VLSI Multi-Commodity Routing Optimization Problem

A project by:

Osmond Xie - Andy Toulis - Kevin Lo - Nima Rabiee Nia

.

Introduction

Integrated circuits, such as microprocessors, consist of a large number of transistors that need to be connected optimally in order to reduce the chip size and increase performance. The approach that is taken in designing and producing these circuits starts by breaking down the circuit into smaller pieces called nodes. Pairs of these nodes called "nets" are connected in a process called routing. One approach to routing is to determine all possible node connections and provide a weight or cost to each "edge". An optimization problem can then be formed by minimizing the total cost of all paths taken to connect required nets. The main constraint in this problem is the amount of times that paths can overlap. This is also known as the amount of "layers" present in the microchip. In VLSI design, a new layer must be added every time a path overlaps. This is because physical overlaps will result in shorted circuits. In the case of this problem, the layers are predetermined by the manufacturer and the objective is to find viable and optimal routes within the layers.

For example, net 1 connects nodes at positions B and H and net 2 connects nodes at positions D and F. Ideal paths for the nets are B-E-H and D-E-F. Both nets pass through position E.

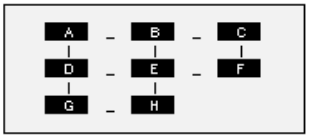


Figure 1 -Proposal Sample Problem

This is only allowable because a multi-layer system will let paths pass over each other rather than truly intersecting. In the figures below, the routes for the red and blue pairs needs two layers as they would cross if only one was used.

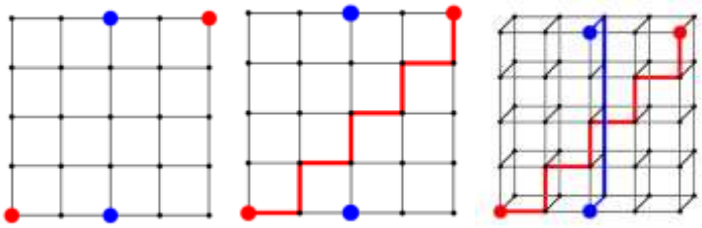


Figure 2 - VLSI Circuit Layers (Nahabedian, 2007)

Data

The data used in the model consisted of 131 nodes and 374 edges gathered from an existing VLSI data set as shown in Figure 3. Each node represents a viable position in the microprocessor wherein a transistor can be placed or through which a route can be passed.

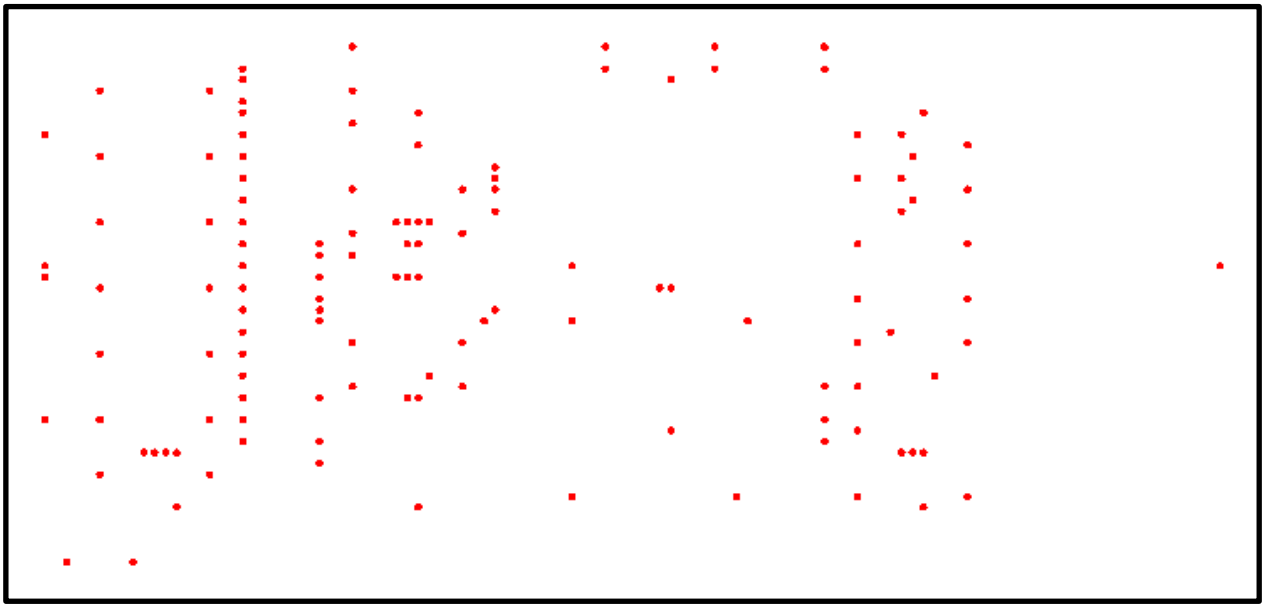


Figure 3 - VLSI Data Set (math.uwaterloo.ca, 2015)

