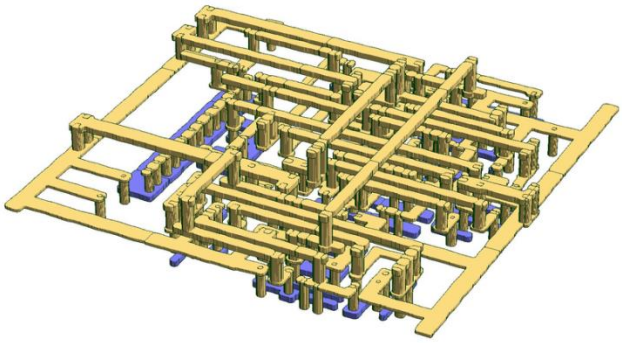
VLSI Routing flow based on Topological Patterns

College Mentor: Samy Zafrany

Introduction

Routing is the process by which chip components (such as transistors or standard cells) are interconnected. The routing flow starts after placing the transistors (or standard cells) at their optimal positions. Connectivity is achieved by laying out vertical and horizontal metal segments (called **wires**). Connecting points from two adjacent layers is achieved by vertical segments (called **vias**). Physical routing of an electrical network consists of one or more such wires and vias, which implement connections between the network end-points. Manufacturing constraints as well as design considerations impose strict rules on the feasibility and correctness of routing.



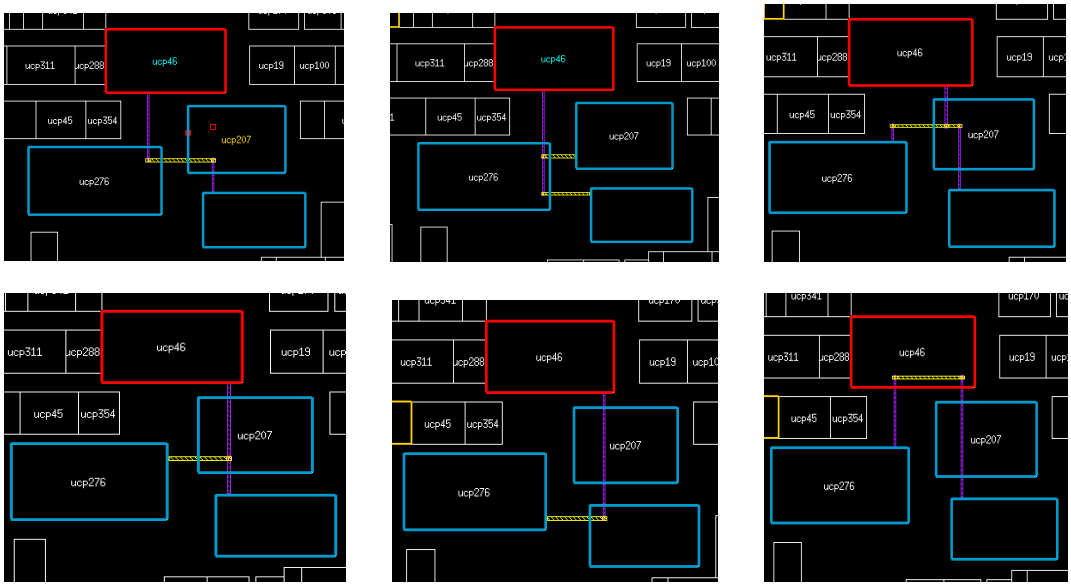
Today's "place and route" process is highly challenging due to the enormous number of electrical networks that must be routed within a very limited area (tens of thousands and even hundreds of thousands nets, where each net consists with several wires). This is a well known NP-hard problem which lacks practical analytical solutions. Thus, all industrial methods for solving this problem are heuristic, far away from being optimal, but strive to be good enough for marketing (highly optimized solution are prohibitive due to violation of the "time to market" rule). One of the most prevalent heuristic is to use several routing iterations; each iteration is a refinement of the former. At the first iteration, which is named **"Global Routing"**, we start with a "crude" and highly "dirty" routing solution (it usually consists of many electrical shorts and design rule violations). But at the next iterations we try to refine and resolve shorts and other type of violations iteratively until we obtain a fully legal and clean routing. Later stages are usually called **"detailed routing"** stages.

