Automated, Inter-Macro Channel Space Adjustment and Optimization for Faster Design Closure

K K Praveen Kumar

VLSI & Embedded System

Indraprastha Institute of Information Technology Delhi

Under the supervison of Dr. Alexander Fell

A thesis submitted for the degree of

Master Of Technology

May 2016

Supervisor

Dr. Alexander Fell, IIIT-Delhi (Internal)
Sachin Mathur, ST Microelectronics (External)

Examiner

Dr. Hitesh Shrimali, IIT-Mandi (External)
Dr. Sneh Saurabh, IIIT-Delhi (Internal)
Dr. Sujay Deb, IIIT-Delhi (Internal)

Date of Defense - June 16, 2016

I dedicate my work to those who believe in ethics, standards, honesty, integrity and hardwork as pillars of success beyond the end result...

Acknowledgements

I would like to thank Dr. Alexander Fell for the enthusiastic discussions, his guidance, support, encouragement, insight and suggestions without which the work would not have achieved fruition.

I would like to extend my thanks to Sachin Mathur for his valuable support, his immense knowledge in the physical design domain, insights during the meetings and for pushing me to achieve the necessary goals which formed the basis of my thesis.

Further, I would like to thank Neeraj Kumar, Sumit Kumar Singh and Tamal Sinha for helping me out with the understanding of the design metrics, flow and challenging me with questions which ultimately helped me for the better understanding of the physical synthesis flow. Special thanks to Rajesh Gupta for validating the results for Design-B.

Last but not the least, I wish to thank Dr. Sujay Deb and Sri Harsha Gade for introducing me to the domain of physical design.

Abstract

Achieving optimal floor-plans during the physical synthesis flow is an iterative and resource intensive process and its quality has a significant impact on subsequent synthesis stages in terms of runtime and quality of results. This problem intensifies due to the abundance of macros in advance technology nodes which poses challenges in the physical design flow, especially in the floor-plan stage. It has resulted in an excessive number of channels among macros that need to be spaced carefully and optimized as they consume placement and routing resources. The work presented here is two-fold: First, a tool is introduced for automatic channel space adjustment. Second, the impact of channel space minimization on the quality of results along with runtimes are investigated. Experimental results for two complex partitions of a taped out design, Design-A and Design-B, each with 3M instances including 225 and 205 macros respectively, are presented. The results indicate an existence of an optimum channel spacing in which a 35% and 124% reduction in turn-around-time is observed with same or better quality of results, when compared to the taped out version of the same.

Contents

Abstract iv

Contents v

List of Figures vi

List of Tables viii

1 Introduction 1
- 1.1 Related Work . . . 2
- 1.2 Organization . . . 2

2 Channel Spaces 3
- 2.1 Resource Estimation . . . 3
- 2.2 Experimentation . . . 4
  - 2.2.1 Methodology . . . 4
  - 2.2.2 Observations . . . 5
  - 2.2.3 Formulations . . . 6
- 2.3 Automation . . . 7

3 Results 9

4 Conclusion and Future Work 14

References 15

A Preliminaries 17

B Quality Indicators 22

C Area Reduction 24

D Impact within Channels 26

List of Figures

2.1 Different macro placement configurations with horizontal channels (HC) and vertical channels (VC). 5
2.2 A typical macro cluster with annotated channels as per configuration shown in figure 2.1a. 5
2.3 Layer utilization under different macro porosity condition for macro configuration shown in figure 2.1a. 6
2.4 Wire length distribution with CDF. 6
2.5 Computational representation of a macro cluster of configuration shown in figure 2.2. 7

3.1 Floor-plans of designs used for case study. 10
3.2 A macro cluster of Design-A. Channels are adjusted using the tool. 10
3.3 Core Area Utilization and Average Channel Congestion vs CDF. 10
3.4 Overflow (Horizontal and Vertical) vs CDF. 11
3.5 Total negative slack and violating paths (setup and hold) vs CDF. 12
3.6 Design Rule Checks vs CDF 12
3.7 Wire length vs CDF 12
3.8 Runtimes of the designs vs CDF 13
3.9 Overall synthesis effort vs CDF 13

A.1 Placement resources 17
A.2 Typical cell placement 18
A.3 Routing resources 18
A.4 Routing grid 19
A.5 A typical floor-plan 20
A.6 Different types of core area 21

B.1 Different core area utilizations 22

C.1 Different versions of Design-A 24

D.1 Shorts (black crosses) within channels as CDF is varied in Design-A - I 27
D.2 Shorts (black crosses) within channels as CDF is varied in Design-A - II 28
D.3 Shorts (black crosses) within channels as CDF is varied in Design-B - I 29

D.4 Shorts (black crosses) within channels as CDF is varied in Design-B - II 30

List of Tables

3.1 Design statistics 9

A.1 Disparity in resource consumption between a macro and a standard cell 20

C.1 Normalized runtime comparisons of different versions of Design-A 25

Chapter 1

Introduction

In a VLSI design cycle, the physical synthesis is a process which transforms the netlist (mapped to a given technology node such as 28nm) into actual physical layouts/geometries for fabrication while meeting the design constraints such as timing, area, power and routability [1, 2]. It is composed of floor-planning, placement, clock tree synthesis and routing stages. The quality of the design at each stage has a significant impact on the succeeding stages [3] in terms of runtime of the EDA tool.

Scaling has enabled complex designs as shrinking technology nodes has resulted in integration of more functionality to the existing and current designs. Advancing technology nodes add more metal layers, varied metal thickness, complex design rules etc. [4] ultimately impacting the physical synthesis flow which is computationally intensive. To alleviate these complexities, today's complex SoC are partitioned into several blocks which undergo physical synthesis flow in parallel thereby optimizing the available compute and human resources.

Floor-planning is the first step in the physical design flow and one of the emerging challenges in this stage is the growing number of macros, IPs and embedded memories in the design [5, 6]. Abundance of macros has resulted in an excessive number of channels that need to be spaced carefully and optimized. The channels are intrinsic to macro placement and space is reserved by placement blockages to prevent an interference of the standard cells that might impact the closure of the design (especially routing). Macros are stacked along the periphery of the chip [6, 7] and their placement is responsible for the creation of the core area which is utilized by succeeding stages of the physical synthesis flow.

The channels present among macros, consume placement and routing resources. The knowledge of resource utilization in these channels is conceived after placement and global routing which, depending on the design complexity, is compute intensive. Manual spacing of channels involves multiple iterations of the placement and global routing stages to achieve an optimal solution. In order to minimize the impact of an iteration on the turn-around-time, optimal solutions are never achieved in general. Also, manual channel space adjustment is unreliable and tedious due to vast amount of channels that exist in a design.

In order to minimize the impact of manual spacing on the turn-around-time, a tool to accomplish an automated channel spacing, is proposed. In addition, to optimize the channel spaces, spacing formulations are further developed based on experimentations on most recurring patterns of macro placement. For this purpose, a router originally derived from the work in [8], has been extended.

1.1 Related Work

The physical synthesis domain consists of computationally intensive problems. Advanced technology nodes along with increased number of instances add more complexity to the problem space which results in increased runtime to achieve optimal solutions. Runtime reduction while maintaining same or better quality of results with better design methodology or flow, is an active research area [9].

Most of the work relating to the handling of macros, falls into the category of mixed-size placement with the primary objective being the removal of overlaps (legalization), optimization (for routability) and provision of better macro placement in terms of wire length and runtimes [6]. Of these, the existence of channels between macros is addressed in [7, 10], albeit briefly. In either case, the technique of inflating the macro sizes is used to either reserve routing resource around a macro [7] or block the narrow channels (in case of fixed macro) [10].

To the best of our knowledge, explicit evaluation of the impact of channel spaces and its minimization on the entire physical synthesis flow has not been studied, which forms the basis of the work presented here.

1.2 Organization

The report is organized into four chapters. Chapter 1 introduces the reader to the report and briefly explains the problem, its impact and related work. Chapter 2 describes the spacing formulations developed along with the tool and its algorithm. The experimental results are presented in chapter 3, while chapter 4 concludes the report.

Chapter 2

Channel Spaces

The channels between macros consume placement and routing resources (see appendix A.1 for further information). Routing resources are further constrained due to macro porosity. After macro placement, the channels are allowed to exist so as to provide enough placement resources for insertion of buffers to prevent any *expected* timing issues and to provide enough routing resources to prevent any *expected* congestion. Under-utilization of these resources creates unused spaces or white-space while over-utilization results in congestion. Congestion impacts design closure while white-space indicates a room for optimizing the design. Usually, the spacing is performed manually on selected channels based on the number of pins, global routing data or from experience during floor-planning, keeping the aspect of design closure. The algorithmic behavior of the subsequent routing stage is not taken into account.

2.1 Resource Estimation

The state-of-the-art technique to estimate the resource utilization, especially routing resources, requires global routing which can be only invoked after the placement of the design is complete. The channels which are congested or with white-spaces, are identified and their spacing is accustomed to the required resources. This trivial approach has drawbacks, foremost of which, is the channel adjustment itself. Manual alteration of channel space involves a revision of the macro placement. Since macros consume a significant amount of placement and routing resources (see appendix A.2.1 for further information), any relocation of macros or channel space adjustment impacts its surrounding resource allocations extensively. Likewise, if the macro is placed at the periphery of the core area, the recovered resources in the core area might result in a completely different floor-plan and a placement solution (see appendix A.2 and A.2.2 for further information). Both of the aforementioned facts render the global routing solution irrelevant for all the other channels except the adjusted channel. In order to accurately adjust the channels, placement and global routing needs to be performed iteratively for each channel which is infeasible. Both placement and global routing are runtime intensive and their reiterations need to be avoided. A routing resource estimation without placement of the design and hence global routing, require the development of spacing formulations and heuristics.

2.2 Experimentation

Initially, to analyse the approach of spacing channels between macros without performing placement and global routing, a hypothesis was developed and tested on Design-A (see table 3.1) with the primary objective of achieving routing closure.

