UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Multi-Level Partitioning Methodologies and their Applications in Modern IC Design

Diploma Thesis

George Raphael Goudroumanis

ggeorgios-r@uth.gr

Supervisor: Christos P.Sotiriou

Volos, September 2023

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Μεθοδολογίες Πολυεπίπεδης Κατάτμησης Κυκλωμάτων και οι Εφαρμογές τους στη Σχεδίαση Ολοκληρωμένων Κυκλωμάτων στις Μέρες μας

Διπλωματική Εργασία

Γιώργος Ραφαήλ Γκουντρουμάνης

ggeorgios-r@uth.gr

Επιβλέπων: Χρήστος Σωτηρίου

Βόλος, Σεπτέμβριος 2023

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Approved by the Examination Committee:

Supervisor **Christos P.Sotiriou**
Professor, Department of Electrical and Computer Engineering,
University of Thessaly, chsotiriou@e-ce.uth.gr

Member **Georgios Stamoulis**
Professor, Department of Electrical and Computer Engineering,
University of Thessaly, georges@e-ce.uth.gr

Member **Fotios Plessas**
Professor, Department of Electrical and Computer Engineering,
University of Thessaly, fplessas@e-ce.uth.gr

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Acknowledgements

I would like to express my heartfelt gratitude to Professors G. Stamoulis, F. Plessas, and C.P.Sotiriou for their invaluable support and guidance throughout the completion of my master's thesis. Each of them played a significant role in shaping the course of my research and academic journey. Specially professor C.P.Sotiriou mentorship extended beyond academia, and I am truly grateful for the personal and professional growth I've experienced under his guidance.

I would like to offer a special word of thanks to a dear friend and valuable mentor of mine, N. Sketopoulos. His unwavering support and encouragement as long as his patience, and genuine interest in my work were pivotal in overcoming various technical and personal challenges during our collaboration started long before this thesis. Furthermore, I would like to wholeheartedly thank T.Asimaki and my colleagues and friends from CASLAB who were always there to support me both technically and spiritually throughout my journey so far.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

George Raphael Goudroumanis
ggeorgios-r@uth.gr

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Diploma Thesis

Multi-Level Partitioning Methodologies

and their Applications in Modern IC Design

George Raphael Goudroumanis

ggeorgios-r@uth.gr

Abstract

This research was carried out during my master thesis dissertation and presents an innovative Multi-Level Partitioning algorithm specifically designed for VLSI circuits. As a divide-and-conquer framework, this algorithm is capable of partitioning large scale designs of million of gates into a manageable number of groups. This process main targets are to reduce the connectivity of the produced groups, also known as cutsize, while preserving of the partitions area balance in minimum execution time. However, the established partitioning frameworks such as MLpart [1], hMETIS [2] [3], KaHyPar[4] [5] [6], PaToH [7] SpecPart [8] [9] and GAP [10] [11] do not produce a result aware of the other VLSI characteristics such as timing and the following Physical Design steps mend to be executed afterward producing suboptimal results regarding these aspects. On the other hand, the introduced algorithm was developed as part of a greater PnR tool aiming to assist on various aspects of the ASIC flow.

To support our claims, we developed an extensive experimental methodology comparing results of forty-two (44) designs obtained by the following academic contests, DAC 2012[12], ISDP 2005/6[13], ISPD 2011[14] and ICCAD 2015[15], ICCAD 2022/23 [16]. Note the last contest, i.e. ICCAD 2023, was for 3D Integrated Circuits, where we took place. Moreover, we compare 2 industrial designs and 5 open source large scale designs namely b19 [17], Leon3mp [17], Netcard [17], jpeg_ecoder [17] and vga_lcd decoder [17]. The presented results include all aforementioned partitioning tools, exploring some of their tuning parameters based on the values proposed in their paper. There are three prisms under which we evaluated the algorithms, each one focuses on one aspect of the ASIC flow. The first category compares the results based on the classic metrics attached to partitioning evaluation, cutsize, area balance ratio, execution time. The second focuses on the timing analysis of the design, introducing the top thousand delay and slack paths' distribution. The third part, presents our results in the

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

context of a 3D CAD PnR tool measuring the total cost produced by an external, unbiased evaluator.

To perform all these experiments and conclude our evaluation, we had to integrate all tools including ours in a grater framework enabling the communication with the outer world. This way, we developed the necessary code to combine both our tool and the other ones with an internal framework supporting industrial formats files parsers and a database able to provide the necessary function to load and evaluate the results of all tools based on the introduced metrics. Furthermore, this suite includes an integrated static timing analysis engine, which is mandatory to evaluate the results for timing driven operations. The code of our tools and the wrappers needed to integrate the other tools with the general framework was developed in C/C++ programming language. On the other hand, the scrips to extract the experiments and evaluate the results were developed in BASH, TCL and Python.

As regarding the first comparison section, the comprehensive analysis of the results establishes our algorithm as an exceptional option to partition large circuits into few and loosely connected sub-circuits. This is evident in the partitions it generates, boasting a cutsize three to twelve times smaller and an area balance ratio seven times to thirty times lower. Remarkably, these advancements are achieved while maintaining a relatively equal execution time, under an hour, compared to the other tools. Considering the second point of comparison, after the data analysis the results also proved that our approach produces far more suitable groups to address the timing driven placement challenge. We safely came up to this conclusion based on the top thousand delay and slack paths' fragmentation reduction by four times to seven times in both cases. As regarding the third comparison point, related to 3D design flow, we compare the results based on the produced tier vias and the achieved tier utilisation ratio.

Keywords:

Electronic Design Automation (EDA), Multi-Level, VLSI, ASIC flow, 3D Chip Design, Timing Driven, Cloud Computing, Machine Learning,

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Διπλωματική Εργασία

Μεθοδολογίες Πολυεπίπεδης Κατάτμησης Κυκλωμάτων και οι Εφαρμογές τους στη Σχεδίαση Ολοκληρωμένων Κυκλωμάτων στις Μέρες μας

Γιώργος Ραφαήλ Γκουντρουμάνης

ggeorgios-r@uth.gr

Περίληψη

Αυτή η έρευνα πραγματοποιήθηκε κατά τη διάρκεια της μεταπτυχιακής μου διατριβής και παρουσιάζει έναν καινοτόμο αλγόριθμο κατάτμησης πολλαπλών επιπέδων ειδικά σχεδιασμένο για κυκλώματα VLSI. Ως ένας αλγόριθμος διαίρει και βασίλευε, είναι ικανός να διαχωρίσει κυκλώματα μεγάλης κλίμακας εκατομμυρίων πυλών σε έναν μικρότερο πιο διαχειρίσιμο αριθμό ομάδων. Κύριοι στόχοι αυτής της διαδικασίας είναι η μείωση των συνδέσεων μεταξύ των παραγόμενων ομάδων, γνωστή και ως cutsize, με ταυτόχρονη διατήρηση παρεμφερούς εμβαδού των ομάδων σε ελάχιστο χρόνο εκτέλεσης. Ωστόσο, τα καθιερωμένα εργαλεία κατάτμησης όπως τα MLpart [1], hMETIS [2] [3], KaHyPar[4] [5] [6], PaToH [7] SpecPart [8] [9] and GAP [10] [11] δεν παράγουν αποτέλεσμα με βάση τα υπόλοιπα χαρακτηριστικά των κυκλωμάτων VLSI όπως ο χρονισμός και τα επόμενα βήματα Φυσικής σχεδίασης που πρέπει να εκτελεστούν στη συνέχεια, παράγοντας μη βέλτιστα τελικά αποτελέσματα σε σχέσει με τις αντίστοιχες μετρικές. Από την άλλη πλευρά, ο προτεινόμενος αλγόριθμος αναπτύχθηκε ως μέρος ενός ευρύτερου εργαλείου PnR.

Για να υποστηρίξουμε τον ισχυρισμό μας, αναπτύξαμε μια εκτεταμένη πειραματική μεθοδολογία συγκρίνοντας τα αποτελέσματα σαράντα δύο (42) ακαδημαϊκών κυκλωμάτων από τους ακόλουθους διαγωνισμούς 2D CAD, DAC 2012[12], ISDP 2005/6[13], ISPD 2011[14] and ICCAD 2015[15], ICCAD 2022/23 [16] στον οποίο και συμμετείχαμε, 2 βιομηχανικά και 5 ανοικτού κώδικα κυκλώματα μεγάλης κλίμακας τα οποία είναι το b19 [17], Leon3mp [17], Netcard [17], jpeg_ecoder [17] και vga_lcd decoder [17]. Τα αποτελέσματα που παρουσιάζονται περιλαμβάνουν όλα τα γνωστά εργαλεία κατάτμησης όπως αναφέρθηκε προηγουμένως, διερευνώντας ορισμένες από τις παραμέτρους τους με βάση τις τιμές που προτείνονται στις αντίστοιχες έρευνες τους. Υπάρχουν τρία πρίσματα βάσει των οποίων αξιολογήσαμε

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

τους αλγορίθμους, καθένα από τα οποία εστιάζει σε μια πτυχή της ροής ASIC. Η πρώτη κατηγορία συγκρίνει τα αποτελέσματα με βάση τις κλασικές μετρικές που συνδέονται με την αξιολόγηση της κατάτμησης κυκλωμάτων, cutsize, area balance ratio, execution time. Η δεύτερη επικεντρώνεται στην ανάλυση χρονισμού του σχεδίου, εισάγοντας ως μετρική την κατανομή των κορυφαίων χιλίων μονοπατιών καθυστέρησης. Το τρίτο μέρος, παρουσιάζει τα αποτελέσματά μας στο πλαίσιο ενός εργαλείου 3D CAD PnR.

Για να εκτελέσουμε όλα αυτά τα πειράματα και να ολοκληρώσουμε την αξιολόγησή μας, έπρεπε να ενσωματώσουμε όλα τα εργαλεία, συμπεριλαμβανομένου του δικού μας, σε ένα ευρύτερο πλαίσιο που επιτρέπει την επικοινωνία με τον εξωτερικό κόσμο. Με αυτόν τον τρόπο, αναπτύξαμε τον απαραίτητο κώδικα για να συνδυάσουμε τόσο το δικό μας εργαλείο όσο και τα άλλα με ένα εσωτερικό εργαλείο που υποστηρίζει αναγνώστες αρχείων βιομηχανικών προδιαγραφών και μια βάση δεδομένων ικανή να παρέχει την απαραίτητη λειτουργικότητα για τη φόρτωση και την αξιολόγηση των αποτελεσμάτων όλων των εργαλείων με βάση τις καινούργιες μετρικές. Επιπλέον, αυτή η σουίτα περιλαμβάνει μια ενσωματωμένη μηχανή στατικής ανάλυσης χρονισμού, η οποία είναι υποχρεωτική για την αξιολόγηση της καταλληλότητας των αποτελεσμάτων μας για λειτουργίες που καθοδηγούνται από τον χρονισμό. Ο κώδικας των εργαλείων μας και των διεπαφών που απαιτήθηκαν αναπτύχθηκε σε γλώσσα προγραμματισμού C/C++. Από την άλλη πλευρά, τα scrips για την εξαγωγή των πειραμάτων και την αξιολόγηση των αποτελεσμάτων αναπτύχθηκαν σε BASH, TCL και Python.

Όσον αφορά την πρώτη ενότητα σύγκρισης, η ολοκληρωμένη ανάλυση των αποτελεσμάτων καθιερώνει τον αλγόριθμό μας ως μια εξαιρετική επιλογή για την κατάτμηση ενός μεγάλου κυκλώματος σε λίγα και σποραδικά συνδεδεμένα υπο-κυκλώματα. Αυτό είναι εμφανές στις κατατμήσεις που παράγει, διαθέτοντας ένα cutsize τρεις έως δώδεκα φορές μικρότερο και έναν λόγο ισοζυγίου εμβαδού επτά έως τριάντα φορές μικρότερο. Είναι αξιοσημείωτο ότι αυτές οι εξελίξεις επιτυγχάνονται διατηρώντας έναν σχετικά ίσο χρόνο εκτέλεσης, κάτω από μία ώρα, σε σύγκριση με τα άλλα εργαλεία. Λαμβάνοντας υπόψη το δεύτερο σημείο σύγκρισης, μετά την ανάλυση των δεδομένων τα αποτελέσματα απέδειξαν επίσης ότι η προσέγγισή μας παράγει πολύ πιο κατάλληλες ομάδες για την αντιμετώπιση της πρόκλησης τοποθέτησης με γνώμονα τον χρονισμό δείχνοντας 4 έως 7 φορές καλύτερο αποτέλεσμα.

Λέξεις-κλειδιά:

Electronic Design Automation (EDA), Multi-Level, VLSI, ASIC flow, 3D Chip Design, Timing Driven, Cloud Computing, Machine Learning

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Table of contents

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

List of figures

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

List of tables

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Abbreviations

IC	Integrated Circuit
VLSI	Very Large Scale Integrated circuit
EDA	Electronic Design Automation
ASIC	Application Specific Integrated Circuit
PnR	Placement and Routing
SOC	System On Chip
NOC	Network Of Chips
IP	Intelectual Property
QOR	Quality Of Results

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Chapter 1

Introduction

In the ever-evolving landscape of modern technology, the demand for faster, more efficient, and increasingly complex electronic systems has become an inherent part of our daily lives. From smartphones to autonomous vehicles, from smart cities to advanced medical devices, electronic systems are at the heart of innovation and progress. These systems are the result of complex and highly specialized design processes, requiring meticulous attention to detail and precision. In this context, Electronic Design Automation (EDA) emerges as a critical driving force, empowering engineers and designers to navigate the intricate path of electronic system development.

The objective of this master's thesis is to delve into the realm of Electronic Design Automation, a multidisciplinary field that combines computer science, electrical engineering, and mathematics. EDA encompasses a spectrum of tools, techniques, and methodologies aimed at automating various stages of electronic system design, from conceptualization and specification to physical realization and verification. EDA's fundamental goal is to expedite the design process, enhance its accuracy, and facilitate the creation of increasingly sophisticated electronic systems that meet the demands of today's technology-driven world.

In detail, this work analyses in depth the partitioning step of the Multi-Level EDA flow, a significant but underappreciated factor of the broad EDA flow. Due to the fact that the processing power requirements during the early stages of EDA in the digital design industry were met by the simultaneously ongoing growth of computers, this part of EDA has remained rather unexplored. However, in modern times, the rapid increase of components inside an IC, forces even the most capable and cutting-edge supercomputers to yield because of the enormous amount of computations needed for the design and simulation of the circuit.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

1.1 Electronic Design Automation (EDA)

EDA is a software industry which is basically used for designing electronic systems such as integrated circuits and printed circuit boards. EDA tools enable engineers and designers to model, simulate, and test electronic systems digitally before physical prototypes are built. This significantly reduces the cost of product development by minimizing the need for expensive hardware prototypes and repeated testing cycles. As a result, companies can bring innovative products to market more efficiently and cost-effectively. On top of that, this time-to-market advantage is particularly crucial in fast-paced industries like consumer electronics and telecommunications, where being the first to market can translate into a competitive edge and higher profitability.

Under the umbrella of EDA software are included a comprehensive suite of tools, methodologies, and processes crucial for the efficient and effective design, verification, and optimization of electronic systems. Starting by the translation of logical circuit descriptions into physical layouts for ICs and PCBs, EDA allows engineers to predict and analyze the behavior of electronic systems using the rest assets of the suit including logic synthesis, timing analysis, and power analysis tools. Furthermore, design for manufacturability (DFM) and design for testability (DFT) tools are integral components of EDA, focusing on ensuring that designs can be produced reliably and cost-effectively while maintaining high test coverage and efficient fault detection. Last but not least, a list of simulation tools are provided analyzing the thermal and electromagnetic profile of the circuit. Of course for different circuits types (3D, NoCs) and applications (Space, Low Power) there are multiple flows and variations of these tools ensuring the high-quality standards.