The purpose of this engineering design project is to introduce a simpler paradigm for the **global routing** stage in which net topologies are restricted to a small, simple, and preferable set of topological patterns. Presumably, this strategy will help us to minimize the number of shorts and violations (as much as possible). We believe that by restricting ourselves to a smaller set of "good" topological patterns (and giving up the majority of other traditional patterns), we will be able to simplify the global routing algorithms and obtain better performance (run time and memory, and thus better "time to market"). Traditionally, the global routing stage is done by maze-routing techniques which require long and

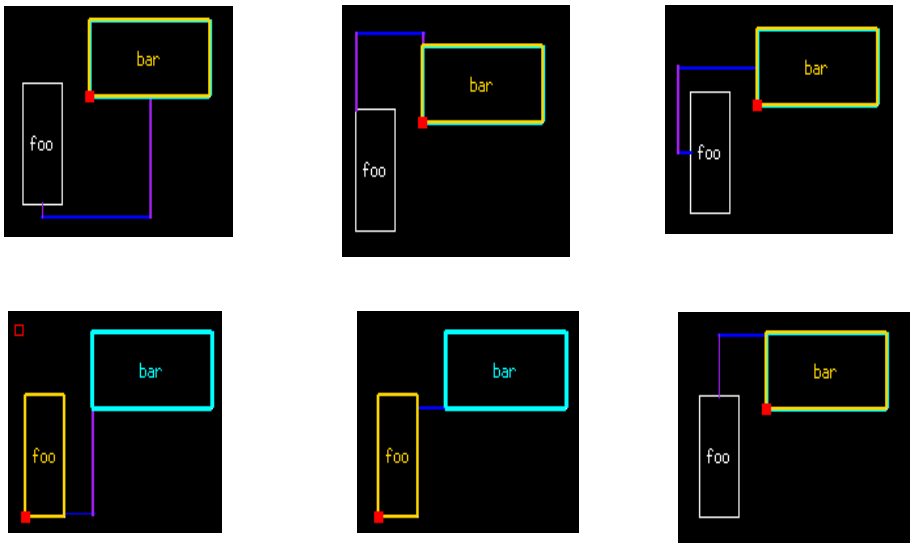
expensive computing resources (mainly due to the large area). We expect that by restricting ourselves to a small set and controlling the type of topological patterns we are allowed to use, will also minimize the refinement efforts at the detailed routing stages and final cleanup (even if we have to repeat the process due to inability to reach an adequate detailed routing). Although there is more than one technology to physically implement a chip (e.g. FPGA, Gate Arrays, automatic synthesis) this project is focused only on the custom design method as described above (which also involves human manual work).

Examples

In the next diagrams we see 6 different routing topologies on a single electrical network (with 1 driver and 3 receivers)



Examples of two-cell routings:



Inputs

The input of our routing problem is a given floor-plan of functional blocks (VLSI cells) and a list of electrical nets (netlist) which connect subsets of cells in this floor-plan. Cells may not contain information regarding precise location of terminals, so wires just need to reach their area in order to connect. The input also contains process-specific data and rules, such as number of routing layers we can use, and other manufacturing constrains.

System Goals

Our goal is to generate a Global routing for all nets in our netlist with as little violations as possible and with a greater chance for enabling the subsequent detailed routing stages to complete successfully. For that purpose we will introduce several metrics for measuring routing quality (such as area congestion, number of violations, number of shorts, etc.), and try to minimize them with respect to probable averages.

The role of the routing process output may differ depending on the design stage. In early design stages the output is used as an estimation of future routing, and consequently as an input to congestion and timing analysis tools, and as a key player in floor-planning considerations. In late design stages the output may be used for solving real connectivity problems, and therefore reach the final stage of the chip fabrication.

Due to time limits and resources we will confine ourselves to obtaining a working prototype and not a fully functional product (which will require more time and resources than the student has for his project). We will therefore use the Python scripting language which is well-known for its rapid prototyping capabilities, and its Tkinter GUI environment which is very easy to learn and use in short time.

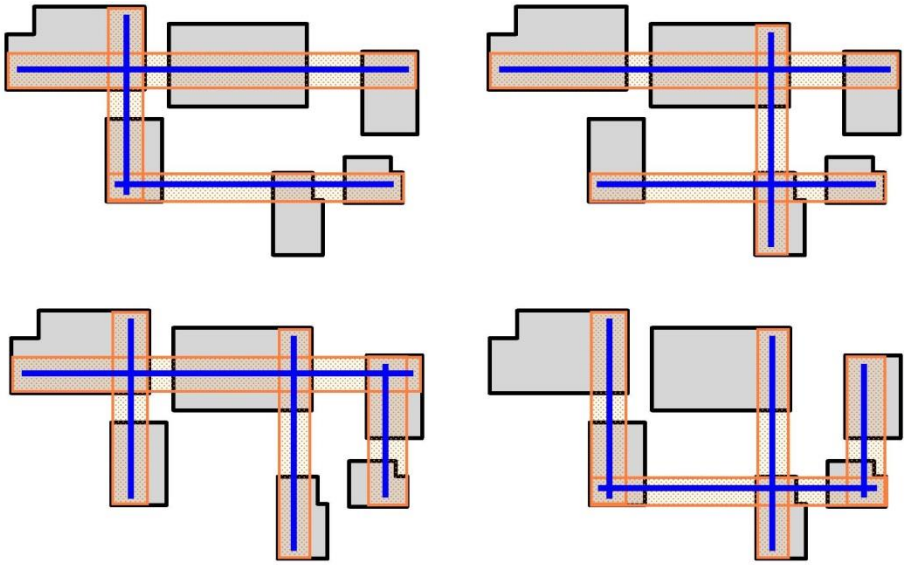
Student Responsibilities

Student should acquire sufficient background and expertise for:

