine the increase in overall wire length after the addition of a delay line pattern. A wire can have more than one original section.

2.2. Mathematical model of a meander segment delay line

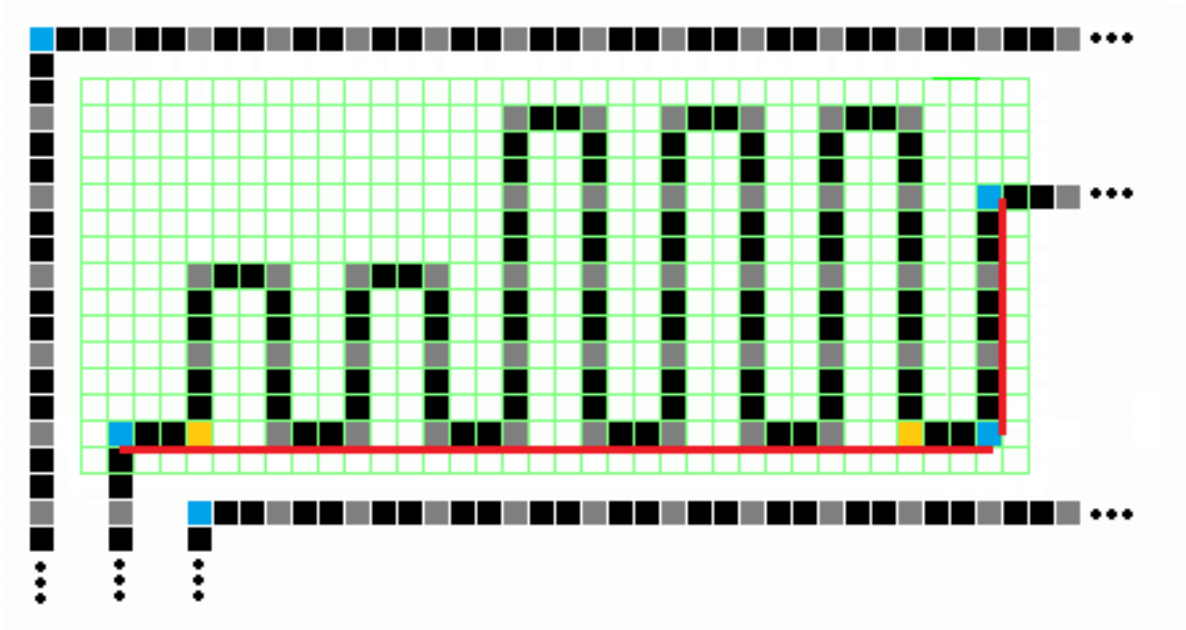


Figure 2.2. : Meander segment delay line in free space (green grid)

The modeling of the serpentine pattern of a meander segment delay line in a given free space is relatively easy under the assumption only a single wire has to occupy the space. In Figure 2.2. two perpendicular segments (red) of the middle out of three wires share a free space (green grid) for the growth of a meander segment delay line. The free space has a width W and a height H which are both multiples of ε. A single meander segment has width w_{msg} and height h_{msg} which are also both multiples of ε. The number of meander segments which can be grown inside the free space can be found by calculating the floor function: $\lfloor W/w_{msg} \rfloor$. The height of every meander segment can thereafter be adjusted separately at will, as long as it stays within the boundary of the free space. It is important to note that the length to height

ratios of the free space do not have any influence on the meander segments within. The only existing constraints are:

$$\text{min. } W \geq \text{min. } w_{msg} \geq 2\varepsilon$$

$$\text{min. } H \geq \text{min. } h_{msg} \geq 2\varepsilon$$

This and the fact that every single meander segment is described by two independent variables (w_{msg}, h_{msg}) offers great flexibility but can become a drawback with too great of an amount of variables even for computers, i.e. efficiency problems in program execution of a PCB router.

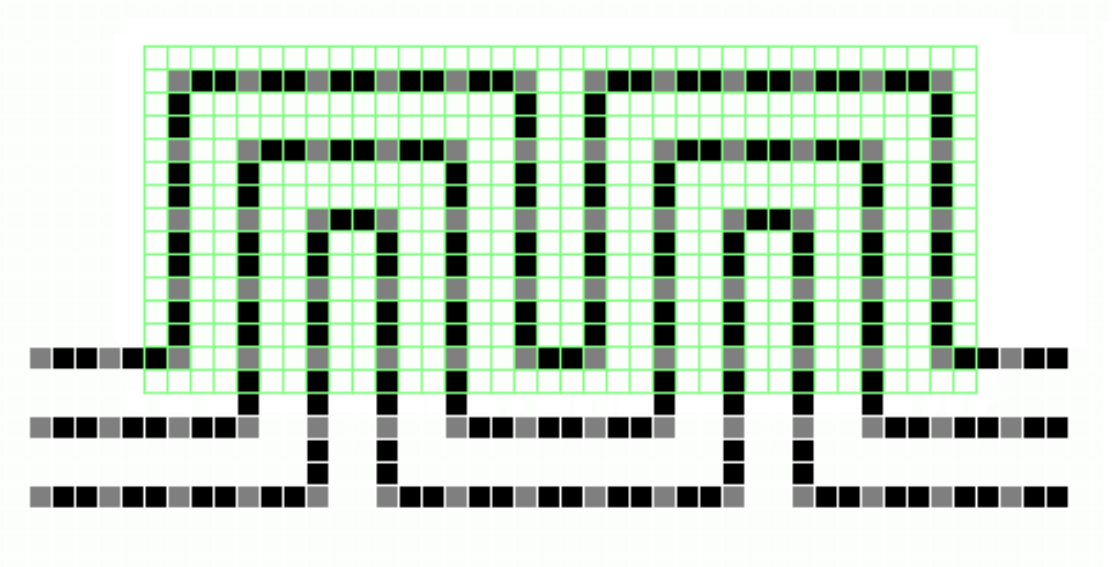


Figure 2.3.: Meander segment wire group in free space (green)

In addition, depending on the PCB router, multiple delay lines can exist in a given free space. An examples is shown in Figure 2.3. and an associated comprehensive model was introduced in [9]. It handles the number of meander segments in free spaces and the dependency between multiple wires but will not be considered further in the scope of this thesis.

2.3. Mathematical model of a flat spiral delay line (corner cell)

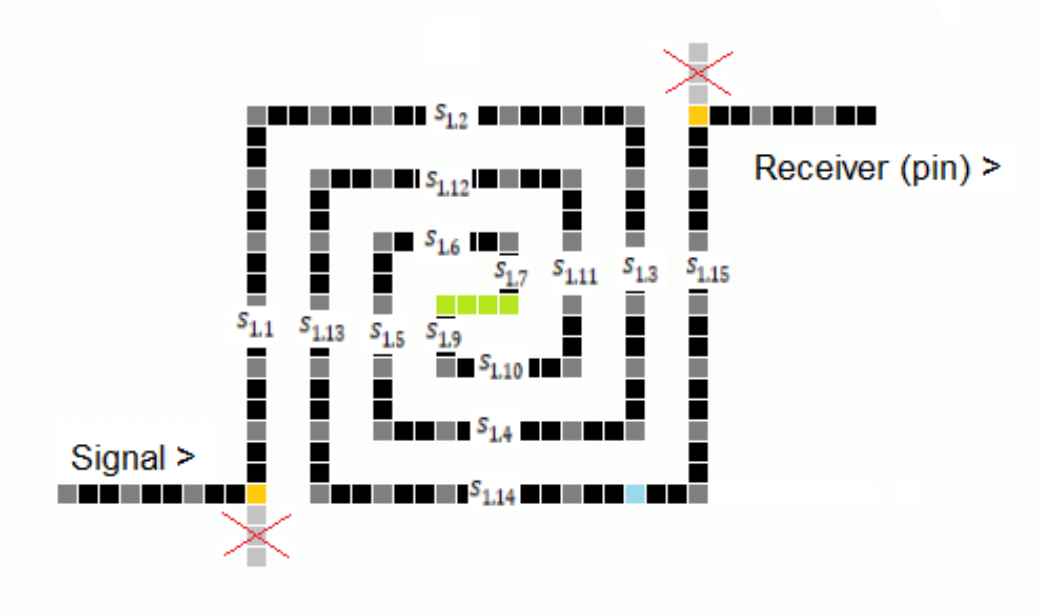


Figure 2.4.: Flat spiral delay line (15 segments)

Figure 2.4. depicts an example flat spiral delay line which consists of 15 wire segments ($s_{1,1}$ to $s_{1,15}$). It fills a free space of width W and height H where both are multiples of ε. The delay line begins at the orange point (x_1, y_1) and ends at the orange point (x_2, y_2). This flat spiral can be described with a formula which offers the spiral's wire length as the result.

$$L_{SPIRAL} = (p+q) + 2\sum_{i=1}^{n}\Big(p + \varepsilon\big(1 + 2(i-1)\big)\Big) + 2\sum_{i=1}^{n}\Big(q + \varepsilon\big(1 + 2(i-1)\big)\Big) - 2\varepsilon \qquad (2)$$

In order to explain the formula it can be separated in four parts:

1) $(p+q) + \cdots$

p is the length of the innermost line segment $s_{1,8}$ (i.e. $p = |s_{1,8}|$) in the center of the spiral (in Figure 2.4. colored green) and has a minimum value ε (i.e. $p_{min} \geq \varepsilon$). Except for this *unique segment* every other segment has a *sibling (wire) segment* of same length:

$s_{1,1} = s_{1,15}$; $s_{1,4} = s_{1,12}$; $s_{1,7} = s_{1,9}$

$s_{1,2} = s_{1,14}$; $s_{1,5} = s_{1,11}$;

$s_{1,3} = s_{1,13}$; $s_{1,6} = s_{1,10}$;

q equals the length of the sum of segments $s_{1,7}$ and $s_{1,9}$ (i.e. $q = |s_{1,7}| + |s_{1,9}|$) and has a minimum value 2ε (i.e. $q_{min} \geq 2\varepsilon$).

2) $\cdots + 2\sum_{i=1}^{n}\Big(p + \varepsilon\big(1 + 2(i-1)\big)\Big) + \cdots$

The first term which includes the sigma symbol is the sum of all the line segment lengths parallel to the unique segment. Each of these segments can be considered to have the length of this segment (p) plus an odd number of ε. Considering the spiral in Figure 2.4. the consecutive terms of the sigma expression would be:

$i = 1 \rightarrow p + \varepsilon$;

$i = 2 \rightarrow p + 3\varepsilon$;

$i = 3 \rightarrow p + 5\varepsilon$.

n which is the break condition for the sigma can be described as the number of line segments parallel to the unique segment divided by 2. This is the number of sibling segment pairs or line segments of different length parallel to the segment with length p. For the spiral in Figure 2.4. n equals 3. Finally, in order to not miss the length of the sibling segments from the overall length of the delay line the whole sigma term needs to be multiplied by 2.

3) $\cdots + 2\sum_{i=1}^{n}\Big(q + \varepsilon\big(1 + 2(i-1)\big)\Big) + \cdots$

The second term which includes the sigma symbol is the sum of all the line segment lengths perpendicular to the unique segment except the two segments making up q (i.e. $s_{1,7}$ & $s_{1,9}$).

This means it is the sum of all the line segment lengths parallel to the two segments $s_{1,7}$ and $s_{1,9}$. Each of these segments can be considered to have the length q plus an odd number of ε. Comparing this term with the previous one it is obvious that they are the same except for p being substituted by q. This means, that under the assumption that the form of the spiral does not change, the same conclusions can be drawn here as in the previous point (explanation concerning form will follow).

