Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

CAS-2019-4518888

M.Sc. Thesis

Area Minimization of the DTB Multiplexer

Reynaldi Cangga Putra B.Sc.

Abstract

The DTB Multiplexer is a component within an NXP chip called the BAP3. DTB Multiplexer provides a testing functionality for the chip. This component is purely combinational, and requires no clock, however this makes the component wiring-costly. This high wiring requirement leads to the area constraint imposed by the wiring demand rather than cell area, and this also leads to the DTB multiplexer reducing the placement area available for other modules. With this method, the wiring area is going to be estimated as the amount of congestion, which would cause detour in the design which results in extra wiring. Here, the DTB multiplexer is placed by external method instead of using the place and route tools usually used by the design team. Instead, the placement is done on MATLAB which is later ported to the place and route tools using script. The placement algorithm implemented in MATLAB is primarily based on two algorithm, Dplace for initial preplacement, which in turn utilizes diffusion preplacement algorithm, and modified C-ECOP for the congestion reduction. More detailed congestion estimation done by using an additional routing estimation algorithm which is based on One-Steiner routing algorithm. The result indicates that the modified C-ECOP can be used to reduce congestion, thus wiring area when paired with a good initial placement algorithm, but the initial placement algorithm and detailed congestion estimation algorithm with One-Steiner could be further improved, and further work is needed to integrate the result with commercial place and route tools.

Keywords: **placement, wiring area, congestion, overflow, quadratic placement, wirelength**

Area Minimization of the DTB Multiplexer
A Chip Component with High Wire Density and Congestion

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Computer Engineering

by

Reynaldi Cangga Putra B.Sc.
born in Bandung, Indonesia

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Microelectronics & Computer Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Area Minimization of the DTB Multiplexer"** by **Reynaldi Cangga Putra B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: March 29, 2019

Chairman: _____
prof.dr.ir. A.J. van der Veen

Advisors: _____
prof. dr. ir. Rene van Leuken

_____
ir. Paul Wielage

Committee Members: _____
prof. dr. ir. Arjan van Genderen

_____
ir. Arjan Leeuwenburgh,

Abstract

The DTB Multiplexer is a component within an NXP chip called the BAP3. DTB Multiplexer provides a testing functionality for the chip. This component is purely combinational, and requires no clock, however this makes the component wiring-costly. This high wiring requirement leads to the area constraint imposed by the wiring demand rather than cell area, and this also leads to the DTB multiplexer reducing the placement area available for other modules. With this method, the wiring area is going to be estimated as the amount of congestion, which would cause detour in the design which results in extra wiring. Here, the DTB multiplexer is placed by external method instead of using the place and route tools usually used by the design team. Instead, the placement is done on MATLAB which is later ported to the place and route tools using script. The placement algorithm implemented in MATLAB is primarily based on two algorithm, Dplace for initial preplacement, which in turn utilizes diffusion preplacement algorithm, and modified C-ECOP for the congestion reduction. More detailed congestion estimation done by using an additional routing estimation algorithm which is based on One-Steiner routing algorithm. The result indicates that the modified C-ECOP can be used to reduce congestion, thus wiring area when paired with a good initial placement algorithm, but the initial placement algorithm and detailed congestion estimation algorithm with One-Steiner could be further improved, and further work is needed to integrate the result with commercial place and route tools.

Keywords: **placement, wiring area, congestion, overflow, quadratic placement, wirelength**

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor Prof. dr. Ir. Rene van Leuken for the continuous support of my master thesis study and research, for his patience and generosity such that I was able to complete my thesis despite my unsuccessful effort in the first year.

I would also like to thank my NXP supervisor Ir. Paul Wielage for his patience guidance while I was in NXP such that I gained a lot more knowledge in ASIC design, again despite the previously unsuccessful effort.

Besides my advisors, I would like to thank Ir. Arjan Leeuwenburg. for providing me the chance to do my internship at NXP for my thesis, and thoroughly helped me out during the worst times during my thesis, provided me a new outlook, and gave me a second chance to complete my thesis at NXP Semiconductos.

I would also like to thank my thesis committee member including Prof. dr. Ir. Arjan van Genderen for providing me the opportunity to defend my thesis by giving me questions and comments.

Last but not the least, I would like to thank my parents, my family, and my friends for supporting me, financially and emotionally throughout my entire study program in Delft University of Technology,

Reynaldi Cangga Putra B.Sc.
Delft, The Netherlands
March 29, 2019

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Today's automotive industry products such as cars are highly computerized, and the chips made for automotive industries require a high degree of robustness and reliability, as failure of any of a chip's components might result in injury or even death of the occupants of a vehicle equipped with said chip. One of the heavily computerized components in a car is its entertainment systems. Entertainment systems use power amplifier to give the desired level of audio output. These amplifiers are controlled by digital chips as well, and in order to achieve the desired level of reliability, the chips must possess a measure of testability and diagnostics. In the event of failure, the failure must be identified through root cause analysis thus some measures are required to enable root cause analysis.

Root cause analysis, can be done invasively or non-invasively. To do this invasively would mean to cut-open the chip itself, which takes a lot of time, while delivering the root cause analysis result is often time-critical. It is desirable to have a means to pinpoint the point of failure quickly and non-invasively, similar to the idea of the implementation of built-in-self-test capability. In the BAP3 chip design, one of the implementations of this measure for the digital section of the chip is the use of Digital Test Bus (DTB). Unfortunately, the use of DTB will incur additional costs, particularly the die area wire density.

1.2 Goal

In many chips, it is desirable to have a capability to perform tests in the chip as part of the debugging purposes and root cause analysis. The more test and diagnostics measures we can implement to the chip, the more reliable the chip will be. However, it will also incur more resource cost for the chip, and minimizing test and diagnostics cost while maintaining as much testability as possible is a balancing act and is subject to research.

There are two methods to implementing the test and diagnostics measures: Invasive and non-invasive. Invasive methods involve prying open the chip, thereby destroying it, while non-invasive uses hardware built into the chip. These measures however, are not always interchangeable, and some of them needs to complement each other, and for running a test and diagnostics of design behavior, there is a need for non-invasive measure built into the chip.

In our case, this built-in non-invasive test and diagnostics measures has created a significant resource requirement problem for the design of the chip, and we would like to minimize the cost of our test and diagnostics measure implementation on our chip.

The obvious advantage to doing this of course, is the reduction of the required size for the floorplan, thus increasing production yield. If the floorplan has been designed, it is no longer possible to increase the yield per wafer, however, the benefit that comes with saving area requirement can still manifest elsewhere.

With reduced resource requirement for this, we can expand the capability of the test and diagnostics measure, increasing the observability of the chip. With smaller area required by the test and diagnostics block, we can also allocate more area to other modules, and we can implement more reliability measures on other blocks, such as more decap cells, increasing reliability and potentially allowing higher clock frequency.

This thesis will explore methods to reduce the cost imparted by the DTB on the chip. This thesis consist of 5 main sections: Introduction, Literature Research, Methodology & Implementation, Results, and Conclusions.

1.3 Brief Hardware Description

The BAP3 is a multi-channel audio power amplifier with digital input and diagnostics capabilities for automotive usage. The chip contains both analog and digital elements. The middle section of the chip contains the digital section of the chip. Digital section of the chip in Figure 1.3 shows the layout of the original BAP3 chip. The DTB multiplexer is marked by the blue ellipse.

1.3.1 Digital Section of BAP3

The BAP3 chip can be broken down to three main components: Digital section, Analog Mixed Signal Section, and Analog section, where the amplifiers are located, and are built in several powerstages.

Our main interest lies in the digital section of the chip. The digital section of the chip consists of many modules controlling the Analog and Analog Mixed-Signal IPs in the chip. Each of the analog powerstages are controlled by a single control loop in the digital section of the chip which, again, consists of several submodules.

These control loops are connected to a manager which can be programmed to control the chip as whole. The manager can also determine the operating mode of the chip, and as part of the testability measure, the manager incorporate a test mode in order to diagnose the fault within the chip.

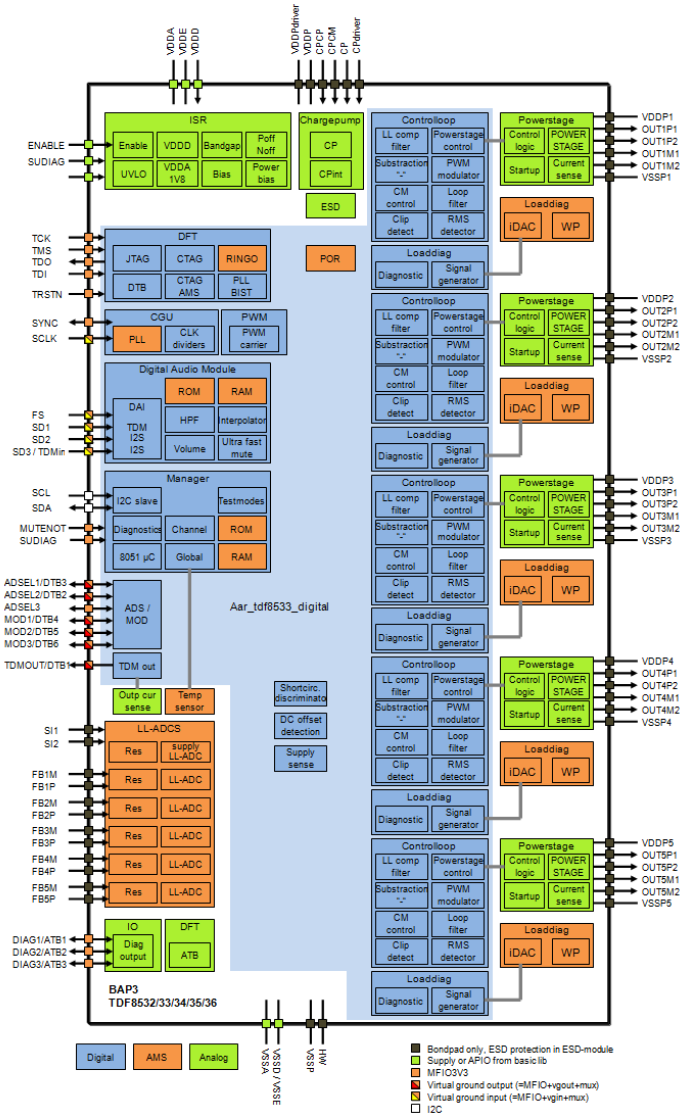


Figure 1.1: BAP3 circuit blocks illustration[29]

1.3.1.1 Implementation of Debugging Functionality

In a test mode, the chip has to be able to diagnose the fault within the design. We would like to be able to pinpoint defects in our chips to be able to debug the chip in case of a failure in our Root Cause Analysis. The measure is implemented in the form of test bus. There are two kind of test buses in the chip: Digital Test Bus and Analog Test Bus, as the name implies implemented as measure as test and debugging for the analog section and the digital section of the chip respectively.

The focus of this thesis is the digital test bus. The test bus has to be able to provide input for every module meant to be tested and output the result in real time. This requires a combinational asynchronous module that is connected to every testable module.

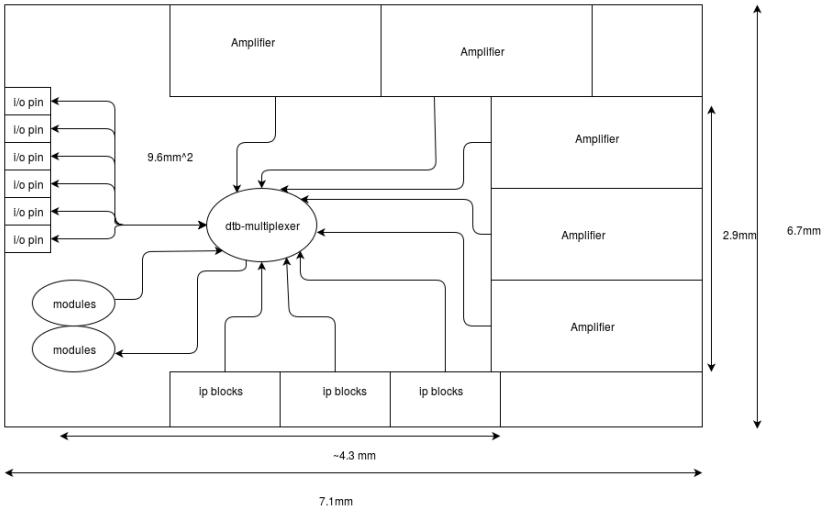


Figure 1.2: BAP3 simplified floorplan diagram. The powerstages are arranged together at the top and right side. These powerstages receives input from the digital section of the chip. There are some digital IP at the bottom section of the chip, namely several ADCs and ISR.

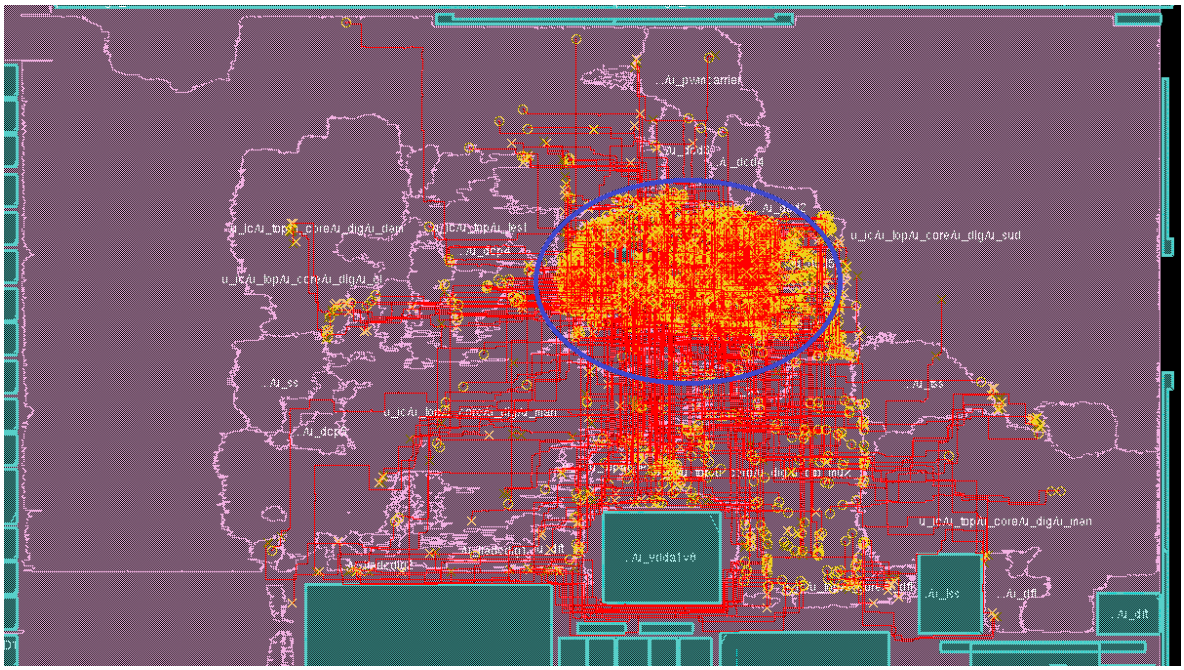


Figure 1.3: The DTB multiplexer contain input and output pins. Here the pins can be seen as the yellow o and x markers within the DTB modules, with x being input for the DTB and o being output of the DTB, with red lines representing net connections. o and x markers outside the DTB represents driver pins or pins driven by the DTB respectively.

- **DTB Multiplexer** For this application, it is desired to have a real-time ability to test the input of the modules in the chip, so we have a complete Data-Test Bus (DTB) that feeds input and takes output to and from each of the desired modules

Process	ABCD9-Power - using the STI=1.0um and STI=2.0um HVNMOST devices - C14_SC_7_CORE_H - C14_SC_15_CORE_GO2
Die size	7.102 * 6.677 mm^2 for 5 channel device (47.42 mm^2)
Complexity	9.6 mm2 digital NESRAM 6k UHDROM 48k 16 mm2 analog 20 mm2 High voltage and Power
Characteristics	Max 50V pin voltage (Chargepump) Max 10A pin currents (Class-D outputs) Frequency ~50MHz main clock frequency, ~100MHz (locally)
Package	HLQFP100 (SOT470...), dedicated leadframe - enlarged cavity - package lead 100 connected to exposed diepad - Chipcoat
Bonding	50um / 30um (mixed bonding) copper wire - separate bond and probe pads - using bondpads with 130um / 90um opening - No Bond On Active
Thermal aspects	Max 175°C die temperature Booster : 20W maximum peak dissipation (5 channels) Headunit : 7W maximum peak dissipation (5 channels) for 7% of lifetime acc. Mission profile
Testplatform	LTX-C Vertical probing VP2.1

Table 1.1: Portion of the datasheet of the BAP3 chip.[29]

in the chip, this data test bus serves as an alternate input or output to or from the intended modules. The relationship between the test buses and the test buses are depicted in Figure 1.4.

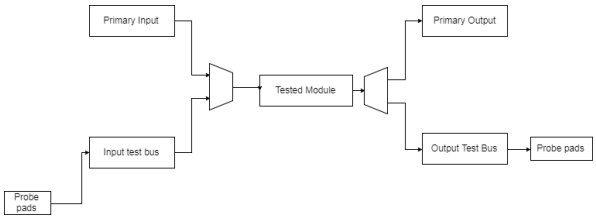


Figure 1.4: Simplified diagram of DTB multiplexer data flow.

Depending on the number of modules we want to test, this relationship might get significantly more complicated as the number of inputs and outputs increases accordingly.