Hypothesis 1 *Given a legal macro placement, for a macro of M pins, a channel space of $\frac{M}{2}$ at the port-side between adjacent macros suffices the routing requirement for the macro under consideration.*

The hypothesis is developed considering each pin of a macro as a net. The routing resources are halved to accommodate the fact that the connected instances are placed together and assuming they are placed on either side of the macro. Since spacing was accomplished manually, only horizontal channels were adjusted. The design tested under hypothesis condition achieved routing closure.

Assessing the routability of the macros without placement of the design necessitates the development of a router which performed detailed routing such as the one introduced in [8]. This router has been extended to emulate the routing conditions of the technology node. Specifically, the developed router supported six layers for routing using the vertical-horizontal-vertical (VHV) routing model.

2.2.1 Methodology

Iterative or annealing techniques using channel space as cost and routability as a constraint were not used for optimization due to their time complexity and the scale of the design. Instead, the routability of the most recurring placement pattern of macros which are shown in figures 2.1 and 2.2, are verified using the developed router.

The patterns such as the one in figure 2.1a, were re-created as shown in figure 2.3. In this figure, the port of macro-1 is shown at the bottom and the blue region depicts resource blockages (in terms of macro-2 and other macros). To test the routability of this pattern, the sources are assigned on the port-side of macro-1 while the targets were distributed in the core area (refer to figure 2.1a). This is realized to maximize the connectivity of a macro with standard cells. Each pin of the macro is assumed as a two-point net as macro porosity within the channels would prevent its fanning out. Also, the macro porosity is initially extended up to the fifth layer to ascertain the exact required routing resource. Later, it is reduced to the fourth layer to emulate the technology node under test. The routing resources within the channels are sized akin to the technique used in [7] for inflation with a few modifications: Instead of inflating the macro sizes by the required amount of routing resources before their placement, the channels (after obtaining a legal macro placement from the EDA tool) which are allowed to exist, are spaced based on the pin count of the macro, its relative location within a cluster and its orientation. For a net to be routed completely, it requires at least two layers with alternate routing preferences, i.e. a vertical layer followed by a horizontal layer. This fact is

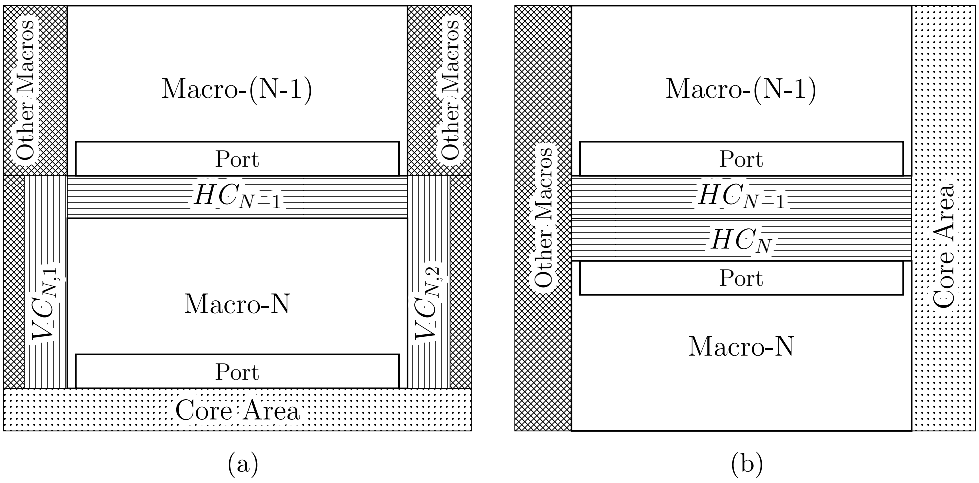


Figure 2.1: Different macro placement configurations with horizontal channels (HC) and vertical channels (VC).

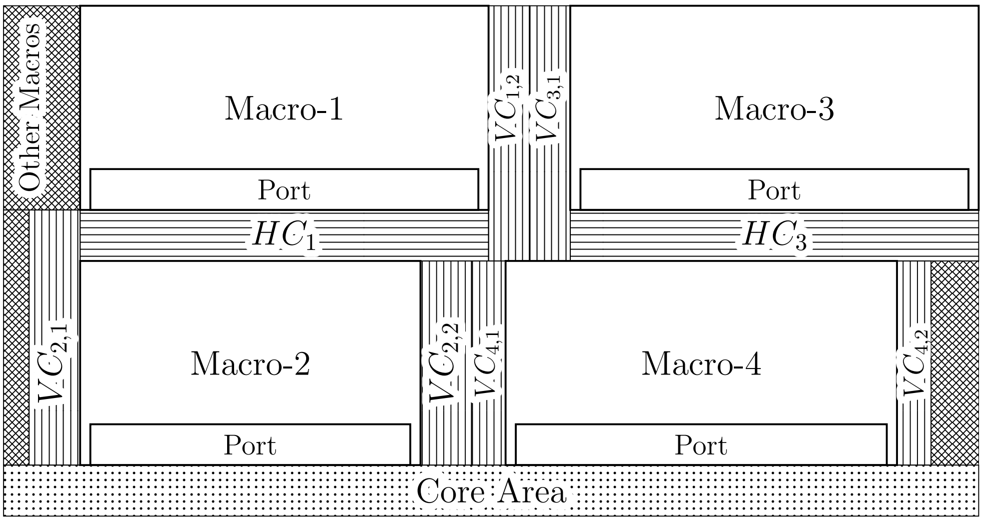


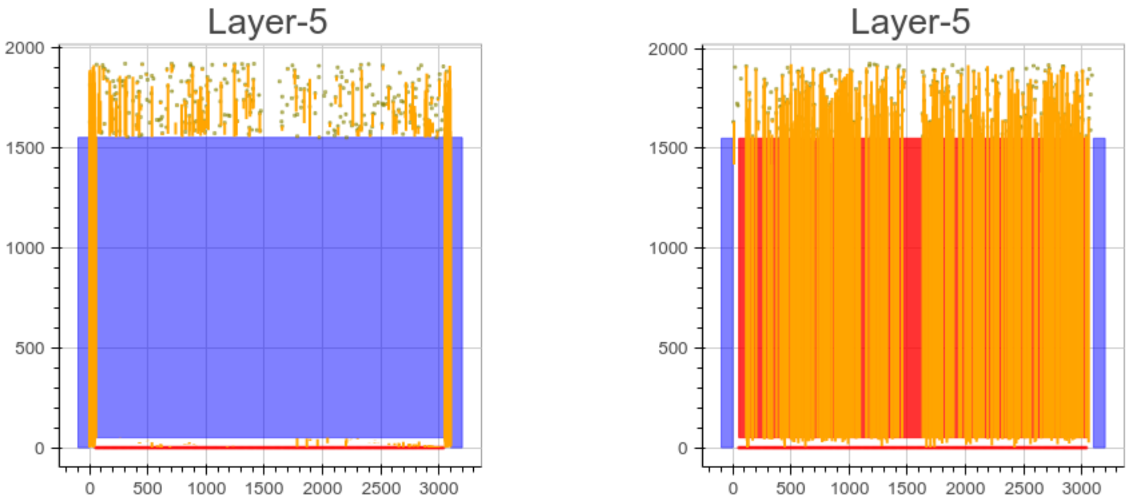
Figure 2.2: A typical macro cluster with annotated channels as per configuration shown in figure 2.1a.

used to constraint the spacing, hence reducing the routing resources (within the channels), with *Channel Division Factors* (CDF) in increments of two. Hence, for the 28nm technology node, CDF varies from two to six as there are six signal routing layers available.

The routability of macro placement pattern depicted in figure 2.1a is tested as this is observed to be the primal configuration. Other configurations can be derived from the primal configuration by stacking macros vertically similar to figure 2.1a, including orientation constraints (figure 2.1b) or the dimensions of the macro (figure 2.2).

2.2.2 Observations

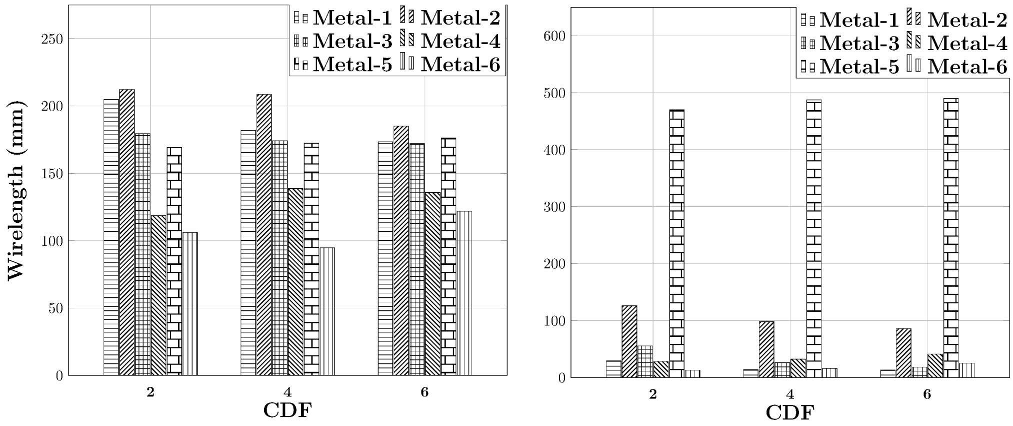
The first observation is that given the exact routing resources in channels, routing closure was achieved. From the wire length distribution across different layers (figure 2.4a and 2.4b), it can be observed that, after reducing the macro porosity by a layer, the nets preferred direct



(a) Macro porosity up to the 5th layer (Minimal routing resources)

(b) Macro porosity up to the 4th layer (28nm)

Figure 2.3: Layer utilization under different macro porosity condition for macro configuration shown in figure 2.1a.



(a) Macro porosity up to the 5th layer.