In this sector, businesses like Cadence Design Systems Inc., Synopsys Inc., Siemens EDA, ANSYS, and Xilinx are directly involved. However, they and their affiliated companies, have spread their network of engineers across the globe, having sites almost at every capital city, with major presence in the United States, United Kingdom, China, and Middle East. Based on 2021 numbers the revenue of Electronic Design Automation software market was over eleven billion (11.36B$) dollars, and it is estimated that by 2030 this number will reach the twenty-five billions (25.70B$). Additionally, the semiconductor industry revenue in 2021 was fifty hundred ninety-five billions (595B$), and it is forecasted that by 2024 it will reach six hundred thirty point nine (630.9B$) billions.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

1.2 Novel ASIC Design flow

EDA software is an entire collection of tools assisting engineers to create high-quality chips. These tools are combined into the ASIC design flow which, based on the circuit characteristics, can vary from quite simple, as presented in Figure 1.1, to rather complicated. It is cautious to proceed deeper into the topic's fundamentals in order to have a better understanding of it.

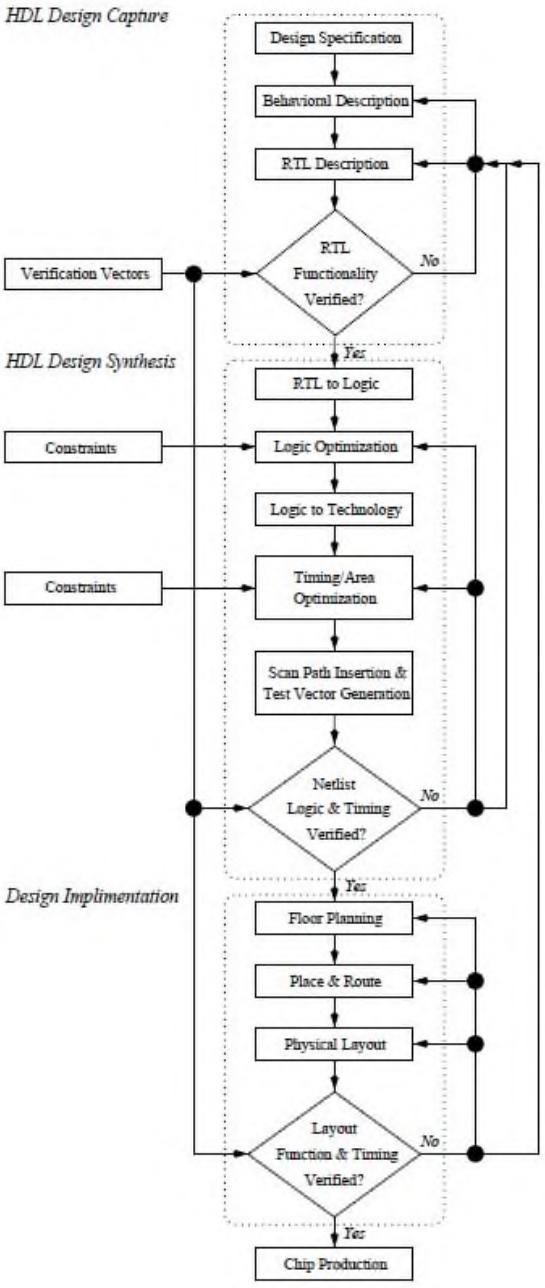


Figure 1.1: Abstract flow chart of the Application Specific Integrated Circuit Design flow [18].

The process begins by describing of the chip requirements and functionalities in a high level hardware description language, such as Verilog, providing the engineer an initial glimpse of the chip's behaviour. Following that, the produced description must be translated from high level commands into gate level representation. This step is called synthesis and alongside with the translation aims to create a directed circuit graph which do not violate the longest path limitations, maximum area and maximum power consumption limitations. Due to the earliness of this stage, the information to check these violations is harvested from the Process Design Kit (PDK) which is used for the particular design. By the end of this step, a file called netlist is created, which will be used afterwards in the following steps. Finally, by performing a behavioural simulation using this file as input, the front end of the flow is considered finished.

Advancing towards the rest of the flow, as presented in Figure 1.2, the first step is the

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

floorplan of the chip. During this step, the shape of the chip is determined alongside with the placement of the pins based on which the communication with the environment is achieved. The shape and the pin placement is usually predetermined, but there are cases where the chip is partitioned into submodules and each one of them could be handled as independent blocks, keeping these attributes flexible. The next step is the power planing of the chip, during which the supply and ground nets are created. This step is in charge to create the power grid, ensuring that the appropriate voltage value will reach all the circuit gates, while at the same time the power consumption of the chip will be preserved at the lowest point.

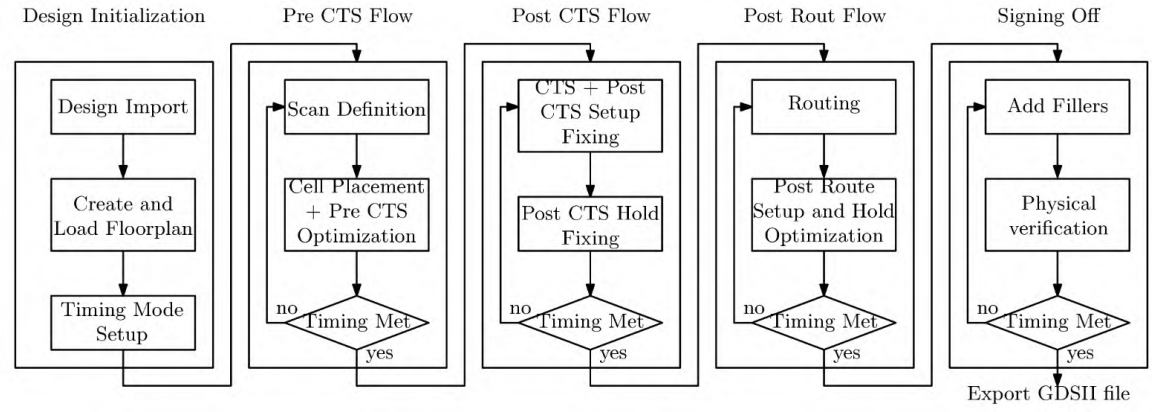


Figure 1.2: Detailed flow chart presenting the physical design process of the ASIC flow [19].

The next step, reaching the end of the ASIC flow, is the placement of the standard cells and macros. Throughout this step, the cells and macros are going to be placed inside the core of the chip, trying to maintain the minimum wirelength and routing congestion. That means that the cells must find a sweet spot in which the connectivity lines among the gates instances are as small as possible while the congestion occurred from their intersections is also limited. It is rather undeniable that the NP hard problem of the design placement requires sophisticated and complex algorithms to achieve a high-quality solution, which is going to be the base of the rest PnR flow. Following the original solution, several post placement techniques are used to address issues such as cell legalization or to prepare the solution for future steps such as clock tree synthesis and optimizations, as well as area and timing recovery.

The other half of the back end flow is the routing of the design. This phase is segmented in three subsequent phases, namely clock route, global route and detail route. The first one considers only the clock network, including the cells added during the clock tree synthesis, as mentioned before. The importance of the clock accuracy at the arrival time of the pulses in the flip-flops, is the reason why this special net is routed before any other. As expected

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

in literature there are also many post-processing algorithms aiming at different issues. The second phase performs a quick and dirty routing of the gates to get a better assessment of the chip wire congestion. The third and final step performs the detail routing of the chip, where all the rules and guidelines of the PDK must be followed. Such rules effecting the spacing of the metals, the directions of the metals, the maximum density of metals in a specific region and the minimum overlap of the wires during the change of direction.

Proceeding to the end of the flow, after the successful place and route of the circuit, it is time to evaluate the result and ensure its functionality. There are several aspects which might affect the functionality, the manufacturability and the testability of the chip. Thus, it is important to use a Static Timing Analysis engine which will analyse the circuit and will report among other important information if the longest timing path violates the requested clock period. After that, the verification of the geometry must be performed to ensure that the produced result can be manufactured using the specified PDK. Finally, the Design Rules Check (DRC) analysis must be applied to ensure that there is no obvious threat to the chip functionality. Of course there are many other check points, verification algorithms even entire tools to verify that the produced layout has the same output with the initial simulation and that after the manufacturing the chip will have the anticipated behaviour.

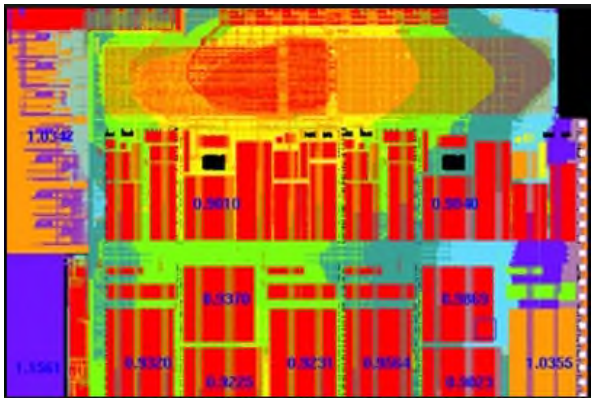


Figure 1.3: Screenshot from ANSYS RedHawk thermal analysis tool [20].

Without further ado, the final phase of the ASIC flow is reached. Here the engineer has to verify the thermal, electromagnetic and test coverage profile of the chip. So, utilising highly sophisticated and complex tools, firstly must verify that the chip's test vectors reach over 99.5% of the chip possible inputs. After that, the maximum allowed temperature of the chip will not be exceeded during chip's operation, causing catastrophic failure. Finally, it must be verified that the electromagnetic behaviour of the chip is nominal and will not jeopardize the integrity of its signals by cross talking and will not affect the surrounding systems.

From the brief abstract presented above, it is clarified that the ASIC flow is an extremely complicated and time-consuming collection of steps which often are repeated many times

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

before the extraction of the final product which will be sent to be printed. So it is of great importance to speed up this flow while ensuring its high quality result, if we are to continue developing larger, more complex and powerful chips to sustain our society's tremendous evolution.

1.3 Multi-Level ASIC flow

1.3.1 Introduction of Multi-Level flow in EDA

The concept of Multi-Level or hierarchical ASIC design flow has been around for decades. Even though it's challenging to pinpoint the exact first appearance of this approach, it can be traced back to the early days of ASIC design, when engineers started grappling with the growing complexity of their designs. One notable milestone in the evolution of multi-level design methodologies was the emergence of Hardware Description Languages (HDLs) like VHDL and Verilog. These languages provided a standardized way to discretise the circuit in blocks based on the logical functionality at various levels of abstraction, facilitating the hierarchical design process.

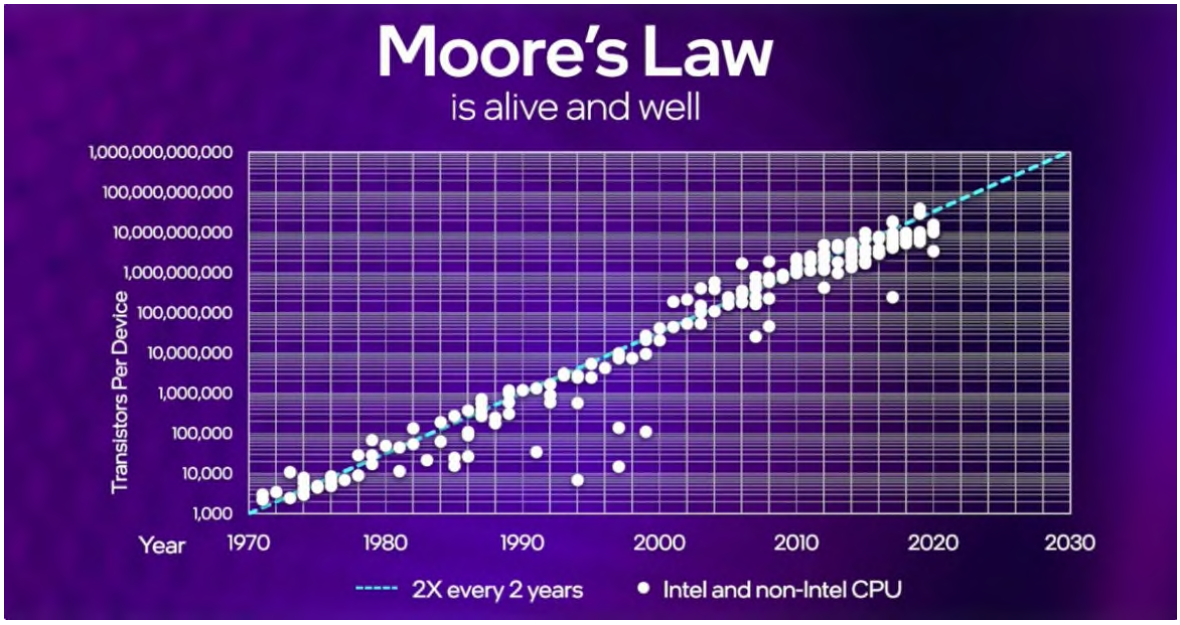


Figure 1.4: Presenting the number of transistors used in produced well-known chips from 70s until now, perfectly aliened with the prediction of Gordon Moore's law [21].

However, as chips became progressively more complex, their logical functions could not be divided into balanced loosely connected submodules. Thus, various algorithms and metrics

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

were emerged in this flow by the researchers of the time, aiming to create a specified number of area balanced sparsely connected groups of instances, dividing the circuit into smaller relatively equivalent blocks. The first approaches, in the early 1980s, did not produce substantial results, able to establish the algorithmic partitioning of the chip as standard practice. The reasons of this outcome are located in the significant computational time required to produce the results, alongside with the lack of developed tools to utilise this result

In our days, the complexity of the chips combined with the billions of devices placed in a chip renders the algorithmic partitioning rather necessary. Based on Moore law, the amount of instances placed within a chip will be doubled every six to eighteen months. As presented in Figure 1.4 the trillion transistor circuits are not so far, which means the even the initial partitions of the chip most probably will must be partitioned again in order to process them in reasonable time, as each one of these will contain millions of instances. In essence, the Multi-Level ASIC design flow in a few years will stand as a cornerstone in the semiconductor industry, streamlining complex design processes and permitting the production of cutting-edge electronic devices and systems with trillions of devices.

1.3.2 Multi-Level ASIC flow steps

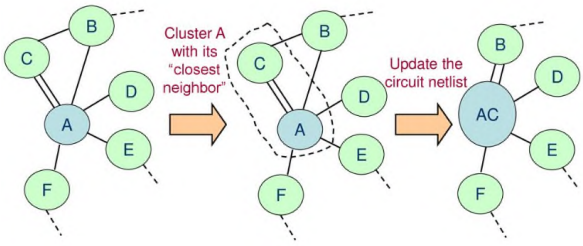


Figure 1.5: Presenting the intuition of VLSI circuit clustering algorithm [22].

As the Multi-Level ASIC design flow in the forthcoming years will be an essential step to address the larger circuits, it is prudent to present an outline of its steps in order to become acquainted with it. The flow starts with a process called clustering or coarsening, aiming to group the heavily connected instances of the circuit reducing the instances from many millions, billions or even trillion of devices into a few hundred thousand groups. The reason that the flow is called multilevel is that this step gradually merge the instances into bigger objects, creating levels of abstraction, trying to avoid the merging of large objects leading to unbalanced groups. Usually in literature and in this work, these groups are referred to as clusters.

The second step of the flow is called partitioning or uncoarsening and its objective is to further reduce the clusters into a specified number of groups, trying to simultaneously

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

preserve the area balance and reduce the intergroup connectivity. The produced objects are commonly known as partitions. Even though both of these steps aim to reduce the number of instances, there are key differences distinguishing the algorithms apart and compelling the order of their execution. The first algorithm groups the instances based on their connectivity, aiming to reduce their count in to a much smaller number while preserving the area balance. On the other hand, the second algorithm targets to create a specific amount of groups with minimum connections between them with the area balance intact. The stricter policy of the second algorithm makes it quite inefficient when a significant amount of objects needs to be taken under consideration, making the first operation mandatory towards the completion of its task.

In some cases, a third step on that flow can be added as a post process optimisation step targeted on the specific metric which the engineer needs to improve. Such metrics could be the timing of the circuit, the power consumption, the inter partition connectivity or the area ratio of the groups. Some of the algorithms to address the previous metrics are the reduction of delay path fragmentation, the separation of power hungry cells into different partitions, the cell replication and the incremental moves of objects from partition to partition respectively.