4) $\cdots - 2\varepsilon$

This last part of the equation allows for the spiral to be inscribed inside a rectangular with as little wasted area as possible and also allows for an easier connection to the original section. These two ε are subtracted from the outermost sibling segment pair (i.e. $s_{1,1}$ and $s_{1,15}$). In Figure 2.4. these are the two short wire parts crossed out red.

Taking all of the above into consideration and assuming

$$|s_{1,8}| = p = p_{min} = \varepsilon \qquad \text{and}$$

$$|s_{1,7}| + |s_{1,9}| = q = q_{min} = 2\varepsilon$$

the total length of the flat spiral delay line in Figure 2.4. would be:

$$L_{SPIRAL} = (\varepsilon + 2\varepsilon) + 2\sum_{i=1}^{3}\Big(\varepsilon + \varepsilon\big(1 + 2(i-1)\big)\Big) + 2\sum_{i=1}^{3}\Big(2\varepsilon + \varepsilon\big(1 + 2(i-1)\big)\Big) - 2\varepsilon$$

$$L_{SPIRAL} = \varepsilon + 2(2\varepsilon + 4\varepsilon + 6\varepsilon) + 2(3\varepsilon + 5\varepsilon + 7\varepsilon)$$

$$L_{SPIRAL} = 55\varepsilon$$

2.4. Mathematical model of a flat spiral delay line (straight cell)

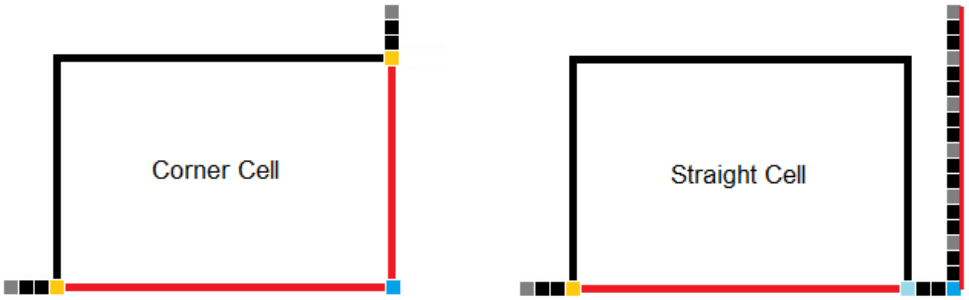


Figure 2.5. : Corner cell and straight cell (red – original sections; orange/light blue – spiral start/end points; dark blue – bending points)

In the previous section a flat spiral delay line was mathematically described, however, all of the deduced conclusions only apply in case of a corner cell, where the start and end points are located as shown on the left in Figure 2.5. (also Figure 2.4.; marked with orange points). Because only horizontal and vertical (no diagonal) routing should be considered in order to ease the problem complexity, it is impossible to inscribe a spiral of that form into the free space belonging to a single available wire segment. This changes if there are two available wire segments perpendicular to each other with overlapping free spaces (e.g. segments colored red in Figure 2.5 on the left). However, only focusing on these cases is too limiting for a modern PCB router, hence, new considerations had to be made.

The solution is to inscribe the flat spiral pattern with an altered form inside a straight cell which only requires the free space of one available wire segment, as shown in Figure 2.5. on the right in red. The flat spiral delay line will have a form as shown in Figure 2.4. with the starting point at the orange point (x_1, y_1) and ending point at the light blue point (x_3, y_1). Furthermore, this solutions can be extended to cover even the previous case of overlapping

free spaces with minimal waste area compared to a corner cell solution (i.e. also shown in Figure 2.5.). The altered formula looks as follows:

$$L_{SPIRAL} = (p+q) + 2\sum_{i=1}^{n}\Big(p + \varepsilon\big(1+2(i-1)\big)\Big) + 2\sum_{i=1}^{n}\Big(q + \varepsilon\big(1+2(i-1)\big)\Big) - 2\varepsilon - \\ - \Big(q + \varepsilon\big(1+2(n-1)\big)\Big) \quad (3)$$

What changed from formula (2) is the addition of the last term. This term subtracts the length of one of the segments in the longest sibling segments pair, either $s_{1,1}$ or $s_{1,15}$, from the overall length of the delay line. However, it is necessary to realize that due to the topology of the spiral in Figure 2.4., that is - the longest sibling segments are parallel to the two sibling segments making up q (i.e. $s_{1,7}$ and $s_{1,9}$), the term uses the q variable. If the topology was changed, for example the spiral consisted of only 13 segments, started with segment $s_{1,2}$ and ended with $s_{1,14}$, then the longest sibling segments would be parallel to the unique segment (i.e. green segment $s_{1,8}$) and the term would use the p variable instead.

In order to decide whether to choose p or q, first one must consider one of the aims of using the flat spiral delay line pattern, namely, high routing density, i.e. occupying as little free space as possible. Similar as in the case of meander segment delay lines where this was achieved by keeping the widths of meander segments constantly minimal and only adjusting their heights, it is possible to set one variable of the spiral to have constantly minimal value while only adjusting the other. What needs to be kept constant and at its minimal value is the q variable and only p should be altered. The reason is that altering p does not generate wasted space inside the spiral while changing q does. Increasing q beyond its minimum

value (i.e. 2ε) will create wasted space between the innermost sibling segment pair parallel to the unique segment and the unique segment itself.

When drawing a spiral in the free space it is necessary to consider cases where $h_{spiral} < H \to \infty$ and $w_{spiral} \leq W$. Filling such a free space with a spiral, requires it to first grow until $w_{spiral} = W$ is reached before the spiral area grows in height. To accomplish that the unique segment of initial length $p = p_{min}$ needs to be aligned parallel to the original section (e.g. in Figure 1.3. $s_{1,11}$ was parallel to the section between two orange points). Next, by increasing p by ε, w_{spiral} will also increase by ε. The overall length of the delay line will be increased by $2n\varepsilon$. When $w_{spiral} = W$, p will reach a maximum value p_{max} and cannot be increased anymore. To continue increasing overall wire length the height h_{spiral} needs to be increased (so long as it possible i.e. $h_{spiral} < H$). This can be done by increasing the number of sibling segments (i.e. increasing n). However, doing that will decrease the previous p_{max} by 2ε. After repeated cycles of setting $p = p_{min}$, increasing p in steps of ε until $p = p_{max}$ and setting new h_{spiral} (i.e. decreasing p_{max}), at one point p_{max} will be equal to p_{min} or will not be able to decrease anymore and still be bigger than p_{min}. The spiral will occupy an almost square shaped free space and it will not be possible to further increase p while the unique segment is parallel to the original section. Therefore, if the overall wire length is still not matched, it is necessary to change the orientation of the unique segment so it is perpendicular to the original section. Afterwards p can be increased again in steps of ε, until $h_{spiral} = H$, while overall wire length will increase by $2n\varepsilon$ in every step. The number of sibling segments (n) as well as the width of the spiral ($w_{spiral} = W$) will both remain constant.

Looking back at all these observations, it seems reasonable to define a new variable which will inform the PCB router whether the unique segment is parallel or perpendicular to the original section. This variable shall be z and the following will apply:

$$z = 0 \Leftrightarrow \text{unique segment parallel to original section} \quad (4)$$

$$z = 1 \Leftrightarrow \text{unique segment perpendicular to original section} \quad (5)$$

In case of straight cells this has significant importance. There are two possibilities to begin inscribing a spiral into such a cell. First, is to treat the starting point (x_1, y_1) as a bending point and have the wire segment $s_{1,1}$ go perpendicular to the original section. This was already shown in Figure 2.4. The second possibility is to have $s_{1,1}$ go parallel to the original section as was shown in Figure 1.3. At first glance these two seem different: signals travel in opposite directions through the spiral. However, in practice they can be treated the same because one is just the mirror image of the other. They are occupying the same area of the available free space and the change in signal direction does not affect its crosstalk characteristics. This will be further explained in the next chapter.

2.5. Mathematical model of a flat spiral delay line (generalized)

In order to generalize the formula and cover both straight and corner cells as well as include the form of the spiral, other than the z variable, an additional variable k should be defined. This variable will also be binary and will decide whether there is a corner or straight cell:

$$k = 0 \Leftrightarrow \text{corner cell} \quad (6)$$

$$k = 1 \Leftrightarrow \text{straight cell} \quad (7)$$

The final, generalized form of formula (3) looks as follows:

$$L_{SPIRAL} = (p+q) + 2\sum_{i=1}^{n}\Big(p + \varepsilon\big(1+2(i-1)\big)\Big) + 2\sum_{i=1}^{n}\Big(q + \varepsilon\big(1+2(i-1)\big)\Big) - 2\varepsilon - \\ - \Big[q\left(\frac{p}{q}\right)^{z} + \varepsilon\big(1+2(n-1)\big)\Big]k \tag{8}$$

The last term can be switched on or off depending on variable k and which longest sibling segment has to be subtracted is decided by variable z. In order to make it easier for the PCB router to determine how to inscribe the spiral in the free space and which of the sibling segments from the longest pair to remove in case of a straight cell, it was determined to set segment $s_{1,1}$ always perpendicular to the original section. Hence, it becomes a constant and following conclusions can be made:

$$z = 0 \Leftrightarrow q\left(\frac{p}{q}\right)^{z} = q \Leftrightarrow \text{unique segment perpendicular to segment } s_{1,1} \tag{9}$$

$$z = 1 \Leftrightarrow q\left(\frac{p}{q}\right)^{z} = p \Leftrightarrow \text{unique segment parallel to segment } s_{1,1} \tag{10}$$

Finally, a formula for n was deduced:

$$n = \left\lfloor \frac{w_{spiral} - q\left(\frac{p}{q}\right)^{z} + k\varepsilon}{2\varepsilon} \right\rfloor, \tag{11}$$

as well as a formula for the increase in overall wire length (L_{WIRE}) due to addition of a delay line (i.e. comparing original segment length to result of formula (8)):

$L_{WIRE} = L_{SPIRAL}$ – Manhattan dist. between starting point (x_1, y_1) and end point (x_2, y_2)

$$L_{WIRE} = L_{SPIRAL} - (|x1 - x2| + |y1 - y2|) \tag{12}$$

3. Motivation

3.1. Meander segment speed-up effect explanation

After many years of being the most widely adopted delay compensation method in the IC design and engineering community, it can be said that high density meander segment delay lines have also become the most thoroughly researched delay line patterns. There exist very comprehensive studies of their delay characteristics as well as many PCB router designs which help automate the routing process ([8]-[12]).

Paper [10] explains how an additive accumulation of crosstalk noise between consecutive meander segments within a meander segment patterned wire will occur. It terms the accumulation as being *synchronous* and eventually leading to the crosstalk voltage surpassing the threshold voltage of logic switching causing a *speed-up effect*. The result is a mismatch of signal arrival times, even though wires are physically the same length (i.e. length-matched but not delay-matched). In [11], a method of moments (MoM) was used on meander segment wire patterns to present a full-wave model which approximates the delays of individual wires and includes mode coupling and corner effects. In contrast, in [8] the authors are able to predict the delays qualitatively using a finite-difference timing domain (FDTD) method. Finally, in [12], the wire delay is being controlled using a linear model which adjusts the number of meander segments on a fixed-shape wire and in [9] further improvements are made with a post-processing method which reduces crosstalk by making a more even distribution of meander segments on individual wires as well as the board while also

adjusting their parameters, such as their widths and heights, individually. In Figure 3.1. below, the unprocessed output of a PCB router is shown in (a) and a refined routing diagram in (b).