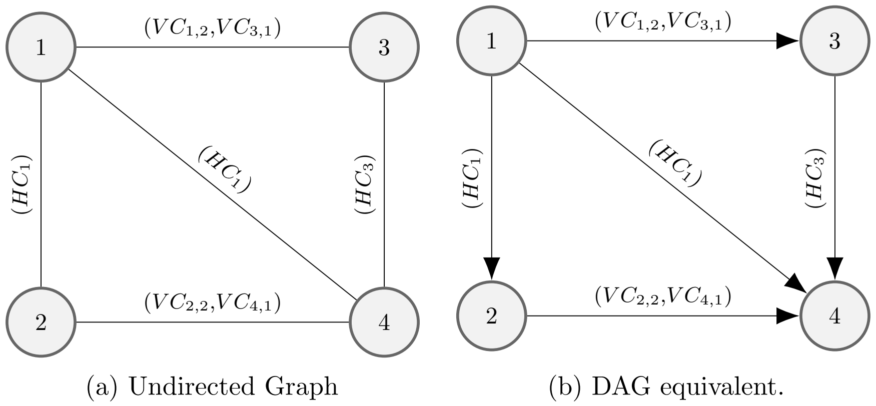
(b) Macro porosity up to the 4th layer.

Figure 2.4: Wire length distribution with CDF.

traversal over the macro (using the fifth vertical layer) instead of detouring around channels to minimize the wire length irrespective of channel space constraints (figure 2.3b and 2.3a). Both of these observations can be attributed to [11]. Finally, constraining channel increases the channel congestion.

2.2.3 Formulations

Equations 2.1 and 2.2 represent the formulations for horizontal and vertical channels respectively. Equation 2.1 can be considered as a natural extension of the inflation methodology described in [7]. Equation 2.2 is aware of the neighbours and the dimensions of the macro. Both of these formulations are used by the proposed channel adjustment tool without iterative



(a) Undirected Graph (b) DAG equivalent.

Figure 2.5: Computational representation of a macro cluster of configuration shown in figure 2.2.

annealing techniques.

$$HC_N = \frac{M_N \times P}{CDF}. \tag{2.1}$$

$$VC_{N,L} = \frac{W_N \times P \times (M_{N-1} + \ldots + M_1)}{W_{N-1} \times CDF} \tag{2.2}$$

In above equations, M_N corresponds to M pins of N^{th} macro. P is the pitch of the technology node. CDF is channel division factor. W_N corresponds to width of N^{th} macro. HC_N and $VC_{N,L}$ are the horizontal and vertical channels of the N^{th} macro while L is either 1 or 2 indicating left or right vertical channel, respectively. Unlike in [7], the numerator of equation 2.1 and 2.2 is not divided by all available layers as all are not available for signal routing in actual industrial designs. Spacing the channels using all available layers, impacts the synthesis flow severely as evident from the results presented in chapter 3. Likewise, the CDF is provided as an input to the tool.

2.3 Automation

A legal macro placement as shown in figure 2.2, is translated into a graph $G(V, E)$ by the tool, in which V represents the set of macros and E the set of channels as shown in figure 2.5a. The graph is partitioned to obtain subgraphs which represent macro clusters such as the one shown in figure 3.2a. Channel spacing is performed in each subgraph using graph traversal techniques which can be considered as a compaction problem. The location of each subgraph is determined relative to the core by inspecting the orientation information of all the vertices which is used to determine the direction of graph traversal (see figure 2.5b). Using the indegree information, the vertices near to the chip boundary are identified and traversed till there are no more neighbours in the given direction. During the vertical traversal, the horizontal channels are spaced using equation 2.1 while equation 2.2 is applied to the space allocation of the vertical channels during the succeeding horizontal traversal. The spacing constraints in terms of CDF

Algorithm 1 Horizontal Channel Spacing Algorithm

```
Require: Legal Macro Placement (from EDA tool), CDF
Ensure: Altered Macro Placement
procedure MacroToGraph($G(V, E)$)
    $MacroGraph \leftarrow \varnothing$
    for each $v \in V$ do
        $N \leftarrow$ findNeighbors($v$)
        addVertex($v$, $MacroGraph$)
        for each $n \in N$ do
            addVertex($n$, $MacroGraph$)
            addEdge($n$, $v$, $MacroGraph$)
        end for
    end for
    return $MacroGraph$
end procedure
procedure ChannelSpaceV($MacroGraph$, $CDF$)
    $MacroCluster \leftarrow$ partitionGraph($MacroGraph$)
    for each $m \in MacroCluster$ do
        if inDegreeVert($m$) $= 0$ then
            $start \leftarrow m$
            updateCoord($start$, 0, $MacroGraph$)
            $N \leftarrow$ findVertNeighborBFS($start$)
            for each $n \in N$ do
                updateCoord($n$, $HC_N$, $MacroGraph$)
            end for
        end if
    end for
    return $MacroGraph$
end procedure
```

can be provided to the tool by the user. All macro clusters can be spaced with the same or different spacing constraints depending on the design. The pseudo code for the horizontal channel spacing is listed in algorithm 1. The algorithm for vertical channel spacing is similar to algorithm 1, but uses equation 2.2 for spacing instead. The time complexity of the algorithm, hence the tool, is dominated by neighbourhood search which is $\mathcal{O}(V^2)$. The time complexity of the channel spacing is $\mathcal{O}(V + E)$. Also, the proposed tool is generic and can be integrated with the flow developed with any commercial EDA tool.

Chapter 3

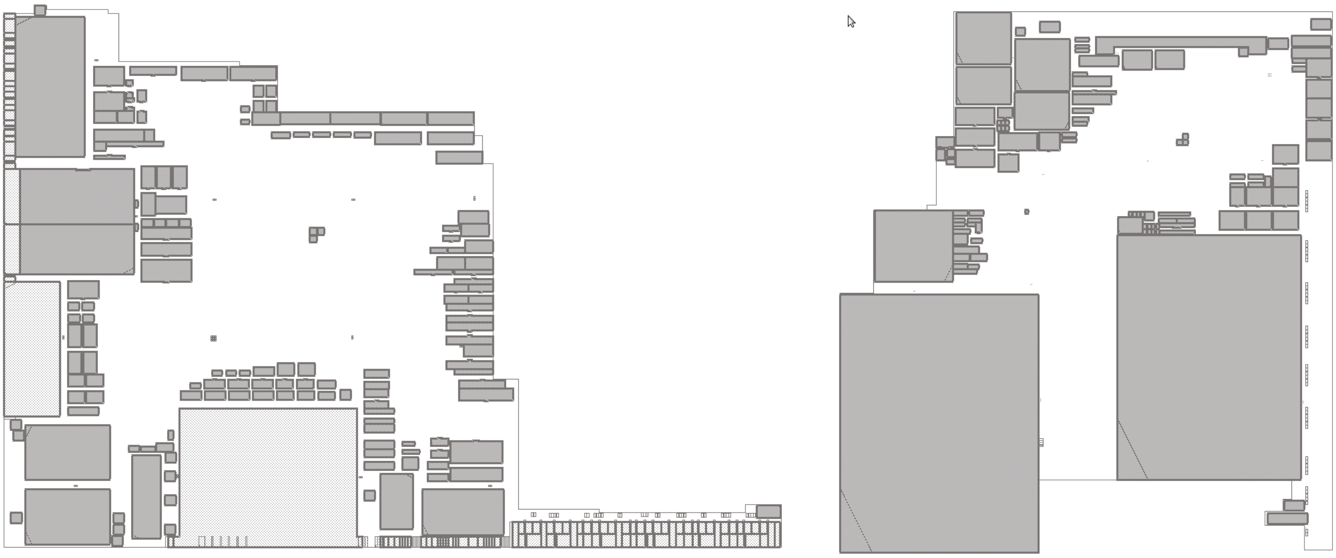
Results

Two designs, Design-A and Design-B (figures 3.1a and 3.1b), are used as a case study. The designs statistics are given in table 3.1. Design-A had a large number of movable macros while Design-B had a complexity impacting its closure during its tape out. Since the designs are already taped out, for a fair comparison, the channel between the movable macros is adjusted. Each of the designs and its iterations with reduced channel spacing in terms of CDF is pushed through the physical synthesis flow using the latest iteration of the state-of-the-art EDA tool from Cadence. The recovered resources from the channels are either re-allocated to the core area or to the neighbouring block resulting in area reduction. The results presented in this chapter involve resource re-allocation from the channels to the core area while the results involving area reduction are presented in appendix C. Due to highly uncorrelated dimensions of the macros in the designs, initially, the macro clusters are individually handled and all clusters are spaced using a common CDF. The runtimes of each stage along with the quality of results (see appendix B) are recorded from the reports generated by the EDA tool and presented. In each of the results, CDF=1 corresponds to taped out version. CDF is varied from 2 (maximal resource in channel) to 8 (minimal resources) as shown in figure 3.2.