1.4 Resources needed for placement and routing

Looking at the chip from another perspective, we have to keep in mind that a chip is also constrained by the size of the die it is etched on. The die of the chip is $47mm^2$ and out of this, the area allocated for the digital core of the chip is around $7mm^2$. The BAP3 uses 5 routing layers, 2 horizontal and 2 vertical layers with last 1 layer used for power distribution. And the chip needs to fit with all these constraints. We are also not allowed to alter the floorplan of the chip.

1.4.1 Resource used by routing and how it affects area

While routing is not done on the substrate level, its presence above the cells does affect cell placement. Each cell has a predetermined position of pins to be used as contacts. The presence of such pins might affect the routability of the lowest layer of the metal, and the effect will propagate to the metal on higher layers as well. This in turn will affect the viability of placing other cells close to the aforementioned cell. A cell with plenty of nets above it might not necessarily need a lot of substrate area to occupy, but other cells might not be able to be placed close to it due to the presence of many nets routed above it.

1.5 Problem description

Digital Test Bus is an important component contributing to reliability in BAP3 chips as explained earlier. But it is important to realize that this component does not contribute to the actual functionality of the chip itself. In a way the Digital Test Bus can be seen as merely an overhead, which is a dead weight in terms of actual functionality and the resource taken by the chip, thus we want to minimize the resource required by the Digital Test Bus as much as possible.

The Digital Test Bus can be broken down into two modules. The control module and the observe module. The Digital Test Bus is highly problematic as it involves long wiring and major congestion points. The cell density of this module is much lower than other modules thus we want to address the constraint imposed by wiring area, so we can increase the cell density, thus making better use of the placement area and reducing the overall size of the module.

We first need to find a metric that can quantify the possible area savings from an improved design. This is important because we are not allowed to alter the floorplan in any shape or form.

The issue with the DTB multiplexer is, due to the nature of its combinational design involving thousands of multiplexer with several hundred inputs, the design is wiring-intensive. The consequence to this is that DTB requires larger-than-desirable routing area, whilst its cell placement are utilization is vastly below average (around

20-30% instead of 60+% average) which reduces the area available for other modules.

Our objective is then is to reduce the area requirement imposed by the routing requirement so that it more closely matches the area needed by the cells alone, thus improving utilization and freeing up additional resources to be used by other modules in the chip.

1.6 Brief DTB Hardware description

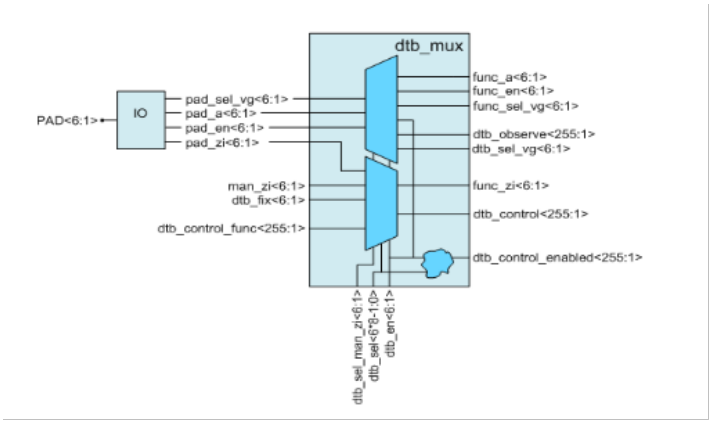


Figure 1.5: DTB Multiplexer block diagram.[29]

The DTB multiplexer consist of two primary modules: Control Module and Observe module. As the name implies, during the test mode, Control module serves as the module that provides the input to each module that we want to test, overriding the original input to the module under test, while the Observe module is used to observe the output of the module that we want to test. In the current BAP3 chip, there are six sets of control and observe modules so that it is possible to monitor or control 6 modules concurrently. These modules receive input or gives output via 6 IO pads that are connected to both modules.

Each DTB multiplexer pads can be controlled by a 9 bit selection-line, as there are six sets of DTB, the whole selection line bus is 54 bit wide, with each dtb corresponds to dtb_sel<0:8>, dtb_sel<9:17> etc. If the selection bus, dtb_sel, of a pin is set to all zero, then the pin set to functional mode where functional signals, are routed straight to the output pads.

Both the pads and observe and control modules are equipped with enable signals. 6 bits for the pads, and 9 bits for both control and observe module.

1.6.1 The Observe Module

The structural description of the observe module is in essence a multiplexer tree. Each of the 418 input signals come from different modules in the digital part of the chip and

multiplexed into a single output at the end.

To observe the signal of a pin, the enable signal of the corresponding pin which is one of the six dtb_en pins must first be set to 1, and dtb_sel must have an input other than all zeros. The desired signal can then be observed in the output pad of the selected DTB.

To observe high speed signals a virtual ground output pad can be selected, with dtb_sel_vg, instead of the normal MFIO pad. A maximum of 6 signals can be observed and/or controlled in parallel. A functional input signal is tied to a fixed value, via dtb_fix, if the functional pin is used as debug output pin.

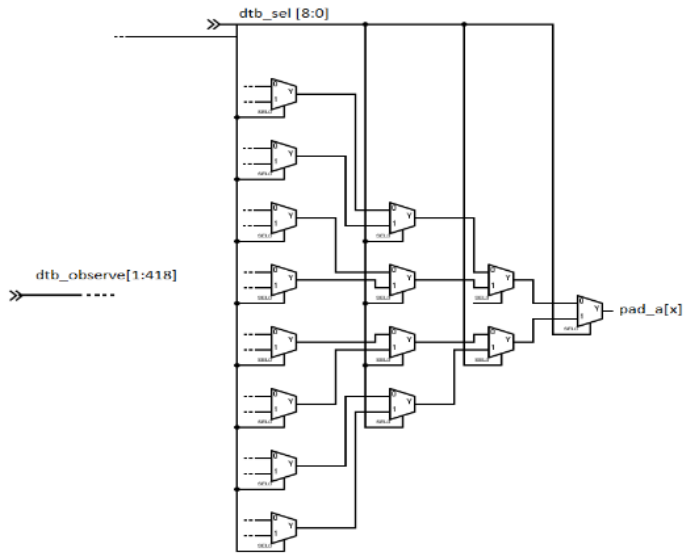


Figure 1.6: DTB Multiplexer observe module block diagram.

1.6.2 The Control Module

The control module enables overriding an input to certain chip component under test to observe the component behavior in real-time.

If the selection bus for a pin does not equal zero and this pin is set to debug input (for example, for pad 6, dtb_en<6> =1), the selected debug signal, dtb_control<selection>, can be controlled by a debug pin. If the corresponding dtb_sel_man_zi bit is enabled for this pin the manager can control the selected debug pin.

The selection between normal functional control and control from the pin can be done inside the dtb_mux, or inside the functional block. If the debug mux is in the functional block, signal dtb_control_enabled<selection> must be used to control this mux.

Literature Research 2

2.1 ASIC EDA workflow

As the number of transistors goes up, it is becoming less and less practical to design a chip manually at transistor level. At VLSI level, Electronic Design Automation was devised to tackle this issue. The Electronic Design Automation or EDA follows a top-down procedure consisting of several steps which can be seen in Figure 2.1 [1].

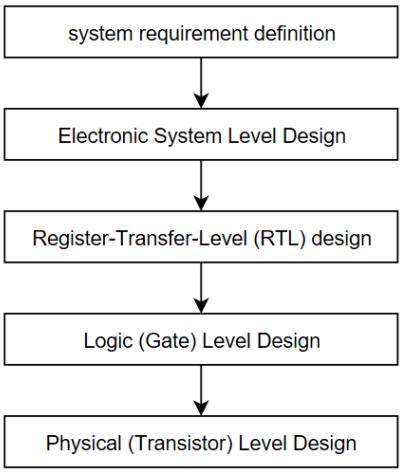


Figure 2.1: ASIC EDA Flowchart

- **System requirement definition**
  System requirement definition involves collecting data from customers and business plan alike to come up with the broad solution to the issue presented.

- **Electronic System Level**
  Electronic System Level (ESL) design involves defining which component of the solution belongs to the hardware or the software, and in addition to this, ESL involves performing simulations and cost estimations along with design-space exploration in order to come up with a solid knowledge basis for making informed design decisions.

- **Register-Transfer-Level**
  Register-Transfer-Level (RTL) design involves translating the hardware component defined in ESL design stage into a design language that can be processed by the EDA tools. RTL design level has two possible input: structural description of a design, or behavioral description of a design

- Structural description of a design describes the interconnection between the component of a design. The result of this translation is known as a netlist.
- Behavioral description of a design describes the reaction of a design given a specific input or stimuli. This description is given in an algorithmic way. RTL translation of this description would describe the behavior of the design in a flow, clock cycle by clock cycle, this way, EDA tools are able to understand the behavioral description of the design.

- **Logical Level Design**
  Logical Level Design involves translating the RTL description into logic gates. Design of this level is mostly done automatically by EDA tools which use their own specific algorithms. This level consists of two steps:

  - **Synthesis**
    This process translates RTL description into gate description that is independent of the technology used in the physical implementation.
  - **Mapping**
    This process maps the gates described in synthesis results into corresponding cells, or components that are available in the technology used, and would match the functionality of the gates in question.

- **Physical Design Level**
  Physical Design Level is a really important step in EDA flow as this highly influences the area and power requirement of the final design, as well as the overall performance. A great amount of research has been dedicated into this aspect of EDA due to its importance. This level can be further broken down into several steps, as can be seen in Figure 2.2.

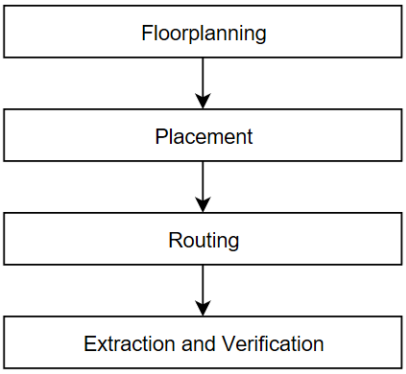


Figure 2.2: Physical Design Steps Flowchart

  - **Floorplanning**
    Floorplanning involves partitioning a into several areas of which only certain modules can be placed within. Floorplanning gives the designer an early feedback for chip area, delay, and congestion estimates.

- **Placement**
  Placement involves placing components on the chips substrate. This is an extremely important step as it heavily affects routability, performance, heat distribution and power requirement.
- **Routing**
  Routing process determines the exact path the wire connections (nets) between cells take on the metal layers above the placed cells. Routing must abide to a set of rules given by foundries in order for the chip to be able to be manufactured. A well-routed chip must have 100% routability in the final result, as well has minimizing wirelength and meeting timing requirements.
- **Extraction and Verification**
  Extraction and Verification involves evaluating the chip based on the given design rules and checking its full functionality [1].

2.2 Scope of the discussion

In order to achieve an acceptable result within the given time limit, we need to focus on only select aspects of EDA design flow. We will not delve deeper into ESL design level, nor the System Requirement Definition, as these pertain to the functionality of the chip, and our effort is geared towards minimizing our design objective without altering the functionality of the chip itself.

We will also not delve deeper into Logic-Level Design step. Although altering the gate structure of the BAP3 might produce a better result, most of the synthesis algorithm that has been researched pertain to timing and cell area optimization, in addition to this, the combinational nature of the structure of the DTB multiplexer leaves little room to be altered as seen on Figure 1.6, thus the time available should be better spent researching other aspects that can be more easily improved.

We will thus focus on the Physical design level of the EDA flow. However we will not delve deeper into Floorplanning level as the floorplan of the BAP3 chip is already fixed. We will also not focus on routing as the routing algorithm used by EDA tools are proprietary and we dont have any means to alter the algorithm. However, we will use some aspect of the routing process in order to improve our effort in the Placement stage, which is where our effort will be primarily focused on, as placement and routing are heavily intertwined with each other.

2.3 Placement

Placement is an important step in the ASIC VLSI design flow. Placement heavily affects both the performance and cost of the chip. A chip with good placement will have shorter interconnect thus lowering delay and increasing possible clock speed. A good placement also improves the routability of a chip, which means we can increase the number of cells contained in a single chip, improving yield. Routability

is the most important metric here as will be obvious later. Furthermore, as mentioned before, placement also affects heat distribution and power consumption of a chip.

Placement can be divided into 3 steps: Global placement, Legalization, and Detailed Placement [1].

- **Global placement**
  In global placement, cells are placed in a flexible manner, and the aim for the placement is to come up with an optimal rough placement of the cells within the chip. As such, the wirelength model used are correspondingly simpler to improve efficiency of this process, thus the resulting placement result might not be immediately routable, possibly having design rule violations such as overlapping cells.

- **Legalization**
  Legalization makes minute adjustment to the position of placed cells or modules in order to meet Design Rule Checks. Legalization does not take cost functions minimization into account, it is only geared towards meeting design constraints.

- **Detailed placement**
  Detailed placement is mostly an iterative process limited to cells or modules on a specific region. The cells and modules inside this local region are further perturbed to achieve optimization while maintaining the solutions within design constraints. All of these are done while cells and modules on other regions are kept fixed.

2.3.1 Placement Objectives

Placement as a process requires some metric in order to evaluate the quality of the process. There are several methods to evaluate the quality of the placement of a chip. Some of the more commonly used ones are total wirelength. Wirelength is always going to be an important placement objective regardless of the design goals of a chip. This is because wirelength optimization often (although not necessarily always) indirectly leads to better optimization of routability, performance, and power consumption [1].

2.3.1.1 Wirelength

Wirelength is commonly used as an objective for placement as it often leads to better optimization of other objective. Wirelength optimization often (though not necessarily always) results in better timing and less power consumption and better routability. A highly routable design will require less detours, and thus will result in less wiring area requirement. Over the years there have been several wirelength models devised, with each offering different advantages in terms of accuracy and computational complexity.

- **Half-Perimeter Wirelength**
  The simplest and least computationally intensive wirelength model, HPWL predicts that the wirelength of a net is simply half the total perimeter length of a box enclosing all the pins of a net. This prediction is accurate for nets with only

2 or 3 pins, but not very accurate for nets with larger number of pins as it tends to underestimate the wirelength of these nets.

- **Minimum Rectilinear Spanning Tree (MRST)**
  MRST predicts that the wirelength of a net is the sum of all shortest possible Manhattan distances between each pins (nodes) required to connect them all. It is more accurate than the HPWL, but it requires more complex computation, with the best achieved time complexity of $O(NlogN)$ [16]. MRST also still falls short in terms of accuracy. Like the HPWL, it is accurate for nets with two pins, but is inaccurate for 3-pin nets or more. Contrary to HPWL, this approach tends to overestimate net length.

- **Steiner Rectilinear Minimum Spanning Tree(SMRST)**
  This approach is derived from MRST. Instead of simply trying to estimate the Manhattan distance of the minimum wirelength needed to connect all the pins, this approach adds virtual pins (nodes), called Steiner nodes/points. This points, when placed in correct location, can trunk wires to connect different nodes instead of connecting the nodes in a roundabout way, which results in detours.

  More importantly, Steiner Rectilinear Minimum Spanning Tree (SMRST) is able to predict more accurately how nets are going to be routed through the bins in the chip. This way, we will have a better picture of how much routing resources will be needed by the connections between cells. This can be a powerful tool in predicting how the routing can contribute to the overall size of a module in a chip.

  The disadvantage of both the MRST and SMRST is that both metrics basically overlap with the routing domain in physical design stage. This means that, to calculate these metrics, one has to use the same techniques used in the routing stage, which means more computational complexity is going to be incorporated in the placement stage. For example, calculating SMRST using One-Steiner algorithm which will be presented in later section, requires a time complexity of $O(N^2logN)$ compared to $O(N)$[7] for HPWL which simply calculates the Manhattan distance of a net.

2.3.1.2 Routability

Routability is an objective that is very hard to quantify, as it depends on the routing algorithm used to route a design, and the resulting routing solution. A way to get a good measurement for this metric is to simply perform global routing, but this is too computationally expensive for early placement process, which often involves multiple iterations. By using MRST or SMRST, we can get a rough idea of how the eventual routing result will look like without having to run a full routing, and we can use it as the basis for our optimization effort.

Congestion is a metric that partially quantifies routability. Once we obtained a routing estimation result from MRST or SMRST, we can then evaluate how many wires cross any given area at once. If a particular area of a chip is subject to high amount of wire crossing beyond the available routing resources (metal layers) around

its vicinity, then the area is said to be congested. Congestion is measured in overflows in both vertical and horizontal directions. Overflow is the amount of excess nets that cannot be routed due to the lack of wiring resources. Congestion also determines the area requirement for wiring as detours caused by congestion will result in increased wiring area requirement.

A net between two cells that would nominally be crossing a congested area will be forced to take a detour around it. This complicates the routing effort and affects area requirement negatively. One possible approach to reduce congestion is by allocating white space on the floorplan. White space is a placement directives which gives lower module density cap to regions that are expected to see more wire crossings (more congestion). However allocating white space requires changes to the floorplan, thus another method is needed to reduce congestion.

2.3.1.3 Other Objectives

Aside from the two objectives mentioned above, placement also affects interconnect delay, power consumption, and heat distribution. Cell placement directly impacts the interconnect between cells and thus affecting delay, the DTB multiplexer however, is not subject to timing constraints, thus this aspect is can be overlooked. Besides, there are plenty of other factors affecting the delay as well, such as routing and driver strength, thus making accurate performance estimation impossible at placement stage.