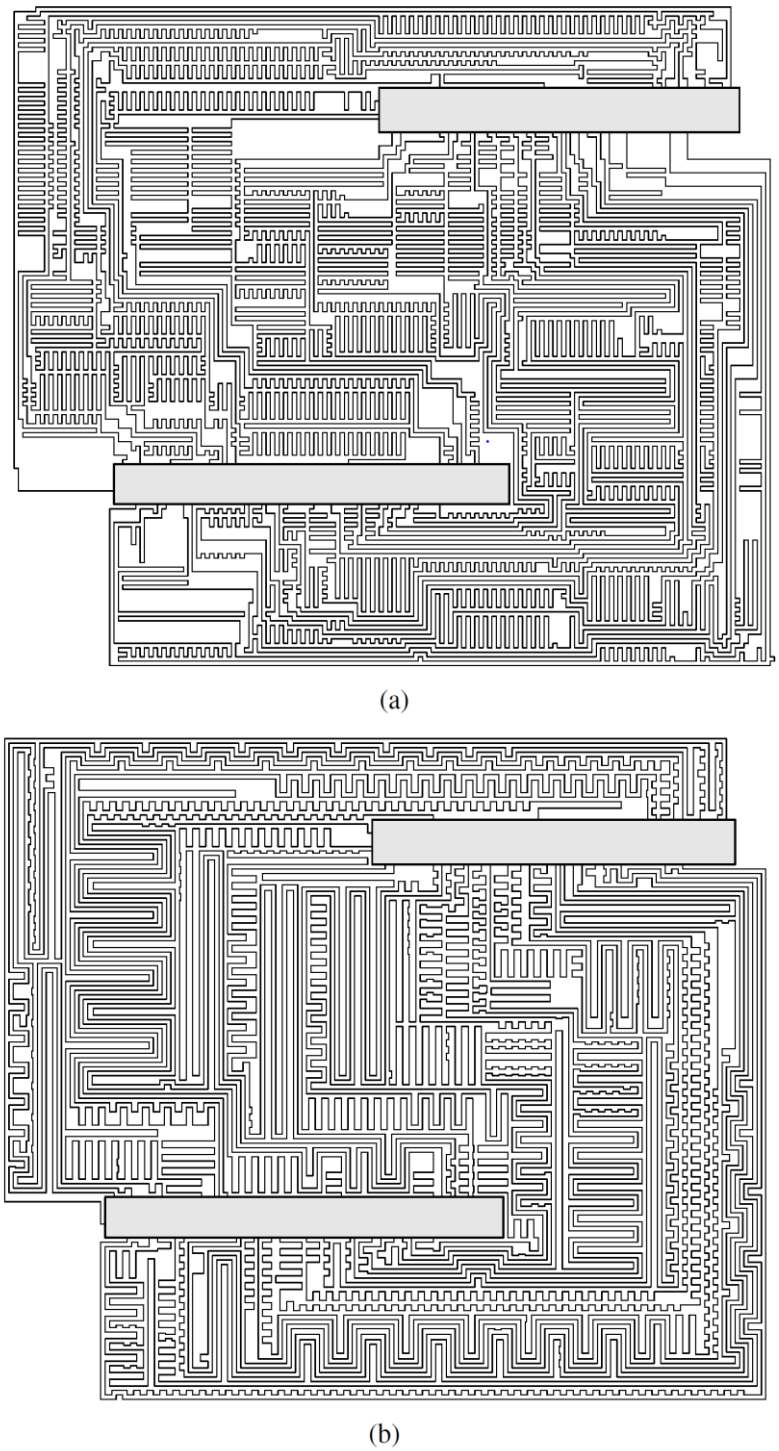


Figure 3.1. : Meander segment based routing (comparison) (a) Original routing (b) Refined routing (Source: [9] Fig. 9)

Trying to explain the speed-up effect in still more detail, let's first look again at Figure 1.2. which depicts a pattern with five meander segments, each consisting of 2 vertical wire segments (segments 1 to 10). Assuming that at time $t = 0$, the main signal switches at the bottom of segment 1, called *near-end* (NE) when comparing to standard transmission lines, it will propagate and reach the top of segment 1, called *far-end* (FE) when comparing to standard transmission lines, in time t_h. However, the switching of the main signal at $t = 0$ simultaneously stimulates crosstalk voltage at the NEs of all the other wire segments. Because in the scope of this thesis only forward propagating crosstalk is of interest (i.e. crosstalk propagating towards the receiver), only the induced crosstalk noise in segments 2, 4, 6, 8 and 10 has to be considered at $t = 0$. On the other hand at time $t = t_h$, the same signal will induce a crosstalk voltage on the FEs of all other segments but only the segments 3, 5, 7 and 9 are of importance. The reason for selecting only some segments is because the time it takes the signal to travel on the horizontal segments (t_s) is negligible ($t_s \ll t_h$) for cases which are within the scope of this thesis. Even if t_s is being considered (e.g. meander segments in wire groups), t_s and t_h could be surmised as $t_d = t_h + t_s$. Hence, after time $t = t_d$, when the main signal reaches the FE of segment 2, the stimulated crosstalk triggered by the main signal in $t = 0$ will also reach the FEs of their respective meander segments (NE 2 → FE 3, NE 4 → FE 5, NE 6 → FE 7 and NE 8 → FE 9). At this point new crosstalk voltage will be stimulated and superposed on the existing and previously induced crosstalk voltage. This continues at every NE and FE of every meander segment as the main signal propagates through the wire. Finally, it might happen that the accumulated crosstalk voltage surpasses the threshold voltage of logic switching before the main signal's arrival.

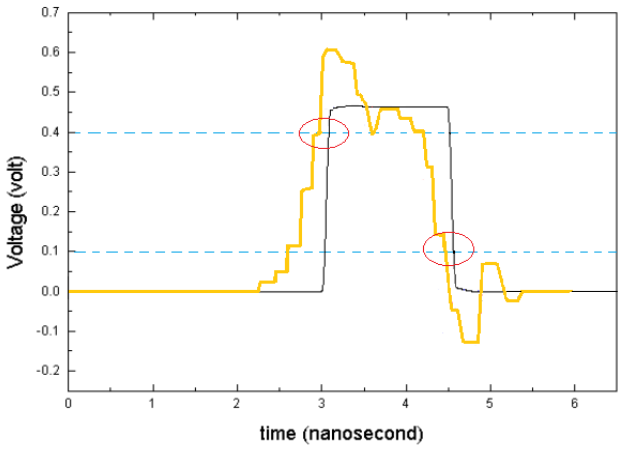


Figure 3.2. : Speed-up effect visualization (black – input signal; yellow – noise afflicted output signal; blue dotted lines – signal switching thresholds; red circles – speed-up effect)

The denser the routing and higher the clock-speeds of a design the more pronounced the negative effects of that accumulation become. Because reducing the clock speed of a single wire or a whole bus is rather difficult and will have an adverse impact on system performance, most PCB designers try eliminating the crosstalk by sorting the meander segments as even as possible on the board and increasing separation between lines [9]. However, no matter how one designs meander segment delay lines, if the PCB area does not increase, the crosstalk, which is inversely proportional to the separation between the lines, will remain.

3.2. Flat spiral delay line delay characteristics

Alternate designs, which force the crosstalk to accumulate *asynchronously* (i.e. crosstalk accumulates not in a additive fashion but distributive over a longer period of time) came into focus after the dimensions of PCBs reached smaller scales. One such design is the already

mentioned flat spiral delay line. Its main advantage over other designs is that it minimizes the periodicity of the transmission line routing as much as practicable, whilst also spreading the crosstalk noise.

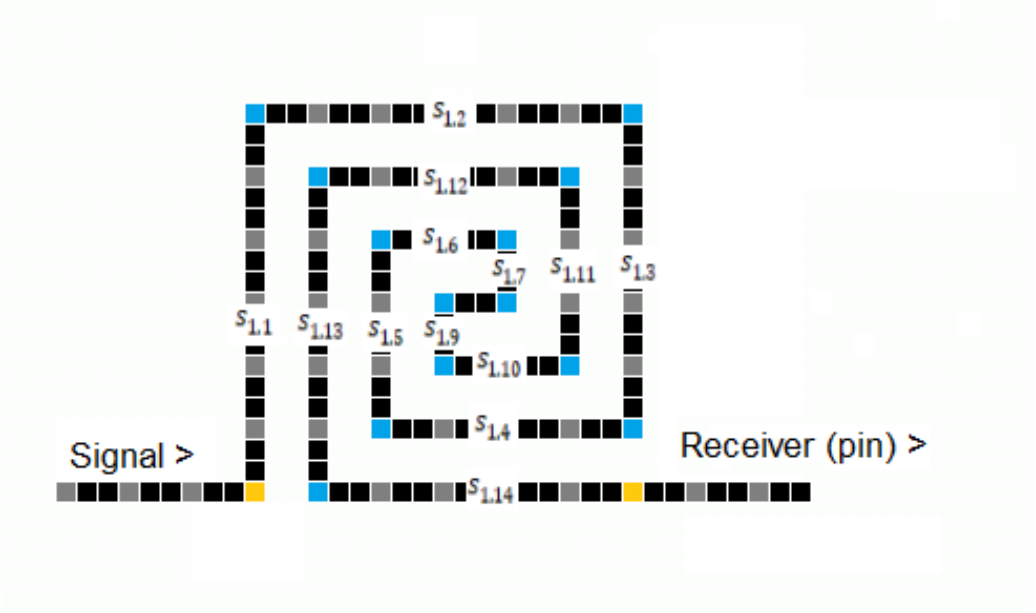


Figure 3.3. : Flat spiral delay line (14 segments)

In order to explain the delay characteristics of a flat spiral delay line the following paragraphs will rely on the spiral shown above. The spiral consists of 14 segments labeled $s_{1,1}$ to $s_{1,14}$ of which 9 are of different lengths and 5 of them have a sibling wire segment. Furthermore, the spiral has 13 bending points which shall be labeled $b_{1,1}$ to $b_{1,13}$ and are colored blue in the picture (e.g. $b_{1,12}$ is the connecting point between $s_{1,12}$ and $s_{1,13}$). There are also a starting and end point (i.e. orange points). Finally, let's initially assume that coupling between non-adjacent lines is negligible and that at time $t_0 = 0$ a switching signal arrives at the starting point. Once this signal is present a forward propagating crosstalk will be induced at the bending point $b_{1,13}$. This crosstalk will arrive at the receiver after the time it takes for it to travel the segment $s_{1,14}$, but also a long time before the main signal arrives. More accurately, the main signal will arrive a period which equals the time it needs to propagate the entire spiral minus the last segment later than the mentioned induced crosstalk. Assuming the time

delay through a segment $s_{1,j}$ is t_j, this means the crosstalk arrives at time $t = t_0 + t_{14}$, while the main signal arrives at $t = t_0 + t_1 + t_2 + t_3 + t_4 + t_5 + t_6 + t_7 + t_8 + t_8 + t_9 + t_{10} + + t_{11} + t_{12} + t_{13}$. When the main signal arrives at bending point $b_{1,13}$ at time $t = t_0 + t_1$, firstly, the previously induced crosstalk will have already arrived at the receiver because $t_{14} < t_1$ based on their respective wire lengths, and secondly a new crosstalk will be induced at bending point $b_{1,12}$. This crosstalk will arrive at the receiver $t = t_0 + t_{14} + t_{13}$ later but still by $t = t_0 + t_1 + \cdots + t_{11} + t_{12}$, sooner than the main signal. This trend continues until the main signal arrives at $b_{1,6}$, i.e. close to the center of the spiral, where the crosstalk will happen with neighboring bending points, i.e. close to main signal itself. However, because the distances are very small in the center of the spiral (i.e. segments have smaller lengths), the crosstalk cannot accumulate fast enough to cause the speed-up effect to appear. The moment the main signal leaves the center of the spiral, for example, arriving at bending point $b_{1,10}$ or any other following bending point, no forward propagating crosstalk will be generated between adjacent wires. There will be backward propagating crosstalk with point $b_{1,3}$, but this is not a concern in the scope of this thesis. A resulting signal wave-form, based on the findings in [8], can be seen in Figure 3.4. It is notable that the crosstalk noise arrives a long time before the unadulterated main signal pulse and there is even a noiseless period between the crosstalk and the edge of the pulse (i.e. time when only backwards propagating crosstalk happens). This means the flat spiral delay line can provide an almost complete isolation of the accumulative noise from the unadulterated pulse.