Figures 3.3a and 3.3b show the quality of results along with variations induced due to different channel spacing (in terms of CDF) in the design after placement and global routing. In Design-A, the overall core area utilization (see appendix B.1 for further information) reduces by 13.3% while for Design-B it reduces by 6.02% when compared to their taped out versions with minimal (CDF=8) channel spacing. The average channel congestion increases as channels are constrained and vary by a maximum of 35.8% and 7.5% for Design-A and Design-B when CDF=1 and CDF=8 are compared, respectively. The impact of congestion within the channels due to the variations in CDF are presented in appendix D. The overflow (see appendix B.2

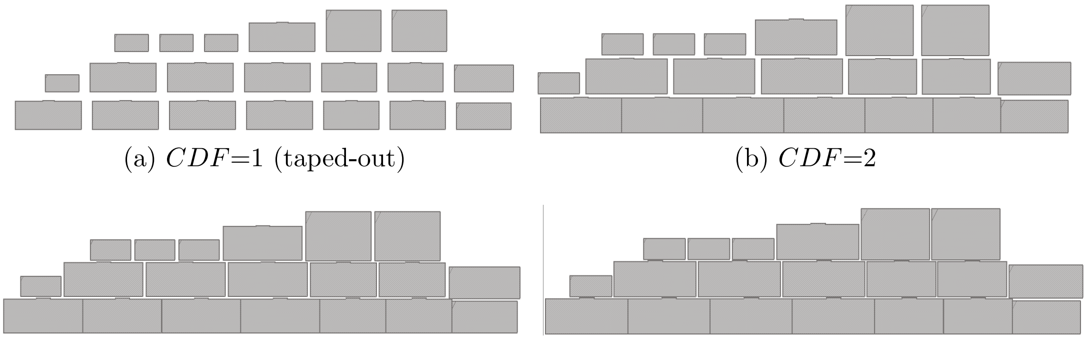
Design	Cells (M)	Movable Macros	Fixed Macros	Nets (M)	Channels
A	3	131	94	3	736
B	3	65	140	3	569

Table 3.1: Design statistics



(a) Design-A (b) Design-B

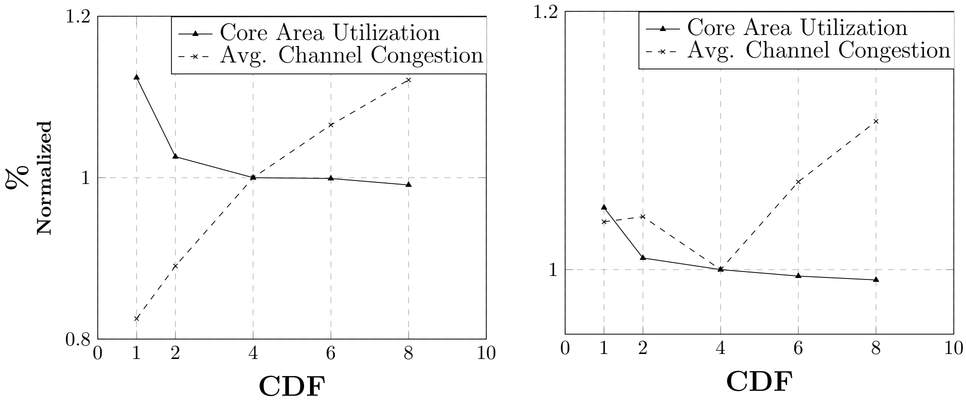
Figure 3.1: Floor-plans of designs used for case study.



(a) *CDF*=1 (taped-out) (b) *CDF*=2

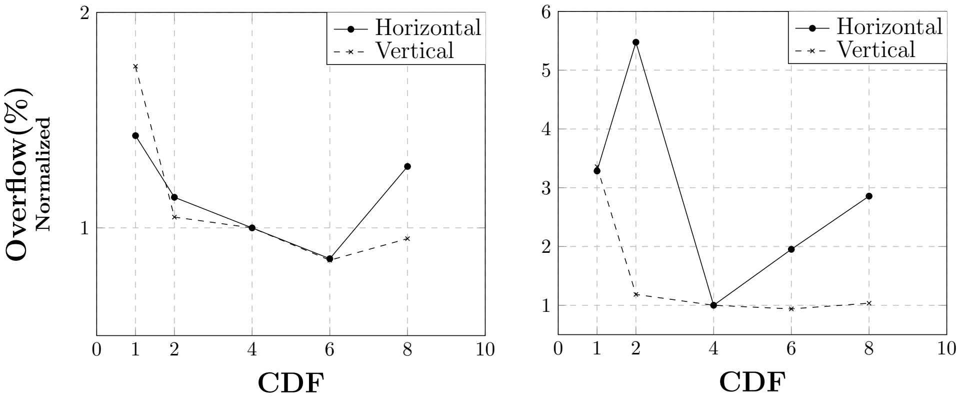
(c) *CDF*=4 (d) *CDF*=6

Figure 3.2: A macro cluster of Design-A. Channels are adjusted using the tool.



(a) Design-A (b) Design-B

Figure 3.3: Core Area Utilization and Average Channel Congestion vs *CDF*.



(a) Design-A

(b) Design-B

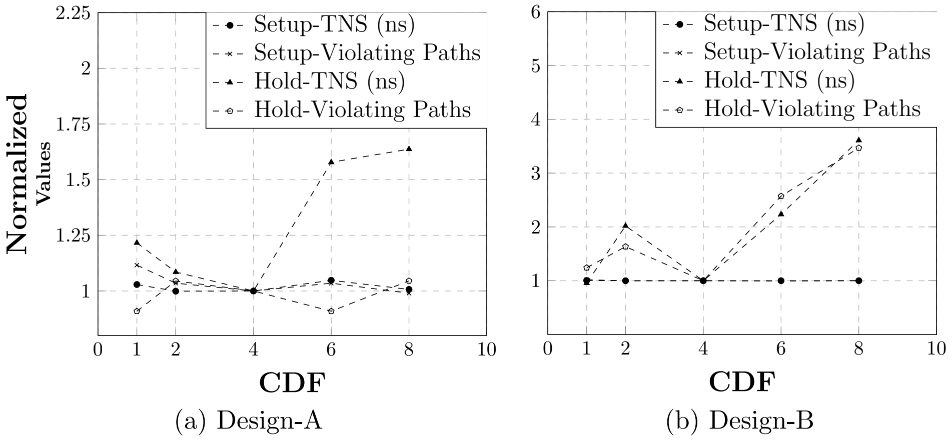
Figure 3.4: Overflow (Horizontal and Vertical) vs CDF.

for further information) metrics are shown in figures 3.4a and 3.4b. In Design-A, the overflow (horizontal and vertical) show a decreasing trend till CDF=6 post which values increases. A similar trend is observed in Design-B till CDF=4. In either case at CDF=4, the overflow reduces by 42.8% and 75% for horizontal and vertical directions while it reduces by 228% and 235% for the same in Design-B which indicates that the design can achieve faster closure in terms of routing.

Figures 3.5a and 3.5b show the variations in setup and hold timings in terms of total negative slack along with violating paths (see appendix B.3 for further information) for each. The results for setup and hold timings for the designs were obtained after clock tree synthesis and routing respectively. At CDF=4, Design-A achieves an overall reduction in total negative slack from the hold perspective by 21.6% along with 10% increase in violating paths when compared to taped out version. For Design-B, hold timings worsen by 4.4% along with 24% reduction in violating paths. In Design-A, the total negative slack for setup observe a dip of 2.9% at CDF=4 with 11.6% reduction in violating paths. In Design-B, the variations in total negative slack and violating paths for setup timings are negligible. Improvement of these values can be attributed to the overall decrease in core utilization resulting in availability of additional resources for buffers insertions to meet the timing constraints of the design. As channels are further constrained i.e. the resources within the channels are reduced, the timing worsens either in terms of total negative slack, violating paths or both.

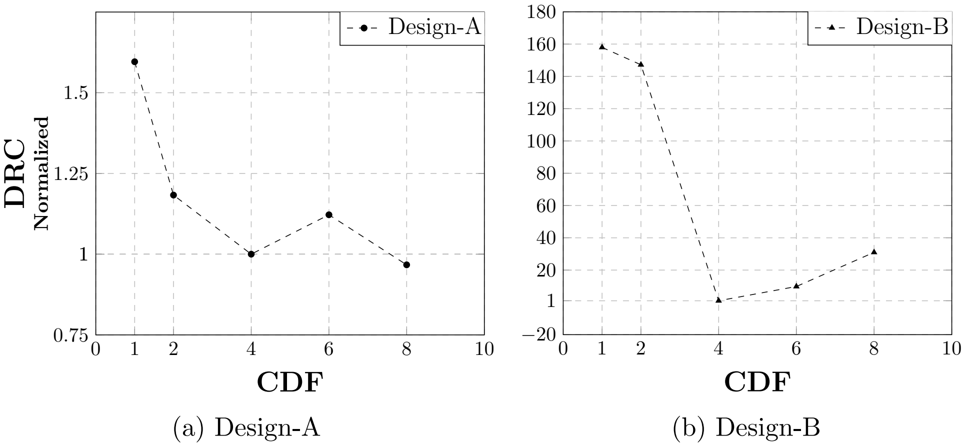
Figures 3.6a and 3.6b, show the total design rule checks (see appendix B.4 for further information) for the designs. It decreases for $CDF \leq 4$ as core utilization decreases and then increases within channels. The wire length of the designs are shown in figure 3.7. When compared to taped out version, Design-A at CDF=4 achieves a reduction of 59% in design rule checks while wire length increases by 3%. Similar results are seen for Design-B. At CDF=4, Design-B achieves an overall reduction of design rule checks by a factor of 150 with a wire length improvement of 4%. In either case, lower number of design rule checks directly translates to a reduced effort required to achieve closure of the design.

Figures 3.8a and 3.8b, show the variation of runtimes as channel spaces are reduced. For



(a) Design-A (b) Design-B

Figure 3.5: Total negative slack and violating paths (setup and hold) vs CDF.