2.3.2 Placement Steps

Placement is an NP-complete problem as demonstrated by Garey and Stockmeyer [17], that even placing two cells connected by a degree-2 net in a straight line is already an NP-complete problem, thus the available solutions to placement problem is usually heuristic. Devising a heuristic solution for the entire placement process is decidedly unwise, thus placement process is divided into several steps as to allow for better results and reducing the complexity of the process itself. Placement is divided into 3 steps: Global Placement, Legalization, and Detailed Placement.

2.3.2.1 Global Placement

Global placement is arguably the most important and influential step in the whole placement process. In global placement, the cost function of the placements objectives are minimized. To do this, global placement algorithm deals with the entire circuit, and is not constrained by more detailed constraints such as module overlaps. Global placement can be roughly divided into 3 technique classifications: Stochastic placers, bipartitioning placer, and analytical placer.

- **Stochastic Placers**
  This placement method uses randomized approach such as simulated annealing to prevent the placer from getting stuck in local minima. The placer starts with randomized cell placement, and moves the cell position based on simulated annealing. One of the most well-known example of this approach is Timberwolf. [25]

This method produces excellent results for smaller circuits, but is very inneficient for larger circuits.

- **Bipartitioning Placers**
  Bipartitioning placer recursively divides a chip into two, and the dividing line between the two new partition is called the cut line. How the chip is divided is determined by the cut cost. Cut cost is basically the number of connection that would cross the cut line and the placer will try to find a position for a cutline that will minimize the amount of connections crossing the cut line. This indirectly minimize the wirelength and congestion [26][27].

  This type of placers are generally seen as fast but simplistic, requiring less computation but producing less desirable results than stochastic placer or analytical placer. One example of this type of placer is Capo. Capo is optimized for congestion minimization but utilizes white space thus requires changes to the floorplan [4].

- **Analytical Placers**
  Analytical placement method attempts to describe the placement problem in an analytical way, such that it describes the cost function and constraints as equations which describe the cells and modules' position in a coordinate system. The cost function used is the wirelength, while the constraints are imposed by module overlap and other design rules specific to the design. To use analytical placement approach, however, a simple HPWL wirelength model is not applicable. This is because HPWL is not convex nor smooth, thus cannot be used to find minima. The non-overlapping constraint is also non-convex and not differentiable.

  A way to tackle this issue is by using wirelength model other than HPWL. The wirelength model used needs to be convex and differentiable. Analytical placers can thus be broadly be classified into two class, based on the wirelength model used: Quadratic Wirelength Placers and Non-linear Placement model [27][28], which uses to Log-Sum-Exp wirelength model [20].

  - **Quadratic Placers**
    Quadratic placers uses quadratic wirelength model as its cost function. As the cost function is quadratic, it is convex and differentiable, and can be reduced to a set of linear equations that can be efficiently solved by LP solvers. A disadvantage of using quadratic wirelength is that it frequently overpenalizes long wires, and this is not always desirable depending on the design objectives, such as when routability is preferred over performance.
  - **Non-Overlapping Constraints**
    Another problem to be tackled in building an analytical placer is satisfying overlap constraints. This is a more complex task than to find a convex differentiable wirelength model for the cost function. Many placers have their own unique way to address this problem and often more ingenuity can be found in this aspect of placement algorithm than the wirelength model aspect. There are several methods used to satisfy this constraints. One of most popular one is the *Move* and *Hold Forces*.

* **Move and Hold Forces**
  One method to satisfy non-overlapping constraint is by using Move and Hold Forces. Hold force is defined as a constant force that serves as the inertia of a cell that makes them less prone to be moved when the placement is redone.

  Move force on the other hand is meant to move and diffuse the module from dense area to less dense area. The strength of the force is dependent on the density of the region the modules are in. The denser the region, the stronger the move force is. This can be derived using Poisson distribution to model the density distribution [15].

  The Move and Hold Force however, can also be substituted with other means, such as Diffusion Process [6].

– **Net model for higher-pin nets in analytical placement**
  Analytical placers are naturally easy to implement for nets with only two pins, but for any net with more than 2 pin, a representative net model is required to transform the net into a set of two-pin connections that can be used by the minimization equations. The two basic net models that are often used are *clique* and *star* model. A combination of both is called the *hybrid* net model.

  * **clique model**
    In clique model, all the pins in a net are connected to each other, forming a complete subgraph. The weighting on each of the link must be set properly to represent the actual edges coming out of the pins of the net, for example, a n-pins net with weight w must adjust the weight of its constituent 2-pin nets to $n/(w-1)$ [10].

  * **star model**
    In Star model, one virtual node is created in the middle of all the pins. With a proper net weights, this can represent the same net as clique model [11][12]. The star model increases the number of nodes in the net representation, thus increasing the number of linear equations to solve, however, it eliminates a great number of the edges used to represent the net. This is especially useful in solving nets with especially large number of pins.

    Using a matrix representation of the linear equations, it can be seen that the matrix representation of star network is much sparser than the clique network representation. This can have favorable effect of reducing the runtime of a solver as linear-programming solver is often much more affected by how sparse the matrices are than the dimension of the matrices themselves.

  * **hybrid model**

Taking the equivalence of both models into consideration, a hybrid net model [12] uses the clique model for nets with 2 or 3 pins and star model for more pins. This serves as the basis of anchor cell in DPlace.

2.3.3 Dplace Global Placement Algorithm

Luo and Pan came up with a quadratic placement algorithm called DPlace [2]. Dplace is optimized for wirelength and overlap minimization, both of which positively contributes towards reducing wire density in our design.

2.3.3.1 Overview

As explained earlier, quadratic placement seeks to minimize the quadratic wirelength cost function. This however will result in significant amount of cell overlap. Most quadratic placers employs force in its placement to overcome this problem. This force is interpreted as adjustments in either the **A** matrix or **b** vector of the quadratic equation, which will be explained later in section 2.3.3. Force directed placement can be divided into two categories: Constant Forces placement and Fixed Point Forces placement.

Constant Forces placement only makes adjustments to the **b** vector of the quadratic equation. This is advantageous in terms of runtime complexity, as the **A** matrix only needs to be calculated once at the beginning. Constant Force placement however can be quite unstable depending on the A matrix, and can often produce a really large shift that push cells out of the chip bounds.

Fixed Point forces Placement addresses some of the problem with the Constant Forces placement by adding virtual fixed connection between cells, thus producing a more stable result, however, this method is both computationally intensive as it involves recalculating the **A** matrix every iteration, and if the virtual net requires careful weighting. If the virtual net is weighted too heavily, it will immobilize the cells connected to it, whereas if the net is weighted to lightly, it will basically produce similar result to Constant Forces placement with added computational cost

Like most other quadratic placement algorithm, Dplace uses quadratic wirelength model instead of the simpler HPWL as its primary cost function. Unlike most other quadratic placement algorithm, however, Dplace does not use force to address the overlapping issue, freeing it from possibly troublesome issue of net weighting. Instead, DPlace uses **diffusion preplacement** and **anchor cells**.

2.3.3.2 Quadratic Placement Method

As mentioned in the previous section, quadratic placers uses quadratic wirelength as a cost function for its placement. Quadratic Wirelength has been briefly discussed in section 2.3, but here we shall delve into the deeper details. Quadratic wirelength between cells i and j, is defined as $w_{ij}((x_i - x_j)^2 + (y_i - y_j)^2)$ where w_{ij} is the weight of

the net between i and j. To illustrate this, suppose we have a circuit consisting of two pads, two movable cells, and, and three two-pin nets, as seen on Figure 2.3

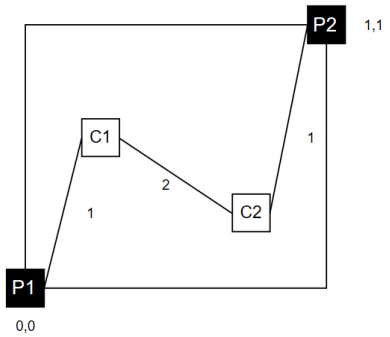


Figure 2.3: A circuit consisting of two pads, two movable cells, and, and three two-pin nets, connecting P1 to C1, C1 to C2, and C2 to P2. The net between C1 and C2 are weighted twice that of the other nets.

As there are three nets, we have three sets of quadratic equation:

$$\begin{aligned} Q_{wl_1} &= 1((x_{c1} - 0)^2 + (y_{c1} - 0)^2) \\ Q_{wl_2} &= 2((x_{c2} - x_{c1})^2 + (y_{c2} - y_{c1})^2) \\ Q_{wl_3} &= 1((x_{c2} - 1)^2 + (y_{c2} - 1)^2) \end{aligned} \tag{2.1}$$

It should be noticed that there are no terms containing x multiplied by y. This is means that the x terms and y terms can be separated, and regrouped according to their axes.

$$\begin{aligned} Q_{wl_X} &= 1(x_{c1} - 0)^2 + 2(x_{c2} - x_{c1})^2 + 1(x_{c2} - 1)^2 \\ Q_{wl_Y} &= 1(y_{c1} - 0)^2 + 2(y_{c2} - y_{c1})^2 + 1(y_{c2} - 1)^2 \end{aligned} \tag{2.2}$$

To find the minimum value of these equations, we simply need to find their derivatives, and thanks to the fact that there is no term containing x multiplied by y, this

can be done rather easily with partial derivatives

$$\begin{aligned}
(\partial Q_{wl_X})/(\partial x_{c1}) &= 2(x_{c1}) + 4(x_{c2} - x_{c1})(-1) + 0 \\
&= 6x_{c1} - 4x_{c2} = 3x_{c1} - 2x_{c2} \\
&= 0 \\
\\
(\partial Q_{wl_X})/(\partial x_{c2}) &= 0 + 4(x_{c2} - x_{c1})(1) + 2(x_{c2} - 1) \\
&= 6x_{c2} - 4x_{c1} - 2 = 3x_{c2} - 2x_{c1} - 1 \\
&= 0 \\
\\
(\partial Q_{wl_Y})/(\partial y_{c1}) &= 2(y_{c1}) + 4(y_{c2} - y_{c1})(-1) + 0 \\
&= 6y_{c1} - 4y_{c2} = 3y_{c1} - 2y_{c2} \\
&= 0 \\
\\
(\partial Q_{wl_Y})/(\partial y_{c2}) &= 0 + 4(y_{c2} - y_{c1})(1) + 2(y_{c2} - 1) \\
&= 6y_{c2} - 4y_{c1} - 2 = 3y_{c2} - 2y_{c1} - 1 \\
&= 0
\end{aligned} \tag{2.3}$$

These system of linear equations can be written in matrix form as

$$\begin{aligned}
\begin{bmatrix} 3 & -2 \\ -2 & 3 \end{bmatrix} \begin{bmatrix} x_{c1} \\ x_{c2} \end{bmatrix} &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
\begin{bmatrix} 3 & -2 \\ -2 & 3 \end{bmatrix} \begin{bmatrix} y_{c1} \\ y_{c2} \end{bmatrix} &= \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\end{aligned} \tag{2.4}$$

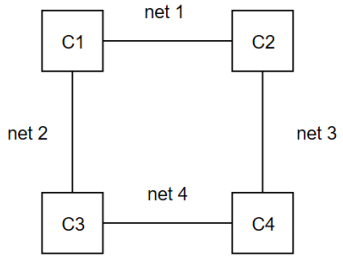
Solving these equations will yield $x_{c1} = y_{c1} = 0.4$ and $x_{c2} = y_{c2} = 0.6$. Instead of having to write all the quadratic equations for each net and evaluating the partial derivative of each equation, the matrix form above can actually be obtained from the Laplacian matrix of the circuit and the coordinates and weight of the pad connections. Thus the minimization function can very simply be written as

$$\mathbf{A}x = \mathbf{b_x} \; and \; \mathbf{A}y = \mathbf{b_y} \tag{2.5}$$

Where A is the Hessian(Laplacian) matrix obtained from the netlist, x and y are the cell position vector to be obtained on their respective axes. And b_x and b_y is the pad vector which is obtained by element-wise multiplication the pads position on their respective axes and the weight of their connection to movable cells.

To start, a netlist is turned into a hypergraph with cells represented by nodes and nets represented by edges. This can in turn be represented by Laplacian matrix such as the one in Figure 2.4

A problem with hypergraph and matrices is that matrices cannot represent a net with more than two degrees succinctly, thus multi-degree nets are often represented



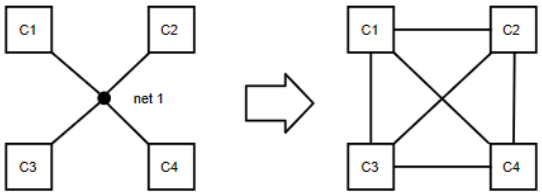
(a)

$$\begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & 0 & -1 \\ -1 & 0 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$

(b)

Figure 2.4: (a) four-cell circuit circuit with each cell connected to its adjacent neighbors and (b) its Laplacian matrix representation

by other net models that are briefly discussed in section 2.3. Traditionally, nets are represented by the clique model, whereas every cells connected to a single multi-degree nets are represented as being connected to one another, essentially forming a complete subgraph for the cells.



(a)

$$\begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

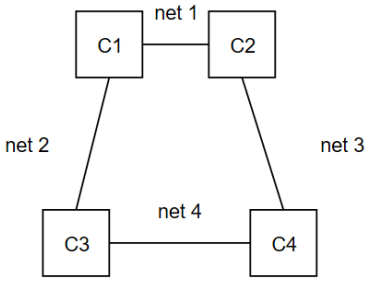
(b)

Figure 2.5: (a) A graph representation of a 4-cell circuit connected with a single degree-4 net, and its transformation to clique net model (b) its Laplacian matrix representation

- **Connection weighting**
  Important nets that have stringent timing constraint should be placed close together, and this can be represented by increasing the weight of the nets in question. This means the net has higher cost and will be minimized more extensively than nets with less weight.

- **Pad location vector**
  The $\mathbf{b}_x$ and $\mathbf{b}_y$ vectors are primarily obtained from the pad location on the circuit. They represent the constants in the derivative equations. Each row of the vector corresponds to a cell in the Hessian matrix, and we obtain the value of each row by multiplying all the x(for $\mathbf{b}_x$) and y(for $\mathbf{b}_y$) position of the pads connected to the cell corresponding to the row in the $\mathbf{b}$ vector with the weight of their connections. The example of this can be seen in Figure 2.7

For a circuit with a million cell, then we need an $\mathbf{A}_y$ matrix with a dimension of a million by a million as well as a $\mathbf{b}_y$ vector one million element long as well. This might

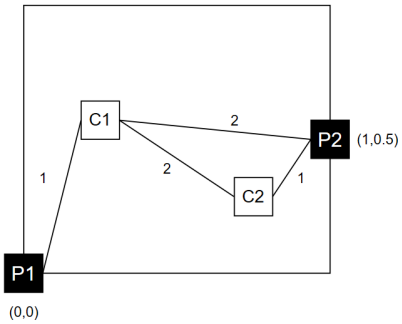


(a)

$$\begin{bmatrix} 3 & -2 & -1 & 0 \\ -2 & 3 & 0 & -1 \\ -1 & 0 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$

(b)

Figure 2.6: (a) Reduction of the length of a net with stringent timing constraint by increasing the weight of the net. (b) its Resulting Laplacian matrix representation



(a)

$$A = \begin{bmatrix} (2+2+1) & -2 \\ -2 & (2+1) \end{bmatrix}$$

$$B_x = \begin{bmatrix} 1(0)+2(1) \\ 1(1) \end{bmatrix}$$

$$B_y = \begin{bmatrix} 1(0)+2(0.5) \\ 1(1) \end{bmatrix}$$

(b)

Figure 2.7: (a) A four-cell circuit where more than a single cell is connected to one pad. (b) Weighting of the Laplacian matrix **A** and **b** vectors based on connections to the cell. The elements of both **b** vectors are obtained by summing the product of the weight of each net connected to a pad and the corresponding cell and the x or y position of each of the connected pad.

seem big, but the since a single net rarely has more than a few hundred connections at most, the resulting matrix will be sparse, thus despite solving for the position vector by means of matrix inversion should require a time complexity of $O(N^2 logN)$[30], the number of actual data processed not all the elements of the N by N matrix, but the number of nonzero elements in the matrix, and thus for computer solvers such as MATLAB, the complexity is scaled accordingly[31].

2.3.3.3 Diffusion Preplacement

- **Overview**
  Diffusion preplacement is a crucial component in Dplace placement algorithm. Diffusion Preplacement allow cells to spread out and reduce overlaps from the quadratic placement result. Diffusion preplacement was proposed by Ren et al. [5]. The diffusion preplacement is inspired by other processes common in semicon-

ductor industry such as dopant-diffusion process on chip substrate [19]. In diffusion process, materials from a high-concentration area moves to low-concentration areas.

Cells in diffusion moves based on the gradient of the cell concentration of their current position. A good analogy to this is how a ball rolls down a hill based on the slope of the steepness of its position. The diffusion process is expected to reach a steady-state once the particle distributions are evenly uniform [5].

A similar gradient-based cell-spreading method has also been tested by the developers of DPlace [21], and it yields similar result in terms of maintaining cell order, but DPlace uses this diffusion preplacement method instead.