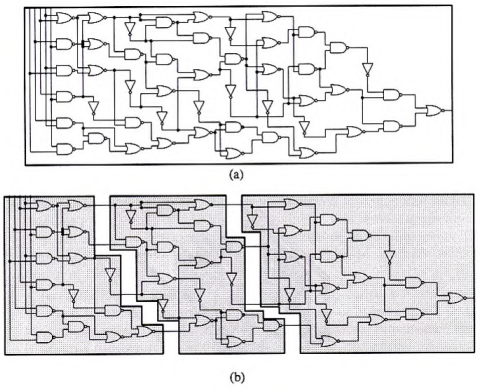


Figure 1.6: Presenting the intuition of VLSI circuit partitioning algorithm [23].

1.4 Multi-Level ASIC design Flow applications

Continuing to the next flow steps, the engineer must perform a Multi-Level placement algorithm. This one will place the partitions as if they were standard cells, and then it will proceed to each one of the partitions to place their enclosed objects. This algorithm is much quicker and scalable compared to the novel placement algorithm, which will try to handle the entire circuit at once. Considering the scalability of this process, each one partition could be distributed into a distant server to complete the novel ASIC flow as an independent chip and then recombined with the other partitions into a predefined floorplan as puzzle pieces. This

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

could significantly reduce the back end elapsed time, saving valuable time for chip testing and evaluation.

One more interesting application of circuit partitioning can be located in 2.5D and 3D ASIC design flow. This new technology aims towards the integration of multiple ICs in the same substrate connected as a NOC or one on top of the other respectively. Starting from the first approach, which is already in use the lasts decades, the separation of the ICs is performed based on the modules' hierarchy creating IP blocks. The second idea is much newer and aims to reduce the distance of inter die routes as they increase significantly the circuit delay due to their thickness. In both of these technologies, the separation of the circuit either in regions or in tiers respectively can be performed by the designer based on the netlist modules. However, during the latest years the integration of multiple technologies, i.e. coexistence of 130nm devices alongside with 22nm devices, arises new challenges in this flow regarding the timing closure and power consumption of the chip which could be addressed by an algorithmic partitioning approach. Both applications are further analysed in the following chapters, accompanied by comprehensive experiments using industrial and academic designs.

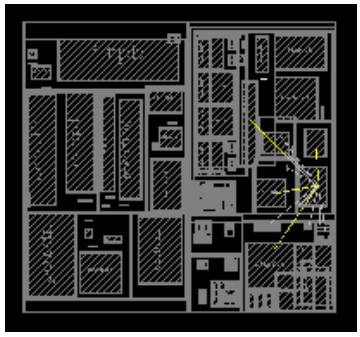


Figure 1.7: Mock floorplan in an IC layout editor window [24].

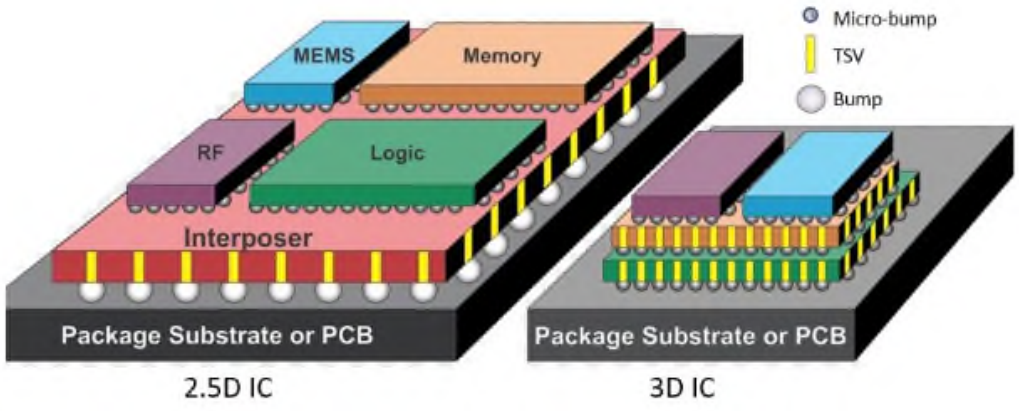


Figure 1.8: 2.5D versus 3D IC designs [25].

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

1.5 Thesis Outline

Despite, the substantial value of the Multi-Level flow, the tools targeted to support that cause are limited. The most known are the MLpart [1], hMETIS [2] [3], KaHyPar[4] [5] [6], PaToh [7] SpecPart [8] [9] and GAP [10] [11] which will be further analysed in the following chapters. The contribution of this work is to introduce a new partitioning tool targeted entirely to VLSI circuits. This one consists of two updated clustering and partitioning methodologies, one for each step of the multilevel flow respectively, and a post-processing optimisation algorithm.

The rest of the thesis is organised as follows. The second chapter delves into the necessary background knowledge to keep up with the thesis and then focuses on the other related works and tools, analysing their advantages and disadvantages. Following that, it will be presented the existing work related to the previous referred applications of chip partitioning, accompanied by the description of their flow and their basic algorithms. The next chapter, manifests the contribution of this work by turning the spotlight on the developed algorithms and heuristics integrated on the existing infrastructure to manage outperform the existing state-of-the-art tools. To endorse our claims, the next chapter includes the results and the experimental methodology, confirming the superiority of our approach regarding the reviewed metrics. The final chapter contains the remaining work which should be done in order to construct a complete bullet-proofed tool, proceeded by the conclusions of this analysis.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Chapter 2

Background

2.1 Introduction

The background chapter serves as an educational portal to our in-depth investigation of multi-level circuit partitioning, a domain that supports the development of cutting-edge electronic systems. This chapter sets the foundation for our analytical journey by explaining the key terms and definitions required to understand the complexities of this topic. We begin this illuminating journey by delving into the following key points.

In our first category, we build the groundwork by defining basic terms. This involves introducing Directed Acyclic Graphs (DAGs) and Hypergraphs as fundamental representation tools in circuit design. We distinguish between $Nets$ and $Flylines$, two fundamental yet nuanced design aspects. Furthermore, we define the roles of the integral $Nodes$ and $Components$ that comprise the graph representation of circuits. Following that, we go into the idea of Vcycle, investigating its application in partitioning strategies. We also investigate the semantics of $Clusters$ and $Partitions$, giving light on their function in BackEnd design. Finally, we discuss interpretations such as fanout, cliques, and routes, which influence partitioning techniques, as long as the timing-oriented aspects like slack and gate delay.

Understanding these terminologies will provide readers with the necessary language and basic knowledge to navigate the complex and dynamic world of multi-level circuit partitioning. These ideas not only serve as stepping stones for our later assessments of partitioning approaches and optimization tactics, but they also equip us to deal with the changing issues given by modern electronic systems. We can start the analysis with the confidence that a solid foundation will pave the way for innovative solutions in this ever-changing industry.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

2.2 Terminologies and Definitions

2.2.1 Graph Representation

In the field of Very Large-Scale Integration (VLSI) circuit design, graphical representations play a pivotal role in modelling the complex interconnections and dependencies between various components of an integrated circuit. Two primary graphical models often used for this purpose are Directed Acyclic Graphs (DAGs) and hypergraphs. These representations have distinct characteristics, and understanding their differences is essential for selecting the most suitable model for a particular design task.

A Directed Acyclic Graph (DAG) is mathematically represented as a set of vertices and directed edges [37]. The set of vertices is denoted as V and consists of unique elements, expressed as $V = \{v_1, v_2, v_3, \ldots, v_n\}$, where v_i represents the ith vertex, and n is the total number of vertices. The connections between vertices are represented by a set of directed edges, denoted as E, where each directed edge is an ordered pair of vertices indicating the direction of the connection: $E = \{(v_i, v_j) \mid v_i, v_j \in V\}$. Importantly, a DAG is characterized by its acyclic nature, meaning there are no closed loops or cycles within the graph. This acyclic property is expressed as a condition ensuring that no sequence of directed edges can return to the same vertex. The mathematical representation of a DAG allows for precise analysis and manipulation, making it a fundamental concept in various mathematical and computer science contexts, including graph theory and data structure implementations.

DAGs are widely employed in VLSI circuit design due to their simplicity and efficiency in capturing the hierarchical and sequential nature of circuits [38]. A DAG consists of nodes (vertices) and directed edges (arcs) connecting them. In the context of VLSI, each node typically represents a logical component or a cell, while the directed edges represent the logical or data flow between these components. One of the key characteristics of DAGs is their acyclic nature, which means there are no closed loops or cycles in the graph. This acyclic property is particularly important because it ensures that signals do not encounter infinite feedback loops, guaranteeing predictable and finite signal propagation times. This is crucial in VLSI design, where accurate timing analysis is essential to prevent issues such as signal skew and metastability.

A hypergraph is mathematically represented as a set of hyperedges and vertices [37]. The set of vertices is denoted as V and consists of unique elements, expressed as $V =$

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

$\{v_1, v_2, v_3, \ldots, v_n\}$, where v_i represents the ith vertex, and n is the total number of vertices. In contrast to a standard graph, a hypergraph includes a set of hyperedges, denoted as E, which connects more than two vertices. Each hyperedge is represented as a subset of vertices, and the set of hyperedges can be expressed as:

$$E = \{e_1, e_2, e_3, \ldots, e_m\}$$

Here, each e_i is a subset of vertices, indicating the complex relationships that may involve multiple components simultaneously. The flexibility of hypergraphs is particularly valuable when dealing with non-hierarchical and complex connections in various contexts such as VLSI circuit design and relational databases. Hypergraphs do not have the acyclic property found in Directed Acyclic Graphs (DAGs), and this flexibility allows for the representation of cyclic dependencies, shared structures, and multiple connections. The mathematical representation of a hypergraph provides a foundation for understanding complex relationships and is a vital concept in mathematical modeling, data analysis, and various fields where non-binary relationships are significant.

As regarding VLSI circuits, hypergraphs, on the other hand, provide a more expressive and flexible representation for VLSI circuits compared to DAGs. In a hypergraph, nodes are still used to represent components, but edges are replaced by hyperedges, which can connect more than two nodes. A hyperedge can represent complex interconnections that may involve multiple components simultaneously. The flexibility of hypergraphs is valuable when dealing with more intricate aspects of VLSI design, such as shared buses, buses with multiple drivers, or components with multiple inputs or outputs. Hypergraphs allow for a concise representation of these complex relationships, making them particularly useful in scenarios where DAGs might become convoluted due to multiple connections.

The most fundamental difference is that DAGs are acyclic, while hypergraphs are not constrained by this property. Hypergraphs allow the representation of cyclic dependencies, which can be beneficial in some scenarios, but may also introduce complexities that need to be carefully managed. Hypergraphs are more complex and expressive than DAGs due to the presence of hyperedges. This complexity can be an advantage when modelling intricate circuit structures, but can also make analysis and manipulation more challenging. DAGs are generally more straightforward and intuitive for representing hierarchical and sequential relationships, while hypergraphs provide greater flexibility when dealing with non-hierarchical and complex connections, albeit at the cost of increased complexity.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

2.2.2 Physical Design Oriented definitions

In the realm of digital circuit design, the notions of circuit fanout, paths, and gates cliques play crucial roles in understanding and optimizing the behaviour and structure of complex circuits. These concepts provide essential insights into signal propagation, logical pathways, and efficient circuit organization. In this section, we delve into the definitions and significance of circuit fanout, paths, and gates cliques.

Circuit fanout refers to the number of logic gates or components that a single output signal can drive or feed into. In other words, it quantifies the capacity of a signal to be distributed to multiple destinations within a digital circuit. A high fanout implies that a signal is distributed to many gates, which can potentially lead to issues like signal degradation, increased propagation delay, and power consumption. Conversely, low fanout values indicate a more localized signal distribution, which can be advantageous in reducing signal integrity concerns and improving circuit performance.

In the context of digital circuits, a path represents a logical sequence of interconnected gates and components that connect an input to an output. Paths are instrumental in understanding the signal flow and logical dependencies within a circuit. They help in analysing propagation delays, critical paths, and overall circuit behaviour. Identifying and optimizing critical paths is essential for ensuring the efficient operation of a digital circuit, especially in applications where timing constraints are critical.

The half-perimeter wire length is a metric used to measure the total wire length required for interconnections in a digital circuit. It is computed as half of the sum of the

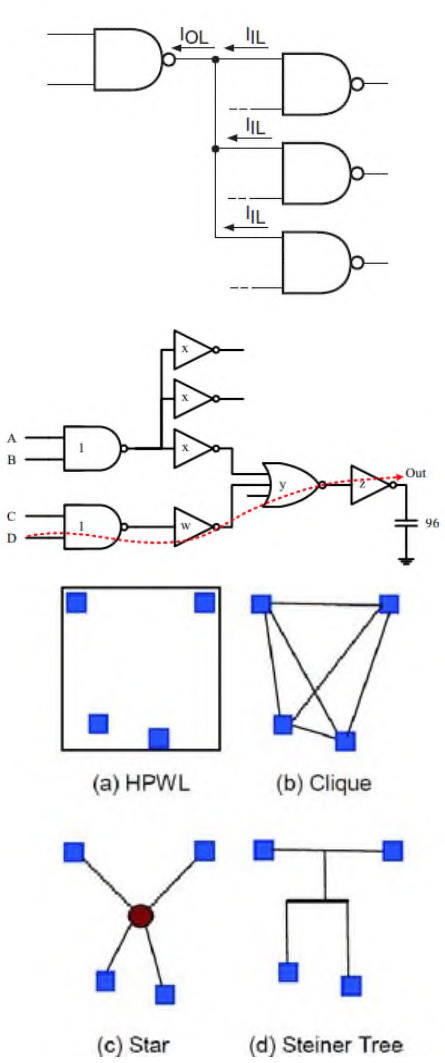


Figure 2.1: a) Simple fanout of 3 NAND gates starting be a same type gate driver. b) The red line presents the longest path of the sub-circuit. c) The top left sub figure presents the Half Perimeter Wire Length of the net. The other images present alternative methods of estimating the net wire length [26].

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

width and height of the layout or the integrated circuit. The half-perimeter wire length is a valuable indicator of the wire's spatial requirements and plays a role in minimizing wire congestion, which can be especially critical in high-density integrated circuits.

Design Rule Checks (DRCs) are a set of rules and constraints that ensure that a digital circuit's physical layout adheres to the fabrication technology's capabilities and specifications. DRCs encompass guidelines related to minimum feature sizes, spacing, and clearances between components. Verifying compliance with DRCs is a crucial step in the design process to avoid manufacturing defects and ensure the physical correctness of the circuit layout.

2.2.3 Timing Analysis Oriented definitions

Continuing, in digital circuit design, timing constraints are pivotal in ensuring that a circuit operates correctly and reliably. They provide critical guidelines for managing signal timing, and several metrics, such as Total Negative Slack, Worst Negative Slack, and gate delay, are employed to assess and optimize circuit performance. In this section, we delve into these essential concepts and their roles in digital circuit design.

Timing constraints are a set of guidelines and specifications that dictate when signals must arrive at their intended destinations within a digital circuit. They encompass parameters like setup time, hold time, clock-to-q delay, and clock frequency. These constraints ensure that signals meet the required timing specifications and allow for correct circuit operation, completing a set of instructions in the appropriate time period.

The first one is, $TotalNegativeSlack$ (TNS) is a crucial metric used to evaluate the overall timing performance of a digital circuit. It quantifies the total amount by which the actual signal arrival times exceed the required timing constraints. A positive TNS indicates that all signals meet their timing requirements, while a negative TNS signifies that some signals are failing to meet the constraints. Addressing negative TNS is essential to prevent issues like signal skew, data loss, or incorrect circuit operation.