The analysis so far ignored coupling of non-adjacent lines, however, even if included, it should be easy to see that crosstalk noise will accumulate in a distributive fashion and not synchronously because it does not increase only at fixed time points (i.e. in contrast to meander segments' near-ends and far-ends).

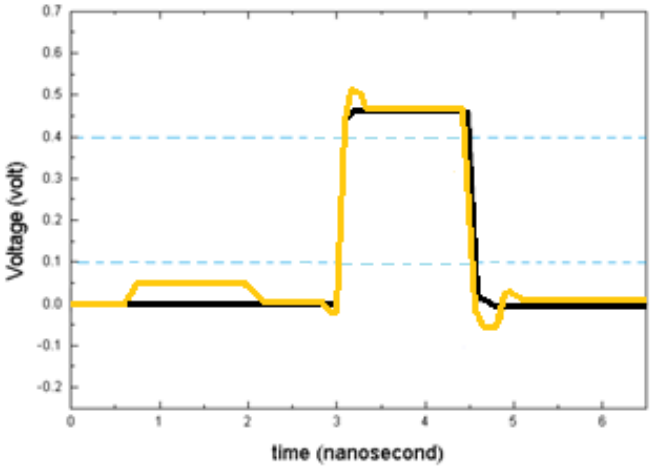


Figure 3.4. : Flat spiral delay line delay characteristics (black – input signal; yellow – noise afflicted output signal; blue dotted lines – signal switching thresholds)

4. Problem formulation

4.1. Input

The algorithm, which will be discussed in the next chapter, and the resulting implemented PCB router, discussed in chapter 6, both take sets of points, defined by their x and y coordinates, and the upper and lower length bounds for all given wires, as their inputs.

The first set of points denote the corner points of the overall available routing area. For example, if the routing area is rectangular, there will be four points in this set.

The next set of points are the coordinates for the corner points of the components, which are fixedly placed within the routing area. There can and should be more than one set of four points in this second category, meaning there can be more than one rectangular component placed in the routing area (i.e. at least 2 sets). Although the algorithm will check whether the provided points are really located within the routing area, the responsibility for the correctness of the inputs should be with the user (i.e. there are no checks, if points provided by user describe overlapping components).

The third category of points, for which coordinates should be provided, are pins. Each provided set in the third category should be linked to a set from the second category, i.e. each component should have at least one pin. Whether the sets are linked is determined by the correlation of their coordinates. Because pins are expected to be located on the edges of the rectangular components, at least one coordinate of a pin, either x or y, should be equal to

either the x or y coordinate of at least two of four points belonging in a set from the second category. Moreover, each pin should also be provided with an integer index, which also stands for the wire index. Every index can at most appear two times (i.e. it is not possible to connect 3 pins with one wire).

Finally, the last inputs are the mentioned upper and lower length bounds for each wire, i.e. belonging to each index.

4.2. Output

The output of the algorithm and the PCB routing program, are collections of points and their coordinates. There will be as many collections, as there were wires defined in the input (i.e. provided indices). Furthermore, there is no defined amount of how many points there are in a collection. Each point within a collection will be a bending point. Only by knowing two consecutive bending points (i.e. two consecutive points in a collection), it will be possible to deduce, how the wire segment is oriented within the routing area. In other words, an output collection is an ordered set of points.

4.3. Objective

The main objective is to connect a number of electronic components which are fixed in a routing area with defined boundaries, are on a single layer of PCB and each have a defined number of pins. Furthermore, the wires should be length-matched and delay-matched using flat spiral delay patterns and be routed as compact as possible, i.e. wasted space should be minimized. Finally, the algorithm and the implemented router should be an efficient one, so it can compete with existing modern PCB routers.

4.4. Constraints

The PCB router in development should satisfy the following constraints:

- Routed wires should meet their given length specifications expressed in ε. (i.e. lower bound < wire length < upper bound)
- Delay line segments should have a flat spiral pattern and should not exceed the routing boundaries.
- The routing area is rectangular.
- Only two components with 10 pins each (ordered clockwise on one component and counter-clockwise on the other component) are placed within the routing area.
- Wire crossings are not allowed (redundant, due to the pin ordering on components, mentioned in the previous point).
- The components are rectangular and have three pins on either horizontal edge and two pins on either vertical edge.
- The special case when one component is placed dead center within the routing area is not allowed (following algorithm and implementation in their current form do not yet support this case).

5. Algorithm

There are ten steps in the algorithm which will be discussed in the following sections.

5.1. Phase 1: Establishing routing area boundaries

This step established dependencies between the different input points and provides the following algorithm with information, whether a point with coordinates (x_1, y_1) is within the available routing area. The idea used here is based in part off the BSG routing scheme developed in [3]. There the authors use a set of vertical (called V-walls) and horizontal (called H-walls) segments to partition the whole routing area into rectangular cells (i.e. each cell has 2 V-walls and 2 H-walls) and then map components, delay line patterns and even pins and wires onto the cells. However, the here discussed algorithm does not go as far. What is done in this step of the algorithm, is giving every created cell an attribute which denotes whether it ok to route within it or not. The size and location of a cell are determined by the positions of the four walls surrounding it, whereas each wall spans two adjacent cells. In order to better understand this step of the algorithm, let's look at the example shown in Figure 5.1. (next page).

There, based on the coordinates the algorithm received as inputs it created a set of adjacent rectangular cells. Afterwards, it added the mentioned attribute to every cell – visualized using green (i.e. cell, where routing is possible) and red (i.e. cell, where routing is not possible).

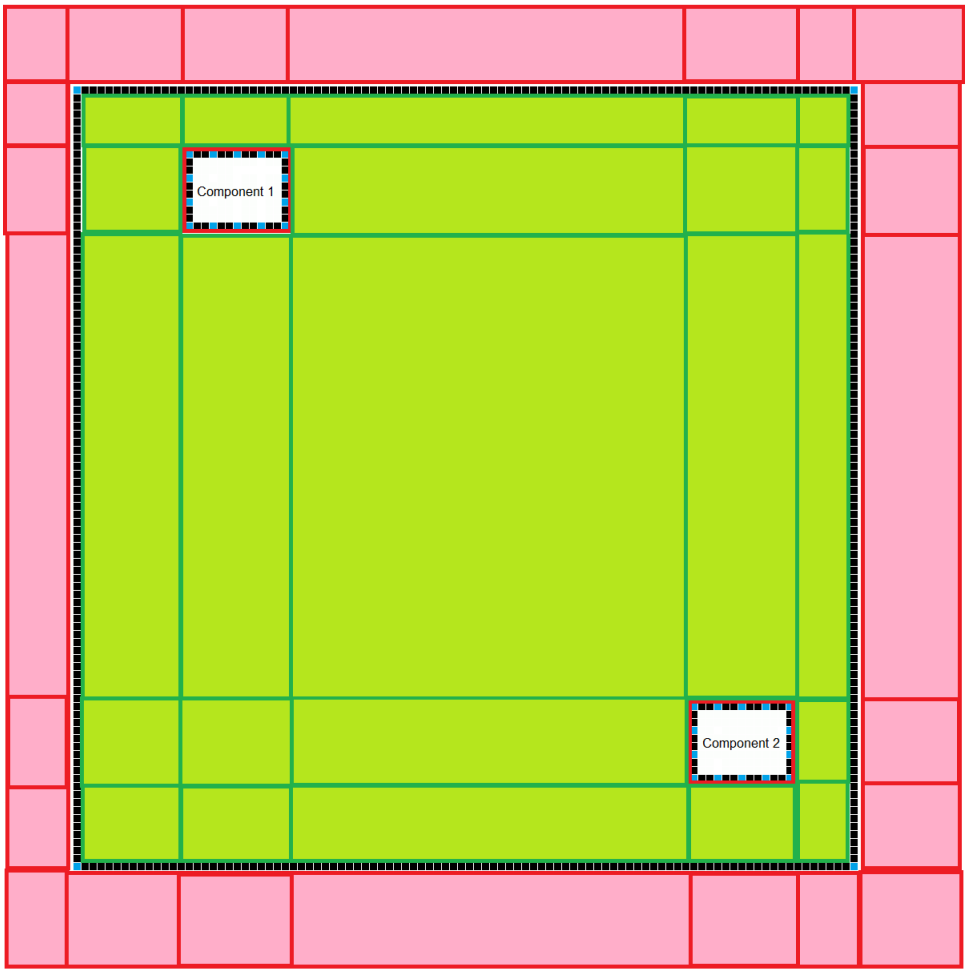


Figure 5.1. : Routing area boundaries visualization (blue points – input points i.e. pins, component corners, routing area corners; darker shades of red/green – BSG cell walls; red – cell, where routing is not possible; green – cell, where routing is possible)

5.2. Phase 2: Simplifying components

This step helps decide, which wire to route first (i.e. which two pins to connect first). There are three sub-steps which need to be taken.

The first sub-step is calculating the Manhattan distance between all the corner points of the two components. There are eight points in total, four per component. The distance between

points of the same component is of no interest, so per one point of a component four computations (i.e. other components corners) need to be made, 16 computations in total. Among the results one will have the smallest value and one will have the largest. These two points which are closest to each other will be called *friends* and the two points which are furthest from each other will be called *enemies*. Due to the geometry of the components, the friend and enemy points belonging to the same component, will be diagonally opposite points of the rectangular component. For each component the edges adjacent to its friend point shall be the vertical and horizontal *friend-lines,* while the edges adjacent to its enemy point shall be the vertical and horizontal *enemy-lines.* There will be pins located on friend-lines as well as enemy-lines. Taking constraints into considerations there will be 5 pins on each one.

The second sub-step is to check, if after taking a step perpendicular to each enemy-line one exits the routing area, i.e. crosses into a red BSG cell. This step shall be called *initial step* and have the size:

$$initial\ step = total\ number\ of\ pins\ on\ both\ enemy\text{-}lines \times \varepsilon \tag{13}$$

If after taking this step one indeed exits the routing area, a shorter step should be taken (i.e. initial step = 5ε; shorter step = 4ε). The step size is each time reduced by one, until it stays within the routing area. However, if even the shortest step (i.e. step of size ε) leaves, the algorithm should terminate because this would suggest a pin is located at a components edge, which is coincident with the routing area boundary. Another reason for the algorithm to terminate would be if the sum of steps on both enemy-lines is smaller than the initial step.

If neither is the case, the discovered step sizes for each enemy-line denote how many pins on enemy-lines can be routed towards friend-lines. To ease the understanding there is an example shown in Figure 5.2. below and an explanation follows.