- **Diffusion Process**
  The relationship between the particle concentration and the diffusion process can be described with the following partial differential equation

$$\partial d_{x,y}(t)/\partial t = D\nabla^2 d_{x,y}(t) \tag{2.6}$$

$d_{x,y}(t)y$ is the particle concentration at point x,y at time (t) D is the diffusivity constant which dictates the speed of the discussion. For the sake of brevity, D will be assumed to be 1 for the rest of the document

The above equation gives us a mean to calculate the instantaneous velocity of each cell in the diffusion process. The velocity vector of a cell in a 2-dimensional diffusion process at any given time is given by the following equations:

$$\begin{aligned} v^H_{x,y}(t) &= -((\partial d_{x,y}(t))/\partial x)/d_{x,y}(t) \\ v^V_{x,y}(t) &= -((\partial d_{x,y}(t))/\partial y)/d_{x,y}(t) \end{aligned} \tag{2.7}$$

Where vector $\mathbf{V_{x,y}}$ of a particle in a diffusion process consist of horizontal velocity element $v^H_{x,y}$ and vertical velocity element $v^H_{x,y}$.

From the equation above, it is then clear that we can calculate the position of each particle at any given time simply by integrating them in respect to . This is given in the following equation:

$$\begin{aligned} x(t) &= x(0) + \int_0^t v^H_{x(t'),y(t')}(t')dt' \\ y(t) &= y(0) + \int_0^t v^V_{x(t'),y(t')}(t')dt' \end{aligned} \tag{2.8}$$

The equations above are enough to model a diffusion process in a continuous environment, however, in practice, we will need to discretize these equation to a form that can be readily accepted by computer simulation such as MATLAB.

- **Discretization**

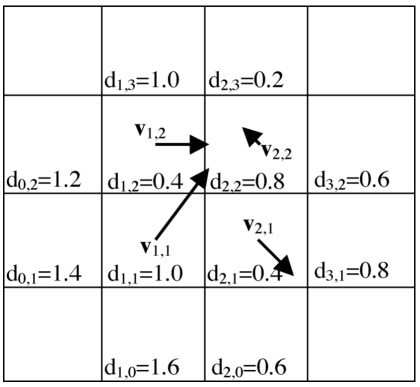


Figure 2.8: Bin velocity is determined by the gradient of the bin density.

A problem with this method is that multiple particles, or cells, can have different position inside a bin, and ideally, each should have different diffusion velocity. This method however, applies the same velocity vector to each and every cells contained in a single bin, which does not lend itself well to the objective of driving the particle away from each other, especially if the number of bins used are relatively low.

Furthermore, two cells that are positioned relatively close towards each other, but happen to be contained on different bins (both located close to a boundary between bins), could have very different velocity vector, and heavily alters the original ordering of the cells, which could give unforeseen results.

To address this problem, this algorithm employs a method called Cell-Velocity Interpolation. With this method, the velocities of the cells that fall within the same bin would be adjusted according to their locations relative to bin center position of the bins they are in, and the bin centers of the adjacent bins closest to the cell.

For the sake of the argument, we will assume all bins to have width and length 1. If a cell has a position of x_c and y_c, and the center of the bin the cell is in is $x_{i,j}$ and $y_{i,j}$, then if $x_c > x_{i,j}$ and $y_c > y_{i,j}$, then the four nearest bin centers to the cell belong to bin (i,j); (i+1,j); (i,j+1); and (i+1,j+1), conversely, if $x_c < x_{i,j}$ and $y_c < y_{i,j}$, then the four nearest bin center to the cell belongs to bin (i,j); (i-1,j); (i,j-1); and (i-1,j-1).

If we let $\alpha = x + 0.5 - \lfloor x + 0.5 \rfloor$ and $\beta = y + 0.5 - \lfloor y + 0.5 \rfloor$, we can use α and β to determine the offset of the cell in question in respect to the bin center it is in. We can then use these offsets to implement a finer adjustment to the velocity of the cell by calculated the weighted mean of the velocity of the cell given by the bin center velocity combined with the velocity of the other 3 nearest bin centers.

These offsets will serve as weighting for each bin in the mean calculation. So the final cell velocity in the case of $x_c > x_{i,j}$ and $y_c > y_{i,j}$ is given by:

$$\begin{aligned} v_c^H &= v_{i,j}^H + \alpha(v_{i+1,j}^H - v_{i,j}^H) + \beta(v_{i,j+1}^H - v_{i,j}^H) + \alpha\beta(v_{i,j}^H + v_{i+1,j+1}^H - v_{i+1,j}^H - v_{i,j+1}^H) \\ v_c^V &= v_{i,j}^V + \alpha(v_{i+1,j}^V - v_{i,j}^V) + \beta(v_{i,j+1}^V - v_{i,j}^V) + \alpha\beta(v_{i,j}^V + v_{i+1,j+1}^V - v_{i+1,j}^V - v_{i,j+1}^V) \end{aligned} \tag{2.9}$$

Or in case of $x_c < x_{i,j}$ and $y_c < y_{i,j}$

$$\begin{aligned} v_c^H &= v_{i,j}^H + \alpha(v_{i-1,j}^H - v_{i,j}^H) + \beta(v_{i,j-1}^H - v_{i,j}^H) + \alpha\beta(v_{i,j}^H + v_{i-1,j-1}^H - v_{i-1,j}^H - v_{i,j-1}^H) \\ v_c^V &= v_{i,j}^V + \alpha(v_{i-1,j}^V - v_{i,j}^V) + \beta(v_{i,j-1}^V - v_{i,j}^V) + \alpha\beta(v_{i,j}^V + v_{i-1,j-1}^V - v_{i-1,j}^V - v_{i,j-1}^V) \end{aligned} \tag{2.10}$$

To illustrate this we can take a cell in bin(3,3). If the cell in question has a position of x=3.6 and y = 3.8, then the closest 4 bin centers are: its own bin center; bin center of bin(3,4); bin center of bin(4,3); and bin center of bin(4,4).

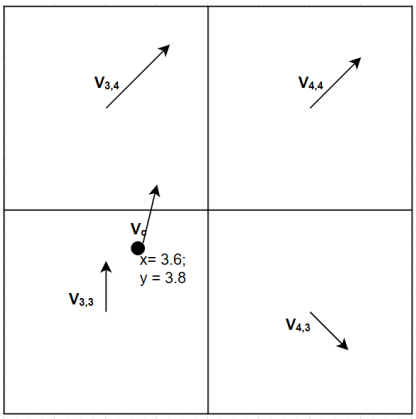


Figure 2.9: Cell velocity fine tuning using the cell position in relation to the bin center position.

$$\begin{aligned} v_{3,3}^H = 0; v_{3,3}^V = 1 \\ v_{3,4}^H = 0.707; v_{3,4}^V = 0.707 \\ v_{4,3}^H = 0.707; v_{4,3}^V = -0.707 \\ v_{4,4}^H = 0.707; v_{4,4}^V = 0.707 \end{aligned} \tag{2.11}$$

From Figure 2.9, we can calculate $\alpha = 3.6 + 0.5 - \lfloor 3.6 + 0.5 \rfloor = 0.1$ and $\beta = 1.8 + 0.5 - \lfloor 1.8 + 0.5 \rfloor = 0.3$.

So using equation 2.9 we can calculate the cell velocity of cell c as

$$v_c^H = v_{3,3}^H + 0.1(v_{4,3}^H - v_{3,3}^H) + 0.3(v_{3,4}^H - v_{3,3}^H) + 0.03(v_{3,3}^H + v_{4,4}^H - v_{4,3}^H - v_{3,4}^H)$$
$$v_c^H = 0 + 0.1(0.707 - 0) + 0.3(0.707 - 0) + 0.03(0 + 0.707 - 0.707 - 0.707) = 0.26159$$

$$v_c^V = v_{3,3}^V + 0.1(v_{4,3}^V - v_{3,3}^V) + 0.3(v_{3,4}^V - v_{3,3}^V) + 0.03(v_{3,3}^V + v_{4,4}^V - v_{4,3}^V - v_{3,4}^V)$$
$$v_c^V = 1 + 0.1(-0.707 - 0) + 0.3(0.707 - 0) + 0.03(1 + 0.707 + 0.707 - 0.707) = 1.19261 \quad (2.12)$$

Thus the cell velocity $\mathbf{v}_c$ is now $\begin{bmatrix}0.26159 & 1.19261\end{bmatrix}$ instead of $\begin{bmatrix}0 & 1\end{bmatrix}$ from the one given by the cell bin velocity.

2.3.3.4 Wirelength Reparation with Anchor Cells

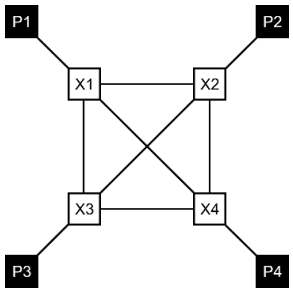
Diffusion reduces the overlaps between the cells inside the placement area. However, this process counteracts the effort of the quadratic placement itself, as spreading cells apart will inevitable increase wirelength, and this is not desirable as we want to start our congestion reduction process from the smallest wirelength possible, minimizing the amount of work we need to do later on.

One way to address this is to repair the diffusion result with the quadratic solver again, but doing this alone means that the cells will collapse back to their pre-diffused state. One solution to this is to fix a number of cells so it will serve as an anchor to prevent too many cells from getting pulled back to their original position before diffusion.

Another way to do this is to use a virtual cell as an anchor instead of actual cell. This method is based on the star net model, where a net can be represented as an extra cell, thus extra elements in the hessian matrix. This way, instead of having a net represented as a complete graph in the hessian matrix, the net is represented as a single cell with two-degree connection to each and every single cell connected to the net in question.

Originally this net model was devised to decrease the amount of computation on the quadratic solver, as it makes the matrix much sparser despite increasing the dimension of the matrix slightly. DPlace utilizes the virtual cell of the star net model [14] not to decrease computational load, but as the anchor instead of actual cells, in practice making the nets the anchor instead of the cells.

As these anchor virtual cells are static, they do not have to be represented in the hessian matrix. Instead, adding static cells is simply equivalent to adding pads in the middle of the placement area instead of around it. This means, for every cell connected to a certain net, we need to add a value to their corresponding elements in $\mathbf{b}_x$ and $\mathbf{b}_y$ vector to reflect the fact that they are now connected to additional pads.

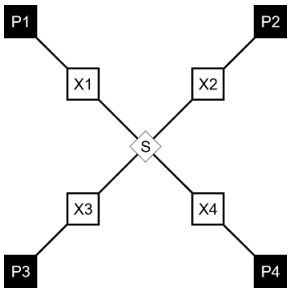


(a)

$$\begin{bmatrix} 5 & -1 & -1 & -1 \\ -1 & 5 & -1 & -1 \\ -1 & -1 & 5 & -1 \\ -1 & -1 & -1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

(b)

Figure 2.10: A circuit with 4 cells, 4 2-degree net connections between the cells, forming a clique, and 4 pads. (b) its wirelength-minimization equation matrix representation



(a)

$$\begin{bmatrix} 5 & 0 & 0 & 0 & -1 \\ 0 & 5 & 0 & 0 & -1 \\ 0 & 0 & 5 & 0 & -1 \\ 0 & 0 & 0 & 5 & -1 \\ -1 & -1 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ s \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ 0 \end{bmatrix}$$

(b)

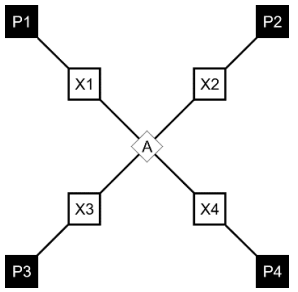
Figure 2.11: The same circuit as Figure 2.10a, but with star net representation (b) its wirelength-minimization equation matrix representation, the dimension has increased by 1 on both axis, but the matrix became much sparser

Not all nets are used as anchors however, as this could make the placement too static. Only nets with degree larger than a certain threshold will be used as anchors. The position of the anchors are calculated as the geometric mean of all the cells connected to the net, and a new hessian matrix and $\mathbf{b}_x$ and $\mathbf{b}_y$ vectors will be created.

The additional value added to the elements of $\mathbf{b}_x$ and $\mathbf{b}_y$ vectors is determined by the position of the anchor cell, which was in turn determined by the geometric center of the net, and the weighting of the anchor, which is equal to the degree of the net minus 1 multiplied by position of the net center.

2.4 Routing

After a chip has been placed, then the components need to be connected to each other according to the netlist. Routing can be classified into two stages, Global routing and Detailed routing. Global routing seeks to create tiles within the circuit containing the cells/module such that each tiles can be routed to each other, while at the same time optimizing the cost function and meeting constraints. After global routing is done, then detailed routing is supposed to actually assign the tracks and vias for the nets.



(a)

$$\begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} p_1 + wC \\ p_2 + wC \\ p_3 + wC \\ p_4 + wC \end{bmatrix}$$

(b)

Figure 2.12: Anchor-cell implementation based on star-net representation (b) its wirelength-minimization equation matrix representation, the dimension is the same as in Figure 2.10, but a weighted constant has been added to the elements of the **b** vector.

Algorithm 1 Anchor-cell construction algorithm

```
for net(i) = 1 to number_of_nets do
  Find the position of the center of net(i)
  if degree of net(i)) > anchor net degree threshold then

    for all cell connected to net(i) do
      Add the cells corresponding element in C vector with ((net degree  1) * position of
      the center of net(i))
    end for
  end if
end for
```

Routing are constrained by both Design Rule Constraint (DRC) and performance constraints. DRC usually pertains to the technology used in the routing while performance constraints will usually pertain to the speed, determined by the critical nets within the circuit/subcircuit in question.

2.4.1 Types of Global Routing

- **Sequential Global Routing**
  Sequential global routing is simple and straightforward, it sequentially route all the nets in the circuit based on a specific order. Sequential routing however, owing to its sequential nature, does not take consideration of the possible ill-effect caused by the routing of the net currently being routed on the routability of other nets. Evidently, the solution quality highly depends on the order of the nets to be routed. Due to its simplicity, it can be run quickly, but will usually yield less desirable routing result compared to concurrent global routing.

- **Concurrent Global Routing**
  As sequential global routing result is highly dependent on the order of nets being routed, there is a desire for another method that is not affected by order of the routing. This method is known as concurrent global routing.

Concurrent global routing with two-pins nets is often modelled as 0-1 Integer linear programming. The objective of the programming is to achieve a routing scheme where all the cells can be routed according to the netlist and the nets involved does not require more tracks than each edge provides. This problem is known to be NP-hard and there are approximation using regular Linear Programming approach by replacing the binary variable of whether a track is required or not (0 or 1) with a real variable that can contain any value in between 0 and 1.

- **Global Routing for higher-pin nets**
  For nets with more than 2 pins, one approach is to decompose the nets into several two-pin connections, and iteratively routing them one by one. One of the simplest way of doing this is by using Minimum Rectilinear Spanning Tree (MRST). Computing MRST can be done with modified Prims or Kruskals algorithm, and this can be done in polynomial time. MRST however, connects all the pins in successive manner, and this way each net will only have a degree of two. This is obviously suboptimal, and simply not realistic to be used as an actual routing algorithm.

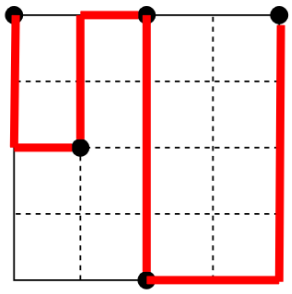
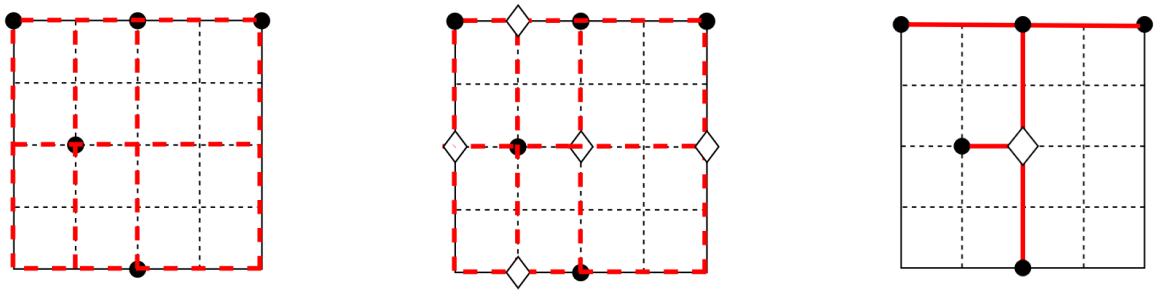


Figure 2.13: MRST result of routing 5-pin net. The resulting wirelength in this example is 16 units.

2.4.2 Spanning Tree Approach

2.4.2.1 Minimum Rectilinear Steiner Spanning Tree

A refinement of the MRST is the **Steiner Minimum Rectilinear Spanning Tree (SMRST)** which connects all the points but with the addition of several virtual nodes known as **Steiner points**. In hindsight, depending on the size of the circuit and cell positions, there is potentially an infinite number of Steiner points to be considered. Fortunately, **Hanan's theorem** proved that there are only a limited number of Steiner points that are worth considering, and these lies within the so-called Hanan grid [8], and we can enumerate all the possible routes through all the Steiner points in the Hanan grid to find the best possible routing scheme. Unfortunately, for higher-degree nets, this would still prove to be too computationally intensive.