Proceeding to the next one, which is $WorstNegativeSlack$ (WNS) identifies the most critical timing violation within a circuit. It represents the most negative slack value among all signals in the design, highlighting the specific signal that is furthest from meeting its timing constraints. Addressing the WNS is of paramount importance because it directly points to the weakest link in the circuit's timing performance. Improving the WNS often leads to overall performance enhancement.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Furthermore, gate delay refers to the time taken by a logic gate to process an input signal and produce the corresponding output. It is a fundamental parameter in digital circuit design and directly influences the signal propagation delay within the circuit. Reducing gate delay is a common optimization goal to minimize signal propagation time and enhance circuit speed.

Finally, Longest path delay, as the name suggests, is the delay along the most extended path in a digital circuit. It represents the maximum time it takes for a signal to travel from the input of the circuit to the output through the longest chain of gates and interconnections. Identifying and managing the longest path delay is essential for meeting overall circuit timing constraints, as it often dictates the circuit's maximum achievable operating frequency.

2.2.4 Vcycle flow

Last but not least, the final definition that we should mention is the V-cycle flow. The V-cycle flow is the main approach, as regarding the multilevel hypergraph partitioning algorithm, and is based on the concept of the multilevel paradigm. This flow aims to partition a hypergraph into k roughly equal parts, with the goal of minimizing the number of hyperedges connecting vertices in different parts. The algorithm consists of three phases: coarsening, initial partitioning, and uncoarsening and refinement.

In the coarsening phase, a sequence of successively coarser hypergraphs is constructed. This is achieved by merging groups of vertices together to form single vertices in the next level coarse hypergraph. There are multiple different algorithms for coarsening, like edge coarsening, hyperedge coarsening and first choice algorithm. These algorithms select pairs of vertices or hyperedges to be merged based on different criteria, such as heavy-edge maximal matching or independent sets of hyperedges. In the initial partitioning phase, a balanced random bisection of the coarsest hypergraph is computed. This partitioning is then carried along in the uncoarsening phase. During the uncoarsening and refinement phase, the bisection is successively projected to the next level finer hypergraph. At each level, an iterative refinement algorithm, such as the Fiduccia-Mattheyses (FM) or Kernighan–Lin (KL) algorithm, is used to further improve the bisection. The Vcycle flow is a powerful and efficient multilevel hypergraph partitioning algorithm. It utilizes innovative coarsening schemes and refinement algorithms to consistently produce high-quality partitionings. The algorithm has been extensively evaluated and compared to other algorithms, demonstrating its superiority in terms of both partitioning quality and runtime.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

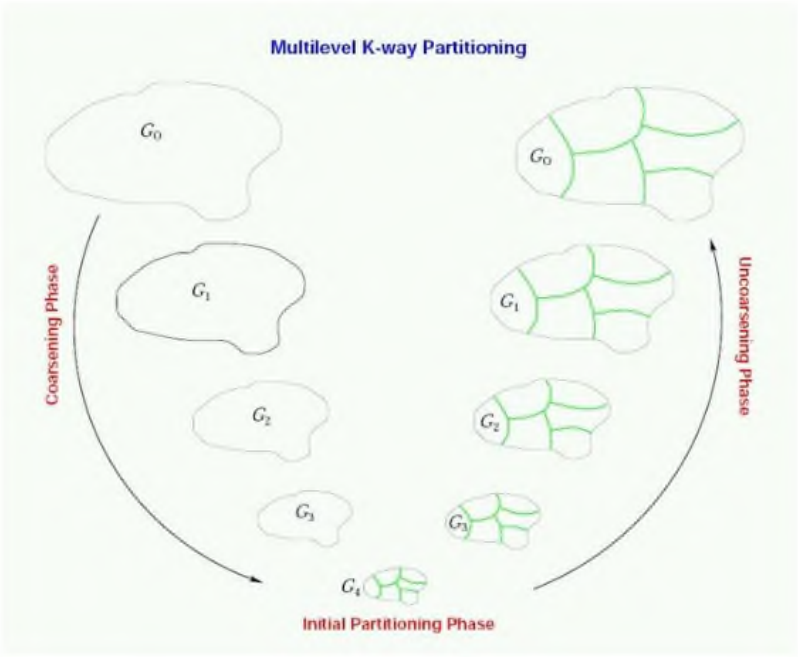


Figure 2.2: The three phases of the multilevel V-Cycle k-way graph partitioning flow [27].

2.3 Multi-Level Placement Application

2.3.1 Algorithm overview and objectives

Multi-Level Placement is a critical phase in the Application-Specific Integrated Circuit (ASIC) design process. Its primary objective is to efficiently place the logical components, such as gates and flip-flops, onto the physical layout of an integrated circuit. Unlike traditional placement algorithms, Multi-Level Placement leverages a hierarchy of abstraction levels to optimize performance, power consumption, and manufacturability simultaneously.

The algorithm aims to achieve several key objectives. Firstly, it optimizes the circuit's timing characteristics by reducing critical path delays and ensuring that setup and hold time requirements are met. Secondly, it minimizes wire length, a fundamental aspect of placement, by carefully arranging components to lower interconnect delays, power consumption, and manufacturing costs. Additionally, it addresses power efficiency by strategically placing components to minimize wire capacitance and dynamic power. Lastly, Multi-Level Placement focuses on ensuring signal integrity, addressing issues like electromigration and voltage drop, which are crucial for circuit reliability and robustness.

Hierarchical placement methods start with a global placement phase, which initializes the initial placement of hiearchy modules. This phase focuses on high-level interconnections

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

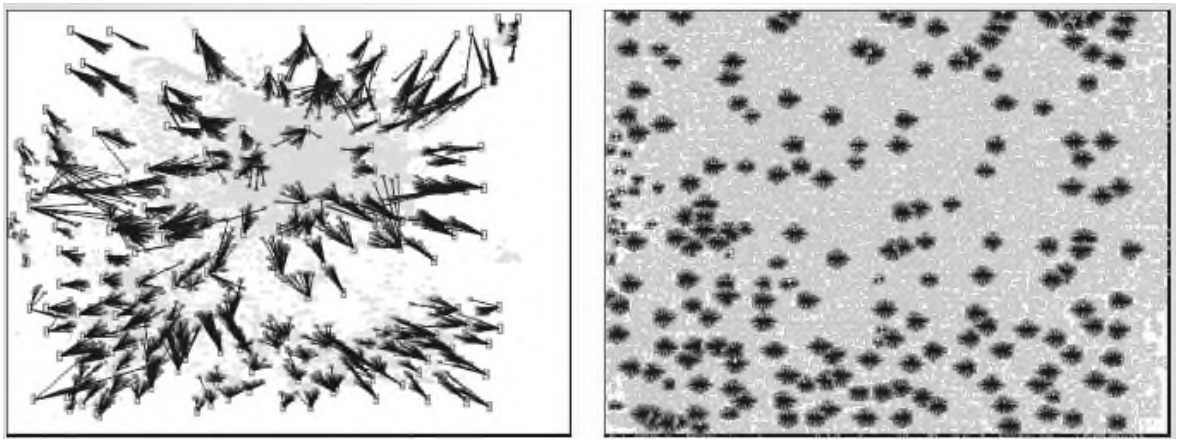


Figure 2.3: The left-hand side image presents the initial positions of circuit cells into the die and the forces represented by the black lines, while the second image presents the final positions of the cells after the force directed algorithm operation [28].

and provides a rough floorplan. Subsequently, the detailed placement phase optimizes the positions of the enclosed components at a finer granularity. It considers local interactions, maintaining legal distances, and meeting design rules. Legalization is another critical aspect, ensuring that the placement adheres to the physical design rules, such as minimum spacing.

2.3.2 Placement Algorithm types

Due to the fact that placement algorithms in the context of digital circuit design play a crucial role in determining the physical locations of logical components, it is more important to discuss the existing algorithm types [39]. There are two common types of placement algorithms are the ”Force-Directed” placement algorithm [40] and the ”Logarithmic Sum-Exponential” (LSE) placement algorithm. These algorithms differ in their approaches and optimization strategies.

The Force-Directed placement algorithm [40] is a physics-inspired approach used in digital circuit design. It views logical components as particles in a system and mimics physical forces to optimize their placement. Components are represented as nodes in a graph, and attractive forces exist between connected components, while repulsive forces act between unconnected ones. These forces are iteratively calculated, causing components to move until a stable and optimized placement is achieved. Force-directed algorithms are often employed in initial placement stages due to their speed and effectiveness in minimizing wire length.

On the other hand, the Logarithmic Sum-Exponential (LSE) placement algorithm [41]

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

$$\gamma \sum_{e \in E} [(log \sum_{v_k \in e} exp(x_k/\gamma) + (log \sum_{v_k \in e} exp(-x_k/\gamma) + (log \sum_{v_k \in e} exp(y_k/\gamma) + (log \sum_{v_k \in e} exp(-y_k/\gamma)]$$

Figure 2.4: During the log sum exponential placement method, such a mathematical expression must be minimized, in order to assign the circuit gates into their optimal positions.

takes a mathematical approach, presented in Figure 2.4, to placement optimization. It seeks to minimize a cost function, typically a weighted sum of wire length and other objectives, through mathematical optimization techniques. LSE placement algorithms are particularly useful when strict constraints are in place, such as minimum spacing or design rule requirements. They leverage convex optimization and a logarithmic barrier function to handle complex placement problems with multiple objectives, ensuring that the placement meets various constraints and trade-offs. A typical evaluation function is presented below in figure 2.4, which must be minimized in order to achieve the optimal gates position.

Of course, there are other approaches such as constraint-based placement [42] which takes into account various design constraints, such as minimum spacing, area constraints, and routing resources. These constraints are explicitly defined and enforced throughout the placement process. Constraint-based approaches are crucial for ensuring the manufacturability and reliability of the layout. One more method called Genetic Algorithms [43] are inspired by biological evolution. They employ techniques like selection, crossover, and mutation to evolve and optimize placement solutions. Genetic Algorithms are useful for exploring a wide search space and are adaptable to various placement objectives and constraints.

It is important to mention that the placement phase is often divided into global placement and detailed placement. Global placement establishes an initial arrangement, focusing on high-level interconnections and overall quality metrics, while detailed placement refines this arrangement to meet design rules and optimize component positions at a finer granularity. Designers often combine these approaches at various stages to meet specific design requirements, enabling the efficient, high-performance, and reliable implementation of digital circuits. The choice of placement method is tailored to the objectives and constraints of each design, allowing for a flexible and adaptable approach in the placement phase.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

2.3.3 Existing Industrial Tools

Hierarchical placement tools are vital for managing the intricacies of modern digital circuit designs, providing the means to break down complex layouts into manageable hierarchical blocks or modules while preserving the design hierarchy. Among the notable industrial tools known for their hierarchical placement capabilities are Cadence Innovus, Synopsys ICC2, Mentor Graphics Olympus-SoC, ANSYS RedHawk-SC, and Magma Talus.

Starting by Cadence, Innovus is a well-regarded tool offering robust hierarchical placement support, enabling multi-level hierarchies and efficient handling of large designs. It excels at achieving a balanced trade-off between runtime and quality, making it a favoured choice for intricate ASIC and FPGA projects. Moving on to Synopsys ICC2, another prominent tool, specializes in hierarchical placement for complex designs. It empowers designers to manage multi-level hierarchies, ranging from block-level to chip-level placement. ICC2 stands out for its proficiency in timing-driven placement and global optimizations, enhancing design performance.

Continuing, Mentor Graphics Olympus-SoC is optimized for system-on-chip (SoC) designs and offers comprehensive hierarchical placement capabilities. It excels in handling hierarchical blocks, enhances scalability, and integrates seamlessly with other EDA tools, making it a preferred option for complex SoC projects. On the other hand, ANSYS RedHawk-SC, primarily a power integrity tool, also features hierarchical placement and optimization capabilities. It emphasizes power-aware placement and is widely utilized in designs with stringent power constraints, such as mobile and IoT devices. Magma Talus is a versatile tool offering hierarchical placement solutions, focusing on hierarchical optimization, clock tree generation, and signal integrity. It is often the choice for larger designs where optimizing hierarchical placement is paramount for overall performance.

2.4 3D Chip Design flow Application

2.4.1 3D Flow Overview and Objectives

A 3D ASIC (Three-Dimensional Application-Specific Integrated Circuit) flow is a design and manufacturing process that involves the creation of integrated circuits in multiple layers or ”stacks” in three dimensions [44]. Unlike traditional 2D ASIC design, where components

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

are placed on a single silicon die, 3D ASICs are designed to have components stacked on top of each other in multiple layers. These layers are interconnected using through-silicon vias (TSVs), allowing for vertical integration. This approach offers several advantages, including improved performance, reduced power consumption, smaller form factors, and the ability to integrate different technologies on separate layers, a concept known as heterogeneous integration.

Through-Silicon Vias (TSVs) are essential for enabling communication between the different layers of a 3D ASIC [44]. TSVs are vertical interconnections that penetrate the silicon layers, facilitating power and signal distribution between the stacked components. Designing and manufacturing 3D ASICs comes with its set of unique challenges. These challenges include managing heat dissipation in a compact space, ensuring precise alignment of TSVs, and optimizing the placement and routing of components in three dimensions. Specialized tools and methodologies tailored for 3D design are required to address these challenges effectively.

3D ASICs have a broad range of applications, particularly in fields like data centres [45], where they can enhance the performance and efficiency of data processing and memory systems. The form factor is significantly reduced in 3D ASICs, making them suitable for portable devices [46]. The manufacturing of 3D ASICs demands advanced semiconductor fabrication techniques. Stacking multiple layers of components requires precision in aligning and bonding the individual dies together. Companies and foundries that specialize in 3D IC technology play a crucial role in this manufacturing process.

Heterogeneous integration is another notable aspect of 3D ASICs [47]. These chips can combine different types of components on separate layers, allowing for the integration of diverse technologies within a single package. As technology continues to advance, 3D ASICs are becoming more prominent, especially in applications where compact size and high performance are essential considerations. This evolution in IC design is shaping the future of semiconductor technology and its applications.

2.4.2 Different flow types

Various techniques are used in the realm of three-dimensional (3D) stacking in integrated circuits, each catering to different design requirements and applications. One such technique is Monolithic 3D Integration (M3DI) [48], which entails the vertical stacking of multiple layers of transistors on a single silicon substrate, interconnected through Monolithic Inter-

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

tier Vias (MIVs) [49], offering notable improvements in performance and power efficiency. Another popular method is 3D IC Stacking, which involves stacking separate dies, each of which contains a whole integrated circuit, on top of one another and connecting them with TSVs[50]. This method allows for the integration of various technologies or functionality on different dies. It is especially useful when various components require independent production procedures. TSV Technology is a critical component of 3D stacking. TSVs are vertical interconnects that pass between silicon layers, allowing communication between different levels or dies. These TSVs can be used in 3D IC stacking [51] to create a dense network of interconnections, allowing for better performance and interconnectivity.

The usage of silicon interposers [52] as a bridge between many dies within a package is introduced by stacking. These interposers include TSV networks and allow for the integration of multiple dies. This approach is ideal for applications that need the integration of numerous dies in a single package, such as high-performance computing or complex networking systems. Meanwhile, 2.5D Stacking is a hybrid of complete 3D stacking and classic 2D techniques [53]. It entails merging numerous dies onto an interposer, which is typically made of silicon. While it does not attain the same vertical density as complete 3D stacking, it is less expensive and more adaptable. 2.5D stacking is widely used in high-performance computing and artificial intelligence applications where form factor constraints are less stringent.

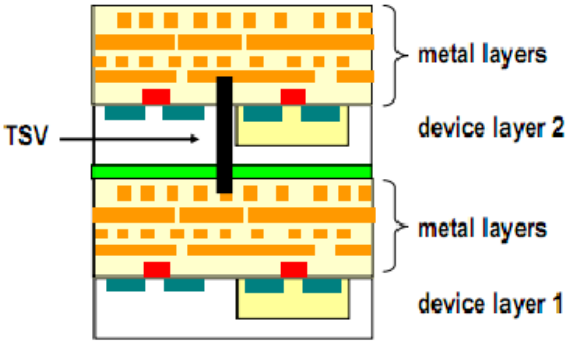


Figure 2.5: A conceptual view of a 3D IC chip, with a through-silicon-via (TSV) used as interconnect between two dies or wafers [29].