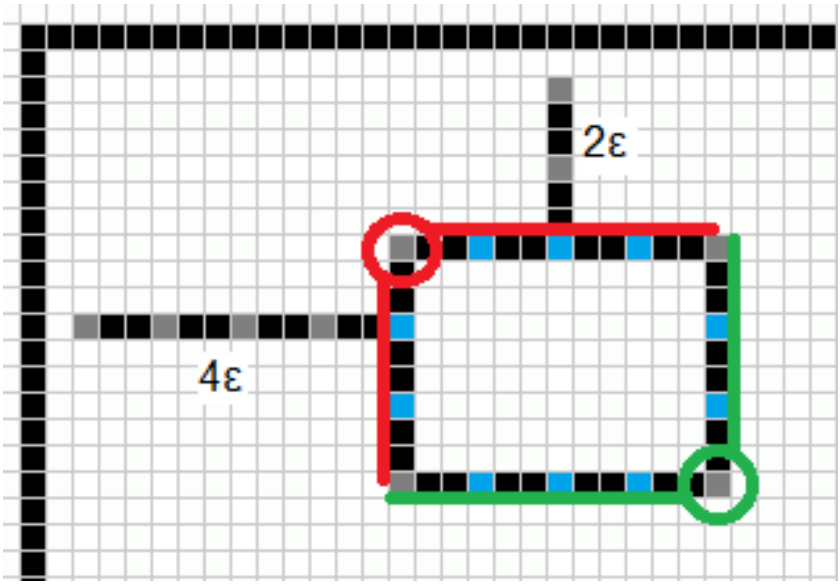


Figure 5.2. : Simplifying components second sub-step (blue points – pins on a component; green circled point – friend point; green lines – friend-lines; red circled point – enemy point; red lines – enemy-lines)

First, let's note that the initial step (i.e. 5ε) was outside the routing boundaries for both enemy-lines. After the appropriate step reductions, it turned out that the router can only take a step of 4ε away from the vertical enemy-line and a step of 2ε away from the horizontal enemy-line. Luckily, the algorithm does not terminate because the sum $4\varepsilon + 2\varepsilon$ is bigger than the initial step (5ε).

The third sub-step of the simplifying components phase is also a simple one. The pin closest to a neutral corner point (i.e. neither enemy nor friend) and belonging to the enemy line with the shorter step is selected. From this pin a wire of length ε is routed perpendicular to the

enemy-line after which a bending point is set and the wire is extended in the direction of the closest friend-line. When it hits the extension of the friend-line another bending point is set. Next, the neighboring pin to the previously selected one is chosen and a wire of length 2ε is routed perpendicular to the enemy-line. Another bending point is set and the wire is again extended in the direction of the closest friend-line until it hits its extension. This process is repeated for all remaining pins which are on enemy-lines until they are connected to the extensions of friend-lines by their respective wires. It is important to remember the lengths of every wire.

Figure 5.3. shows how the example from Figure 5.2. changes after going through the third sub-step.

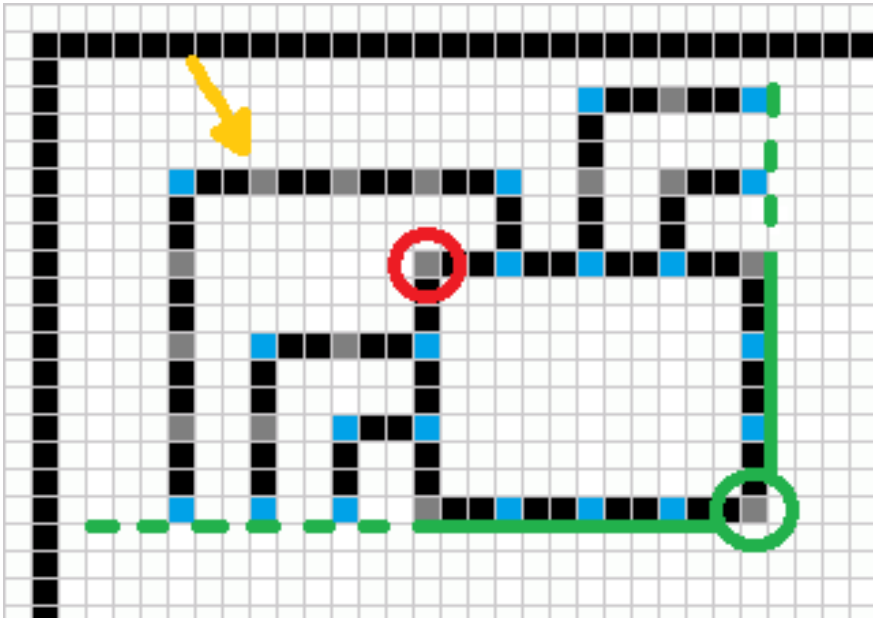


Figure 5.3. : Simplifying components third sub-step (blue points – pins on a component or bending points; green circled point – friend point; green lines – friend-lines plus extension; red circled point – enemy point)

Noteworthy is the middle wire, pointed at by orange arrow. This wire has one more bending point than other wires. This occurs when there are pins on an enemy-line which cannot be routed towards the closest friend-line (i.e. the horizontal enemy-line has 3 pins but its perpendicular step was only 2ε, which means only two wires could be routed towards the closest friend-line). These pins should be left until there are no other free pins which satisfy the regular third sub-step scheme. Then, starting from the pin closest to the enemy point they are routed towards the extension of the available friend-line using the Manhattan Distance.

Finally, there is a special case not yet discussed in this phase. It is the case when two components are of equal size and their geometric centers are on a common mirror line. In this case, in the first sub-step, one will discover that each component has two friend points and two enemy points. However, in truth this is not a problem and the next two sub-steps can be applied just the same if an alteration is made where three edges of the rectangular component are enemy-lines and only one edge is a friend-line (i.e. the one between the two friend points).

5.3. Phase 3: Checking feasibility

The previous simplified components main step and all its sub-steps were applied to both components separately. Afterwards, the wires on the edges of the friend-lines (i.e. the ones whose newly created bending points on the friend-line extensions are furthest away from the friend point) can be named the *boundary wires*. Due to the pin order off the components the boundary wires will have consecutive indices (e.g. if one has index 4, the other has to have index 3 or 5). However, a situation may occur where the boundary wire indices of one component differ from the boundary wire indices of other component. This would create problems while routing using the following steps of the algorithm.

Therefore, in this step the check is performed. If the previously found boundary wires are not the same for both components, the found solution in the simplified components main step's third sub-step is marked as invalid for one component and a new solution is looked for. An example for a different solution to the one presented in Figure 5.3. is shown in Figure 5.4.

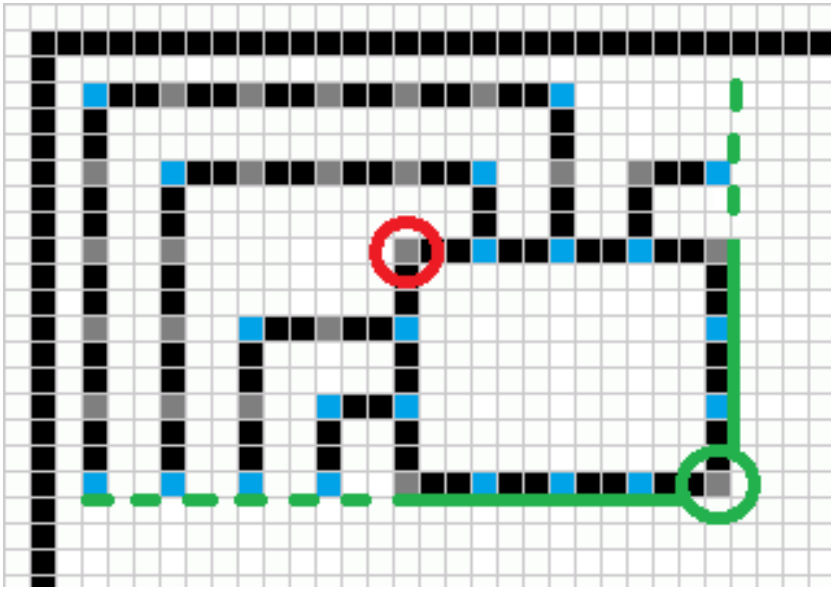


Figure 5.4.: Alternative solution for the third sub-step (blue points – pins on a component or bending points; green circled point – friend point; green lines – friend-lines plus extension; red circled point – enemy point)

Note that now there are four wires connecting to the extension of the horizontal friend-line instead of three and only one connecting to the vertical instead of two. For the shown example this was the only other option but in practice there might be many more solutions, which need to be checked before the right one is discovered. Also the number of solutions, whether valid or not, will increase if the number of pins increases.

Finally, after performing the feasibility check, the routing area boundary has to be modified. To do that, phase 1 has to be performed again.

5.4. Phase 4: Routing wire

In this step two cases have to be considered. First one is routing a boundary wire and the second one is routing any other wire.

The reason why routing a boundary wire is different, is that one must watch more carefully to stay within length requirements. However, in this version of the algorithm, this problem was solved by means of keeping track of the remaining Manhattan distance between the current wire lead position and the destination (i.e. bending point or pin). So after finding the two boundary wire candidates in the previous step one can now choose which one to begin with. There is no difference but the algorithm will always begin with the wire with the lower index.

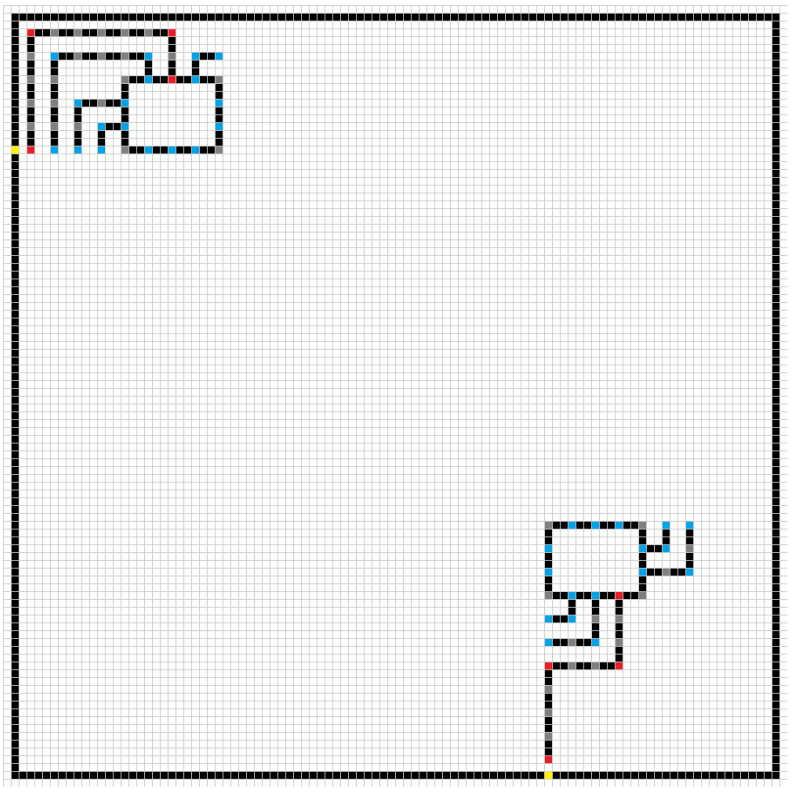


Figure 5.5.: Starting condition for boundary wire routing (blue points – pins or bending points; red points – pins and bending points of the chosen boundary wire; yellow points – closest point on the closest routing area boundary)

Another reason why routing boundary wires is different is that there is no reference wire to route along. However, it is best to find a reference and for the boundary wire this will be the routing area boundary. So when routing the boundary wire the initial direction to choose is towards the closest point on the closest routing area boundary (e.g. Figure 5.5. yellow points). Also it is necessary to route the boundary wire towards the routing area boundary for both components, thus creating different starting conditions than for all the remaining wires.

When routing a new wire the following temporary variables will be created:

l_i – current total length of the ith wire,

$L_{i,MIN},\ L_{i,MAX}$ – length bounds for the ith wire,

$(x_i,\ y_i)$ – coordinates of the current wire lead position,

$\left(x_{i,start},\ y_{i,start}\right)$ – coordinates of a pin or the last bending point,

$\left(x_{i,fin},\ y_{i,fin}\right)$ – coordinates of the destination pin or bending point.