(a) (b) (c)

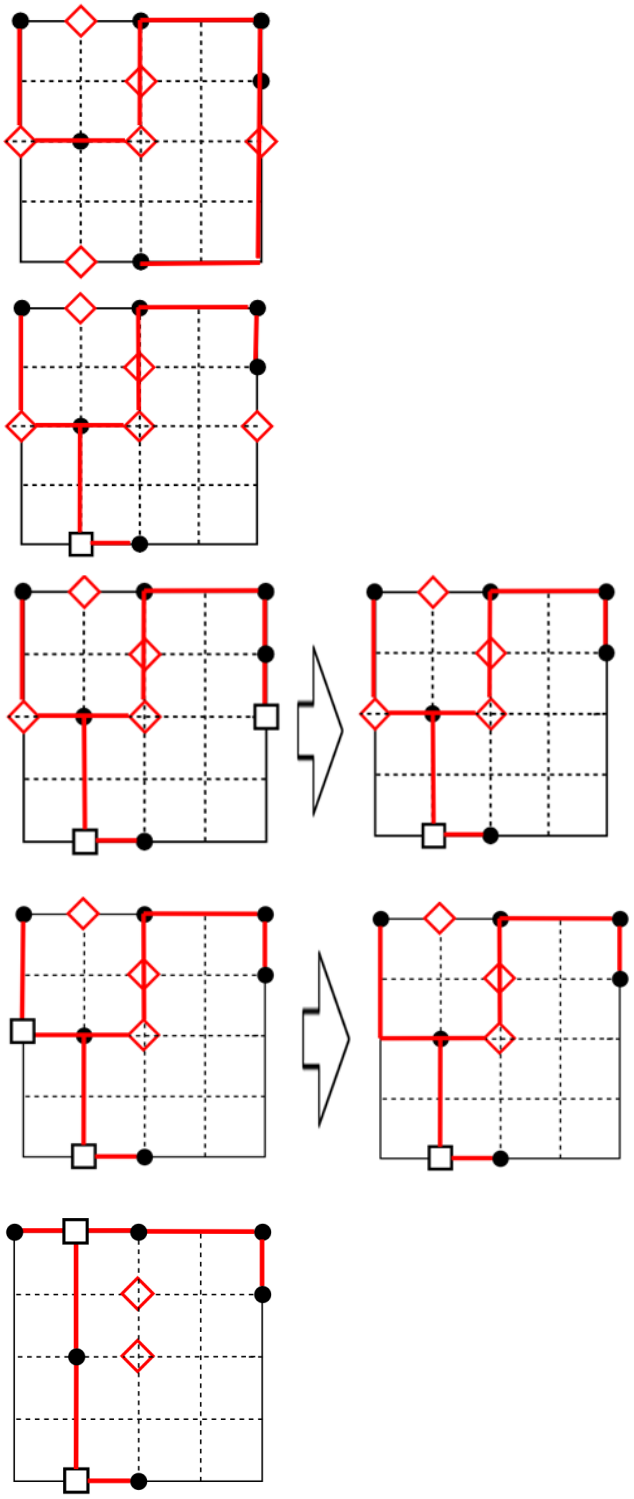
Figure 2.14: (a): Hanan grid of the same 5 cells connected by a net, (b) the resulting possible Steiner points marked by ◊ which are the intersections of the Hanan grid. (c): The resulting MRST routing with an additional Steiner point. The wirelength cost for this routing is only 9 unit.

MRST construction, is an NP-complete problem, however, and many heuristics method have been devised to construct MRST. One of the fastest one is the One-Steiner approach [7].

Algorithm 2 One-Steiner Algorithm

```
C = nodes from all the cells in the design
S = all Steiner points in the design
Create MRST using Prim or Kruskal's algorithm
while C is not empty do
  Find wirelength of (C ∪ x), x ∈ S
  if SMRST wirelength decreases then
    C = (C ∪ x)
    delete x from S
    remove all Steiner points already in C with degree ≤ 2
  end if
end while
Output = MRST(C)
```

- **One-Steiner Approach**
  Based on the Steiner points approach, Kahng and Robins proposed a fast SMRST construction method called the One-Steiner approach. In this case, the routing starts with normal MRST construction with either Prim or Kruskals algorithm, One-Steiner algorithm adds an extra Steiner point to the points already used in creating the MST (cells connected to the net). With the added extra Steiner points, the SMRST length will decrease to the point that if there is no more Steiner points that can be added will give any reduction in length, then the process will be terminated [7]. The general algorithm of the One-Steiner approach is described in Algorithm 2.



Current routing length = 14 units

Add one Steiner point, resulting length = 12 units $<$ 14 units, this Steiner point is **kept**. Current length is now 12 units

Add another Steiner point, resulting length = 13 units $>$ 12 units , this Steiner point is **discarded**.

Add another Steiner point, resulting length = 12 units (no improvement) , this Steiner point is **discarded**.
It must be noted that even if this point was to be added, it will be deleted later anyway as the degree is ≤ 2

Add one Steiner point, resulting length = 10 units $<$ 12 units, this Steiner point is **kept**. Current length is now 10 units

Figure 2.15: shows the example of One-Steiner algorithm, nodes from cells are marked by •, unexplored Steiner points are marked by ◊, and used Steiner points are marked by □. While One-Steiner is a relatively fast routing algorithm, its time complexity is still $O(N^3)$, for a net with high degree, this can take a very long time to complete. A batched variant of the algorithm also exists with a time complexity of $O(N^2 log(N))$.

2.5 Congestion and Area Requirement

With the advances in VLSI design, the number of cells and wiring ever increases. As wires are routed over the cell, this presents another constraints which is wire density, or in other words, congestion, which is separate from the cell density. Routing overflow sometimes contributes to area requirement more than the cells themselves. It is possible to implement heuristics method to improve congestion at the placement level, and this is key to improving routability of the design, and eventually reducing the area requirement imposed by wiring detours [7]. There has been other congestion estimation method based on bounding-box routing model and supply and demand model [22][23][24], but the result has not been very satisfactory[6].

2.5.1 Congestion Minimization with C-ECOP

Wiring area is intrinsically tied to congestion. We need to know the congestion to accurately determine the area required by wiring. By actually doing a trial routing on the design, this method can produce an accurate model on congestion, and thus wire density. However, as have been demonstrated by the One-Steiner wire routing model, the computational cost is too prohibitive to be of practical use in our case. Instead, we need to use a simplified wire crossing model that can still provide an adequate picture of the congestion in the design.

Li et al. proposed a congestion-driven placement method called the C-ECOP for standard cell placement. The method relies on a new routing model and cost function, and relies on ILP for the actual placement stage [6].

2.5.1.1 Routing Model and Cost Function

C-ECOP uses the global bin concept to estimate congestion cost. In this concept, the chip is partitioned into multiple bins during the global placement stage, hence global bins. A wire crossing the edges of this bins will contribute towards the congestion of the design, thus the congestion calculation is tied to each bin. The more bins we have in the calculation, the more accurate the result would be, as would be clear later.

The simplest utilization of the global bin concept is to simply place a design, and let a simple trial router such as bounding box router to try route a placement result. This can give us the general idea of where the wires are going to be, thus the wire crossings between the bins can be calculated and thus we have a rather rudimentary result on congestion in our design.

We can think of each edge of a bin of having a finite amount of routing resource supply, and each wire crossing an edge will impose demand on the said edge. An edge is said to be congested when the routing demand exceeds the routing supply on the particular edge. The congestion of a bin with this method is thus calculated as the sum of

the congestions of all four edges of the bin in question, as seen in Equation 2.13 and 2.14.

$$overflow(e) = \begin{cases} 0 & (d_e \leq s_e) \\ d_e - s_e & (d_e > s_e) \end{cases}$$ [6]

2.13

$$con(b_{ij}) = \sum_{k=1}^{4} overflow(e_k)$$

$$\cos t_c = \sum_{j=1}^{n} \sum_{j=1}^{n} con(b_{ij})$$ [6]

2.14

However this method is not particularly accurate, especially over larger number of bins, as a simple router such as bounding box router does not have any means to ascertain which bins are crossed when a wire between two cells that are not adjacent to each other needs to cross multiple bins to connect them, as can be seen in Figure 2.16

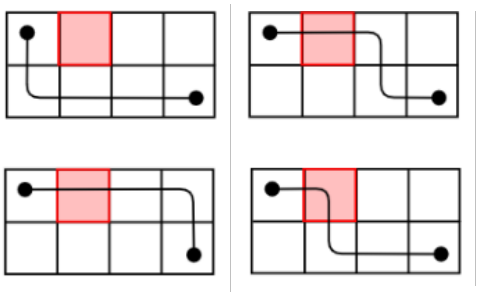


Figure 2.16: Multiple possible routes to connect two cells in two non-adjacent bins

Simple estimation like bounding box router does not take this into account, thus it cant give an accurate prediction on where the congestion is going to be, and on the other hand, as have been mentioned before, running a Steiner-tree routing is too costly for an iterative placement method [9].

Instead of resorting to either of those methods, C-ECOP uses a probability routing that allocates distribution of a wire based on the bin containing the cells that its connected to. Each edge of a bin will have a probability of whether a wire crosses it or not.

To illustrate this we can take an example of a single bin k. This bin is located in an area where all the bins in it have a non-zero probability of being crossed by a wire

going from cell 0 to cell 1. The probability of the left edge of this particular bin being crossed by a wire going from cell 0 to 1 is given in Equation 2.15 .

$$W_{kl} = C^{i_k - i_0 - 1}_{i_k - i_0 - l + j_k - j_0} \times C^{i_1 - i_k}_{i_1 - i_k + j_1 - j_k} / C^{i_1 - i_0}_{i_1 - i_0 + j_1 - j_0} \quad [6]$$

2.15

Where W_{kl} is the probability of the left edge of bin k being crossed by a wire. The generalization of this probability calculation for both left and right direction and taking account of no crossing at all, is given in Equation 2.16.

$$p_l(k) = \begin{cases} & i_0 \leq i_k \leq i_1 \\ w_{kl} & \\ & j_0 \leq j_k \leq j_1 \\ 0 & otherwise \end{cases}$$

$$p_r(k) = \begin{cases} & i_0 \leq i_k \leq i_1 \\ w_{kr} & \\ & j_0 \leq j_k \leq j_1 \\ 0 & otherwise \end{cases}$$

$$W_{kl} = C^{i_k - i_0 - 1}_{i_k - i_0 - l + j_k - j_0} \times C^{i_1 - i_k}_{i_1 - i_k + j_1 - j_k} / C^{i_1 - i_0}_{i_1 - i_0 + j_1 - j_0}$$

$$W_{kr} = C^{i_k - i_0}_{i_k - i_0 + j_k - j_0} \times C^{i_1 - i_k - 1}_{i_1 - i_k - 1 + j_1 - j_k} / C^{i_1 - i_0}_{i_1 - i_0 + j_1 - j_0} \quad [6]$$

2.16

W_{kl} is the probability of the left edge of bin k being crossed by a wire,
W_{kr} is the probability of the right edge of bin k being crossed by a wire.
i_k is the horizontal index of bin k
j_k is the vertical index of bin k
i_0 is the horizontal index of bin 0, where cell 0 is located
j_0 is the vertical index of bin 0, where cell 0 is located
i_1 is the horizontal index of bin 1, where cell 1 is located
j_1 is the vertical index of bin 1, where cell 1 is located

proof: *In global placement, wires are assumed to be traversing the bins in a greedy way in a taxicab geometry, that is given horizontal bin index i and vertical bin index j, the wire will go through these index in a non-descending fashion. This means, if a wire starts from bin (i,j), i = 1 and j = 1, the wire can either go to bin(1,2) or bin(2,1) but not both. This means for a wire traversing n bins horizontally and m bins vertically, we have a set* $i \in \{1, 2, .., n\}$ *and* $j \in \{1, 2, .., m\}$ *thus the number of unique paths a wire can traverse is either* $\begin{bmatrix} n-1 \\ (n-1)+(m-1) \end{bmatrix}$ *or* $\begin{bmatrix} m-1 \\ (n-1)+(m-1) \end{bmatrix}$.

Finding a probability of a an edge of a bin k being crossed by a wire going

from cell c_0 to c_1 is simply a matter of enumerating all the possible wiring routes going from bin 0 to bin k, and multiplying it with all the possible wiring routes going from bin k to bin 1, and then dividing it over the number of total routes from bin 0 to bin 1. This can be done with the previous equation. The difference of probabilities between left and right (and top and bottom) sides is merely 1 bin, which is represented with the -1 on either the first or second term of the equations.

2.5.1.2 Higher-Degree Net Representation

The congestion cost model used in C-ECOP poses a problem. It can only be used on nets with degree of two. There have been other representations of net model other than clique such as star model which replaces the complete-graph representation of the clique model with two-cell connections with the addition of a single virtual cell. Hou et al. uses this star representation to create a routing model which represents a net as a collection of 2-degree connections between a cell and a virtual cell called the net center. The net centers position is calculated as the geometric mean of all the cells connected to that particular net. C-ECOP uses this net model to break down nets in the design into two-degree connections thus enabling the use of its cost function [13].

2.5.1.3 ILP-based Congestion Reduction

Minimizing congestion after a placement usually means trading off wirelength for routability. Thus we need to achieve a good balance between wirelength and congestion. This problem however, is NP-hard, and we need a heuristic method to find a good enough solution. The core of this problem is, that unlike cell overlap problem described in diffusion preplacement, the data available here are not points, but a set of probability of the possible wirings given a set of placement, which means we cant simply use similar gradient-based diffusion as before in a straightforward manner.

C-ECOP uses an iterative method to reduce congestion. It takes input from a placer, such as DPlace, and estimates the congestion using the method described in section 2.5.1.1. A parameter called cell flow tendency is then calculated, which will be used in the ILP placement, where the placement result will be recalculated again for its congestion, and fed back to the ILP until a desirable result is obtained.

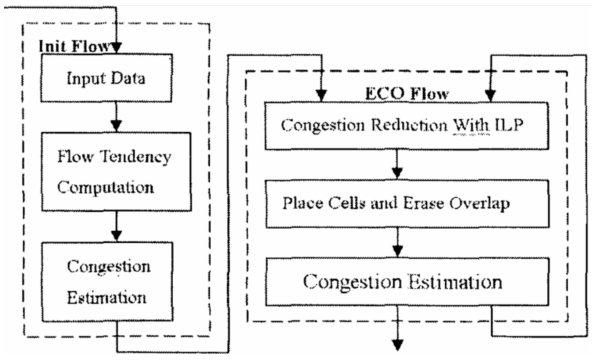


Figure 2.17: C-ECOP algorithm flow.[6].

- **Cell flow tendency**
  Like diffusion preplacement, cells in a heavily congested bin should be moved out in order to reduce congestion in the bin in question, thus reducing routing demand and reducing wire density. The problem is knowing where the cells should be moved to. In diffusion preplacement, cell density gradient provides a direction which the cell would follow at any given iteration, however this is not the case with C-ECOP.

  Instead C-ECOP calculates the direction via cell flow tendency. This parameter indicates how likely it is for a cell to be moved out of the bin it is currently in. Cell flow tendency of a cell c_k in bin(i,j) is calculated as:

$$\begin{aligned} xflow_l(c_k) &= \sum_{net_{kl}} p_l(k) - \sum_{net_{kr}} p_r(k) - \sum_{net_{kc}} \\ xflow_r(c_k) &= \sum_{net_{kr}} p_r(k) - \sum_{net_{kl}} p_l(k) - \sum_{net_{kc}} \end{aligned} \tag{2.17}$$

  For horizontal component, and

$$\begin{aligned} yflow_t(c_k) &= \sum_{net_{kt}} p_t(k) - \sum_{net_{kb}} p_b(k) - \sum_{net_{kc}} \\ yflow_b(c_k) &= \sum_{net_{kb}} p_b(k) - \sum_{net_{kt}} p_t(k) - \sum_{net_{kc}} \end{aligned} \tag{2.18}$$

  For the vertical component

net_{k_l} denotes nets whose net centers lie within the bin to the left of bin(i,j), net_{k_r} denotes nets whose net centers lie within the bin to the right of bin(i,j), net_{k_t} denotes nets whose net centers lie within the bin on top of bin(i,j), and finally net_{k_b} denotes nets whose net centers lie within the bin below bin(i,j).

$xflow_l$ indicates the likelihood of having a reduction of net crossings on the left edge of bin(i,j) from moving the cell from bin(i,j) to bin(i-1,j). A positive $xflow_l$ mean moving the cell in question to the left will result in decreased net crossing in the left edge of bin(i,j), which means a reduction in overflow in that particular edge.

Likewise, a positive $xflow_r$, $xflow_t$, or $xflow_b$ means a probable reduction in overflow in their respective edge on bin(i,j) if the cell is moved according to their respective direction. A negative cell flow tendency, however, indicates that moving the cell in the particular direction would probably not result in reduced congestion.

While calculating the cell flow tendency does gives us the information on where the cells can be moved in order to reduce congestion, this does not necessarily mean that the cell should be moved in the particular direction indicated by its cell flow tendency. If two or more cell have a flow tendency that suggest they should move to the same bin, it is possible that the bin they are moved into ends up being more congested

than the bins the cells originated from. This calls for an arbitrative mechanism that can distribute the cell in a way that would reduce congestion in highly congested bins without causing congestion elsewhere.

Cell movement can be modeled as linear equation where the movement of a cell is described as a term in an integer linear equation. In order to construct this equation, we need to know how much improvement we gain by moving a cell in a particular direction. The probable gain can be calculated as:

$$\begin{aligned} gain_h(c_k) &= \sum_{net_{kl}} p_l(k) + \sum_{net_{kr}} p_r(k) \\ gain_v(c_k) &= \sum_{net_{kt}} p_t(k) + \sum_{net_{kb}} p_b(k) \end{aligned} \tag{2.19}$$

Aside from the gain itself, we need to know where the cell should move exactly before constructing the equation. Cell flow tendency which direction the cells should be moved in one axis, as we can see in Equation 2.19, only one flow in either horizontal or vertical component can be positive at any given time.