1. Modeling VLSI Physical Routing of electrical networks (Wire Polygons) at a scripting level. It also includes abilities to draw (usually small) models on a graphical canvas for model visualization and verification
2. Compute optimized routings that adhere to a pre-selected set of topological patterns
3. Enable the user to formulate various parameters and constrains on the desired routings and know how to handle them properly, and provide advisory information on best topologies from the given list of patterns and other parameters
4. Ability to run routing flows on large groups of electrical networks of diverse kinds (serial or parallel mode).
5. Suggest compact sets of preferred topological patterns for a given network cell placement

6. Provide quality assessments on the outcomes of long global routing flows: accurate information on network resources such as total wire length, timing, resistance, capacitance, congestion, free area, number of shorts, design rule violations, etc.

Here is a suggested idea for exploring a solution:



The dotted orange rectangle represents a "Tunnels Grid" structure which is a sort of a "solution space" for a connecting segment. Routings are generated by "travelling" on tunnel centerlines. The student is of course free to seek other paths for finding a solution.

Outline for Work Stages

The following list is an estimated path on how to proceed to a successful project completion

1. The first thing is to get acquainted with the basic "place and route" concepts, techniques, algorithms, and software tools
   [40 hours]
2. We will be using Python and C++ programming languages and the Tkinter graphical user interface for visualizing VLSI components on top of a TK canvas. The student should invest the needed time to gain intuition, learn and get enough practice in these coding environments before wetting his hands with the real problems
   [20 hours]
3. The student should start with writing simple programs this includes mastering the VLSI GUI drawing tools (such as a basic VLSI canvas), communicating and manipulating VLSI canvas objects. To save time and effort, the student will be provided with basic software libraries which
   [20 hours]
4. As a start, the student should implement a simple inefficient algorithm for generating random routings for a given net and layout of its component cells. The student should be responsible for

generating inputs and checkers for checking algorithm validity and code robustness
[20 hours]

5. Next, the student should be able to understand and model the notion of a topological pattern, and be able to classify given routings to their respective topological patterns. This includes generation of appropriate sets of inputs for the algorithms, and an adequate set of checkers, benchmarks, and regression tests for verifying correctness. Regression tests should cover most of the challenging corner and extreme cases
   [30 hours]
6. The main challenge in this project is handling simultaneously a large group of nets (hundreds or thousands) for conducting a full global routing flow. The student will have to process each net (serially or in parallel) and select the best topological pattern based on a given set of (client supplied) parameters and requirements. The resulting route should be measured against known quality measures and should prove to be adequate. The complexity of the global routing problem is NP-hard, and there's still a lot of room for innovation and improvement. The student will enjoy a large selection of possibilities to choose from (like simulated annealing or evolutionary algorithms).
   [100 hours]
7. For that purpose, the student may have to get acquainted with several advanced heurist algorithms such as probabilistic methods, simulated annealing, evolutionary computing (genetic algorithms), and may adopt one of them for his problem - on a smaller scale if they turn to be too powerful for our problem, or the student can come up with a new more appropriate heuristics for the problem. This is yet very open to many directions and we will be more informative once we approach this stage
   [60 hours]
8. The last thing is to run our algorithm on real life cases (known internet industrial test cases) and see how we're doing in several benchmarks such as run time, memory consumption, floor quality, etc. If feasible, we may try to compare our performance against industrial algorithms if available (although most of them are proprietary and are not publically available)
   [50 hours]

The project is yet open for variations and extensions to other type of algorithms and layout quality constraints, and potential students are encouraged to come up with creative ideas, which can make this project suitable for several students (each focuses on a different algorithm/constraint set).

Development durability: One Semester (450 hours)

Resources

1. http://users.cs.cf.ac.uk/C.L.Mumford/Research%20Topics/layout/Outline.html
2. http://www.engr.uconn.edu/~tehrani/teaching/cad/14_floorplanning.pdf
3. http://vlsicad.ucsd.edu/Publications/Journals/j46.pdf

4. http://www.or.uni-bonn.de/~vygen/files/analyt.pdf
5. http://effbot.org/tkinterbook/canvas.htm
6. http://effbot.org/tkinterbook/tkinter-index.htm
7. https://wiki.python.org/moin/TkInter
8. http://effbot.org/tkinterbook/tkinter-index.htm