Any node is accessible through a combination of other nodes and each node is connected to its immediate neighbour on both x and y planes. A source for a net is the starting point and a sink is the ending point. For example, in net A-B, node A is the source and node B is the sink. There may be other nodes in between them that make up the net. In addition, the three iterations of the problem used six sets of nets where the source and sink were randomly generated and are shown in Appendix B.

Model

Assumptions

Let there be a graph G containing the set of nodes and set of bidirectional edges (arcs) that connect the nodes. The maximum capacity of each edge is set to a constant which represents the number of nets any node can be part of. This is known as the amount of layers in the VLSI microchip and it is predetermined by the manufacturer. In addition, each edge comes with a predetermined value or cost of utilization.

Manufacturers strive to achieve a minimum certain chip thickness because additional layers incur extra cost. Thus working within a predetermined number of layers is a plausible scenario. Additionally node locations and their corresponding edges are determined in VLSI chip design in a process called routing where it determines all possible routes from point A to B by utilizing algorithms and heuristic solutions. Once routes are formed, costs are associated with each route based on variables such as data input, edge location, distance and edge material. The problem solved in this report is the next step in the process wherein optimal routes are determined based on costs and layer restrictions.

Formulation

Decision Variable

$x_{ij}^k = \{0,1\}$ *as the decision variable representing flow of electricity directed from node i to node j in net k*

Parameters

c_{ij}^k *as the cost of directing electricity from node i to node j in net k*

$I_{sink}^k(i) = \begin{cases} 1 \text{ if node i is the sink of net k} \\ 0 \text{ ow} \end{cases}$

$I_{source}^k(i) \begin{cases} 1 \text{ if node i is the source of net k} \\ 0 \text{ ow} \end{cases}$

$adj(i)$ *as the set of the indices of nodes connected to node i*

$D =$ *maximum capacity of each edge*

Program

$$\text{Min} \sum_k \sum_i \sum_{j \in adj(i)} c_{ij}^k x_{ij}^k$$

st

in = out: $$\sum_{j \in adj(i)} x_{ji}^k + I_{source}^k(i) = \sum_{j \in adj(i)} x_{ij}^k + I_{sink}^k(i) \quad \forall i,k$$

edge capacity: $$\sum_k x_{ab}^k + x_{ba}^k \leq D \quad \forall \text{ pairs of adjacent nodes } a,b$$

All x_{ij}^k binary

Explanation

The objective function is minimizing the total cost of routes for a given set of nets in completing the connections required by the microprocessor. The first constraint labelled as "in=out" are constraints that ensure that the flow in and out of a node is equal. However, for sink nodes, since they are the ones where the flow would terminate, the "out" would be given a value of 1. Similarly with source nodes, an artificial "in" would be given a value of 1 as that where the flow begins.

For constraint two labelled as "edge capacity", it specifies that a given edge cannot be part of more than D nets. This simulates and constrains the number of layers in a microprocessor that are available for routing.

The last constraint is a binary constraint that restricts all variables to binary values of 0 or 1. This is due to the fact that the model is a 0-1 integer linear program. Each path or edge is considered either "on" (1), or "off" (0).

Strengths and Weaknesses

One of the main strengths of the model is that it will always be able to find a feasible solution if each pair of points can be connected in one way or another. The model was effective in finding a feasible solution using a greedy method of minimizing the total cost with the given number of layers. If it is not feasible with the number of layers that was specified, the "edge capacity" constraint can be relaxed in order to find how many are required.

However, even though the model is quite efficient in finding solutions, there are a few limitations. Firstly, the program is constrained to the number of layers that was specified by the user and it does not solve exactly how many layers were needed. Secondly, there are a number of assumptions and data required such as all the possible paths of the nodes and the costs associated with each path.

Extensions

The solution currently only outputs an answer to the user if there are a feasible set of paths given the layering constraint. A viable extension of this solution would be to add in a heuristic wherein the number of allowable layers is increased if a feasible solution cannot be reached within the given layers. The maximum amount of layers required should be equal to the amount of nets required. This added heuristic would let the system determine the amount of layers required to implement an optimal path given that edge costs stay the same.