After the starting conditions for the boundary wires are set as shown in Figure 5.5. the following routing steps are identical as for any subsequent wire. First, the current wire lead position has the same coordinates the start position, i.e.

$$(x_i, y_i) = \left(x_{i,start}, y_{i,start}\right)$$

From there one can take a step (i.e. change coordinates of (x_i, y_i) accordingly) of size ε in one of four directions:

$$(x_{old}, y_{old}) \rightarrow (x_{new},\ y_{old}) \mid x_{new} = x_{old} + \varepsilon \quad (14)$$

$$(x_{old}, y_{old}) \rightarrow (x_{new},\ y_{old}) \mid x_{new} = x_{old} - \varepsilon \quad (15)$$

$$(x_{old}, y_{old}) \rightarrow (x_{old},\ y_{new}) \mid y_{new} = y_{old} + \varepsilon \quad (16)$$

$$(x_{old}, y_{old}) \rightarrow (x_{old},\ y_{new}) \mid y_{new} = y_{old} - \varepsilon \quad (17)$$

Of course, not all directions will be valid, for example taking a step “back” when at a pin would end up inside the component which is not a valid green BSG cell. Furthermore, before deciding to take the step, first it is necessary to check the validity of the step directions of that point. The aim is to route along an existing boundary (either routing area boundary or another wire). Keeping that in mind, the point, which is under consideration of being the next step, should have two of its next directions within a valid routing area and one direction outside a valid routing area. The last of the for directions would be returning and should be disregarded. However, if the point, which is under consideration of being the next step, has only one of its next directions within a valid routing area that means it will be a bending point. Finally, after taking every step and before even considering the next one, the algorithm checks if the current position is the destination.

By taking these precautions with the point, which is under consideration of being the next step, it is assured that no missteps are made and the routing is as dense as possible. Also only after reaching the destination the algorithm enters the next main phase. The Figure 5.6. depicts the result of this phase when applied to the previously discussed boundary wire example while adding an obstacle. Green points are the steps taken and among them are visible blue points which the algorithm successfully determined to be bending points.

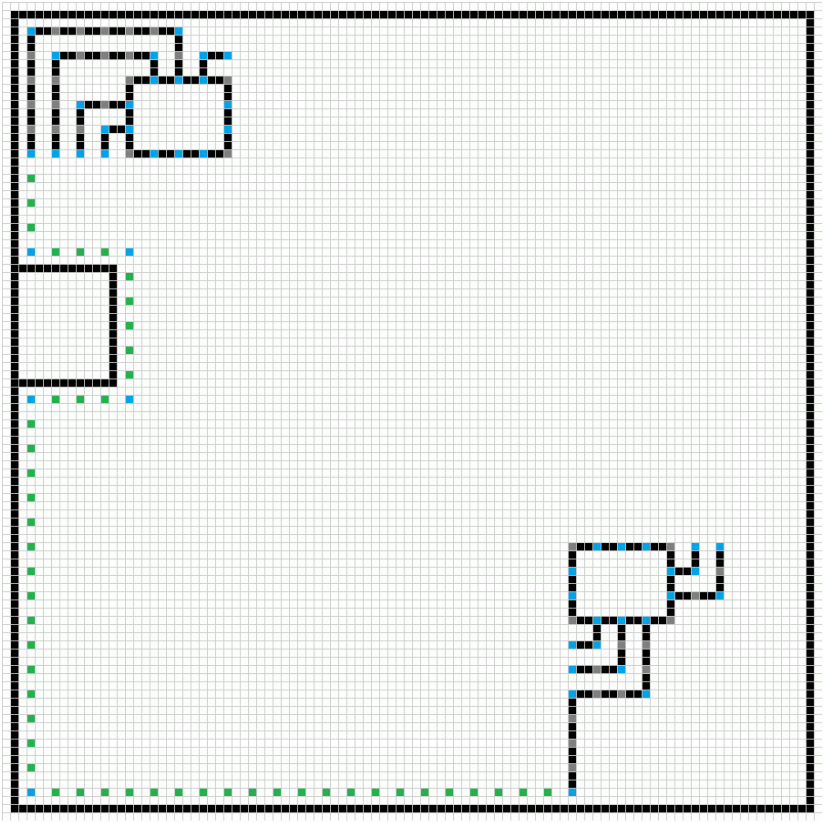


Figure 5.6.: Wire routing stepping visualization (blue points – pins, defined bending points or determined bending points; green points – steps taken every ε)

5.5. Phase 5: Length matching

The final phase of the algorithm is the length matching phase where the flat spiral delay line finally appears. The first thing to do is check whether the already routed wire is within the length bounds or whether it overshot or undershot. If it overshot, the routing of that wire is reset and the same tactic is used as in the case of the boundary wire, i.e. keeping track of the remaining Manhattan distance between the current wire lead position and the destination. If the wire has not yet reached the required length, length matching is required.

In that case, first, the wire is abstracted as a straight line with points on it. The line starts at the pin of one component and ends at the line of the other component. The points on the straight line are in reality the set bending points. So the line will consist of many segments

with their starting and ending points being bending points. Now similar to what was done in chapter 2, where the definitions for free spaces and the mathematical model for a spiral was discussed, first the segments of the line has to be separated into available and unavailable segments. In Figure 5.7. the available segments were marked green and the unavailable segments were marked red for the boundary wire example.

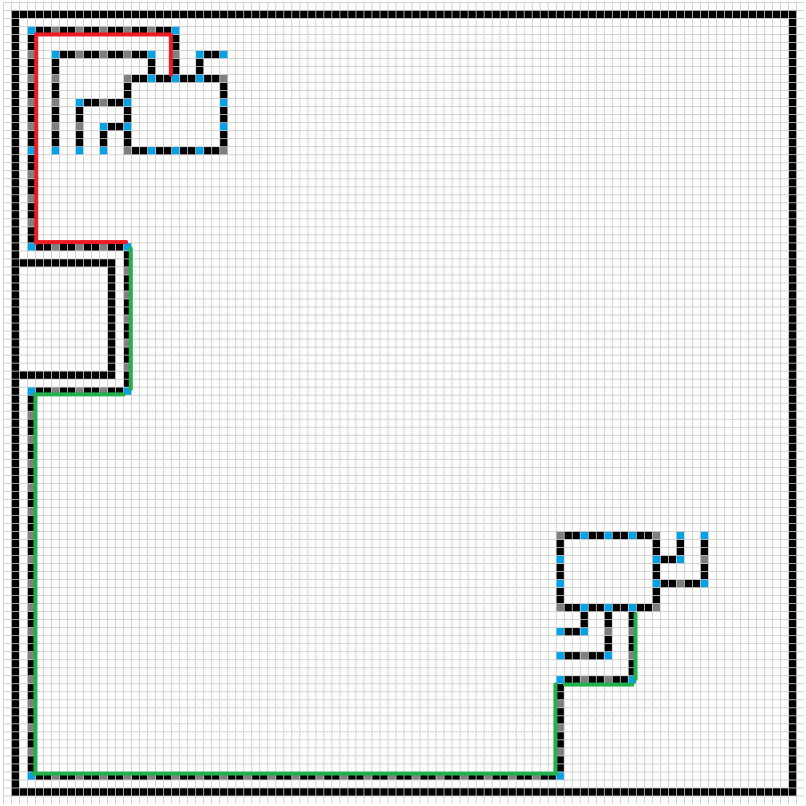


Figure 5.7.: Available (green) and unavailable (red) segments

Next, it is necessary to deduce how big the free spaces for the available segments are. Here, the idea of BSG cells, as previously used in phase 1 of the algorithm, can be recycled. Hence, knowing W and H of the free spaces, it is possible to inscribe the spirals applying their mathematical model. As an additional rule which helps reduce obstacles for future wires the available segments should each have a priority attribute. This will determine where spiral will eventually be drawn. The segments with the highest priority should be the ones in the middle

of the line, i.e. furthest from the pins. The priority decreases the closer a segment is to a pin. Furthermore, if a segment chosen for the spiral is too long, i.e. the width of the available free space is too big, it can be divided into smaller segments by splitting it in half. The split point will have the same importance as a bending point.

After deciding on the spiral, i.e. knowing the number of sibling segments n, the lengths of p, q, h_{spiral} and w_{spiral}, as well as the increase in wire length due to the delay line, the coordinates for the bending points, which create the spiral, can also be calculated.

Finally, after finishing length-matching of the wire the routing area boundary has to be modified. To do that, phase 1 has to be performed again, after which the algorithm needs to jump directly into phase 4 again and route the wire which is adjacent to the last routed one.

6. Implementation

6.1. Initial program implementation

The flat spiral delay line based PCB router described by the algorithm introduced in the previous chapter was programmed using C++ during the course of working on the thesis. However, there were multiple implementations along the line, which grew together with the increased focus of the algorithm. The initial implementation was rather crude programming. It did not focus on efficiency but solely on providing a result. For ease of development the program was split into multiple smaller standalone programs and all of them accessed and wrote the results to external txt files, which they then shared amongst each other.

The first of the small programs used a user generated txt file as data input and before even starting to work on the problem, it had to sort through the unformatted data to find the relevant pieces. The program was doing, what was described in phase one of the algorithm, namely, establishing routing area boundaries based on BSG cells. The output of the file was then fed to another small program.

This one focused only on routing wires. The user had to manually select which wire should be the boundary wire, i.e. the next wire to be routed. The output of this program was yet another txt log file which was shared with the first program and the last of the small programs.

The last of the small programs was using the software Gurobi Optimizer ver.7.5.1, to choose the optimal place and size of the spiral on the wire by solving the mathematical formula of

the spiral while being subject to various constraints. However, the solver oftentimes chose very unique looking spirals, i.e. very wide but not high – almost similar to meander segments, which did not offer the expected advantage of reducing the speed-up effect. Furthermore, when there was little free space but just enough for one last spiral, the solver frequently crashed. Still, when it worked the output it generated was saved to another txt file, which was shared with the first program.

The first program, having received results from the second and third program would than start the cycle anew. Hence, for example, routing ten wires would generate approximately 20 txt log files and the implementation as a whole would be very inefficient in handling them.

6.2. Improved program implementation

That is the reason why an improved version of the implementation was necessary. The development was started from scratch and using an IDE (Visual Studio by Microsoft). Now the program runs smoothly and the majority of efficiency issues stem from the algorithm itself. The program is not split up and it has a command line style UI which allows for providing the necessary inputs. There is no need for saving them in a separate txt file unless it's a bash script. Furthermore, it does not use Gurobi anymore but relies on a precompiled library of spiral patterns with different shapes and sizes. The program then only needs to select the fitting one. However, the output of the program is still just a txt file with a huge number of coordinates.

7. Possible improvements and future research possibilities

7.1. Improvements to the algorithm

There exist some minor possibilities for improvements to the algorithm presented in chapter 5 which do not change it on the fundamental level. However, whether they are really improvements remains debatable without a sufficiently efficient router implementation for comparison. Also because as far as is known, the proposed PCB router seems to be the first modern router focused solely on the flat spiral delay line, there exist no alternative, comparable algorithm.

Still, one improvement would be to use bigger steps than ε (e.g. 2ε or even 3ε) (section 5.4). This change might significantly increase the efficiency of the algorithm by reducing the amount of necessary operations. However, ways of dealing with stepping out in the open or out of routing area bounds have to be considered. A possible solution might be to take consecutively shorter steps until a valid step, i.e. if a step of size 3ε turns out invalid, return and take a step of size 2ε; continue reducing step size until a valid step can be taken. Although this solution seems an improvement, considerations have to be made, what the best step size should be. Even a step the size of 4ε might already become an invalid step which is not detectable by the algorithm without a more fundamental change (i.e. stepping from one valid routing area region into another, whilst stepping over an invalid region e.g. region with a smallest sized spiral). The bigger the step size the more care will have to be taken to avoid

mishaps which might be a greater burden on the efficiency of the algorithm than taking small steps.