Finally, there is die-on-wafer [54] [55]. Stacking is the process of stacking one or more full dies on top of a wafer containing integrated circuits. When particular components, like as sensors or advanced memory, must be incorporated into a wafer containing digital logic, this approach is often used. It allows for the mixing and matching of diverse technologies within a single package. These various 3D stacking approaches enable designers and manufacturers to adjust their integration tactics to satisfy specific design needs, maximize performance, increase power efficiency, and solve form factor concerns. The specific application, design

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

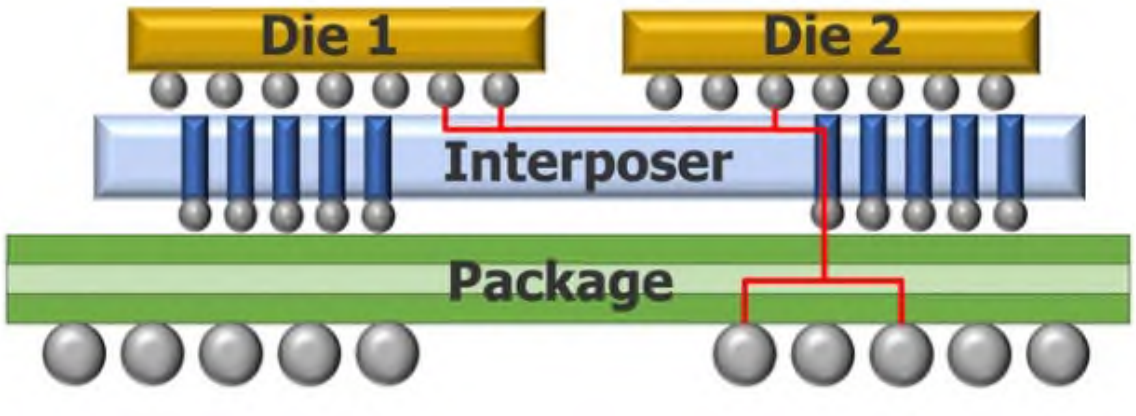


Figure 2.6: 2.5D-IC assembly that includes two substrates (silicon interposer + organic package) [30].

requirements, and accessible manufacturing capabilities all influence the choice of a particular 3D stacking approach.

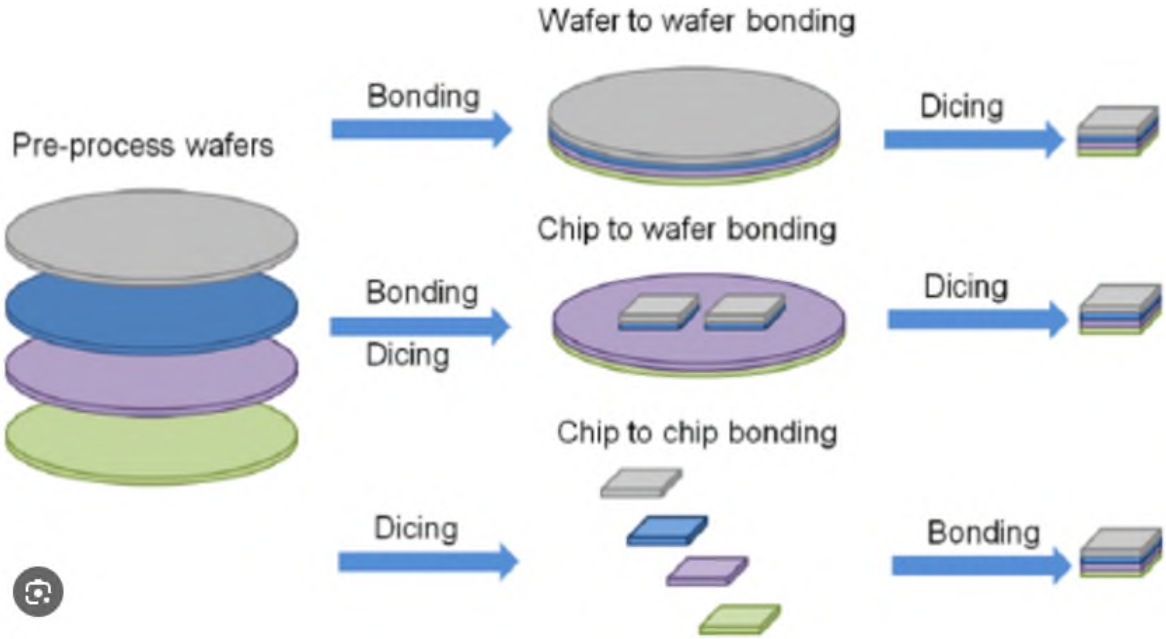


Figure 2.7: Different bonding technologies for 3D Integration circuit according to fabrication approach. [31].

2.4.3 Modern Challenges

Three-dimensional (3D) chip integration, while offering numerous advantages, presents several notable challenges [56] [44]. One of the primary concerns is heat dissipation. As

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

components are stacked vertically in 3D integration, heat generated in one layer can affect the layers above and below, potentially leading to thermal issues, rendering effective thermal management solutions essential to prevent overheating.

Another challenge lies in the design and reliability of Through-Silicon Vias (TSVs), which serve as critical vertical interconnects. Designing TSVs to be both reliable and manufacturable is a complex task. Factors like TSV placement, fill materials, and TSV-induced stress must be carefully addressed to ensure proper functionality. Designing for 3D integration introduces increased complexity, as designers need to consider vertical placement, TSVs, and thermal management in addition to traditional 2D design considerations. Designing for manufacturability and ensuring proper alignment across multiple layers require intricate and precise methodologies.

The manufacturing process for 3D integration is another area of challenge. It involves wafer thinning, die stacking, and microassembly. Precision manufacturing is crucial to ensure that dies are correctly aligned, bonded, and rigorously tested. Heterogeneous integration, which involves integrating different technologies or materials in 3D, can be challenging due to differences in thermal expansion coefficients and material properties. Managing these variations is essential to prevent stress-induced failures. Testing 3D integrated devices is more challenging than traditional 2D chips. Accessing and testing individual layers can be complex, and techniques for ensuring high yield and detecting and repairing defective components are critical. Furthermore, the lack of industry-wide standards for 3D integration hinders interoperability between different vendors and tools. The development of common standards is crucial to promote adoption and ensure compatibility.

Lastly, as devices become smaller and more power-efficient, power delivery and signal integrity can become challenges in 3D integration. Ensuring that power is distributed effectively, and that signals maintain their integrity, is a critical aspect of overcoming these challenges. Addressing these issues requires a collaborative effort between semiconductor manufacturers, designers, and researchers to develop new technologies, tools, and best practices specific to 3D integration. Despite the difficulties, the benefits of improved performance and energy efficiency continue to drive research and innovation in this field.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Chapter 3

Related Work

3.1 Introduction

Partitioning circuits is a key and difficult challenge in the ever-changing environment of electrical design automation. As the demand for increasingly complicated and efficient integrated circuits grows, so does the need for improved circuit partitioning approaches. This chapter serves as the starting point for our research of multi-level circuit partitioning, a topic that is critical in the creation of complex electronic systems.

This chapter aims to provide a comprehensive background for our investigation into multi-level circuit partitioning, with a particular focus on the foundational concepts, historical developments, and the contemporary challenges faced in this intricate field. By understanding the complexities and intricacies of multi-level circuit partitioning, we can pave the way for innovative approaches and solutions that address the ever-growing demands of modern electronic systems. To achieve this, we will delve into the historical evolution of circuit partitioning techniques, the key drivers necessitating its advancement, and the state-of-the-art methodologies that researchers and engineers employ to tackle the challenges presented by today's cutting-edge technologies.

Our journey through this chapter will lay the groundwork for the subsequent discussions and analyses of various partitioning techniques, optimization strategies, and the potential for advancements in multi-level circuit partitioning. It is our hope that this exploration will not only contribute to the scholarly discourse on this subject but also inspire practical, real-world solutions for the design and implementation of complex integrated circuits in an era of unprecedented technological innovation.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

3.2 Multi-Level Clustering

3.2.1 Algorithm Overview and Objectives

Multi-level clustering techniques are critical in the design of Very Large Scale Integration (VLSI) circuits. These algorithms provide a methodical approach to dealing with the complexities and challenges of current semiconductor devices, which are made up of millions, if not billions, of transistors and interconnections. Because of the overwhelming complexity, a disciplined mechanism for grouping circuit components into meaningful groups is required, and multi-level clustering serves this goal well. Multi-level clustering approaches, which are tailored specifically for VLSI design, aim to expedite the design process by facilitating the decomposition of large-scale circuits into more manageable and optimal sub-modules. This hierarchical architecture provides a number of advantages for VLSI designers and engineers, as it streamlines the design process while allowing for a more ordered and systematic approach to addressing the complexity of VLSI circuits.

Multi-level clustering techniques are used for area and power optimization in addition to complexity control. Given the necessity of decreasing chip space and power consumption in VLSI circuits, these methods aid in identifying crucial locations that demand special attention, enabling efficient resource allocation and power distribution.Furthermore, improving signal integrity is a major goal of multi-level clustering in VLSI design. The algorithms aid in the organization of components to minimize signal interference and path lengths, resulting in dependable and high-performance circuitry.

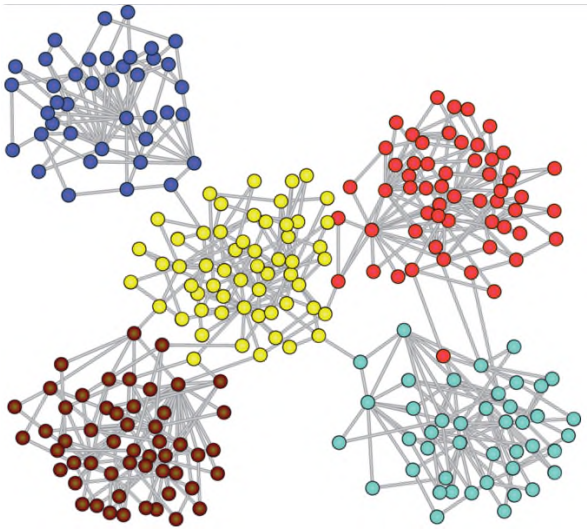


Figure 3.1: General clusters approach on a directed graph [32]

Manufacturability and yield enhancement are also important factors in VLSI design, and multi-level clustering can help with both. These algorithms lead to increased yield, cheaper production costs, and enhanced manufacturability by arranging components in ways that mitigate manufacturing difficulties. Furthermore, multi-level clustering techniques strive to im-

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

prove overall circuit performance. They boost speed, reduce latency, and maximize resource efficiency inside the VLSI circuit by isolating important modules and optimizing their connections. Finally, considering the constant increase in the complexity of VLSI circuits, the scalability of these methods is critical. Multi-level clustering algorithms are built to be scalable, allowing for larger and more complex designs without losing performance or economy, ensuring their relevance in an ever-changing industry.

3.2.2 Algorithm types

Multi-Level Clustering algorithms are particularly valuable in the modern semiconductor industry because all of the above listed objectives are critical for the chip manufacturing process. As a result, it is necessary to discuss the many ways that are currently being employed in industry to handle this ASIC flow stage. The also called hierarchical clustering techniques, can be broadly categorized into a few categories, as presented in Figure 3.2, with the most known of them to be the agglomerative and divisive approaches [57].

The more commonly used of the two is agglomerative hierarchical clustering [58]. It starts with each data point as a separate cluster and then merges smaller clusters into bigger ones. The procedure begins with the assumption that each data point is a separate cluster. The algorithm then iteratively merges the two closest clusters, continuing until all data points are part of a single, comprehensive cluster. One of the distinguishing characteristics of agglomerative clustering is the generation of a dendrogram, which is a tree-like structure that depicts the clustering hierarchy. The dendrogram's branches represent the merging of clusters at various phases. You can determine the amount of granularity in your clusters by visually studying the dendrogram and selecting an acceptable cut-off point. This allows for greater freedom in analysing the data and comprehending the links between data points, making it applicable to a wide range of applications.

Divisive hierarchical clustering [59], on the other hand, adopts a different strategy. It first groups all the data points into a single cluster before repeatedly breaking them up into smaller clusters. Although less popular, this strategy has its advantages in some contexts. The technique divides a cluster periodically into two smaller clusters, eventually resulting in a tree-like structure like a dendrogram but showing the division of clusters. When you assume that data naturally falls into a layered or hierarchical structure, dividing hierarchical clustering can be helpful. But compared to agglomerative clustering, it is frequently computationally

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

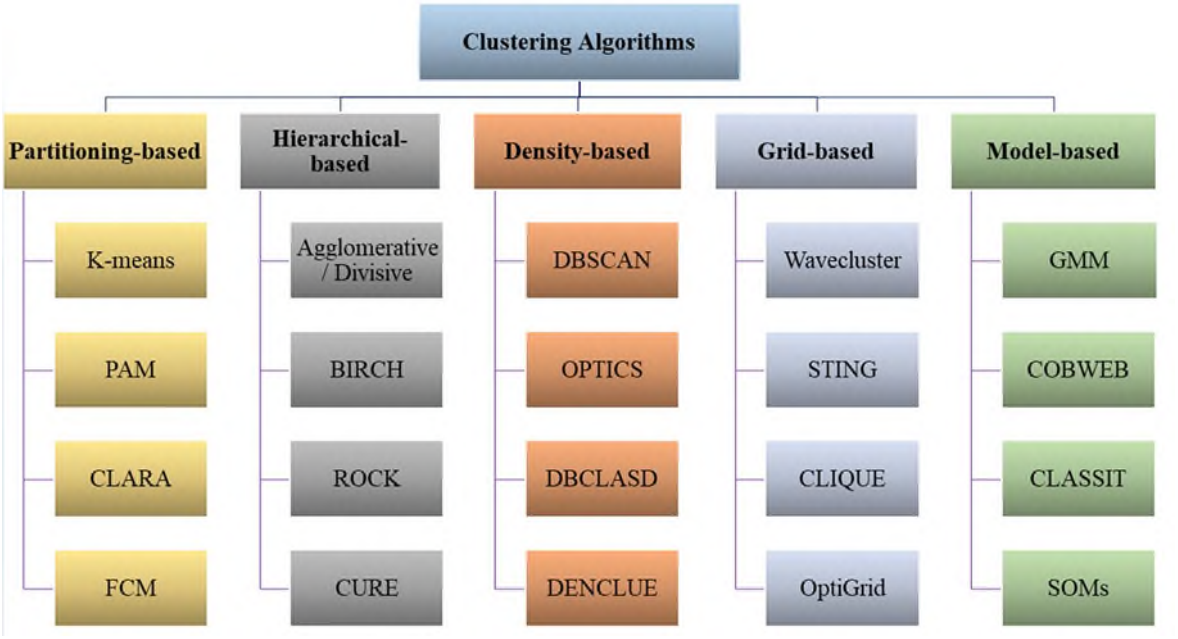


Figure 3.2: Broad classification of clustering algorithms [33].

more demanding and sophisticated, which restricts where it can be used.

Of course, there are more sub-categories addressing the clustering phase, each one exploiting different circuit characteristics. The most well-explored between them are the partitional clustering methods like K-Means [60] partition data into non-overlapping clusters, and density-based approaches such as DBSCAN [61] which excel at discovering clusters with varying shapes and sizes. Another subset of methods is the non-parametric algorithms like Mean-Shift which they find cluster centres by shifting towards high-density regions, exploiting circuits levels, while probabilistic methods like Gaussian Mixture Models model data as a mixture of Gaussian distributions. Finlay, the analysis would be incomplete if the Spectral clustering approach [62] was not included, which employs eigenvalues for cluster formation, utilising the Laplacian matrix of the circuit.