For example, when $xflow_l$ is positive, then $xflow_r$ has to be negative, which means the cell should only be moved to the left. The same goes for the vertical component $yflow_t$ and $yflow_b$. If a cell only have one direction with positive flow in the horizontal axis and none in the vertical axis, the answer is clear cut. However, if this was not the case, we need to determine which axis the cell should be around.

For a cell to be moved around on particular axis, a cell must have a positive cell flow tendency in either direction. Thus, in mathematical terms we can describe it as:

$$x_k = (xflow_l(c_k) > 0 \; or \; xflow_r(c_k) > 0) \tag{2.20}$$

For horizontal axis and,

$$y_k = (yflow_t(c_k) > 0 \; or \; yflow_b(c_k) > 0) \tag{2.21}$$

For the vertical axis.

This conditions however, were described to be overly rigorous and prevented many of the cell movement from happening at all, thus from experimental result performed by Li et. al, they decided to lower the condition from 0 to a negative threshold T_h and T_v. This way, the ILP accepts possible small increase of maximum congestion over the affected bins, in exchange for possibly better results in the next

iteration. The ILP can then mathematically be described as

$$\begin{gathered} minimize\ C_{max}\ subject\ to \\ con_h(bin_{ij}) = \sum_{C_k} m_{kij}(k) x_k gain_h(C_k) \leq C_{max} \\ con_v(bin_{ij}) = \sum_{C_k} m_{kij}(k) y_k gain_v(C_k) \leq C_{max} \\ x_k + y_k \leq 1 \\ x_k, y_k \in \{0, 1\} \\ x_{kij} \in \{-1, 0, 1\} \end{gathered} \tag{2.22}$$

Where $j = 1, 2, ..., N$ and $k = 1, 2, ..., M$. C_{max} is the maximum congestion of any bins in the placement, and $m_{kij}(k)$ is the integer to be determined by the ILP.

Methodology and Implementation

3

3.1 Cost Function

Wiring area is usually obtained after performing the final routing of the design. This is problematic not only because chip routing takes a long time, we also do not have the ability to modify the routing algorithm used by many EDA place and route tools, which in our case, is Cadence Encounter EDI 14.27. Congestion, however, can be estimated early in the placement stage and can provide a good indication of how routable the design is. A design with lower congestion for the same cell area will likely require less wiring area than another circuit with same functionality and cell area but higher wirelength and congestion count. Thus our primary metrics to be used in this thesis is wirelength and congestion.

As mentioned in the previous chapter, our focus will be the placement stage, and our objective is congestion minimization.

3.2 Placement Algorithm Selection

Several global placement algorithms have been reviewed, namely DPlace, Capo and Dragon [3]. Capo is a congestion-driven placement method which might sound like the best solution, however Dragon and Capo requires changes to the floorplan which rules it out of the solution space [3][4].

Instead of relying on Dragon or Capo, another method was devised where a separate process is used to move the cells around to minimize congestion.

So instead we now focus on both Dplace and Kraftwerk. DPlace and Kraftwerk are both a quadratic placers. Quadratic placers tend to yield the best preplacement result when it comes to wirelength and this should give us a good starting point [18]. Kraftwerk yields better HPWL result and was the winner of ISPD 2005/2006 placement contest. Kraftwerk implementation in MATLAB however, did not yield workable result as the spreading mechanism did not work as intended, and took too much CPU time in MATLAB, thus given limited time constraint, we will use DPlace instead.

The algorithm proposed in this effort is designed to minimize wire density by minimizing wiring congestion and wirelength. The flowchart of the algorithm can be can be seen in Figure 3.3.

We will use DPlaces algorithm for our quadratic placement, as DPlace provides good wirelength minimization and should provide a good starting point for minimizing our congestion. To implement our Dplace algorithm, we need to obtain the pad position and the Laplacian matrix.

The pad position in this context refers to the position of the cells on modules outside the DTB Multiplexer which is connected to a net that is also connected to the cells

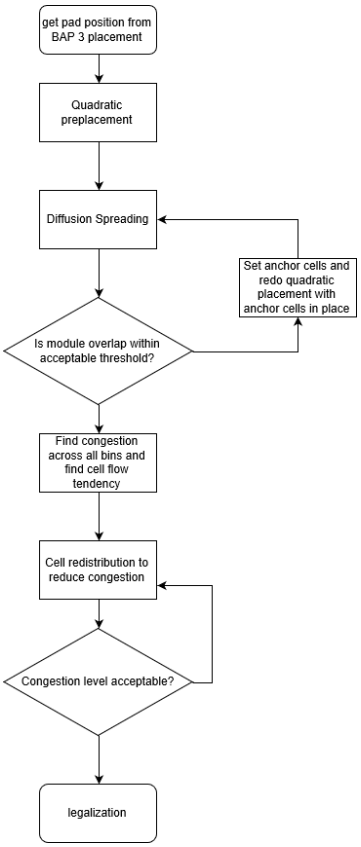


Figure 3.1: Full workflow of the proposed method.

of the DTB multiplexer. The cells outside DTB multiplexer are treated as pads (non-movable objects), as we only want to modify the DTB multiplexer, and they provide a reference point to where the DTB multiplexer should be placed, thus they are going to be used in the **b** vector on the wirelength minimization equation. In total there are 693 pads connected to the DTB multiplexer.

3.3 Quadratic preplacement

We obtain the hessian matrix required for the quadratic placement from the netlist available from the BAP3 synthesis result. A netlist is then turned into a Laplacian matrix using a MATLAB script. As there are no timing constraint in the DTB multiplexer, no different weighting for net between cells needs is used, thus all the edges in the **A** matrix will have weight of -1, whereas the diagonal will be the negative sum of all the edges weight, plus the sum of the weight of the pads connected to the cell corresponding to the element in the diagonal matrix, as explained in section 2.3.3.

The weight of the cell-pads connection, however, needs to be adjusted to create a placement result that would have a similar cell-area as the original DTB multiplexer cell placement. Experimental results show pad-cell net weight of 0.5 should suffice

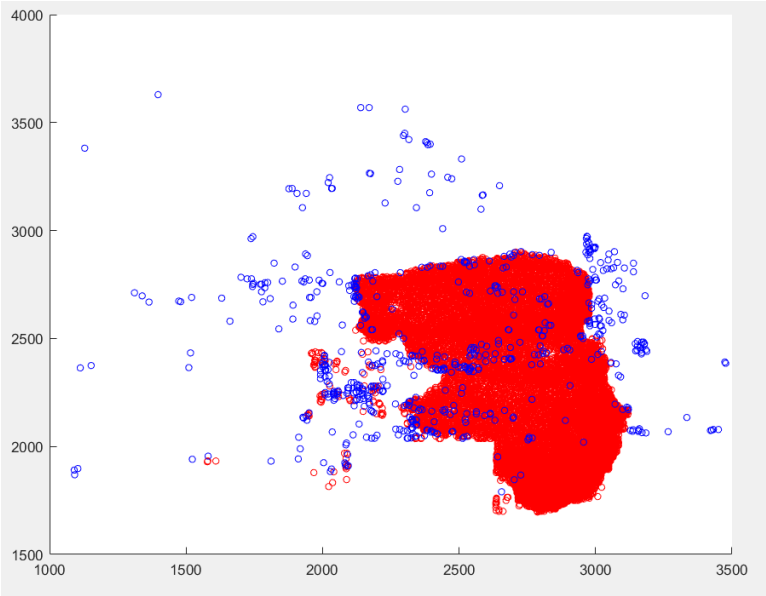


Figure 3.2: The pad position of the DTB multiplexer represented as blue markers, while the original cell position of the DTB multiplexer represented as red markers. The red-shaded area represents the original DTB multiplexer placement area.

to create a placement result which would occupy similar cell area to original BAP placement which is around $0.93mm^2$ This weight is then multiplied element-wise with the pad position of both axes to obtain vector $\mathbf{b}_x$ and $\mathbf{b}_y$. These vectors are then divided by hessian matrix $\mathbf{A}$ to obtain the quadratic preplacement vector $\mathbf{X}$ and $\mathbf{Y}$.

3.4 Diffusion Preplacement and wirelength reparation

After the initial quadratic placement has been performed, we will have a placement result that is highly overlapped. In order to reduce overlap, we need to perform diffusion preplacement on the placement result. The first step to this is to compute the gradient. Dplace uses Forward-Time-Centered-Space (FTCS) method to compute the gradient. However, MATLAB has a built-in function to find bin gradients, thus instead of using the FTCS equation, we can simply substitute it with MATLABs *gradient* command.

Determining the bin size can be tricky. If the bins are too small, then both the memory requirement and process time increases. Bin size that is too small will also contain too few cells to accurately calculate the gradient if a bin. An extreme example of this would be if the bins are smaller than the smallest cell size, then each bin will only count as being filled by one cell, thus we wouldnt be able to calculate the gradient of the bins. On the other hand, bins that are too large will give a very inaccurate result.

Based on experimentations, a bin size that yields acceptable runtime and accuracy is $30\mu m$. All the cells used by the DTB multiplexer from the default synthesis and mapping ranges from $1.024\mu m * 3.584\mu m$ to $9.728\mu m * 3.584\mu m$, which corresponds to

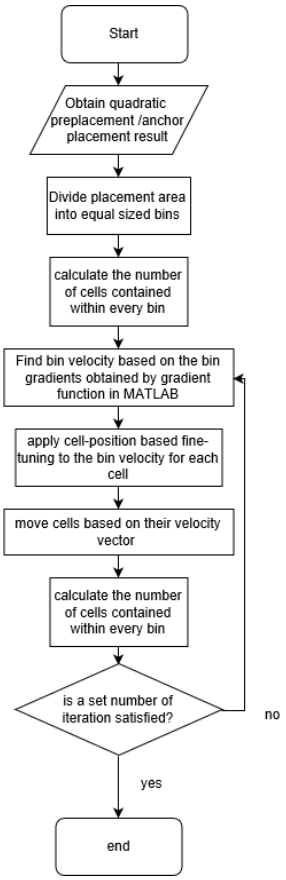


Figure 3.3: flowchart of diffusion placement process.

an area of $3.67\mu m^2$ and $4.86\mu m^2$, and the mean size of the cells are $8.72\mu m^2$. Thus for the cell overlap to be satisfactory, we will set a condition that all bins should contain no more than $\lfloor (10\mu m)^2/8.72\mu m^2 \rfloor = 11\ cells$.

In between iterations, to prevent nets from getting too long, we use anchor cells to repair the wirelength by bringing cells closer to their net centers. The details of this is explained in section 2.3.3, and the flowchart of this process is given in Figure 3.4.

3.5 Congestion Estimation and Reparation

Using the method explained in Section 2.5.1, we will then use the placement result from the D-place flow to calculate the congestion. The C-ECOP method however encounters a problem during the ILP stage. The heaviest congestion of the DTB multiplexer occurs in the right-bottom region of placement area. ILP moved all the cells out of the most congested bin. This reduced the congestion of the bin, but the bin still ended up being the most congested bin in the entire placement area.

As the cost function of the ILP is actually defined as a constraint $con_h(bin_{ij}) =$

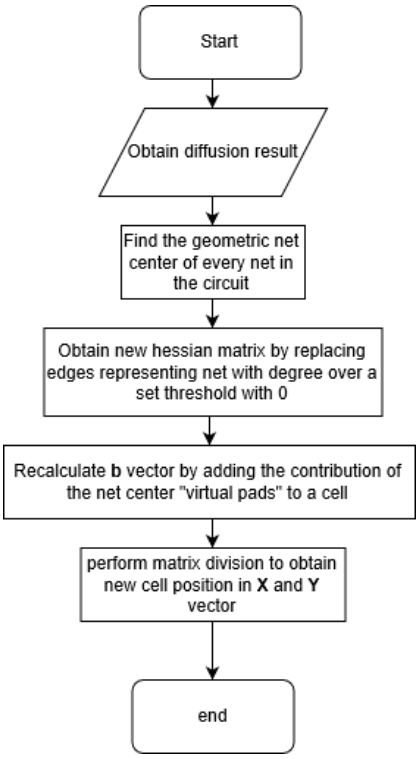


Figure 3.4: Anchor cell wirelength reparation process flowchart.

$\sum_{C_k} m_{kij}(k)x_k gain_h(C_k) \leq C_{max}$ and $con_v(bin_{ij}) = \sum_{C_k} m_{kij}(k)y_k gain_v(C_k) \leq C_{max}$ while the parameters are cell movement, this means if there is no more cell to be moved in a bin, there is nothing that the ILP can do.

Instead we will use diffusion method similar to the one used on Dplace. Cells are moved according to their congestion gradient. Fortunately, instead of having to calculate the gradient, we already know the direction of the cell movement from the cell flow tendency. We will use this instead to move our cells instead of using the ILP. We still move the cells based on the same movement parameters as the ILP, namely the *cell flow tendency* and *cell gain*.

As cells are spread, less and less cells will be advantageous to move. At one point no more cells will have *cell flow tendency* greater than the movement threshold defined in section 2.5.1, thus we stop the cell movement process, and start repairing the overlap and wirelength using the *anchor cell* and *cell diffusion* from Dplace. The flowchart of this process is given in Figure 3.5.

The overflow is then calculated by subtracting the excess of possible wire crossing from the wiring resource available per edge. The technology used in BAP3 utilizes 5 metal layers. 2 vertical, 2 horizontal for net routing, and 1 for power supply. As we are not dealing with power supply, our congestion will be focused only on the four routing layers.

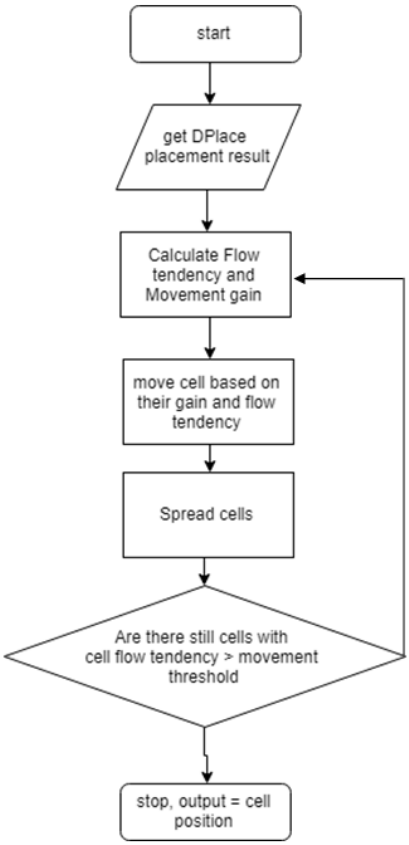


Figure 3.5: Flowchart of the modified C-ECOP algorithm used.

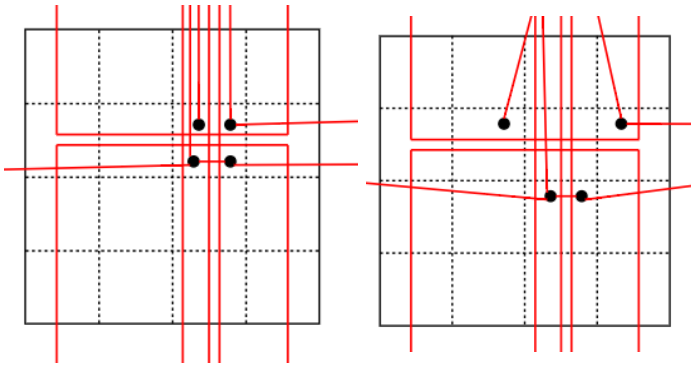


Figure 3.6: Example of failure in ILP-based cell movement. (a) The center-upper-right bin in the placement are is the maximum congested bin, not only because of the cells contained within but also wires crossing it from other bins, even if all the cells are moved out to (b), the bin still end up the most congested. The ILP will have no more cells to move and is stuck.

The pitch for all four layers is $0.512\mu m$ wide. On the other hand, each bin in the

C-ECOP measures $40\mu m$ by $40\mu m$, which is the smallest practical bin size that the PC used in simulation could handle. Thus a single edge can theoretically route at most 78 net crossings, or 156 total net crossing can be handled by a single bin on each axis. As each axis has two layers, this number is doubled to 312 nets. Thus an X or Y overflow of a single bin on means any bin the amount of net crossing subtracted by 312.

3.6 Congestion Re-estimation using One-Steiner Routing Estimation

The congestion of the placement result after C-ECOP cell movement is then going to be evaluated again. This time, in addition to the congestion estimation method of the C-ECOP itself, we will also use the One-Steiner method to perform a routing estimation to give us a more accurate congestion estimation result.

Nets between degree of two and a certain threshold will be routed by the One-Steiner method, thus transforming all those specific nets into two-pin connections. This way, instead of relying on the geometric net center, we will have a set of Steiner points which will aid routing prediction thus improving the accuracy of routing estimation.