Another improvement to the algorithm is checking for the shortest remaining routing distance not just for the first wire but constantly for every wire. This will allow to spot cases where wires might run outside their requested maximum length bound sooner and reduce the computation time (i.e. discover if routed wire is within required bounds the moment it leaves them versus the moment it reached its destination pin). However, while using the Manhattan distance calculation for this task does not pose big issues for the first wire, because the routing area is uncongested, doing so with the remaining wires might require further considerations (e.g. Based on Manhattan distance calculation a currently routed wire has reached its maximum length, but routing it towards the destination pin in the proposed shortest possible way is not possible).

7.2. Improvements to the implementation

For this section the first proposed improvement is formatting the program output in a way machines can read and as such draw the resulting routing automatically. Getting just a huge set of coordinates is not a human readable format. Visualizing it as a routing diagram is the ultimate goal, but due to lack of prior knowledge of available software solutions and limited working time on the making of this thesis, it is beyond its scope. Nevertheless, a good place to start would be the freeware program *Asymptote*, which uses written syntax to draw graphs and diagrams, similar in fashion to the well known LaTeX program, which is used in academia.

Another improvement would be solving the problems which appeared when using the Gurobi Optimizer. As with the previous point, the lack of more advanced knowledge of this software's working principles and the limited working time on the making of this thesis, turned out being a hindrance to writing code which runs smoothly and efficiently. The presented solution works, but is not efficient enough to compete with other PCB routers, which focus on meander segment based routing and are supported by the market. Yet, as long as there is a demand for a solution which reduces or removes the speed-up effect, continued development of this PCB router or similar ones will happen.

7.3. Future research possibilities

Due to this being the final section of the thesis, let's try and look into the future. In the coming years more research into routing topologies, including alternatives to meander segment delay lines and maybe even flat spiral delay lines, will happen. Though, it may also be possible that PCB routers which incorporate both meander segment delay lines and flat spiral delay lines simultaneously in one routing diagram will appear. Another interesting direction for development of the flat spiral delay line based PCB router, is to use it in order to solve problems where more than 2 components are present and all should be interconnected. It becomes challenging because wire crossings will have to be avoided if the routing can only happen on a single PCB layer. Therefore, the last idea for future research is to incorporate findings which allow for automated routing on multiple board layers in the development of flat spiral delay line based PCB routers.

References

[1] M. M. Ozdal and M. D. F. Wong, "A length-matching routing algorithm for high-performance printed circuit boards", *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 12, pp. 2784–2794, Dec. 2006

[2] Y. Kubo, H. Miyashita, Y. Kajitani, and K. Takeishi, "Equidistance routing in high-speed VLSI layout design", *Integr. VLSI J.*, vol. 38, no. 3, pp. 439–449, 2005.

[3] T. Yan and M. Wong, "BSG-route: A length-constrained routing scheme for general planar topology", *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 11, pp. 1679–1690, Nov. 2009.

[4] C.-Y. Chin, C.-Y. Kuan, T.-Y. Tsai, H.-M. Chen, and Y. Kajitani, "Escaped boundary pins routing for high-speed boards", *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 3, pp. 381–391, Mar. 2013.

[5] T.-H. Li, W.-C. Chen, X.-T. Cai, and T.-C. Chen, "Escape routing of differential pairs considering length matching", in *Proc. Asia South Pac. Design Autom. Conf.*, Sydney, NSW, Australia, pp. 139–144, 2012.

[6] T.-M. Tseng, B. Li, T.-Y. Ho and U. Schlichtmann, „Post-Route Alleviation of Dense Meander Segments in High-Performance Printed Circuit Boards", in 13th *ICCAD Conf.*, Taiwan and ISBN 978-1-4799-1071-7, pp. 713–720, 2013.

[7] H.-B. Wu and F.-L. Chao, "Flat spiral delay line design with minimum crosstalk penalty", *IEEE Trans. Compon., Packag., Manuf. Technol. B*, vol. 19, no. 2, pp. 397–402, May 1996.

[8] O. Ramahi, "Analysis of conventional and novel delay lines: A numerical study", *J. Appl. Comput. Electromag. Soc.*, vol. 18, no. 3, pp. 181–190, 2003.

[9] T.-M. Tseng, B. Li, T.-Y. Ho and U. Schlichtmann, „ILP-based alleviation of dense meander segments with prioritized shifting and progressive fixing in PCB routing", *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 6, pp. 1000–1013, Jun. 2015.

[10] R.-B. Wu and F.-L. Chao, "Laddering wave in serpentine delay line", *IEEE Trans. Compon., Packag., Manuf. Technol. B*, vol. 18, no. 4, pp. 644–650, Nov. 1995.

[11] B. Rubin and B. Singh, "Study of meander line delay in circuit boards", *IEEE Trans. Microw. Theory Techn.*, vol. 48, no. 9, pp. 1452–1460, Sep. 2000.

[12] A. Kabiri, Q. He, M. Kermani, and O. Ramahi, "Design of a controllable delay line", *IEEE Trans. Adv. Packag.*, vol. 33, no. 4, pp. 1080–1087, Nov. 2010.

Appendix A

What follows is a display of the working of the PCB router described in this thesis prior to finalizing its working principles (especially phase 2: simplified components), therefore, its considered an appendix.

There are 11 figures, each showing the routing of consecutive wires in the given example. In the example two components of equal size are placed within a rectangle routing area such that their geometric centers are on a common mirror line (special case). The routing area was chosen such that $W = 940\varepsilon$ and $H = 740\varepsilon$. Furthermore, there are 10 wires to route, the wiring pitch is chosen to be 20ε and the wire should have a length between 2400ε and 2500ε.

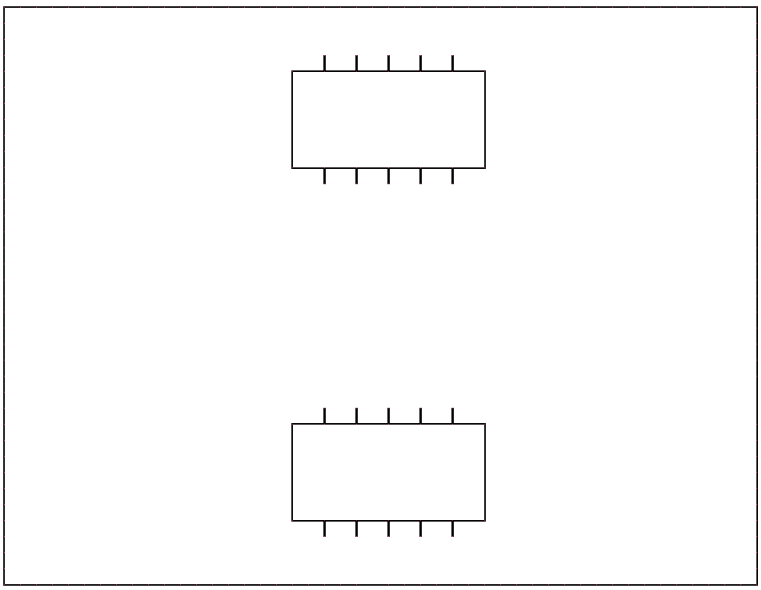


Fig. A1: Routing area

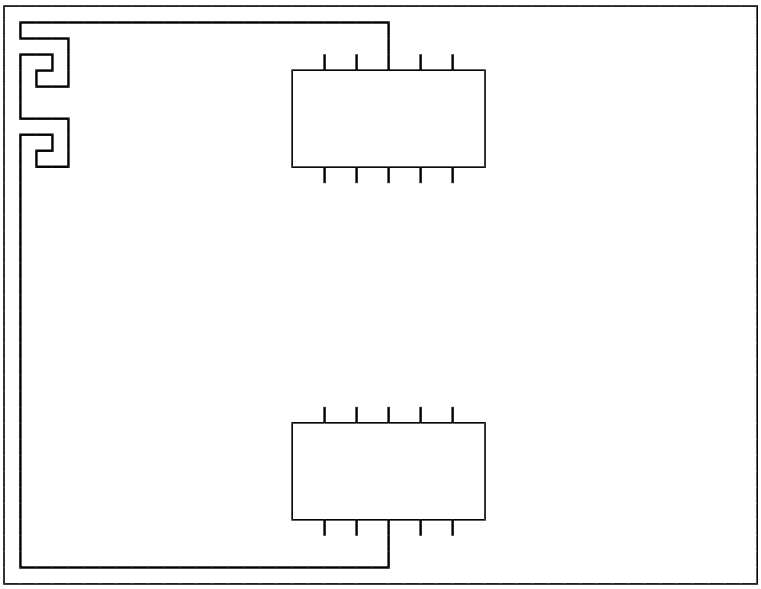


Fig. A2: Routing wire 1

wire length = 1940 (no spiral compensation)

wire length = 2420 (after spiral compensation)

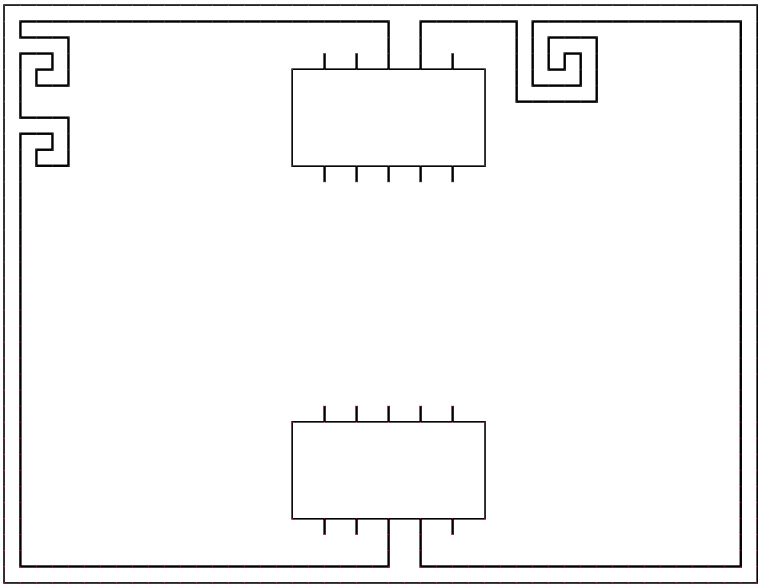


Fig. A3: Routing wire 2

wire length = 1820 (no spiral compensation)

wire length = 2420 (after spiral compensation)

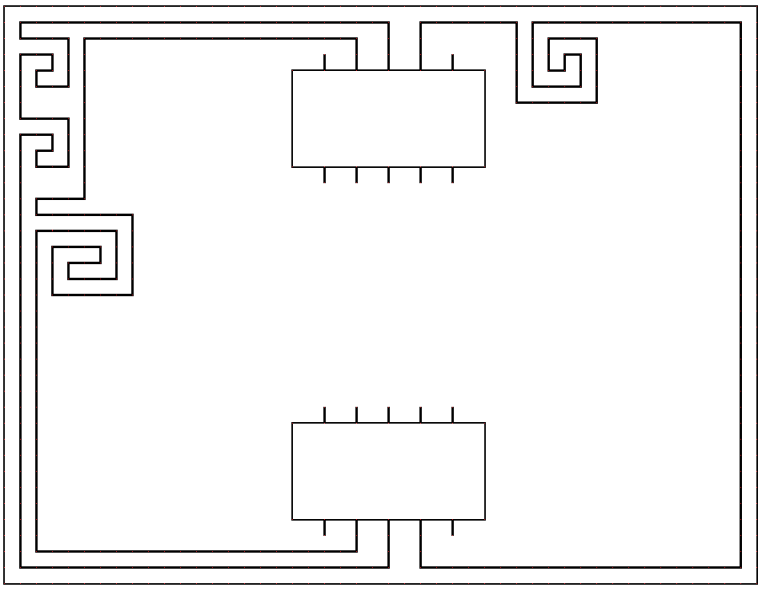


Fig. A4: Routing wire 3

wire length = 1740 (no spiral compensation)

wire length = 2460 (after spiral compensation)