3.2.3 Existing algorithms and tools

In previous paragraphs are presented the objectives and main types of clustering approaches. To complete the presentation of the clustering phase related work, it is necessary to bring forward the most used and well established tools of this area of interest. The first method is called $edgecoarsening$ [2]. In this method, a heavy-edge maximal matching of the vertices of the hypergraph is computed to select the pairs of vertices. These vertices are then

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

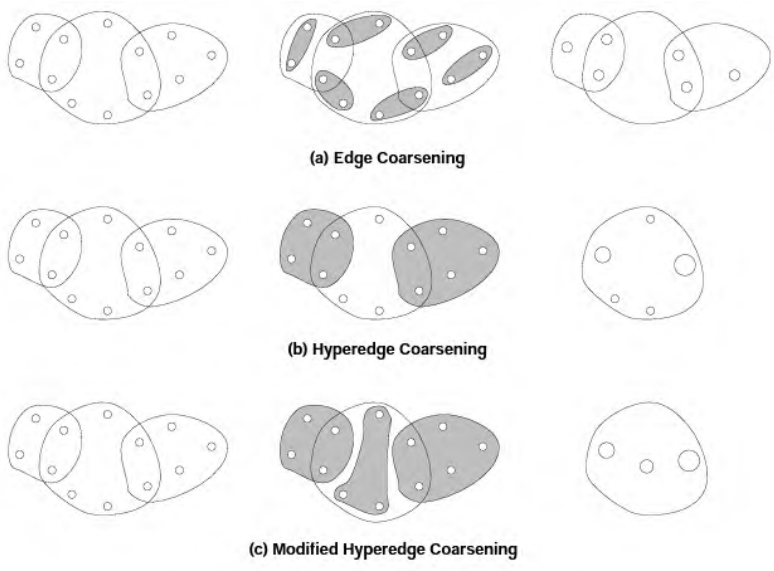


Figure 3.3: Different edge coarsening techniques and the coarsening they induce [2].

merged together to form a single vertex in the next level coarse hypergraph. The heavy-edge maximal matching is a matching that maximizes the weight of the edges in the matching. The weight of an edge is the sum of the weights of the vertices it connects. The matching is called maximal because it is not possible to add any more edges to the matching without violating the matching property. The edge coarsening method is preferred when the hyperedges are relatively small and the weights of the vertices are not too different from each other. This is because the heavy-edge maximal matching may not be able to capture the important structure of the hypergraph when the hyperedges are large or the weights of the vertices are significantly different from each other.

Another similar approach is the First Choice (FH) [63] which is a method used in hypergraph partitioning algorithms to determine how groups of vertices should be merged together in the next level coarse hypergraphs. It starts by creating an empty list of groups and then iterates through each vertex in the hypergraph. For each vertex, the algorithm checks if it is highly connected to any vertex already in a group. If it is, the vertex is added to that group. If not, a new group is created with the vertex as the only member. This process continues until all vertices have been assigned to a group. The resulting groups of vertices are then merged together to form single vertices in the next level coarse hypergraph. The goal of the FirstChoice algorithm is to create groups of vertices that are well-connected within themselves, which can help improve the efficiency of subsequent refinement algorithms in the partitioning process.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

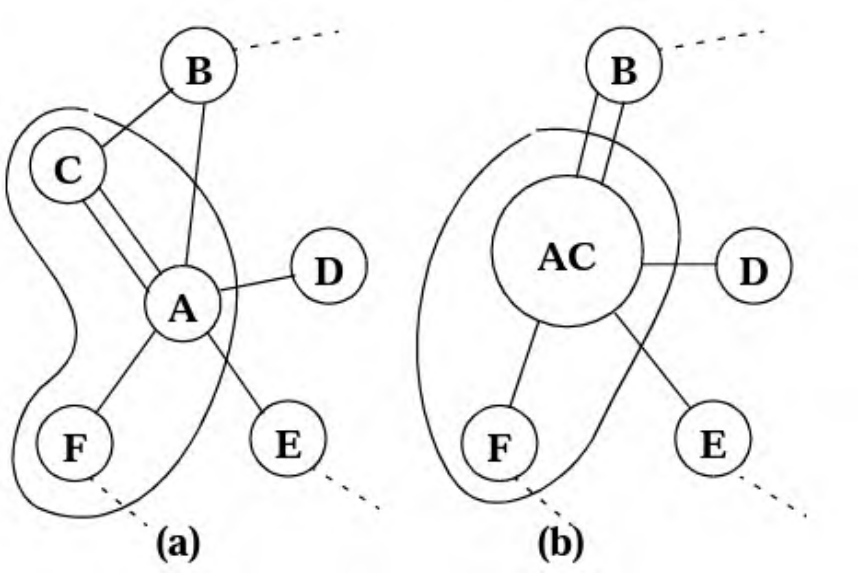


Figure 3.4: Clustering a pair of objects A and C using either the First Choice or the Best Choice [34].

The next clustering algorithm discussed in [34] is called the best-choice bottom-up clustering algorithm. The algorithm starts by initializing a priority queue (PQ) with all objects in the netlist. Then enters a loop where it continues to cluster objects until the target number of objects is reached. In each iteration of the loop, the algorithm picks the top tuple (u, v, d) from the PQ, which represents the pair of objects with the highest clustering score. These objects are then clustered together to create a new object u'. After clustering, the netlist is updated to reflect the new object u' and its connections. The algorithm then calculates the closest object v' to u' and its clustering score d'. This information is inserted into the PQ. The algorithm also includes a lazy-update technique to reduce the runtime. Instead of updating clustering scores for all neighbor objects after each clustering operation, the algorithm marks them as invalid. Only when an object is picked from the top of the PQ, its clustering score is updated if necessary. This lazy-update approach significantly reduces the number of score update operations on the PQ, leading to faster runtime. Additionally, the algorithm includes methods for controlling cluster sizes and handling fixed blocks. Cluster sizes can be indirectly controlled by using a clustering score function that is inversely proportional to the size of the cluster objects. Direct size control can also be applied by imposing hard or soft bounds on the cluster sizes.

Finally, a new approach to effective circuit clustering called RW-ST (Random Walk - Self-Tuning) algorithm is presented in paper [64]. The goal of the paper is to reduce the problem

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

size of layout synthesis algorithms by condensing the circuit netlist. The RW-ST algorithm is based on a random walk in the circuit netlist graph. The algorithm starts by constructing a random walk in the netlist graph. A random walk is a stochastic process that moves from the current module to a random adjacent module. The cover time of the random walk is the expected length of the walk that visits all vertices in the graph. The paper shows that the cover time of a random walk in a d-regular graph of n nodes is $O(n^2)$ and $O(nlogn)$, which means that a single random walk can sample the entire netlist graph.

Following, the algorithm identifies cycles in the random walk. A cycle is a subsequence of nodes in the walk that starts and ends at the same node. The cycles represent potential clusters in the netlist. The algorithm then computes the sameness value for each pair of nodes in the netlist. The sameness value measures the commonality of the sets of nodes visited in cycles originating from each pair of nodes. Based on the sameness values, the algorithm clusters node pairs with sameness greater than zero. The clusters are formed by merging the nodes that have high sameness values. The resulting clusters represent the condensed netlist. The algorithm also introduces a quality measure called DS (Degree-Separation) to evaluate the effectiveness of the clustering. The DS quality of a clustering is the weighted average of the cluster degree and cluster separation. The cluster degree is the average number of nets incident to each module in the cluster, and the cluster separation is the average length of the shortest path between two nodes in the cluster. The higher the DS quality, the better the clustering. The paper presents experimental results comparing the RW-ST algorithm with other clustering methods. The results show that RW-ST consistently produces better clusterings in terms of DS quality. The algorithm is also applied to two-phase Fiduccia-Mattheyses partitioning, and it is shown to improve the solution quality compared to standard FM partitioning.

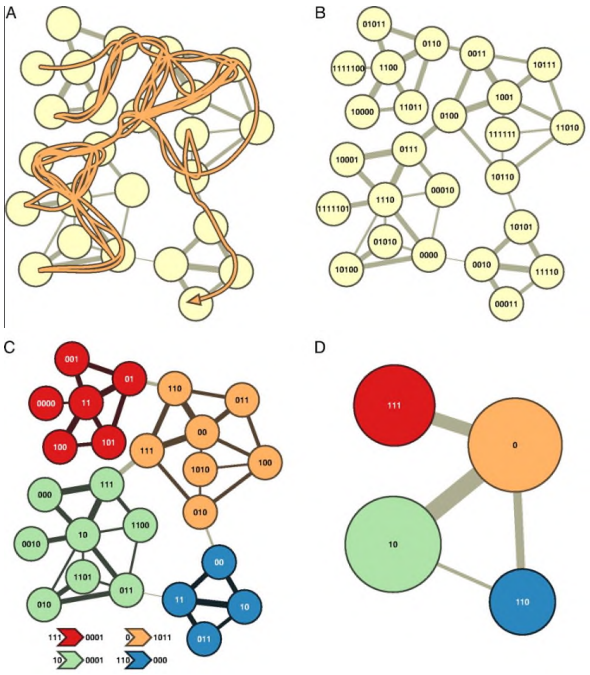


Figure 3.5: Maps of random walks on complex networks reveal community structure [35]

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

3.3 Multi-Level Partitioning

3.3.1 Algorithm overview and objectives

Partitioning algorithms play a crucial role in electronic design automation (EDA) for the optimization and decomposition of complex digital circuits. These algorithms are designed to break down a large circuit into a predefined number of smaller, more manageable sub-circuits, facilitating further optimization and enhancing the efficiency of circuit implementation. In this section, we provide an overview of partitioning algorithms and outline their primary objectives. Multi-level circuit partitioning algorithms operate on the principle of hierarchical decomposition. They divide the original circuit into smaller components in a hierarchical fashion, starting from the entire design and proceeding to smaller granularity levels. This approach is essential for various stages of digital design, including logic synthesis, placement, and routing. The goal is to achieve a partitioning that balances the trade-offs between partition size and complexity, ultimately optimizing the circuit's performance and ease of implementation.

In achieving these goals, partitioning algorithms pursue several key objectives. First and foremost, they seek to minimize the number of cut nets, which represent the connections between partitions. Minimising cutsize, is critical for applications like parallel static timing analysis [65] where each inter-partition connection stands as unconstraint path, introducing notable error in the analysis. Simultaneously, these algorithms strive to minimise the area ratio between the larger and smaller partition to produce area balanced groups of gate level instances. This objective is vital for cloud based operations [66], as the number of instances in each block, which is assigned into a different agent, is proportional to the computational load of each agent. Also, the third target of partitioning algorithms is to minimise critical path fragmentation in order to produce a result suitable for timing driven operations. The final objective is aligned with the first one, assigning an extra notion of criticality into the timing critical paths. These paths are the longest paths of the circuits, which most often have the greatest path delay, significantly affecting the timing closure of the circuit.

3.3.2 Algorithm types

Due to the importance of Multi-Level partitioning in semiconductor industry, there are many partitioning approaches in the literature addressing the previously described objectives.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

The most well-established methods between them are the recursive bipartitioning, the kway partitititioning and the flow based approaches. Furthermore, there are more recent techniques introducing machine learning methodologies towards that cause, showing notable improvements on the quality metrics results.

The recursive bipartitioning algorithm begins with the whole digital circuit and systematically divides it into two roughly equal sub-circuits, hence the ”bipartitioning” designation. The division is performed recursively, meaning that each of the two resulting sub-circuits can themselves be subdivided in the same manner, creating a hierarchical structure of partitions. The primary objective of this method is to minimize the cut, which represents the number of connections (or nets) that cross the boundary between the two sub-circuits. By doing so, it ensures that the logical connectivity of the circuit is maintained while optimizing for performance or other design criteria. This recursive process continues until a predetermined granularity level is achieved, or specific design constraints are met.

Moving on to the second approach, where k represents the number of partitions desired, which is typically specified by the designer. The primary objective of the k-way partitioning method is to create balanced partitions with roughly the same number of components or nodes in each partition. These balanced partitions help optimize various aspects of the circuit, such as performance, area utilization, and manufacturability. The method is often guided by a cost function, which may include minimizing the number of connections between partitions (cut) or optimizing other design criteria like meeting area constraints. The k-way partitioning process can be iterative, where partitions are refined in each step to approach a more balanced and optimized solution. The choice of k can have a significant impact on the quality of the partitioning, as it affects the granularity of the divisions. A smaller ’k’ can lead to finer-grained partitions, while a larger k may produce coarser partitions.

Both of the previous categories often utilise a version of either Fiduccia-Mattheyses (FM) [67] or Kernighan-Lin (KL) [68] partitioning algorithms to create the initial partitions or optimise the final result. Both of these algorithms are presented in Algorithm 1 and Algorithm 2 respectively. Their main idea is to iteratively test tentative moves of objects between partitions to determine the minimum achievable cutsize. However, even though this exhaustive method is effective, it is quite expensive to be introduced in newer circuits. Thus, more modern studies are based on their fundamental idea introducing additional heuristics to reduce their execution time overhead.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Algorithm 1 Fiduccia-Mattheyses (FM) Algorithm

Require: Graph G representing the circuit, Initial partitioning

Ensure: Balanced partitioning with minimized cut size

```
bestPartitioning ← Initial partitioning
minCutSize ← CalculateCutSize(G, bestPartitioning)
moved ← True
while moved do
    moved ← False
    for each cell c in G do
        currentPartition ← PartitionOf(c)
        gain ← CalculateGain(c, currentPartition)
        if gain > 0 then
            for each neighbor n of c do
                gain ← gain + CalculateGain(n, currentPartition) −
                CalculateGain(n, OtherPartition(n))
            end for
            Move c to the other partition
            moved ← True
            Update cut size
            Update balance criterion
        end if
    end for
    if cut size is smaller than minCutSize then
        bestPartitioning ← Current partitioning
        minCutSize ← Current cut size
    end if
end while

return bestPartitioning
```

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Algorithm 2 Kernighan-Lin (KL) Algorithm

Require: Graph G representing the circuit, Initial partitioning

Ensure: Balanced partitioning with minimized cut size

```
bestPartitioning ← Initial partitioning
minCutSize ← CalculateCutSize(G, bestPartitioning)
moved ← True
while moved do
    moved ← False
    for each cell pair a in one partition and b in the other partition do
        gain ← CalculateGain(a, b)
        if gain > 0 then
            Swap cells a and b between partitions
            moved ← True
            Update cut size
            Update balance criterion
        end if
    end for
    if cut size is smaller than minCutSize then
        bestPartitioning ← Current partitioning
        minCutSize ← Current cut size
    end if
end while

return bestPartitioning
```

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Last but not least, the flow based algorithms use network flow algorithms to partition a circuit into two or more subcircuits with balanced weights and minimum net cuts [69]. Network flow algorithms are based on the concept of finding the maximum amount of flow that can pass through a network of nodes and edges, where each edge has a capacity and a cost. A network flow algorithm can also find the minimum cut of the network, which is the minimum capacity of edges that need to be removed to disconnect the network. The goal is to find a partition of the network that balances the weights of the nodes in each subcircuit and minimizes the cost of the edges that cross the subcircuits. Such algorithms are the Min-net-cut partitioning, which tries to minimize the number of nets that cross the subcircuits, regardless of their weights or costs, the Min-cut partitioning, which targets to minimize the total weight or cost of the nets that cross the subcircuits and the Ratio-cut partitioning, which aims to minimize the ratio of the cut size to the subcircuit size, which is a measure of how balanced and compact the subcircuits are.