(a) Design-A (b) Design-B

Figure 3.6: Design Rule Checks vs CDF

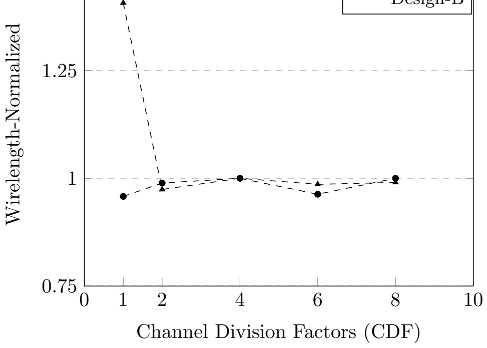
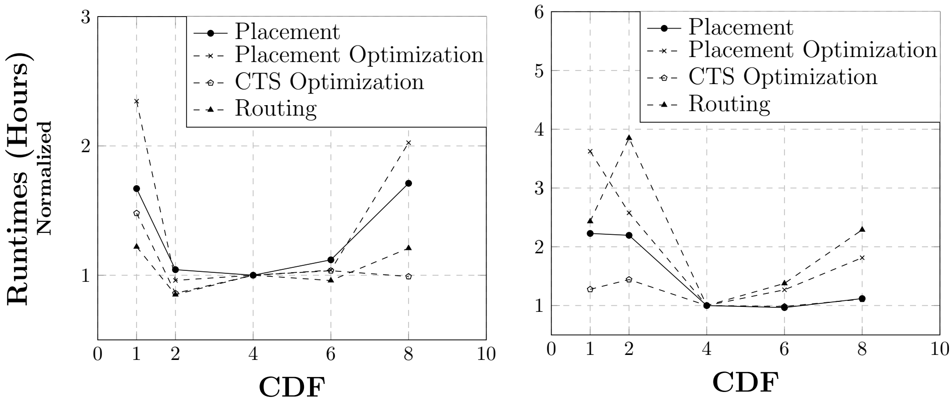


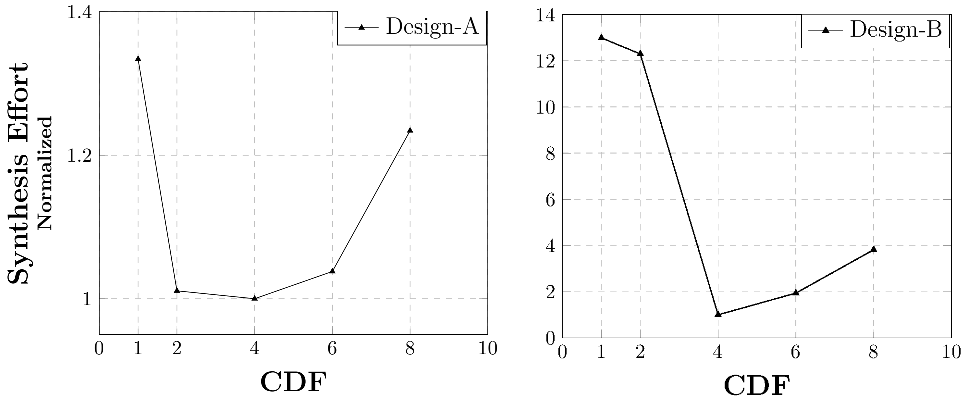
Figure 3.7: Wire length vs CDF



(a) Design-A

(b) Design-B

Figure 3.8: Runtimes of the designs vs CDF



(a) Design-A

(b) Design-B

Figure 3.9: Overall synthesis effort vs CDF

Design-A, the runtimes are minimum at CDF=2 but the quality of results in terms of timing and design rule checks are not optimal. At CDF=4, the runtimes are higher by 10.1% but with better quality of results. For Design-B, the minimum runtimes are observed at CDF=4 along with better quality of results.

Overall at CDF=4, Design-A achieves an reduction of 35% in turn-around-time, while Design-B achieves an reduction of 124% with better quality of results when compared to their taped out versions (CDF=1). In order to visualize the optimum, all results are added and normalized. Since the data is derived from the quality of results and runtimes of the EDA tool of the design at each stage of physical synthesis flow, the data can be regarded as the total combined *effort* required by the EDA tool as well as the design engineer to achieve design closure. Since the change in *effort* is realized by optimizing the channel spaces between the macros, it can be concluded that at the optimal channel spacing (CDF=4), the overall *effort* required by the EDA tool and design engineer is minimal thereby achieving faster design closure. The optimum can be clearly observed from figure 3.9a for Design-A and from figure 3.9b for Design-B.

Chapter 4

Conclusion and Future Work

A tool has been proposed to adjust the channel space among macros. The channels are spaced based on the relative location of a macro within a cluster, its orientation and number of pins. Formulations used for macro inflation were extended, developed and applied to space the channels in the design. Based on the experimentation on two taped out designs, it was found that a channel spacing with CDF=4 provided an optimum condition for the entire physical synthesis flow. The total effort required by the EDA tool or the design engineer to achieve design closure, was observed to be reduced by 34% for Design-A and by a factor of 13 for Design-B. It was also found that abutting the macros to achieve maximal core area has a detrimental effect on the synthesis.

The work presented here is a proof of concept. The experimentations were conducted using the latest iteration of the EDA tool by Cadence using a basic design flow of placement, clock tree synthesis, and routing. Since, the industry uses different tools and different design flows at different stages of the design, one of the possible future work is to validate the observations using a mix of different EDA tools in varied designs.

References

[1] Z. Li and C. J. Alpert, “What is Physical Synthesis ?” *ACM/SIGDA E-Newsletter*, vol. 41, no. vol-41, pp. 1–3, 2011.

[2] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation*, 2nd ed. Norwell, MA, USA: Kluwer Academic Publishers, 1995.

[3] C. Alpert, Z. Li, G.-j. Nam, C. N. Sze, N. Viswanathan, and S. I. Ward, “Placement: Hot or Not?” *International Conference on Computer Aided Design*, pp. 283–290, 2012.

[4] Z. Li, C. J. Alpert, G.-j. Nam, C. Sze, N. Viswanathan, N. Y. Zhou, I. B. M. Systems, T. Group, and B. Road, “Guiding a Physical Design Closure System to Produce Easier-to-Route Designs with More Predictable Timing,” *Dac*, vol. 1, pp. 465–470, 2012.

[5] Z. W. Jiang, H. Chen, T. C. Chen, and Y. W. Chang, “Challenges and solutions in modern VLSI placement,” *2007 International Symposium on VLSI Design, Automation and Test, VLSI-DAT 2007 - Proceedings of Technical Papers*, 2007.

[6] I. L. Markov, J. Hu, and M. C. Kim, “Progress and Challenges in VLSI Placement Research,” *Proceedings of the IEEE*, vol. 103, no. 11, pp. 1985–2003, 2015.

[7] T. C. Chen, P. H. Yuh, Y. W. Chang, F. J. Huang, and T. Y. Liu, “MP-trees: A packing-based macro placement algorithm for modern mixed-size designs,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 9, pp. 1621–1634, 2008.

[8] R. A. Rutenbar, “The first eda mooc: Teaching design automation to planet earth,” in *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, June 2014, pp. 1–6.

[9] Z. Li, C. J. Alpert, G. J. Nam, C. Sze, N. Viswanathan, and N. Y. Zhou, “Guiding a physical design closure system to produce easier-to-route designs with more predictable timing,” in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, June 2012, pp. 465–470.

[10] J. Cong, G. Luo, K. Tsota, and B. Xiao, “Optimizing routability in large-scale mixed-size placement,” *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, pp. 441–446, 2013.

[11] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 3, pp. 346–365, Sept 1961.

[12] K. Golshan, *Physical Design Essentials: An ASIC Design Implementation Perspective.* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[13] L.-T. Wang, Y.-W. Chang, and K.-T. T. Cheng, Eds., *Electronic Design Automation: Synthesis, Verification, and Test.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009.

[14] P. Saxena, R. S. Shelar, and S. Sapatnekar, *Routing Congestion in VLSI Circuits: Estimation and Optimization.* Springer Science & Business Media, 2007.

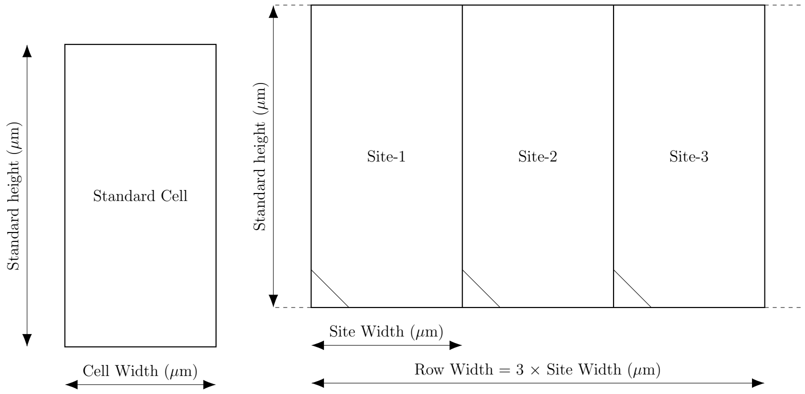
Appendix A

Preliminaries

A.1 Resources

The problem of placement and routing corresponds to the mapping of cells from the structural netlist derived from logic synthesis onto a physical die and to embed physical wires to replace their logical connectivity, respectively. From the computational perspective, the problem space within which the solution for placement and routing is found is the actual physical partition of the SoC. Hence, the chip is composed of resources termed as placement and routing resources, which are explained below:

- **Placement Resources:**
  A chip or its partition is composed of rows which are placement resources. The height of the rows (figure A.1b) is same as the height of the standard cell (figure A.1a) in a given technology node/library. Rows are composed of smaller units called sites.



(a) A standard cell

(b) Rows and sites

Figure A.1: Placement resources

A standard cell can occupy a row and an integral number of sites depending upon its functionality and characteristics (figure A.2).