A second extension to the proposed solution would be to extend it past 6 possible nets. In the current solution, constraints have only been created for 6 nets or less but realistically, VLSI problems can extend far past 6 nets. The problem was only formulated for a 6 nets because of the optimization capabilities of OpenSolver. Constraints and variables increase proportionally to each net added. If the problem grows substantially past 6 nets it is likely that a heuristic solution will have to be devised instead.

Solution(s)

Given the randomly generated net lists (see Appendix B) for each test and a maximum layer constraint of 2, we were able to find valid and optimized paths for each of our 3 tests. These paths only ever overlap twice at any edge in correspondence with the edge capacity constraint.

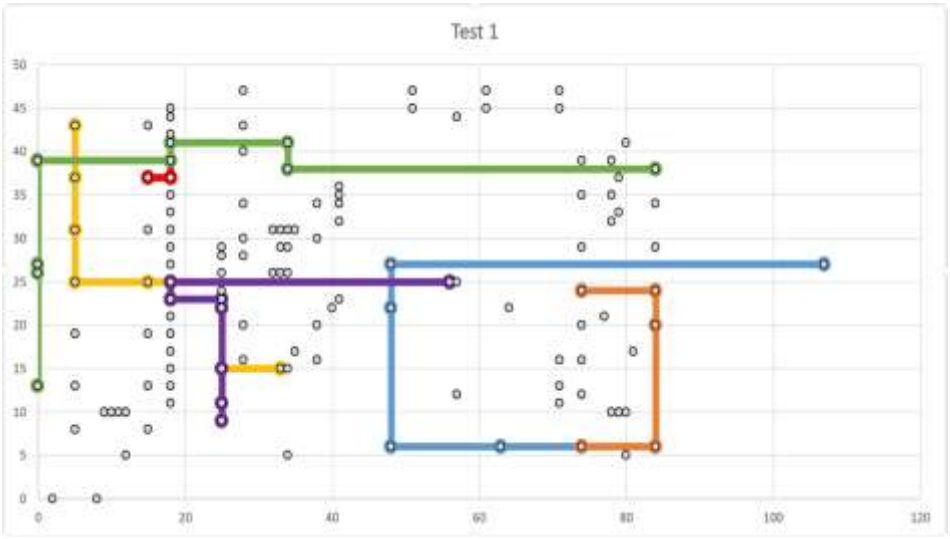


Figure 3 - Test 1 Output Plot

As seen in figure 3, nets may overlap several other nets as in the case of the purple net. However for any given edge from point A to point B, there may only be 2 nets. In the Test 1 figure, the purple net overlaps with both the blue net and the yellow net but only overlaps with at most one at a time for any of its given edges.