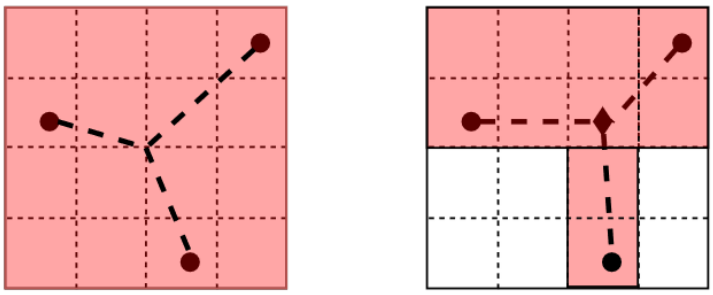


Figure 3.7: One-Steiner two-pin transformation improving C-ECOP congestion prediction. In Figure (a), C-ECOP needs to calculate the net crossing probabilities of all the bins shaded, whereas in (b), the net has been transformed into two-pin nets with the addition of a Steiner point marked by $\Diamond$, thus C-ECOP only needs to calculate the crossing probabilities of the the first two rows as the crossing probability of all the bins between and containing the Steiner point and the bottommost cell are 1.

An algorithm has been devised to create this new graph, called the OSVN (One-Steiner Virtual Nets). As the C-ECOP estimation algorithm takes input in form of netlist (not adjacency matrix), this can easily be altered. This algorithm deletes all nets with degree between 2 and a preset threshold, and replaces them with a greater number of two-degree nets. The flowchart of this process is given in Figure 3.8.

The reason why there is a threshold on net degree used is due to the time complexity of the One-Steiner method which is $O(N^3)$ which means it will take a really long runtime for high-degree nets, and its relative inaccuracy for high-degree nets, making it not so

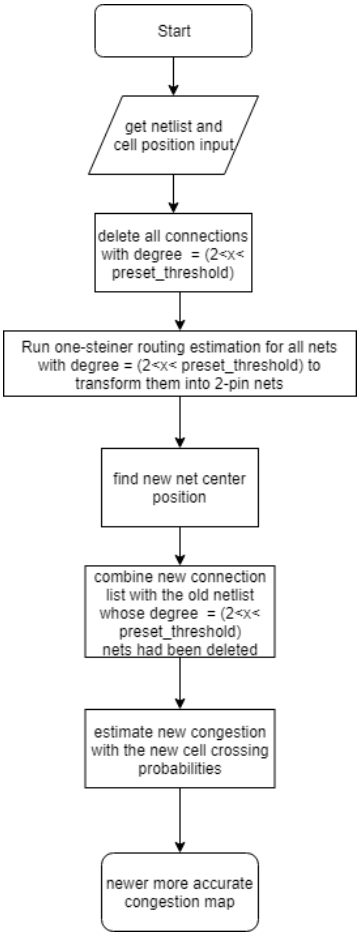


Figure 3.8: One-Steiner Virtual Net algorithm designed to replace connections between degree 2 and a select threshold with actual routing estimation using One-Steiner algorithm.

advantageous over the net-center method. Based on experimental result, a threshold degree of 15 is used as this gives a reasonable runtime.

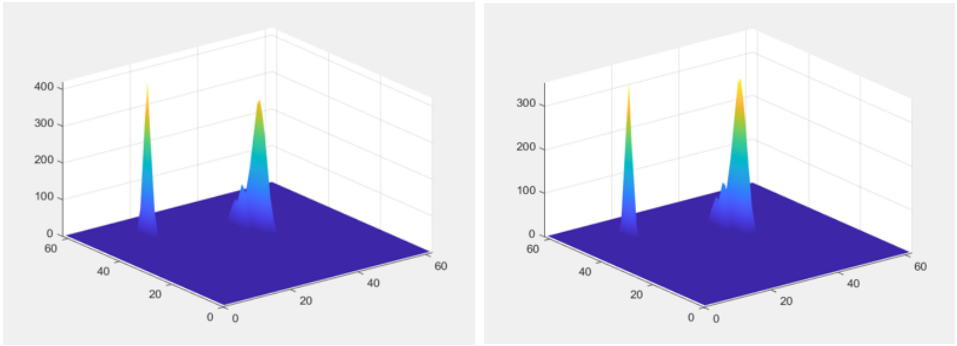
3.7 Encounter Implementation

The MATLAB placement result will be placed in encounter flow after the init stage of the BAP3 place and route script, which means the cells are placed before the automated placement is done by encounter. All the cells will be set as SOFTFIXED to allow encounter to perform detailed placement and legalization according to its DRC while retaining their general order retaining most of the MATLAB placements wirelength and congestion result.

4 Results

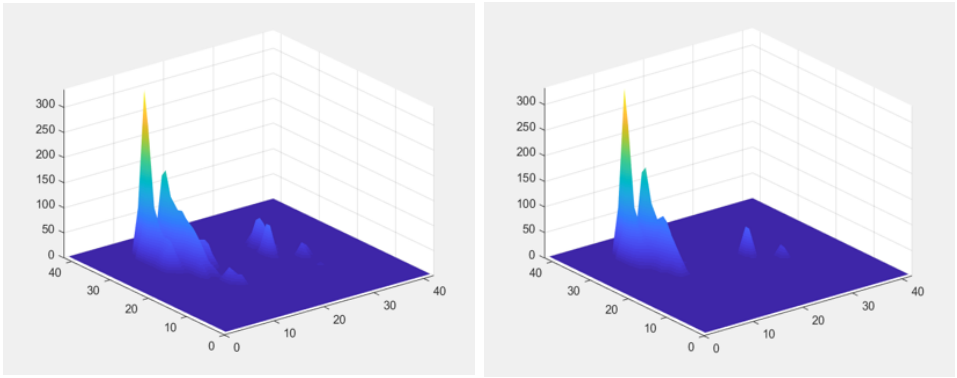
4.1 MATLAB Results

We use MATLAB for most our algorithm implementation including the DTB Multiplexer placement, while we use Cadence EDI 14.27 to place the rest of the chip and also obtain the final actual routing.



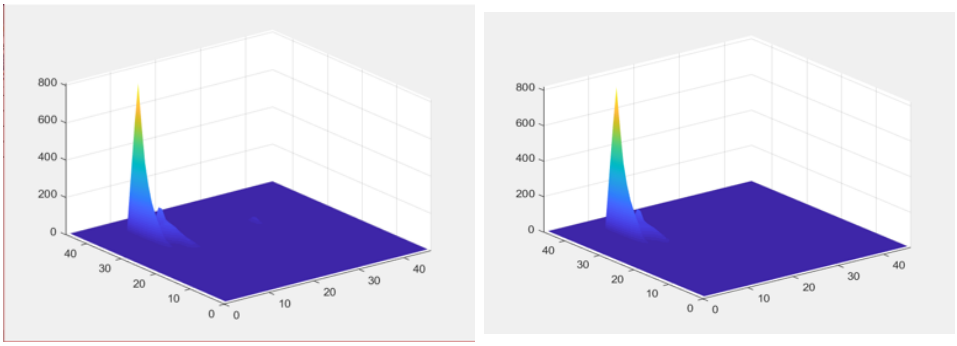
(a) (b)

Figure 4.1: (a): Horizontal Congestion estimation of original DTB multiplexer placement without One-Steiner routing prediction and (b) with One-Steiner routing prediction.



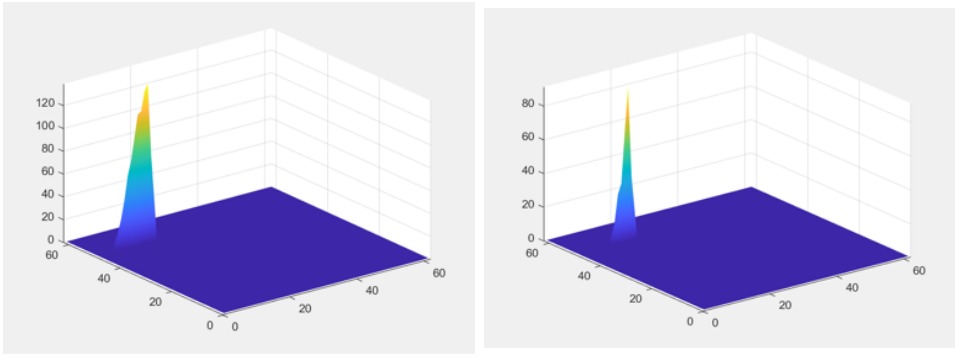
(a) (b)

Figure 4.2: (a): Horizontal Congestion estimation of DPlace placement result without One-Steiner routing prediction and (b) with One-Steiner routing prediction



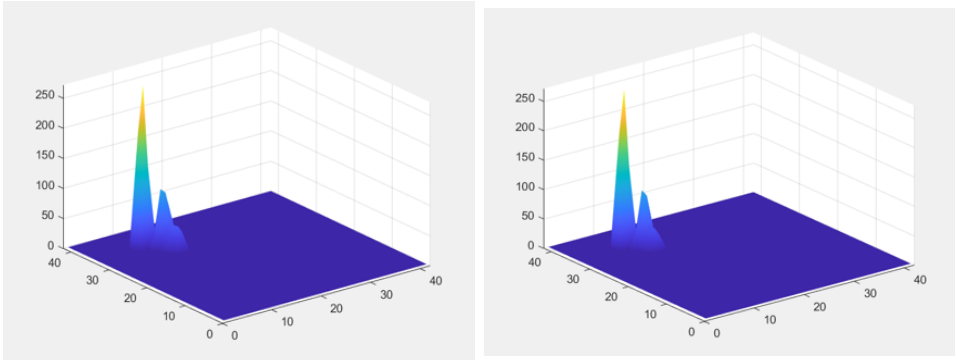
(a) (b)

Figure 4.3: (a): Horizontal Congestion estimation of C-ECOP congestion reparation result without One-Steiner routing prediction and (b) with One-Steiner routing prediction.



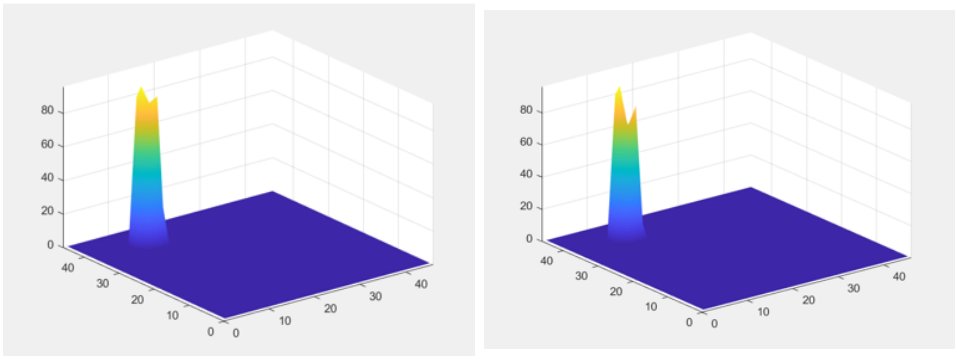
(a) (b)

Figure 4.4: (a): Horizontal Congestion estimation of original DTB multiplexer placement without One-Steiner routing prediction and (b) with One-Steiner routing prediction.



(a) (b)

Figure 4.5: (a): Horizontal Congestion estimation of DPlace placement result without One-Steiner routing prediction and (b) with One-Steiner routing prediction



(a) (b)

Figure 4.6: (a): Horizontal Congestion estimation of C-ECOP congestion reparation result without One-Steiner routing prediction and (b) with One-Steiner routing prediction.

Placement	**Total Estimated Wirelength (HPWL),** μm	**Total Horizontal Overflow (C-ECOP estimation)**	**Total Horizontal Overflow (C-ECOP + One-Steiner estimation)**	**Total Vertical Overflow (C-ECOP estimation)**	**Total Vertical Overflow (C-ECOP + One-Steiner estimation)**
Original	3.08094e+06	5971.89	5437.01	915.91	270.81
Dplace	4.43033e+06	5991.83	4832.36	1184.36	1081.12
Dplace + C-ECOP	4.35496e+06	5498.83	4789.35	691.61	609.99

Table 4.1: Total estimated wirelength and total overflow results

Placement	**Maximum number of cells per** ($10\mu m * 10\mu m$ **bin), must not be** $>$ **11**	**Maximum Horizontal Overflow (C-ECOP estimation)**	**Maximum Horizontal Overflow (C-ECOP + One-Steiner estimation)**	**Maximum Vertical Overflow (C-ECOP estimation)**	**Maximum Vertical Overflow (C-ECOP + One-Steiner estimation)**
Original	10	420.77	354.19	139.09	91.42
Dplace	11	335.94	332.99	271.77	271.77
Dplace + C-ECOP	11	813.27	820.33	95.93	96.29

Table 4.2: Cell density and maximum overflow results

Results indicate that C-ECOP cell rearrangement improves upon the DPlace

placement result, with 8.22% reduction in horizontal congestion, and 41.6% reduction vertical congestion when measured with C-ECOP congestion estimation. With added routing estimation using One-Steiner Algorithm, the difference is less impressive with 0.89% reduction in horizontal congestion and 43.57% reduction in vertical congestion.

The maximum horizontal congestion after the C-ECOP rearrangement has risen considerably. This is caused by the replacement of the ILP in the C-ECOP algorithm with diffusion-based cell rearrangement based on cell-flow tendencies, which means there is a possibility that in the process of moving cells away from congested region, too many cells are moved in into a same non-congested region, subsequently making that region congested. However, this is an acceptable alternative to the ILP which is unable to deal with congested regions with no cells to be moved out, as has been explained in section 3.5.

The disparity in horizontal congestion measurement can be attributed to the fact that One-Steiner algorithm's routing estimation decomposes higher-degree nets into two-degree nets that are shorter in length than the original net. As the C-ECOP probability calculation uses geometric net center as a point of reference, this means if a Steiner point of a net connecting closely-positioned cells fall inside the same bin of one cell, the number of connections are going to be vastly reduced, as given in the example in Figure 4.7.

This is not a problem if the bins are sufficiently small, however given our memory limitation, the bin size we used did not lend itself well to this issue. This will require modification to the algorithm to use the position of the Steiner point as the net center instead of the net center of a two pin nets. However, this could not be implemented in time.

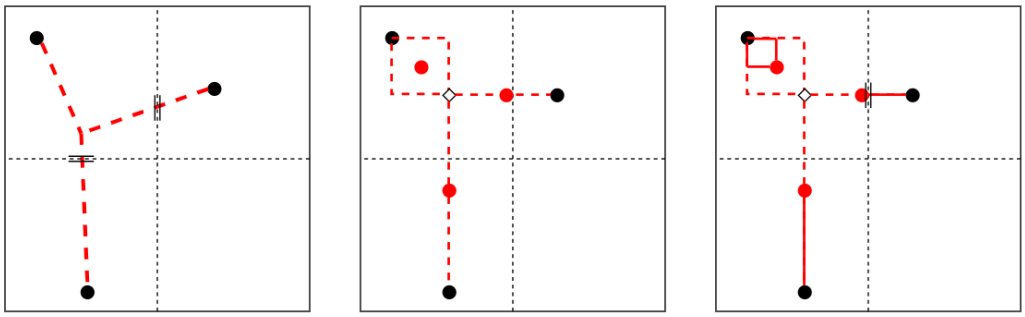


Figure 4.7: (a) A degree-3 net connecting 3 cells. The bin crossings are denoted by parallel lines. In this case, we expect 2 bin crossings, thus both the upper left bin and upper right bin will have 1 horizontal crossing, while both the upper left bin and lower right bin will have 1 vertical crossing. (b) By adding Steiner points, the routing become more determined, however due to C-ECOP net-center method (denoted by red dots), the nets are practically only half the length of what it should be, thus resulting in C where the vertical crossing between upper-left and lower-left bins is gone.

Between the DPlace + C-ECOP placement result and the original DTB multiplexer placement, measured by C-ECOP congestion estimation, the DPlace + C-ECOP placement resulted in 7.9% reduction in horizontal congestion, and 24.47% reduction in vertical congestion. However, when estimated using C-ECOP + One-Steiner congestion estimation, this resulted in 11.91% reduction in horizontal congestion, but 225.24% increase in congestion. Again, this is due to unreliability of the C-ECOP + One-Steiner method for large binsize.

It must also be noted that despite being able to achieve lower congestion, the resulting wirelength is greater than the original of the DTB multiplexer original placement. This can have negative effect on the actual wiring area as less wirelength means, even though a net has to take a detour due to congestion, the resulting area requirement will be lower. For the time being, this can only be tested by performing routing using actual place and route tool.

4.2 Encounter Routing Results

Encounter placement using EDI 14.27 yields disappointing results. The cells placed after init stage and before placement stage could not be routed due to spacing violation on virtually all BAP3 component despite all cells having been set to SOFTFIXED and theoretically EDI should be able to move the cells around to conform it to its DRC, however this does not seem to be the case.