3.3.3 Existing algorithms and tools

Continuing with our analysis, it is necessary to review the existing tools, some of which will be used to evaluate our proposed algorithm. The first and oldest tool is named MLpart, presented in [1], targeted to partition hypergraphs. The MLpart algorithm is a multilevel partitioning algorithm presented in VLSI CAD physical design. It is a fundamental optimization technique that aims to divide the nodes of a hypergraph into groups of approximately equal total weight while minimizing the number of hyperedges that are cut. The algorithm follows a three-step process: clustering, top-level partitioning, and refinement or uncoarsening. In the clustering step, the hypergraph nodes are combined into clusters based on their connectivity, resulting in a smaller, clustered hypergraph. This step is repeated until there are only a few hundred clusters left, creating a hierarchy of clustered hypergraphs.The top-level partitioning step requires an initial solution generation. This is done by assigning nodes to partitions in decreasing order of size using a biased random selection method. The goal is to keep the slacks (the difference between the assigned area and the maximum allowed area) approximately equal while introducing randomness. Once all partitions reach their minimal required cell area, slacks are computed relative to the maximal allowed areas. The top-level partitioning is performed using the CLIP-FM algorithm with the requested tolerance for the original partitioning problem. The best solution from three independent starts is further refined using

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

the LIFO-FM algorithm. CLIP-FM is slower but produces better solutions, while LIFO-FM balances solution quality and runtime. The refinement or uncoarsening stage involves projecting solutions from one level to the next and iteratively improving them using the FM algorithm. This stage may stop before reaching the lowest-level hypergraph, and clustering or refinement may be resumed earlier than usual. The hMETIS partitioning program introduced additional heuristics such as hyperedge removal and V-cycling, which are critical to its performance but require careful tuning. The MLpart algorithm improves on the baseline implementation by introducing several new techniques. One technique is the use of a relaxed move acceptance criterion, which accepts moves that do not increase the violation of balance constraints. Another technique is the randomization of gain computation at the beginning of each pass, which is done by computing gains of legal moves in a random order. The algorithm also includes a preferential placement technique that encourages the movement of nodes adjacent to fixed nodes.

The second and relatively newer tool is hMETIS presented in [2], [3]. The hMETIS algorithm is a multilevel hypergraph partitioning algorithm that aims to find high-quality solutions for partitioning large and irregular hypergraphs. It is designed to scale well to very large hypergraphs and requires relatively small amounts of time. This tool also consists of the basic three Vcycle phases coarsening, initial partitioning, and uncoarsening / refinement. As mentioned before, in the coarsening phase, the algorithm successively reduces the size of the hypergraph by grouping vertices into disjoint clusters and collapsing them into a single vertex. To do so, this process is performed using various coarsening schemes, such as edge coarsening, hyperedge coarsening, or modified hyperedge coarsening. Once the coarser hypergraphs are obtained, the initial partitioning phase begins. In this phase, the smallest hypergraph is partitioned using a bisection algorithm. The bisection algorithm aims to divide the hypergraph into two equal-sized partitions while minimizing the number of hyperedges cut. This initial partitioning serves as the starting point for the subsequent refinement phase. The uncoarsening and refinement phase is where the solution of the smallest hypergraph is projected to the next level finer graph and iteratively refined to improve the quality of the partitioning. The refinement algorithm used in this phase is a variation of the Fiduccia-

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Mattheyses (FM) algorithm. The FM algorithm iteratively tries to find subsets of vertices in each partition that can be moved to other partitions to improve the partitioning quality without violating the balance constraint. This iterative process continues until no further improvement can be made. Throughout the algorithm, randomization is used to select vertices for matching in the coarsening phase and to determine the order of vertex movements in the refinement phase. This randomization introduces some level of randomness into the algorithm, but it also allows for exploration of different possible solutions.

The next tool in our list is the KaHyPar (Karlsruhe Hypergraph Partitioning) [4], [5], [6]. This framework is a high-quality hypergraph partitioning algorithm that aims to divide a hypergraph into balanced and heavily-connected partitions. It employs a multi-level approach, combining various heuristics and techniques to achieve superior solution quality. The algorithm consists of several key components and phases. Firstly, KaHyPar uses a semi-dynamic hypergraph data structure that allows efficient vertex and hyperedge deletions and reversals. This data structure is designed to support the partitioning process without considering insertions of additional vertices or nets. To compute the partitions, KaHyPar supports both direct k-way partitioning and recursive bisection (RB) approaches. In direct k-way partitioning, the hypergraph is directly partitioned into k blocks, while in RB, a bipartition of the initial hypergraph is computed recursively until k blocks are obtained. KaHyPar employs two preprocessing techniques to improve the partitioning process. The first technique is pin sparsification, which reduces the number of pins (connections) in the hypergraph to speed up the overall process. The second technique is community-aware coarsening, which infers information about the community structure of the hypergraph to guide the coarsening process. The coarsening phase reduces the size of the hypergraph by merging vertices and hyperedges to create a coarser representation. KaHyPar uses a coarsening algorithm that restricts contractions to blocks of the previous solution and uses either the old or a newly computed solution as the initial par-

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

tition. For generating the initial partition, KaHyPar employs a portfolio-based approach. It uses multiple algorithms or heuristics to create different initial partitions and selects the best one based on certain criteria. To refine the initial partition, KaHyPar utilizes a localized local search algorithm. It applies V-cycles, which involve n-level coarsening and refinement, to improve the solution quality. The flow-based refinement phase further improves the solution quality by optimizing the connectivity metric. KaHyPar uses flow algorithms to compute maximum flows in the hypergraph and adjusts the partition accordingly. In addition to these components, KaHyPar incorporates a memetic algorithm, which is a genetic algorithm that also employs local search. It evolves a population of solutions using recombination operators with more than two parents, ensuring that the offspring is no worse than the parents. This allows for extensive exploration of the global solution space.

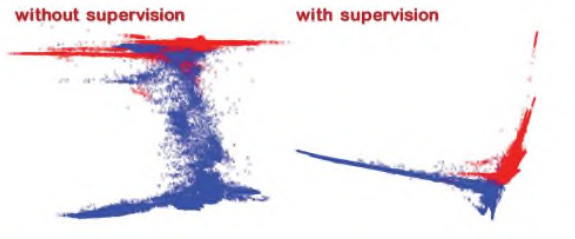
Another worth mentioning tool, is the PaToH [7] (Partitioning Tool for Hypergraphs) which is also a hypergraph partitioning specialised tool. The PaToH starts by coarsening the original hypergraph into a sequence of smaller hypergraphs. This coarsening is achieved by merging disjoint subsets of vertices into clusters, where each cluster forms a single vertex in the coarsened hypergraph. The weight of each vertex in the coarsened hypergraph is equal to the sum of the weights of its constituent vertices in the original hypergraph. The net set of each vertex in the coarsened hypergraph is the union of the net sets of its constituent vertices. After the coarsening phase, the algorithm proceeds to the initial partitioning phase. Here, a bipartition is found for the coarsest hypergraph using various initial partitioning techniques. The goal is to find a balanced bipartition that minimizes the cutsize. PaToH includes different random partitioning methods as well as variations of the Greedy Hypergraph Growing (GHG) algorithm for this step. Finally, the uncoarsening phase begins, where the bipartition found in the previous step is projected back to the original hypergraph. This projection is achieved by assigning the constituent vertices of each cluster in the coarsened hypergraph to the same part in the original hypergraph. The resulting partition is then refined using iterative improvement heuristics based on the Kernighan-Lin (KL) and Fiduccia-Mattheyses (FM) algorithms. These heuristics aim to further minimize the cutsize


PaToH (Partitioning Tool for Hypergraphs)

Ümit Çatalyürek[1], Cevdet Aykanat[2]
[1]The Ohio State University, Columbus, OH, USA
[2]Bilkent University, Ankara, Turkey

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

by swapping or moving vertices between parts while maintaining balance. Throughout the algorithm, PaToH provides various customization options, such as different coarsening and refinement algorithms, as well as parameters to control the balance and cutsize objectives. The algorithm also supports multi-constraint hypergraph partitioning, where each vertex has multiple weights associated with it, and partitioning with fixed vertices.



The latest update on partitioning tools is named SPECpart [8], [9], and it is designed by the same authors with MLpart framework. SpecPart is a supervised spectral framework for hypergraph partitioning solution improvement. It addresses two limitations of state-of-the-art hypergraph partitioners: (i) the reliance on local neighborhood structure during hypergraph coarsening without fully considering the global structure, and (ii) the potential stagnation on local minima during refinement heuristics. The SpecPart algorithm consists of several key components. First, it incorporates pre-computed hint solutions into a generalized eigenvalue problem. By solving this problem, SpecPart obtains high-quality vertex embeddings that capture the balanced partitioning objective and global hypergraph structure. This step leverages initial high-quality solutions from multilevel partitioners as hints. Next, SpecPart constructs a family of trees from the vertex embedding. These trees distill the cut structure of the hypergraph and serve as a basis for exploring a large space of candidate solutions. A tree-sweeping algorithm is used to partition the trees efficiently and generate potential solutions. To further improve the initial solutions, SpecPart introduces a novel cut overlay method. It computes clusters by removing the hyperedges cut by any of the initial solutions. The resulting clustered hypergraph is smaller and often contains an improved solution that can be computed optimally using an Integer Linear Programming (ILP) formulation. Finally, SpecPart lifts the improved solutions to a coarsened hypergraph, where an ILP partitioning instance is solved to alleviate local stagnation. This step helps overcome the limitations of refinement heuristics getting trapped in local minima. The SpecPart algorithm has been validated on multiple benchmark sets, including the ISPD98 VLSI Circuit Benchmark Suite, Titan23 Suite, and Industrial Benchmark Suite. Experimental results demonstrate that SpecPart can substantially improve the cutsize by more than 50% compared to leading partitioners hMETIS and KaHyPar for some

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

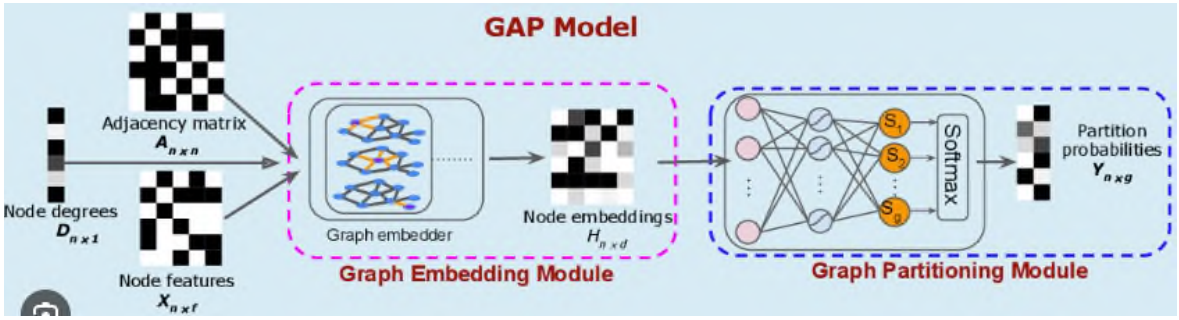


Figure 3.6: Generalizable Approximate graph Partitioning (GAP) [10].

benchmarks. The algorithm's performance is influenced by several parameters, including the number of eigenvectors, the number of trees, the number of best solutions, the number of iterations of ISSHP, the number of random cycles, and the threshold of the number of hyperedges. These parameters can be tuned using autotuning techniques to optimize the algorithm's performance.

Last but not least, we discuss a machine learning approach called Generalizable Approximate Graph Partitioning (GAP) [10], [11]. This algorithm is a deep learning framework designed to solve the problem of graph partitioning. Graph partitioning involves dividing the nodes of a graph into balanced partitions while minimizing the number of edges that are cut across the partitions. This is a combinatorial optimization problem that has been traditionally approached using heuristics and approximation algorithms. GAP takes a different approach by leveraging deep learning techniques. It consists of two main components: the graph representation learning module and the graph partitioning module. The graph representation learning module is responsible for generating node embeddings, which capture the structural information of the graph. These embeddings are then fed into the graph partitioning module, which assigns each node to a specific partition based on the learned representations. The key innovation of GAP lies in its ability to generalize to unseen graphs. Unlike traditional approaches that optimize the partitioning for each individual graph, GAP is trained on a set of graphs and can then be used to produce performant partitions on unseen graphs. This generalization is achieved by learning the representation of the graph while jointly optimizing for the partitioning loss function. This allows GAP to adapt to different graph structures and produce efficient partitions across a wide variety of graphs. To train the GAP model, a differentiable loss function is defined that represents the partitioning objective. This loss function uses a continuous relaxation of the normalized cut, which is a commonly used metric for evaluating

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

the quality of graph partitions. The network parameters are then optimized using backpropagation, allowing the model to learn to generate balanced partitions with minimum edge cut. In the experimental evaluation, GAP is compared against hMETIS, a widely used graph partitioning algorithm. The performance of GAP is evaluated on both real and synthetic graphs, including widely used machine learning models, scale-free graphs, and random graphs. The results show that GAP achieves competitive partitions while being up to 100 times faster than hMETIS. Furthermore, GAP demonstrates its ability to generalize to unseen graphs, producing partitions with low edge cut and high balancedness.

All the presented tools have the V-cycle flow in common, either as the main flow, following its steps one by one, or as a general approach employing only the ideas of coarsening and refinement steps as presented in GAP and KaHyPar. This persistence of the literature on this flow led us to decide to utilise this flow on our tool also. The second worth mentioning point of this presentation is that none of these frameworks utilise any other circuit characteristics to produce the results apart from the circuit graph connectivity. This is a major issue considering that the partitioning phase is at the early stages of the ASIC flow, because the following steps will use a suboptimal result regarding timing, power and other related constraints. To address this issue, our approach, by taking as input the industrial PDKs formats, encompass all the gate characteristics during the initial separation and optimisation process of the clustering and partitioning phase.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Chapter 4

Our Contribution

4.1 Introduction

In previous chapter was thoroughly analysed the importance of Multi-Level flow. However, it is rather obvious that most of the existing algorithms and methodologies are outdated or insufficient to address the complex challenges posed by the ever-evolving semiconductor landscape. The statistics and Moore's Law predictions suggest that sooner than later the Multi-Level flow will be integrated in standard ASIC design flow as it will become mandatory. This highly possible outcome led us to research this topic and come up with a portfolio of solution in areas of interest aligned with our expertise. Thus, based on the literature and on the industry feedback, we observed that one of the most complicated and crucial steps was the initial partitioning of the chip. Given that this step affects the placement phase, which affects all the other afterwards, it is of paramount importance to produce high quality results.

The current chapter presents a cutting edge partitioning tool able to tackle advanced modern semiconductor challenges following the clasic V-cycle flow. Starting the analysis, this section discusses two multilevel clustering methods developed and optimised by our team and also exhibits various optimisation steps which are implemented or will be in the foreseeable future, as they would have significant impact in the QOR. The next part of this chapter introduces the core multilevel partitioning approach, combining both a recursive bipartitioning and a kway partitioning algorithm to extract area balanced groups of instances with minimum cutsize. This section is the most complex and yet important of this work, as it presents all the introduced execution time and quality results optimisations techniques developed for this thesis.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

4.2 Multi-Level Clustering Phase

Our algorithm, addressing the clustering phase, as every other related approach, is targeted to create fairly balanced, loosely connected groups of objects. Its key difference is that it utilises circuit oriented metrics to assign a notion of criticality to its decisions supporting the rest of the ASIC flow, while, at the same time, is aware of the common practices often used before the engineer reach this step of the flow. Even though, the algorithm outline seems quite simple, there are many critical details to ensure high quality results in minimum execution time regardless of the design. Because of that, it is important to analyse the algorithm, targeting to highlight all these features, rendering it a superior alternative to the most of the current state-of-the-art tools.

Starting from the beginning our clustering algorithm operates at a gate-level netlist, coarsening it to a number of levels, where each clustering level, above the standard cell level, *i.e.* Level 1, contains a set of lower level clusters and unclustered standard cells. Each standard cell or object must be uniquely assigned to a cluster, at each clustering level, creating a hierarchy, which will be used by the multi-level partitioning algorithm described below. Our algorithm is targeted to create area balanced clusters both within the same level and across the levels of hierarchy. Balancing object areas, as much as possible, is also very important, as clusters become the new operational grain. Especially, in force-directed placement for example, as object area is typically a function of the spreading force [39], the more uneven the cluster areas, the more pronounced the spread forces between clusters, reducing sensitivity to cluster-to-cluster connectivity. The second objective of the algorithm is to group instances based on net fanout or timing metrics to assist the following operations of the ASIC flow, *i.e.* the placement and the routing phase. In modern EDA tools, the majority of the algorithms are targeted to reduce the timing violations and the high fanout nets of the design, as both of them jeopardise the performance and the power consumption of the chip respectively. Last but not least, through the clustering phase, the reduction of the adjacency matrix density is required, which indicates the reduction of inter-cluster connections. We could consider each external cluster connection as a dependency, preventing the exploitation of the divide-and-conquer nature of the algorithm and as a result reducing its effectiveness on algorithms such as extraction and signoff timing phases. These steps require each cluster to be as much as possible isolated from the rest of the circuit, as every outgoing connection introduces notable error in their analysis.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

4.2.1 Top Level Algorithm Presentation

To further understand the innovation of this approach, it is wise to analyse the steps of the algorithm to highlight the previous theoretical quality targets through a thorough inspection of them. The toplevel algorithm outline of the algorithm can be found below in this section, as long as the core algorithm growing the clusters.