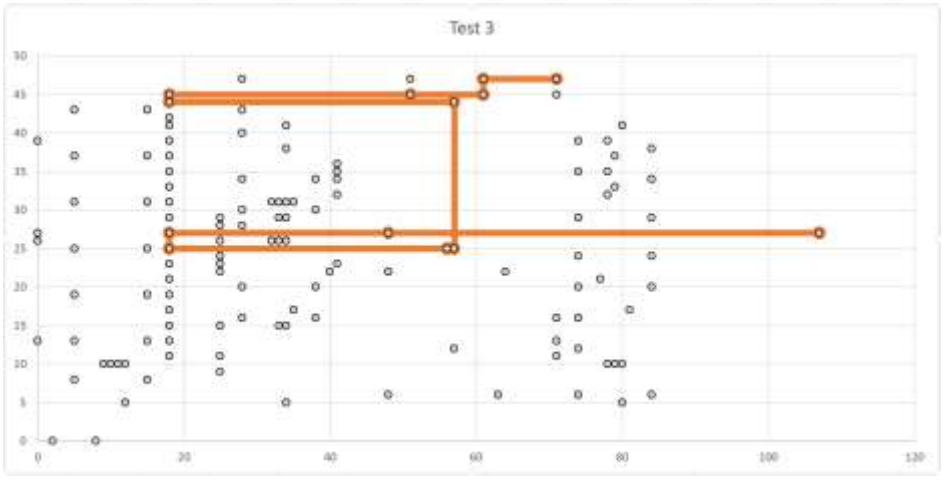


Figure 4 - Test 3 Route (71, 47) - (107, 27) Optimal

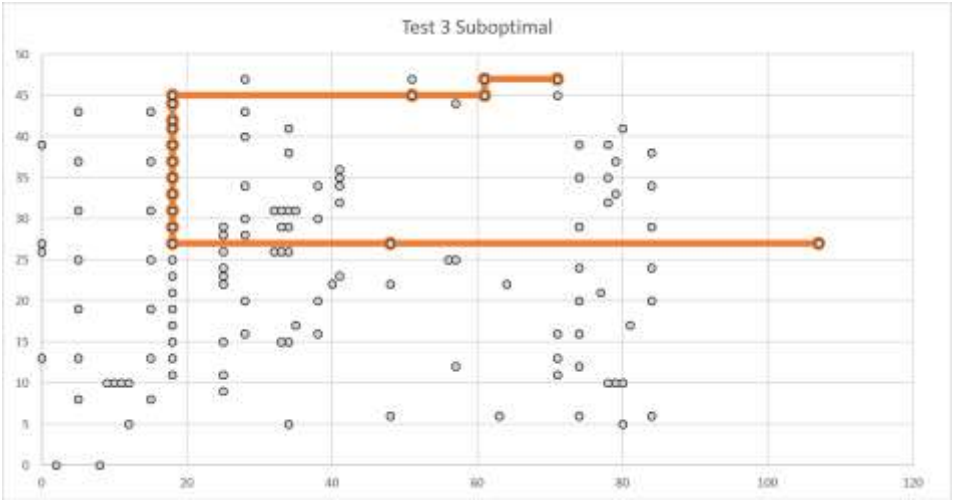


Figure 5 - Test 3 Route (71, 47) - (107, 27) Sub-optimal

Based on solution results, the optimization program is also capable of completing its objective of minimizing the total cost between all paths. The route for net (71, 47)-(107, 27) that was generated by test 3 is displayed in Figure 4. Though the route looks more complicated than that created in Figure 5, it is much more optimal because paths for the latter net have a much higher total cost. The optimized solution has a total net cost of only 90 compared to a net cost of 167 with the latter manually created path.

References

[**1**] Saxena, P., & Shelar, R. (2007). Routing congestion in VLSI circuits estimation and optimization. New York: Springer.

[**2**] Nahabedian, A. (2010, April). A Primal-Dual Approximation Algorithm for the Concurrent Flow Problem. Retrieved November 23rd, 2015: https://www.wpi.edu/Pubs/ETD/Available/etd-042910-160853/unrestricted/nahabedian.pdf

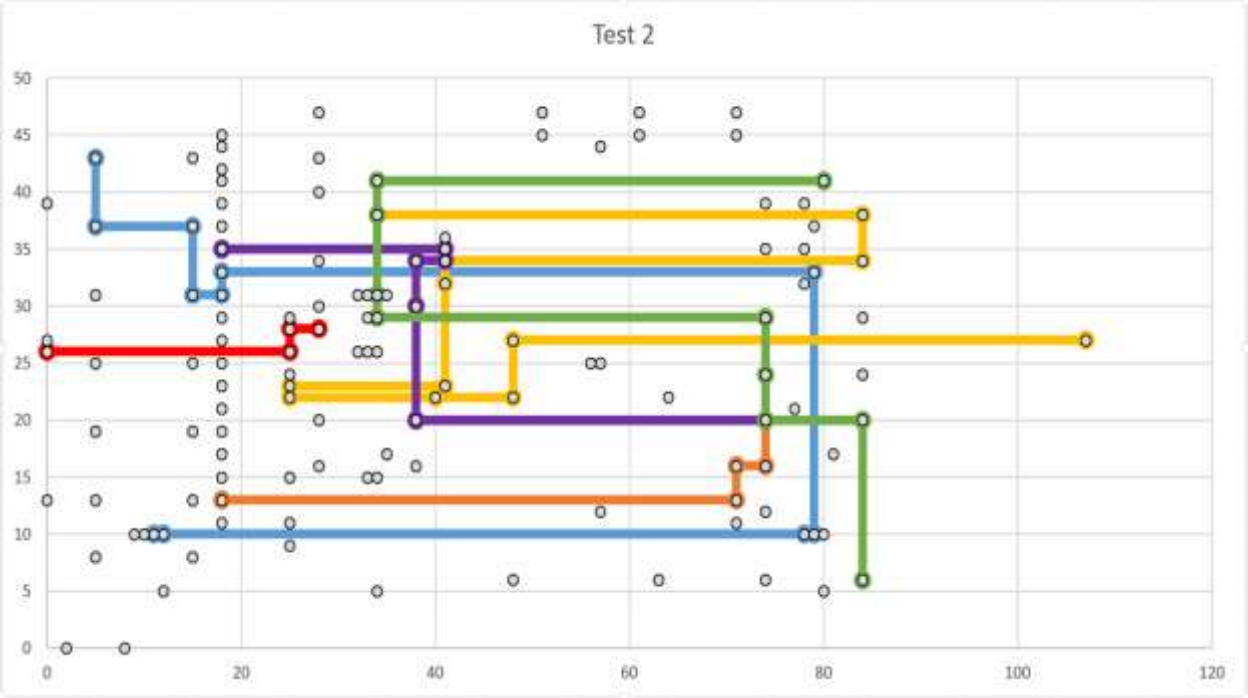
[**3**] uWaterloo. (2013, May).VLSI Data Sets. Retrieved November 23rd, 2015: http://www.math.uwaterloo.ca/tsp/vlsi/