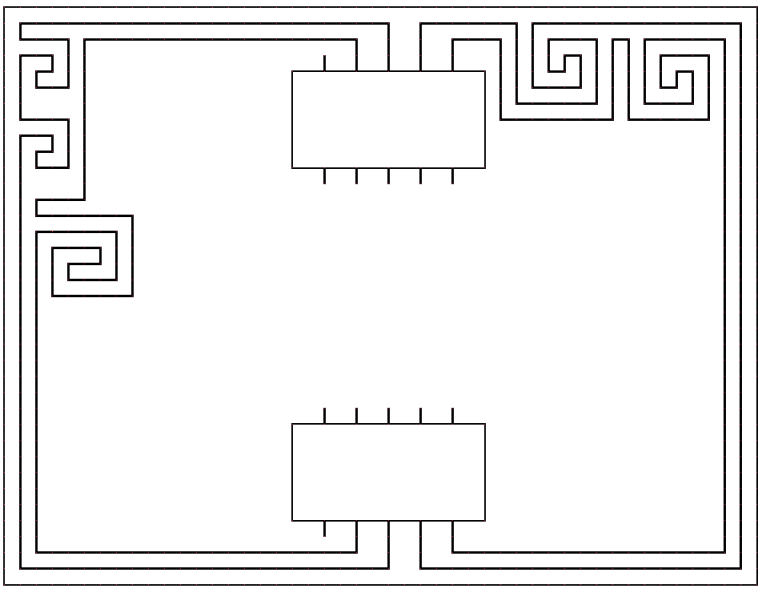


Fig. A5: Routing wire 4

wire length = 1820 (no spiral compensation)

wire length = 2420 (after spiral compensation)

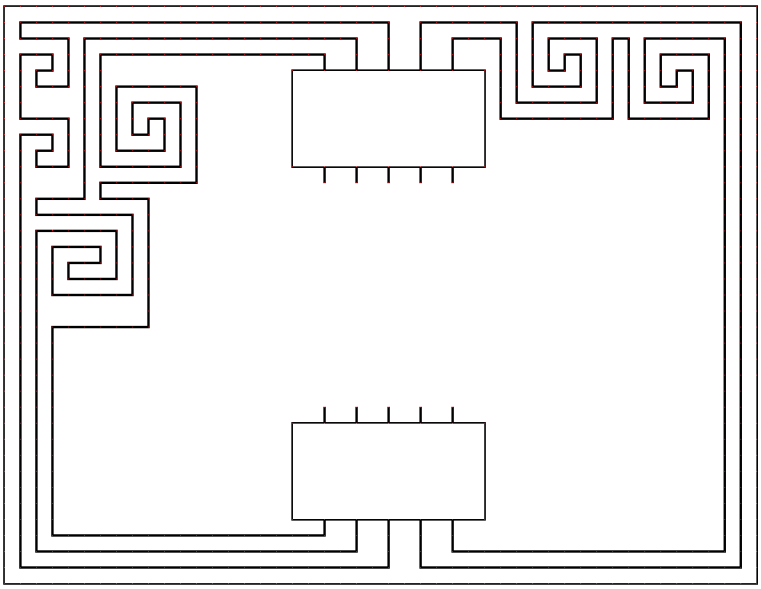


Fig. A6: Routing wire 5

wire length = 1660 (no spiral compensation)

wire length = 2500 (after spiral compensation)

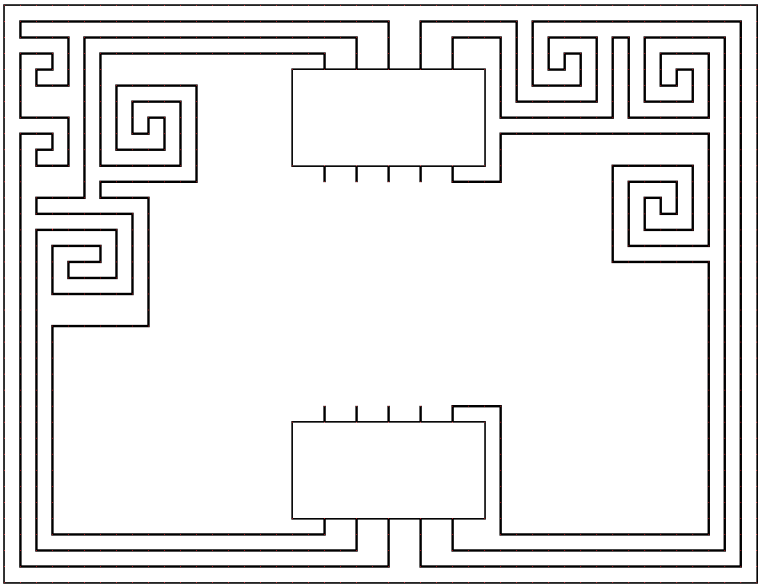


Fig. A7: Routing wire 6

wire length = 1620 (no spiral compensation)

wire length = 2460 (after spiral compensation)

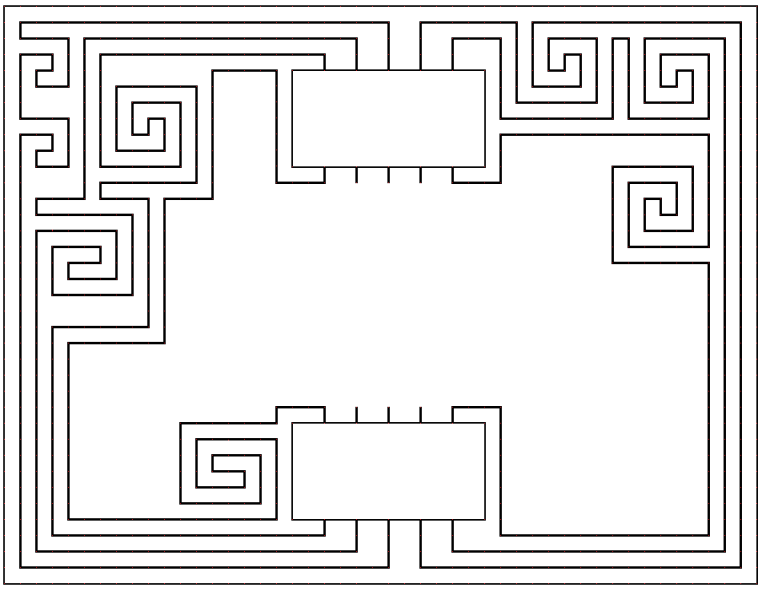


Fig. A8: Routing wire 7

wire length = 1740 (no spiral compensation)

wire length = 2460 (after spiral compensation)

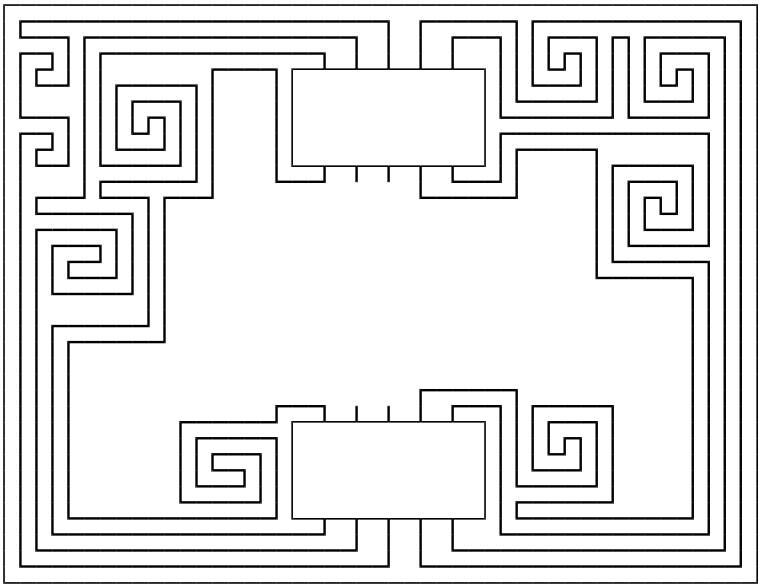


Fig. A9: Routing wire 8

wire length = 1660 (no spiral compensation)

wire length = 2500 (after spiral compensation)

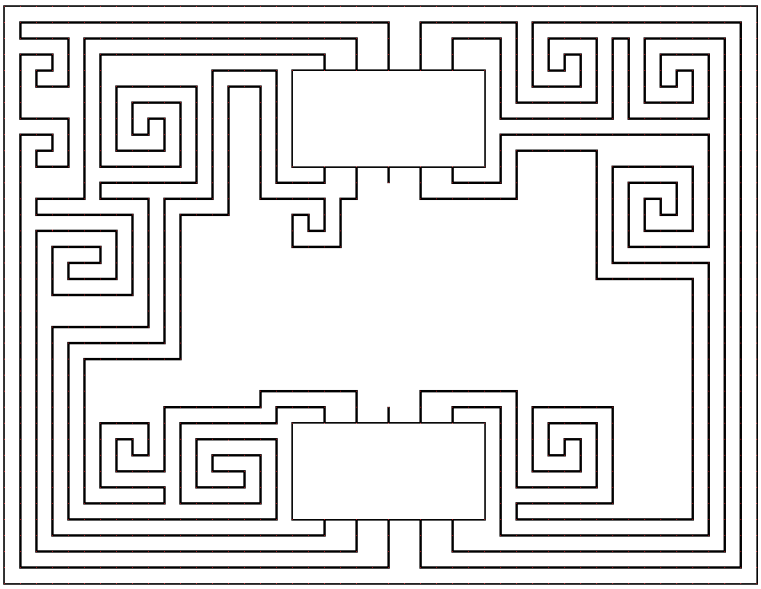


Fig. A10: Routing wire 9

wire length = 1780 (no spiral compensation)

wire length = 2420 (after spiral compensation)

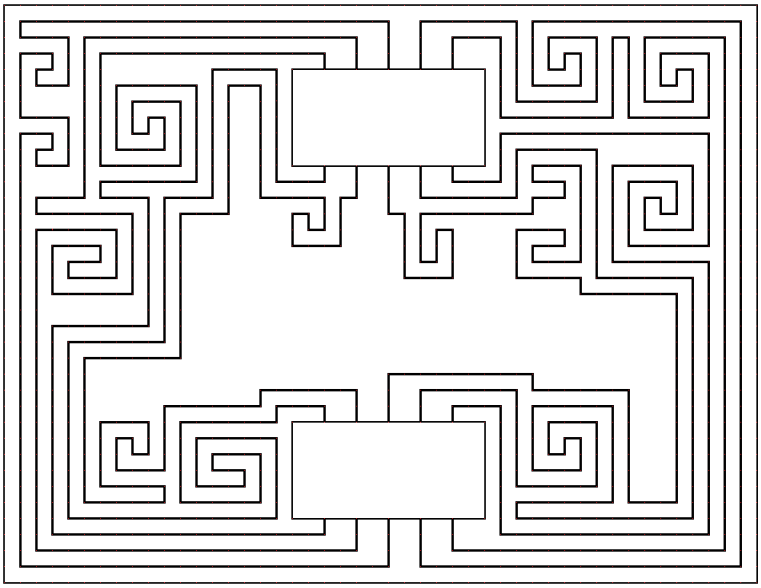


Fig. A11: Routing wire 10

wire length = 1800 (no spiral compensation)

wire length = 2420 (after spiral compensation)