Several enlarged version of the placement (simply by scaling the distance between the cells) also failed. Fencing the area around the DTB multiplexer also did not yield any better result. More investigation is needed to ascertain the cause of this issue

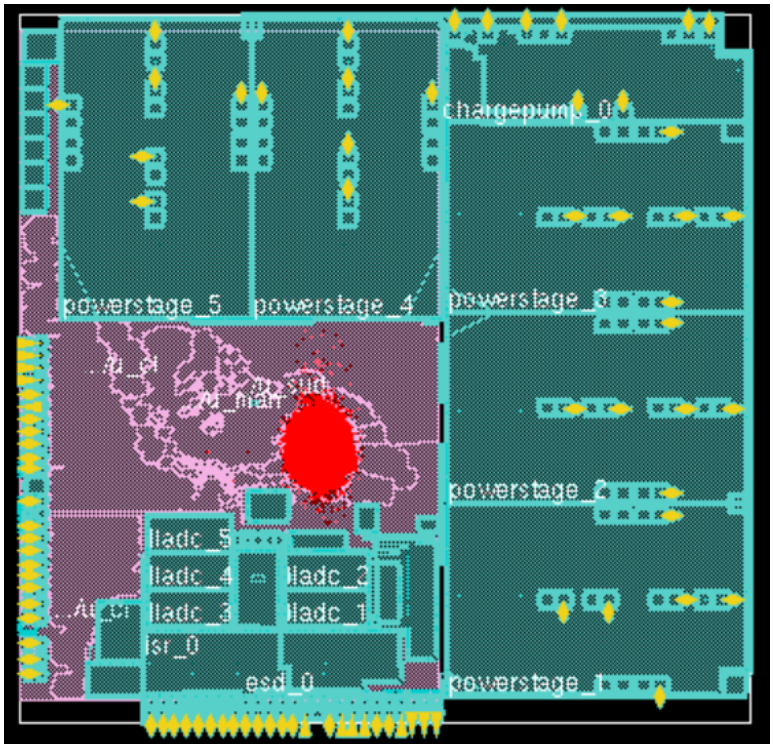


Figure 4.8: First trial place and route in EDI 14.27, the DTB Multiplexer is highlighted in red.

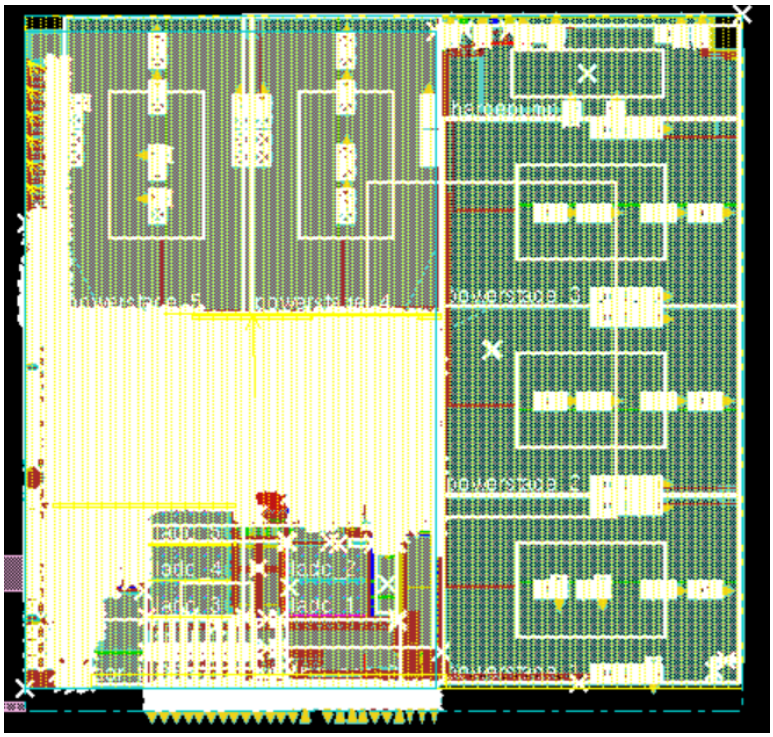


Figure 4.9: Complete routing failure immediately after route stage.

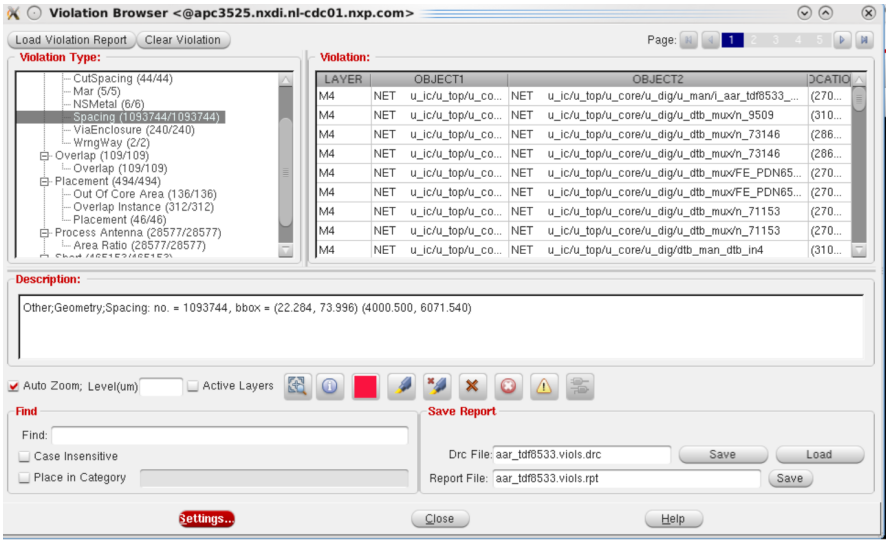


Figure 4.10: Spacing violations, almost exclusively caused by DTB Multiplexer cells.

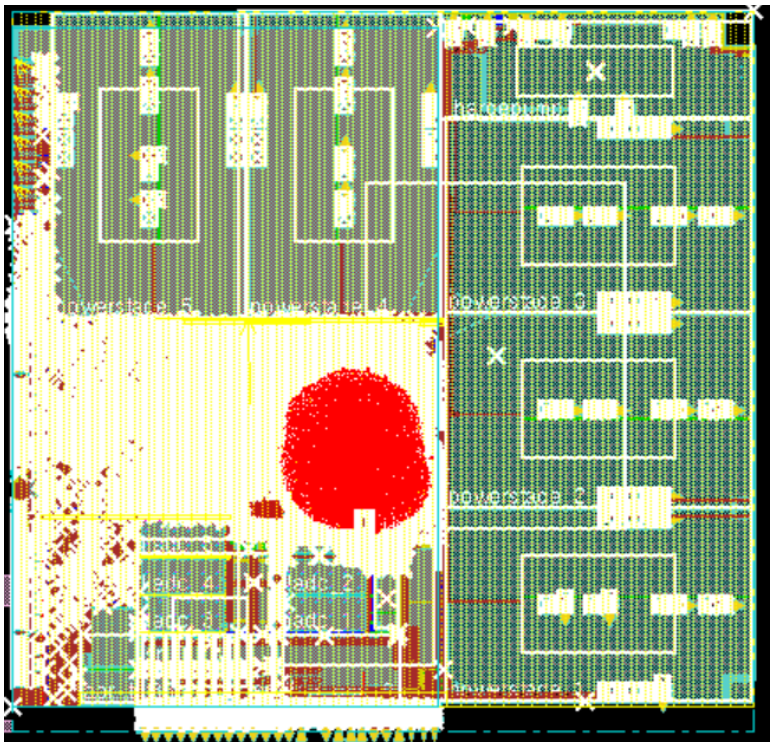


Figure 4.11: 200% scale placement of DTB Multiplexer.

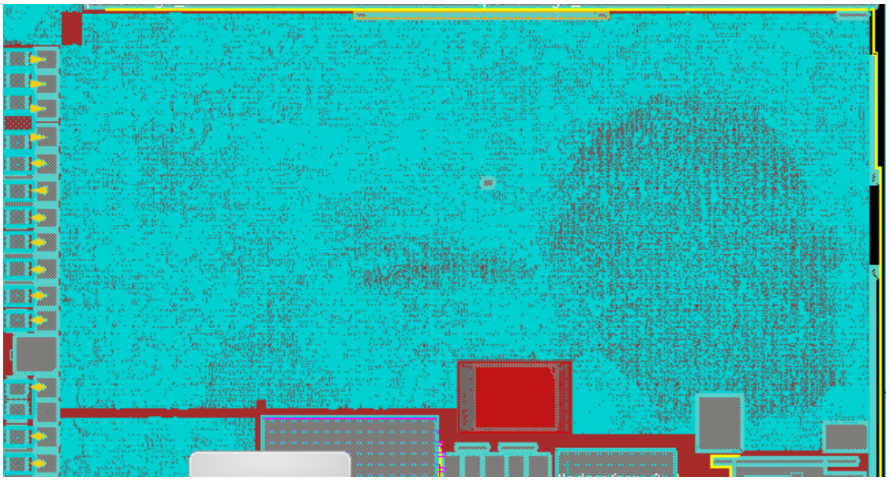


Figure 4.12: Extremely sparse DTB Multiplexer placement with no cell overlap.

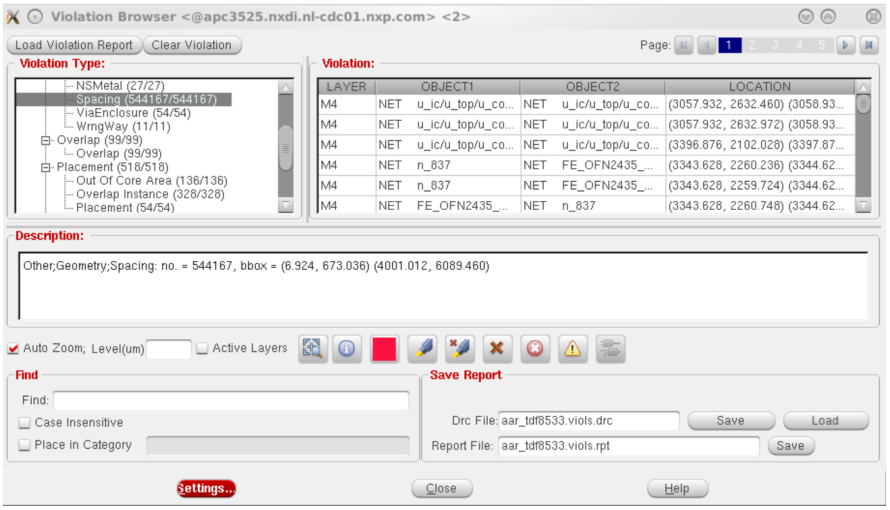


Figure 4.13: The 200% scaling placement still result in extremely high spacing violation caused by DTB Multiplexer, albeit the number is reduced by half.

Conclusion and Recommendations

5

- The modified C-ECOP algorithm is an effective method to reduce congestion in a wiring-intensive design. It was able to reduce congestion on a placed circuit by 8.22% reduction in horizontal direction, and 41.6% reduction vertical direction when measured with C-ECOP congestion estimation. While we cannot directly estimate the resulting wiring area savings from this result, it is likely to be reduced as the congestion reduction did not impact the wirelength negatively.
- C-ECOPs ILP method is unable to deal with a heavily congested bin if most of its congestion did not originate from the bin itself. This is caused by the current cost function which is implemented as a form of constraint. Recommendation for further research is to find a better cost function that combines both the maximum congested bin with neighboring bins to provide cell movement in the event of no movable cells remaining in the most congested region.
- The C-ECOP + One-Steiner congestion estimation algorithm needs further refinement as the current algorithm uses net centers as point of reference to determine wire crossing. Modified algorithm using Steiner points as the point of reference might result in a more accurate estimation, but this algorithm could not be implemented in time due to time constraint. Future recommendation aside from using using Steiner points as reference is a better optimized, batched version of the algorithm that can accommodate even larger degree nets in the prediction, thus improving estimation accuracy even further.
- The Placement algorithm used produced a sub-optimal wirelength. With a better placement tool, it is possible to obtain both wirelength and congestion improvement. Kraftwerk [15] offers better wirelength result compared to Dplace according to ISPD2005 placement contest result [26], given enough time, this could have been implemented instead, thus recommendation for future research is the use of Kraftwerk instead of Dplace to obtain initial placement result.
- More research is needed to know the routing algorithm used by Place and Route tools like Cadence Encounter to prevent total routing failure. Given the proprietary nature of the routing algorithm, it is hard to implement an external placement in the cadence EDI environment, thus recommendation for future research is to perform more routing trial using cadence EDI to come up with a placement solution that does not result in spacing errors.

Bibliography

[1] Wang et.al Electronic Design Automation Elsevier inc. 2009. Burlington, MA, USA.

[2] T. Luo, D.Z. Pan Dplace2.0: A stable and efficient analytical placement based on diffusion, in *2008 Asia and South Pacific Design Automation Conference.*

[3] M. Wang, X. yang and M. Sarrafzadeh, Dragon2000: Standard-cell-placement tool for large-Industry circuits, in *proc. Int. conf. on Computer Aided Design*, 2000.

[4] J.A Roy, D.A Papa, S.N. Adya H.H. Chan, A.N Ng, J.F. Lu, and I. L. Markov Capo: A robust and scalable min-cut Floorplacer. In *proc. ACM/IEEE International Symposium on Physical Design*, 2005.

[5] H. Ren, D.Z. Pan, C.J. Alpert, and P. Villarubia, Diffusion-based placement migration, in *proc. Design automation conf*, June 2005.

[6] Z. Li, W. Wu, X. Hong, Congestion driven incremental placement algorithm for standard cell layout, in *Proceedings of the ASP-DAC Asian and South Pacific Design Automation Conference*, 2003.

[7] A. Kahng, G. Robins, A new class of Steiner tree heuristics with good performance: the iterated One-steiner approach," *IEEE International Conference on Computer Aided-Design*, 1990.

[8] M. Hanan, On Steiners problem with rectilinear distance, SIAM J Appl. Math vol 14, pp 255-265, 1966.

[9] G. Meixner, U. Lauther, "Congestion-driven placement using a new multi-partitioning heuristic", *In Proc. Int. Conf. Computer-Aided Design*, November 1990.

[10] J Kleinhans, G. Sigl, F.M. Johannes, and K. Antreich, GORDIAN: VLSI placement by quadratic programming and slicing optimization,*IEEE Transaction on Computer Aided Design of Integrated Circuit and Systems*,vol. CAD-10, march 1991.

[11] H. Eisenmann and F.M. Johannes, Generic global placement and floorplanning, in *Proc. Design Automation Conference*, 1998.

[12] N. Vishnawatan and C.C.N. Chu; FastPlace: Efficient analyitical placement using cell shifting, iterative local refinement, and hybrid net model, *In Proc. Int Symp on Physical Design*, pp 26-33, 2004.

[13] Wenting Hou, Hong Yu, Xianlong Hong, Yici Cai, Weimin Wu, Jun Gu, W.H. Kao, "A new congestion-driven placement algorithm based on cell inflation", *Design Automation Conference 2001. Proceedings of the ASP-DAC 2001*, pp. 605-608, 2001.

[14] F. Mo, A. Tabbara, and R.K Brayton, A force-directed macro-cell placer in proc, *nt. conference on computer aided design*,p4, EECS, UC Berkeley, November 2000.

[15] P. Spindler and F,M. Johannes, Kraftwerk: A fast and Robust Quadratic Placer Using and Exact Linear Net Model, in *proc. Int. Conf. on Computer Aided Design*, 2006.

[16] L. J. Guibas, B. Chazekke, D.T. Lee, The power of geometric duality in *24th annual Symposium on Foundations of Computer Science*, 1983.

[17] M.R. Garey, L.J. Stockmeyer, D.S. Johnson, Some Simplified NP-Complete Problems, *roc of the 6th Annual Symp. On Theory of Computing*, 1974.

[18] C.J. Alpert The ISPD98 Circuit benchmark suite, In *Proc. International Symposium on Physical Design*, pp.80-85 1998.

[19] Z J. D. Plummer, M. D. Deal, P. B. Griffin, Silicon VLSI Technology: Fundamentals Practice and Modeling, NJ, Englewood Cliffs: Prentice-Hall, 2003.

[20] W.C. Naylor, R. Donnely, and L. Sha, Non-Linear Optimzation System and Method for Wirelenght and Delay Optimization for Automatic Electric Circuit Placer, US Patent 6,301,693,2001.

[21] T. Luo, H. Ren, C. J. Alpert, D. Z. Pan, "Computational geometry based placement migration", *Proc. Int. Conf. Comput.-Aided Des.*, pp. 41-47, 2005.

[22] M. Wang, X. Yang, M. Sarrafzadeh, "Congestion minimization during placement", *Computer-Aided Design of Integrated Circuits and Systems. IEEE Transactions*, [22]vol. 19, no. 10, pp. 1140-1148, Oct. 2000.

[23] M. Wang, M. Sarrafzadeh, "Modeling and minimization of routing congestion, " *Design Automation Conference 2000. Proceedings of the ASP-DAC 2000* pp. 185-190, 2000.

[24] M. Wang, M. Sarrafzadeh, "On the Behavior of Congestion Minimization During Placement," *International Symposium on Physical Design*, pp. 145-150, April 1990.

[25] C. Sechen, A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package," *IEEE Journal of Solid-State Circuits Vol.20*, Issue 2 , April 1985.

[26] G. J. Nam, C.J. Alpert, P. Villarubia, B. Winter and M. Yildiz, The ISPD2005 Placement Contest and Benchmark Suite, in *Proc. Nternational Symposium on Physicla Design*[26], (New York, NY, USA), ACM, 2005.

[27] M.C. Yildiz and P.H. Madden, Improved Cut Sequences for Partitioning-Based Placement, in *proc. International Conference on Computer Aided Design*, 2000.

[28] T. Chan, J. Cong, and K. Sze, Multilevel Generalized Force-Directed Method for Circuit Placement, in *Proc. International Symposium on Physical Design*, 2005.

[29] NXP Semiconductors, "DTB Mux Component Requirement Specification" BAP3 Product Specification, April 2013 [Revised April 2013]

[30] Tveit, Amund., "On the Complexity of Matrix Inversion. ", 2003.

[31] Mathworks, "Sparse Matrix Operations: Computational Complexity," *MATLAB R2019b documentation*, 2019. [online]. Available: https://nl.mathworks.com/help/matlab/math/sparse-matrix-operations.html [Accessed April 15, 2019]