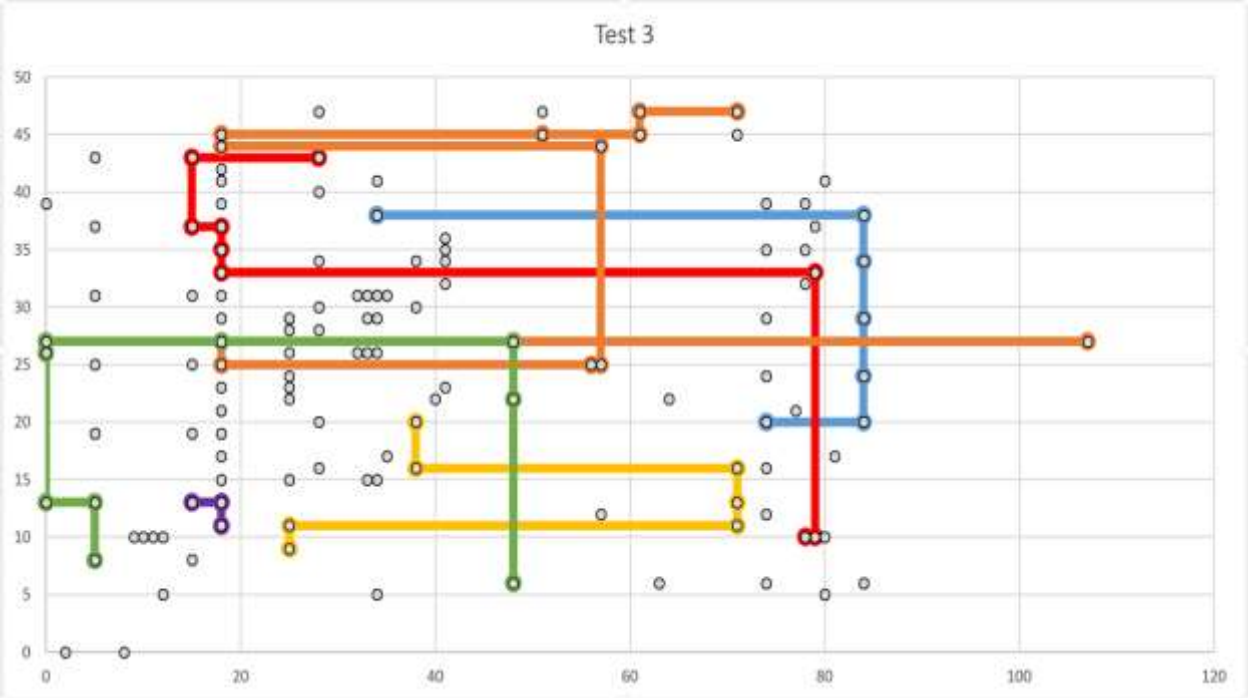
[**4**] SAS. (2012, August). Multicommodity Flow Problem. Retrieved November 23rd, 2015: http://support.sas.com/documentation/cdl/en/ormpug/65554/HTML/default/viewer.htm#ormpug_decomp_examples01.htm

[**5**] Lim, Sung Kyu (2008). Practical Problems in VLSI Physical Design Automation. New York: Springer.

Appendices

Appendix A - Test 2 and 3 Results





Appendix B - List of Nets

(Starting x, Starting y) - (Ending x, Ending y)

	Test 1	Test 2	Test 3
Net 1	(107,26) - (84,6)	(5,43) - (11,10)	(74,20) - (34,38)
Net 2	(74,6) - (74,24)	(18,13) - (74,20)	(78,10) - (28,43)
Net 3	(15,37) - (0,39)	(28,28) - (0,26)	(107,27) - (71,47)
Net 4	(5,34) - (33,15)	(34,38) - (107,27)	(25,9) - (38,20)
Net 5	(25,9) - (56,25)	(74,20) - (18,35)	(15,13) - (18,11)
Net 6	(0,13) - (84,38)	(84,6) - (80,41)	(5,8) - (48,6)

Appendix C - Model Excel File

See attached Excel File.