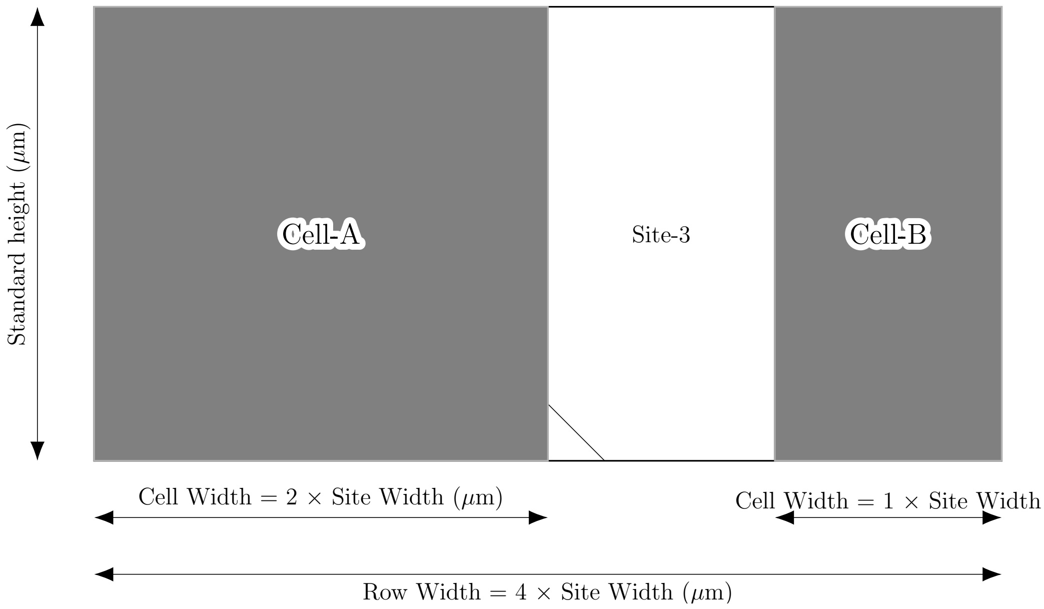
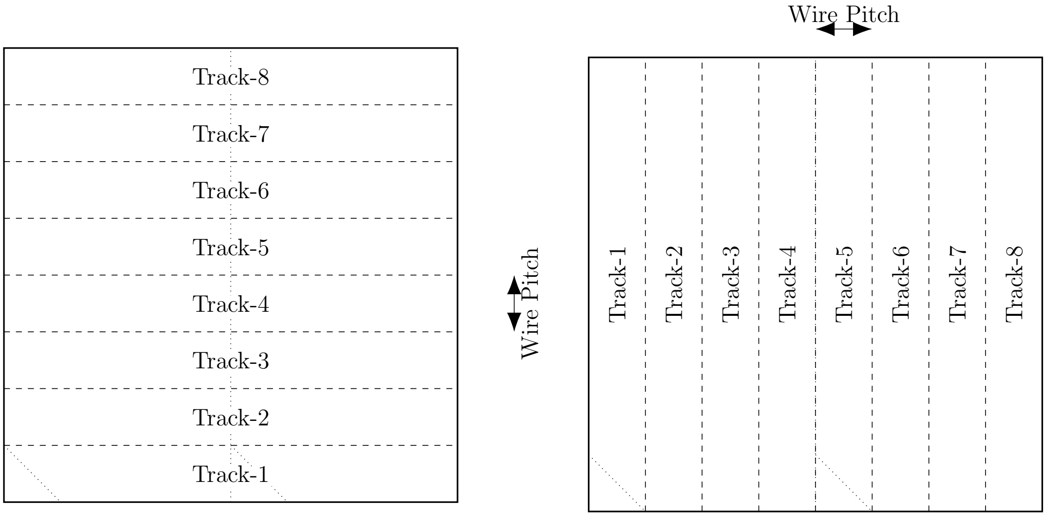

Standard height (μm)
Cell-A
Site-3
Cell-B
Cell Width = 2 × Site Width (μm)
Cell Width = 1 × Site Width
Row Width = 4 × Site Width (μm)

Figure A.2: Typical cell placement

- **Routing Resources:**
  Rows are overlaid with tracks (figure A.3a and figure A.3b), which act as guide for the router to embed the physical wire. The width of the track is based on the pitch or *line2via* spacing [12, 13] of the technology node.



(a) Horizontal tracks on rows

(b) Vertical tracks on rows

Figure A.3: Routing resources

Today's router are based on the preferred direction routing model [8, 13] and designs are composed of multiple layers with alternating preferred direction such as VHV or HVH. This superposition of layers constitutes a routing grid (figure A.4) with an intersection of a horizontal and vertical track constituting a routing cell. Switching between the routing layers is possible by the usage of vias.

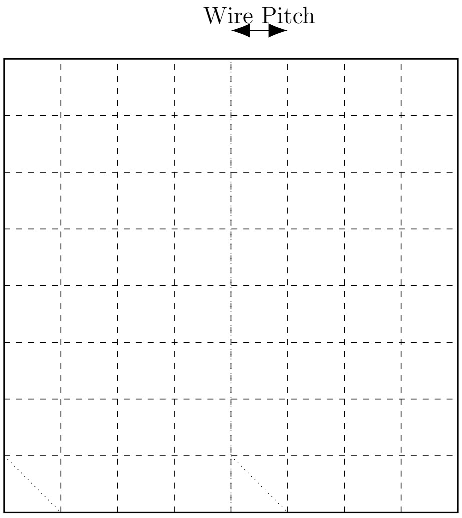


Figure A.4: Routing grid

Usually, a chip or its partition may contain thousands of rows and equivalent tracks along with multiple layers for routing. For example, in the 28nm technology node (standard cell height is 0.8μm), a square partition of height 500μm may have 625 rows with 5000 horizontal and vertical tracks alternating with its multiple layers.

A.2 Floor-plan

Floor-planning is the first step of the physical synthesis flow. In this stage the chip dimensions are determined based on the target utilization and instance in the design. If the design is hierarchical, the dimensions are determined from the top level. In hierarchical flow, the partition area is fixed and cannot be changed during the entire flow, hence, the designs are called as fixed-die designs. Floor-planning involves pad placement, macro placement and power grid definitions with routing (special nets). Once these tasks are performed, the core area, or the area where all the functional and critical logic resides, is defined. Core area utilization is one of the important quality indicators that can be used to assess the quality of the floor-plan. Higher utilization might indicate a congested design which might impact timing and routing closure. The quality of core area is defined by the macro placement. Usually, the macros are stacked along the periphery of the die so as to maximize the core area with minimal discontinuity and maximize the utilization. Continuity of core area refers to the availability of placement and routing resources without any blockage (for routing and placement). Also a core area of 1:1 is desired, but not necessarily available. Continuous resource availability depends on macros and their placement which are discussed in the next section. A typical floor-plan is shown in figure A.5 which is composed of rows. The placement and routing resources can be seen in further detail in figures A.1a, A.1b, A.3a, A.3b and A.4.

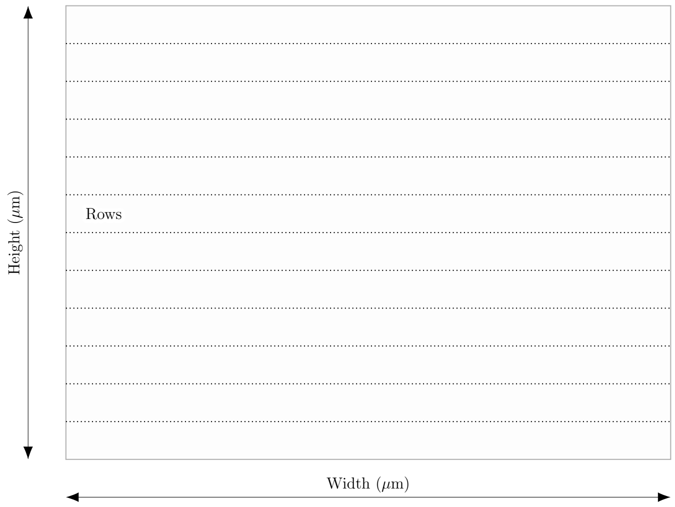


Figure A.5: A typical floor-plan

Cell type	Height (μm)	Width (μm)	Placement Resources (Rows:Site)	Routing Resources (Routing Cells)
Standard Cell	0.8	1	1:1	80
Macro	150	300	187:56100	18x10^6

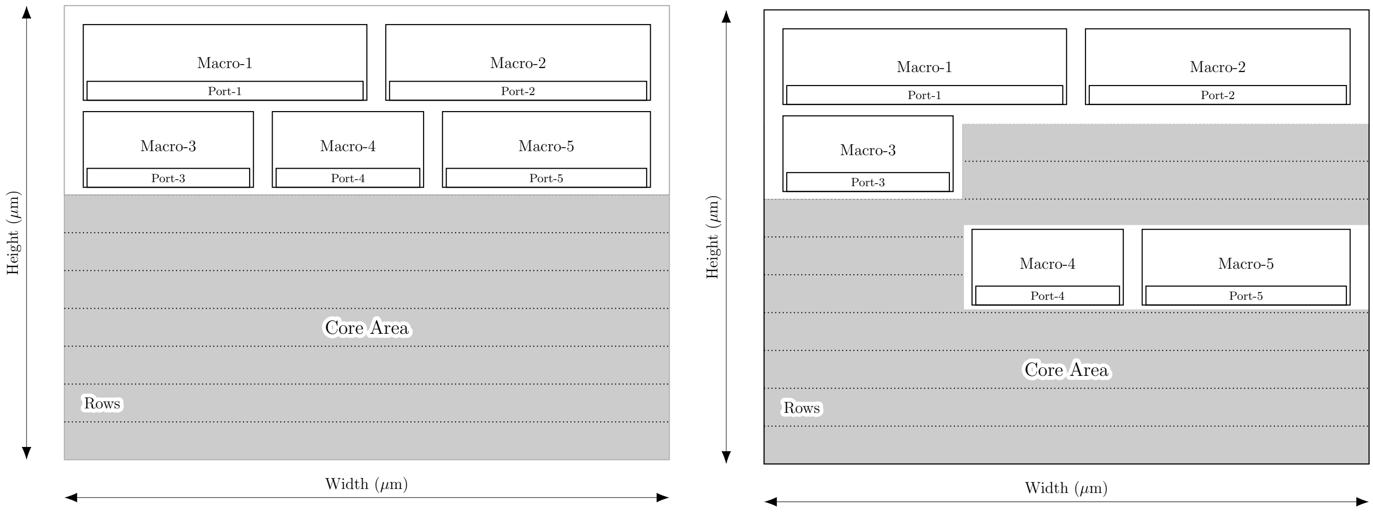
Table A.1: Disparity in resource consumption between a macro and a standard cell

A.2.1 Macros

A macro is a rectilinear object (but most often rectangular) which is constrained by orientation in a given technology node and consume a significant amount of placement and routing resources compared to a standard cell. Also, due to macro-porosity, they block routing resource up to certain layers. For example, in the 28nm node, macro blocks the routing resources up to 4th layer of possible 8 layers. To quantify the disparity in the resource consumption refer table A.1 (standard height SH is 0.8μm and site width assumed is 1μm).

A.2.2 Core Area

Core area, as mentioned before, is the central area in which all the functional logic is placed. Due to the resource disparity that exists between macros and standard cells, it is preferred not to have macros placed in the core area as it would lead to discontinuity in the problem space constraining the tools finding the solution for placement, routing and their optimization routines impacting their run-times and the resulting quality of results. In order to avoid a discontinuous core area, usually as a standard industry practice, the macros are placed along the periphery of the chip. Figure A.6a depicts a floor-plan with macro stacked along periphery while maintaining a continuous core area while figure A.6b depicts a discontinuity in core area due to macro placement within the desired core area.



(a) A continuous core area

(b) A discontinuous core area

Figure A.6: Different types of core area

Once the core area is defined, the utilization value is used as an indicator for assessing the quality of floor-plan as well as to predict its impact on the downstream stages in the physical synthesis flow.

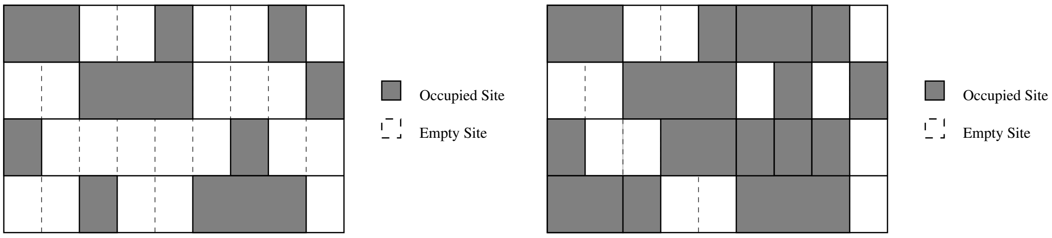
Appendix B

Quality Indicators

The quality of the design is analysed in each stage of physical synthesis flow and are used to address issues that could impact design closure. At the floor-plan stage, core area utilization or density is used as the quality indicator. Once placement is performed, congestion in the core area as well in the channel between macros using overflow metrics are used to assess the quality. After clock tree synthesis, timing slack and number of paths responsible for timing violations are used as quality indicators. After routing, the primary quality indicators are timing slack (hold) and the number of violating paths along with design rule checks (DRC).

B.1 Core Area Utilization

It is the ratio of occupied sites (by standard cells) in the core area to that of total available sites (in the core area). Usually, a lower core area utilization directly correlates with faster runtimes with better quality of results in terms of timing and routing as more resources are available for the EDA tool for achieving better design convergence. Figure B.1a depicts a core area which is less utilized while figure B.1b depicts high utilization of core area.



(a) Low core utilization

(b) High core utilization

Figure B.1: Different core area utilizations

B.2 Congestion and Overflow

Congestion is one of the primary factors that is addressed after placement and its estimate is obtained after performing global routing. During the global routing process, the entire chip is divided into grid cells, g-cells or bins which are composed of tens of tracks [8, 13, 14]. For a given bin, hence, for the entire design, congestion is defined as a ratio of required routing resources (demand) to that of available routing resources (supply) while overflow is the positive difference between the demand and supply. A design is said to be congested when the amount of routing resource required exceeds the available routing resources. A congestion map is generated using different parameters such as total track overflow, maximum congestion and the number of congested bins [14]. Usually, core area is devoid of channels. Channels usually exist between the macros and have very limited placement and routing resources. The nets from the macros are routed across these channels and buffers are placed to meet the timing constraints as the placement resources are reserved for buffers using placement blockages. High congestion within channels would impact routing closure while deteriorating timing. It is always desired to minimize the congestion within the channels.

B.3 Slack and Path Violations

Timing slacks are defined as the difference between arrival and required times [8] of clock signals for a given register to register path. Timing slacks are categorized as positive and negative slack.

- **Positive Slack**: For any given timing path, slack is always desired to be positive. It indicates that the timing constraints for the path has been met and is better by the margin indicated by the absolute value of the positive slack.

- **Negative Slack**: It is not desired and it indicates that the timing constraints for the timing path has not been met and is worse by the margin indicated by the absolute value of the negative slack.

Usually, during the physical synthesis flow, the timing paths with negative slacks need to be fixed. The total negative slack or TNS is an aggregate of all the negative slacks of those paths which have violated the imposed timing constraint. These violating paths are fixed by the design engineers using the EDA tools. Consequently, a lower number of violating paths can be used as an indicator for the better quality of the design.

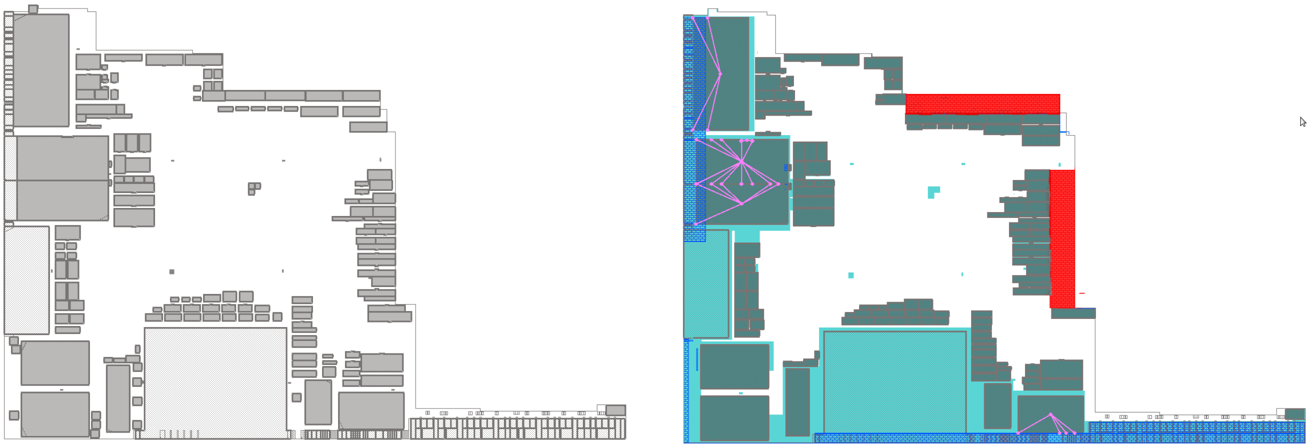
B.4 Design Rule Violations

The number of design rule violations that need to be fixed are one of the primary quality indicators for a design after routing is completed. Design rules are imposed by the foundries for the manufacturability of the chip. Some of the design rules are spacing rules, minimum width rule, via rule etc. Usually, the EDA tools are capable of removing the design rule violations automatically by performing local repairs and the remaining violations which exist after automatic repair need manual intervention which impacts the design closure.

Appendix C

Area Reduction

The results presented in chapter 3 involved reallocation of placement and routing resources from the channels to the core area as CDF was reduced. In area reduction trails (emulating block to block resource reallocation), all the channels are spaced with CDF=4 and the core area utilization is maintained similar to the taped out version (CDF=1) as the recovered resources are not allocated to the core area, instead, the equivalent amount of placement and routing resources are reserved by the usage of placement and routing blockages for all layers in the design at appropriate locations. Figure C.1b shows the area reduced (indicated by red regions) version of Design-A with channels spaced using CDF=4.



(a) CDF=1 with no area reduction

(b) CDF=4 with area reduction (red regions)

Figure C.1: Different versions of Design-A

Of the two designs presented in chapter 3, Design-A achieved routing closure as the overflow was manageable by the tool while Design-B did not which can be attributed to heavy congestion (figure 3.4b) and overflow as the utilization was maintained similar to that of taped out version with reduced resources. The results for Design-A are tabulated in table C.1.

Design Version	**Placement Runtime** (Hours)	**Clock Tree Optimization Runtime** (Hours)	**Routing Runtime** (Hours)
Taped Out (CDF=1)	1.282	1.368	1.046
Optimized (CDF=4)	0.76	0.919	0.857
Area Reduced (with CDF=4)	1	1	1

Table C.1: Normalized runtime comparisons of different versions of Design-A

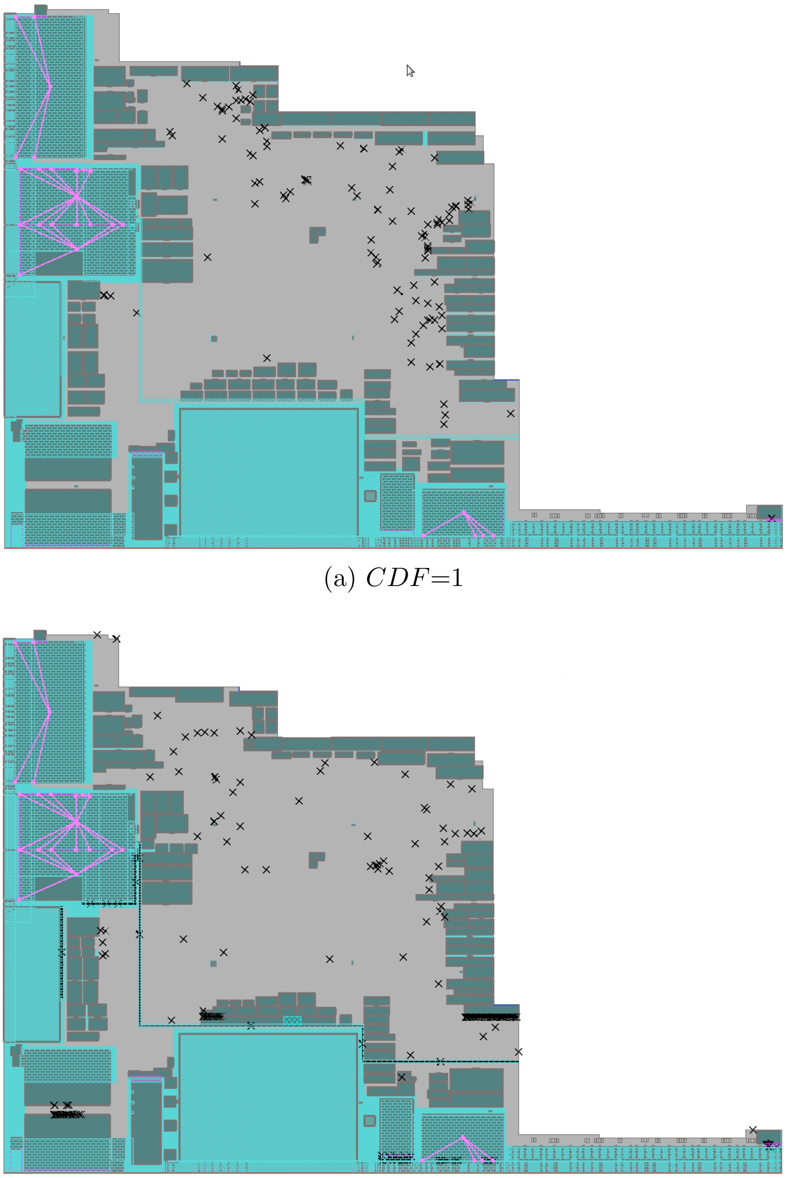
The overall area of Design-A was reduced by 4.1% by the usage of placement and routing blockages (see figure C.1b). Compared to the taped out (CDF=1) version of Design-A, the version with reduced area and channels spaced using CDF=4 achieved placement with run-times reduced by 28%, clock tree optimization with run-times reduced by 36.81% and routing with run-times reduced by 4.6%.

Appendix D

Impact within Channels

The resources within the channels among the macros are usually reserved for buffer insertion and for routing. Three types of net namely power/ground (P/G), clock and signal nets are routed in the given order. P/G nets are routed in floor-plan stage, clock nets are routed in clock tree synthesis stage while signal nets are routed in routing stage of the physical synthesis flow. Metal M1, M2, M7 and M8 were used for P/G routing for Design-A and Design-B. Also, for macros, P/G routing and clock routing used metal four (horizontal) and metal five (vertical).

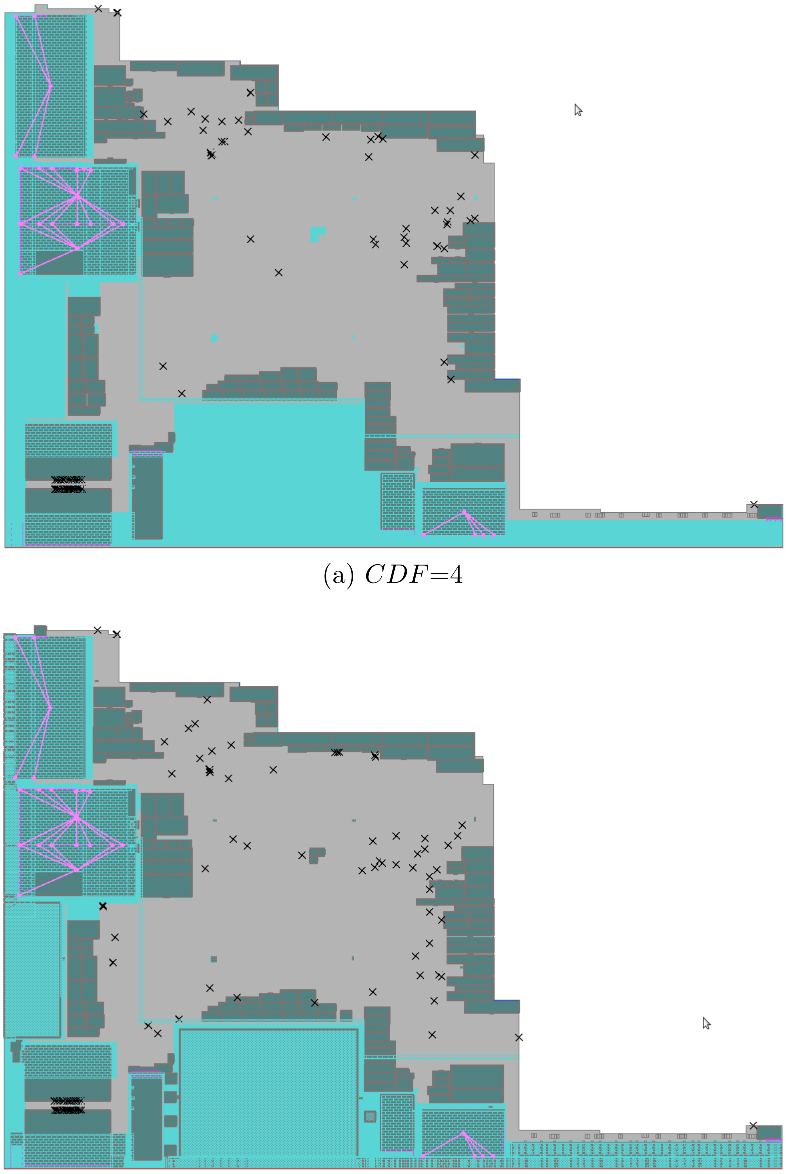
Minimization of channel spaces results in reduced resources within layers one to four which results in increased congestion while timing deteriorates. Figure 3.3 and 3.5 depict the aforementioned trend as *CDF* is varied from two to eight. A congested bin lacks the required resource for routing which results in incomplete routing, opens or shorts [14] which are removed by performing localized repair using the EDA tool along with manual intervention impacting the design closure. The variations in the number of shorts (represented as black crosses) in the design with *CDF* is shown in figures D.1 and D.2 for Design-A and figures D.3 and D.4 for Design-B. In either case the number of shorts is observed to be minimum within channels when channel spacing is optimum i.e. CDF=4 (see figure D.2a and D.4a).



(a) CDF=1

(b) CDF=2

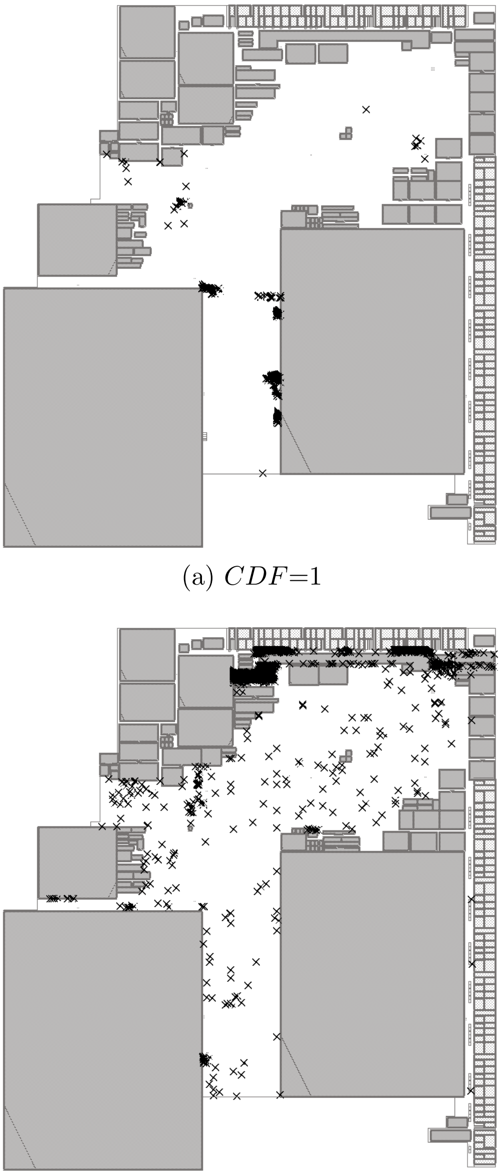
Figure D.1: Shorts (black crosses) within channels as CDF is varied in Design-A - I



(a) CDF=4

(b) CDF=6

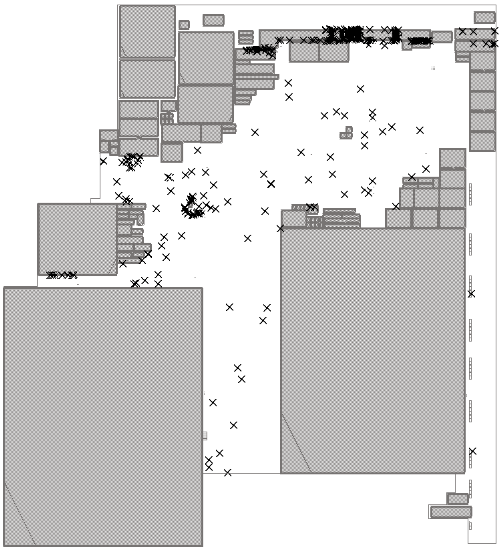
Figure D.2: Shorts (black crosses) within channels as CDF is varied in Design-A - II



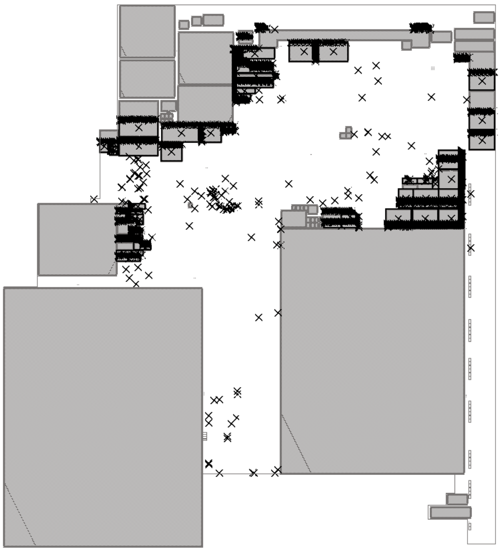
(a) CDF=1

(b) CDF=2

Figure D.3: Shorts (black crosses) within channels as CDF is varied in Design-B - I



(a) CDF=4



(b) CDF=6

Figure D.4: Shorts (black crosses) within channels as CDF is varied in Design-B - II