Initiating the review of the algorithm, clusters are created by assigning standard cells to them, based on a single seed net, which has a notion of criticality, introduced by the previous referred quality metrics, with nets sorted by that critical parameter as shown in the first line of the algorithm. This will typically be fanout, with increasing or decreasing order, however it may also be a timing aware metric as slack or delay. Then, at lines 3-9, and with the initial level set to 1, the function `grow_mlclusters()` is called, which corresponds to the core clustering algorithm, presented in Algorithm 4. The conditional at line 10 checks the clustering termination condition, *i.e.* the current level objects, against the FO parameter. If cluster creation is saturated for the current level, line 8 exits the loop. Then, at line 11, a post-clustering flattening is performed to abolish inferior quality clusters by flattening them at their level.

Algorithm 3 Clustering Algorithm Top-Level

Input: Netlist ($Standard\ Cells$, $Nets$), Sorting Order (O), Final Objects Number (FO), Level Reduction (LR), Upper Area Bound Ratio (UBR), Level Upper Area Bound Ratio ($LUBR$), Minumum Clusters per Level (MCL).

Output: Set of Clusters per Level, up to a computed Maximum Level, satisfying input parameters.

1: SN = `sort_nets`($Nets$, O); // sort nets based on specified order O //
2: l = 1; // level 0 is standard cell level //
3: **repeat**
4: // grow current level clusters //
5: |*clusters(l)*| = `grow_mlclusters`($Netlist$, l);
6: $l = l + 1$;
7: **if** (|clusters(l)| $< MCL$) **then**
8: **break;**
// cluster creation saturated at current level //
9: **end if**
10: **until** (`|objects(`l`)|` $< FO$); // clustering exit condition //
11: `flatten_mlclusters`(*clusters per l*, $maxlevel$);
// post clustering Flattening step to guarantee MNM //

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

4.2.2 Algorithm Parameters Presentation

Before we continue further on the core algorithm, it is wise to take a step back and review the parameters of the algorithm, as they significantly affect the operation of the algorithm and its quality of results. The algorithm has nine tuning parameters responsible to determine the methods which will be used as long as the permissible limits.

Initiating the parameters review, we could not start by the most important of them, called Level Upper Bound Ratio $LUBR$. This variable determines the maximum allowed ratio between the smallest and largest area cluster of the current level. During the operation of the algorithm, this parameter is altered based on the ratio achieved on the previous level and the number of the current level. This modification allows the algorithm to start grouping the objects with more flexibility during the first levels, while as the clusters become bigger and bigger this flexibility must be reduced to avoid the grouping of large heavily connected objects which results in unbalanced clusters.

Parameter Name	**Interpretation**
Level Upper Bound Ratio ($LUBR$)	$\frac{maxareacluster(N)}{minareacluster(N)}$
Upper Bound Ratio (UBR)	$\frac{maxareacluster(N)}{maxareacluster(N-1)}$
Area Bound Type (ABT)	Method evaluating the balancing factor of the clusters
Level Reduction (LR) Ratio	$\frac{\#ofobjects(N)}{\#ofobjects(N-1)}$
Final Objects (FO) Number	termination condition, when objects $\leq FO$
Nets Sorting Type (NST)	Nets sorting criticality type Fanout, Delay or Slack
Nets Sorting Order (NSO)	Nets sorting order Increasing or Decreasing
Minimum Clusters per Level (MCL)	alternative termination condition
Minimum Number of Members (MNM)	post clustering flattening constraint

Table 4.1: Algorithm main Parameters, where value N stands as hierarchy levels

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

The next one, is the Upper Bound Ratio UBR parameter, controlling the area growth of clusters through levels. This parameter is closely related with the continuous one, Area Bound Type ABT. In order to prevent the unpredictable growth of clusters during the hierarchy levels, the first parameter, determined by the user, enforces a specific area balancing factor between the current and previous level clusters. However, due to the fact that it is not that trivial to decide if the average or maximum area cluster of the previous level should be used, the second parameter introduces four metrics to evaluate the area balancing through levels. The first metric is called MAX and considers only the maximum area cluster of the previous level. The second one is called $MIN - MAX$ and creates a range of values based on the maximum area cluster of the previous level, in which the clusters of the current level must be included to be considered as valid. The third method is called $AVERAGE - AVERAGE$ and evaluates the maximum allowed area for the current level based on the average area cluster of the previous level and the average area cluster of the current level until this stage while the final approach, called $AVERAGE - MAX$, utilises the average cluster of the previous level and the maximum area cluster of the current level so far.

Continuing to the next two parameters, named Level Reduction LR and Final Objects Number FO, it controls the amount of levels that will be produced during the clustering algorithm. The first one computes the expected reduction of clusters number in each level, while the second one sets the lower limit of level clusters. Thus, combining those two parameters, we could theoretically predict the produced levels of the algorithm. However, because of the post-processing algorithms performed in each level and various artefacts regarding the connectivity of the design and area balancing of previous levels, this prediction is not guaranteed. Despite that, these parameters work as soft restrictions for the algorithm to produce a high-quality result, creating minimum levels of hierarchy.

The following two parameters adjust the criticality factor assigned to every net of the design, based on which the clustering algorithm will assess and perform the grouping of the gates. Starting by the criterion type, which either will be physical aware `i.e.` net Fanout degree or timing aware `i.e.` gatepin delay or slack, the user can select the mode by assigning the respective value to the NST variable. On top of that, the ordering of the nets will significantly affect the outcome of the algorithm as in some case the criterion should be used as pulling force while in others as pushing force. The user is able to switch between these modes by changing the value of the NSO variable.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Last but not least, the final parameters tune the post-processing algorithm referred as flattening step. As briefly mentioned before, this step aims to demolish small clusters which would jeopardise the divide-and-conquer nature of the algorithm. To determine the amount of objects consisting a small cluster, the MCL sets the lower limit of objects per cluster, while to prevent the annihilation of all level clusters and the MNM value sets the minimum allowed clusters number per level. Also, the last variable is used as early exit condition in case that the algorithm do not succeed to create enough clusters at the current level.

4.2.3 Core Algorithm Presentation

In an abstract perspective, the core clustering algorithm consists of two phases, Phase I is seed creation, while Phase II is clusters fill-in. Phase I completes when the level reduction ratio is satisfied by the number of generated seed clusters or there are no available seed nets remain. The following step will then grow the formed clusters until there are no more gates to group or the area balance constraints disallow any further moves. A thorough pseudocode of the core algorithm is presented in Algorithm 4.

The main loop, lines 4-32, of Algorithm 4 grows clusters one sorted net at a time, to control the area bounds and ensure that the cluster area is balanced as much as possible. If in Phase I, the loop selects j as the current seed net, line 4, in Phase II, lines 5-7, j is the next seed net. Function `get_net_candidate_object`, line 8, identifies a candidate standard cell or object to group with the current net. This corresponds to the lowest area, unclustered standard cell or object of the current net, again to ease the area balancing. If the net is covered, this indicates that the entire net fanout has been clustered, so no candidate has been found. Thus, the inflation of this cluster will stop and the algorithm will continue with the rest of the seed nets.

Upon an unsuccessful candidate, the next attempt will take place when the net is revisited by the for loop, as $netcovered$ will be 1, line 9. If a candidate has been identified, then function `check_area_and_insert` is called, line 12. If the area bounds are violated, by adding the standard cell or object to the cluster, the latter returns a $result$ of 0. In Phase I, the loop will then consider the next net. The algorithm will move to Phase II when the condition of line 29 is satisfied. Phase II exits when all nets set fail to add further standard cells or objects to any cluster, line 23, and the clusters created at the current level are returned, line 24. Note that a successful clustering step resets the failed nets counter, line 17.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Algorithm 4 Multi-level Clustering Core Algorithm, *i.e.* `grow_mlclusters()` function of Algorithm 3.

Input: Netlist ($StandardCells, Nets$), Sorted Nets (SN), Level_Reduction (LR), Upper Area Bound Ratio (UBR), Level Upper Area Bound Ratio ($LUBR$), Minimum Clusters per Level (MCL), Current Level (l)

Output: Set of Clusters at current Level, ensuring area balance between them and aiming for their number to be $\geq (\frac{1}{LR})\times$ previous level Clusters.

1: // $clusterednets$ = list of nets corresponding to current level clusters //
2: // $mlclusternets[j]$ = additional nets associated to cluster of seed net j, related to cluster cell net contents //
3: $phase$ = 1;
4: **for** (j in Sorted Nets SN) **do**
5: **if** ($phase == 2$) **then**
6: j = *next clustered net in clusterednets*; // clusters fill-in phase //
7: **end if**
8: *(netcovered, candidate)* = `get_net_candidate_object`(j, l);
9: **if** ($netcovered == 1$) **then**
10: `continue`
11: **else**
12: // check area bounds, and if satisfied, insert candidate into mlcluster of net j //
result =
`check_area_and_insert`(j, *candidate,* UBR, $LUBR$, l);
// result indicates whether area bounds are satisfied //
13: **if** ($result == 1$) **then**
14: **if** ($phase == 1$) **then**
15: $clusterednets = clusterednets \cup j$;
16: **else if** ($phase == 2$) **then**
17: $failednets = 0$;
// reset failed nets count upon successful clustering //
18: **end if**
19: **else if** ($result == 0$) **then**
20: // clustering candidate of net j failed //
21: **if** ($phase == 2$) **then**
22: $failednets = failednets + 1$;
23: **if** ($failednets == |clusterednets|$) **then**
24: **return clusters;**
// all candidate nets failed; clustering Phase II ends //
25: **end if**
26: **end if**
27: **end if**
28: **end if**
29: **if** ($|objects(l)| \geq (\frac{|objects(l-1)|}{LR})$) **then**
30: $phase$ = 2; // move from phase 1 to phase 2, i.e. fill-in phase //
31: **end if**
32: **end for**

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

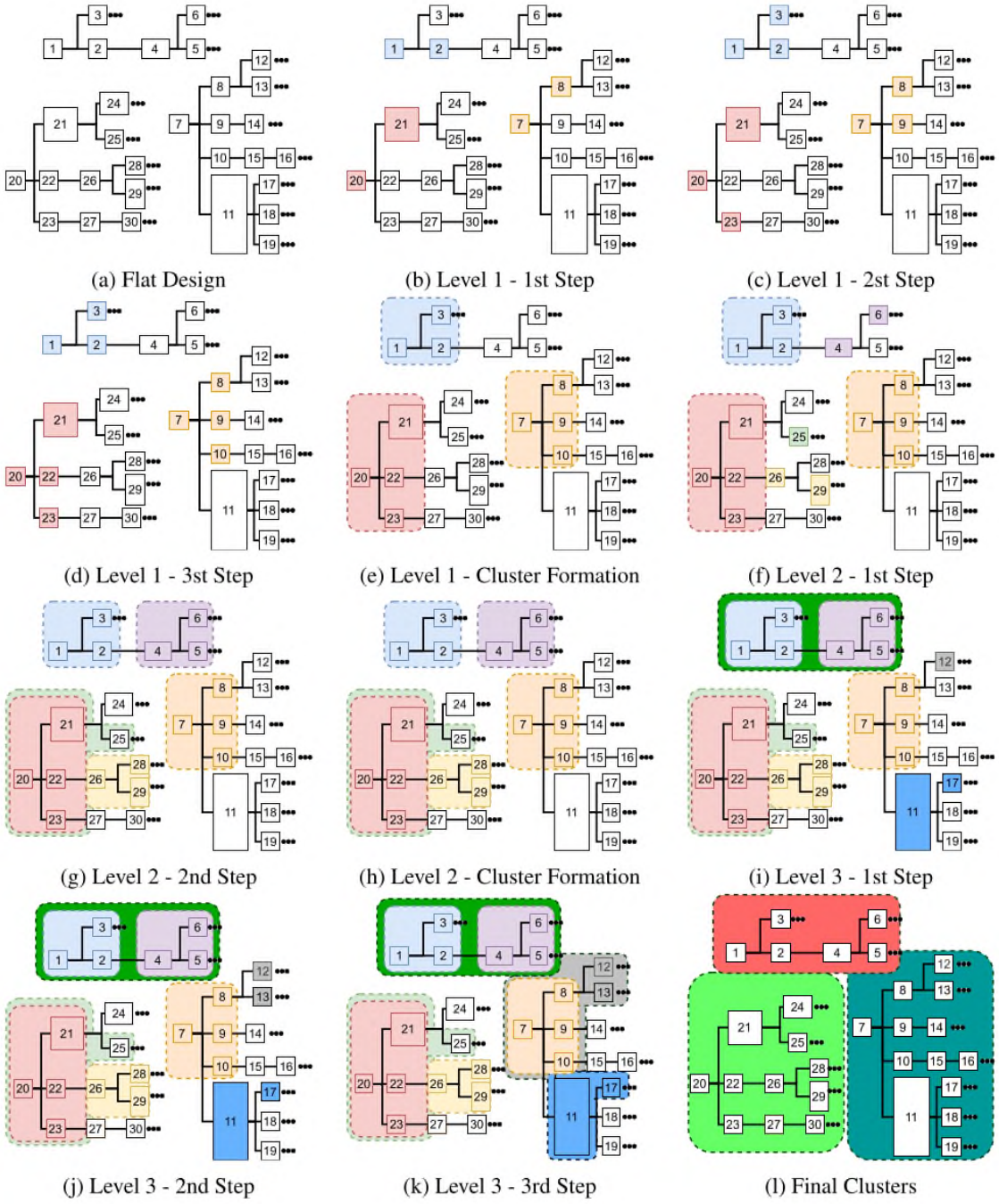


Figure 4.1: Simplified Multi-Level Clustering algorithm operation overview step-by-step [36].

An additional complication is that at every level of clustering, clusters of previous levels will exist. Thus, the total number of objects at level L, line 29 of Algorithm 4, and line 9 of Algorithm 3, include (i) clusters of level L, (ii) any other clusters of any level i, which have no parents up to level L, and (iii) any unclustered standard cells at level L. This set represents the `Total Number of Objects` at Level L. Similarly, at the last level of

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

clustering, FO represents the desired Total Number of Objects at the last level. Note that using the Total Number of Objects at Level L is preferable to considering solely the clusters of Level L, as the former provides a more complete picture of the complexity, However, as mentioned above, the FO criterion may indeed not be satisfied. This may occur if no more clusters may be grown past a certain level by Algorithm 4. As the key difference between clustering and partitioning, the first one is a bottom-up algorithm, which makes it harder to predict and satisfy the exact number of resultant objects at the Top-Level. A brief overview of the algorithm operation is presented below in Figure 4.1.

4.2.4 Post-processing algorithm

As mentioned before, to eliminate poor-quality clusters a post-processing approach has been introduced which either destroys the problematic cluster and releases its children into the level or emerges its children with its parent objects. The algorithm is performed right after the creation of the clusters' hierarchy, `i.e.` line 10 in Algorithm 3. The algorithm completes in three phases, namely Top-down, Bottom-up and Clean up. All of them have the same end goal, approaching it from different perspectives in order to resolve all possible corner cases.

Top-down flattening operates from the last to the first clustering level, identifying clusters which do not satisfy the MNM parameter. If this is the case, members of the cluster may be collapsed, by 1 level of clustering, to reveal their children. This will increase the original cluster's members. Instead of arbitrary collapsing members of the MNM violating cluster, we sort its members by area, and identify its minimum area child. This is done to ensure that we identify the minimum area solution while satisfying the MNM goal. Unfortunately, this phase is not sufficient to satisfy the MNM goal. This is because a cluster may not have any cluster members, but solely standard cells. This necessitates the usage of bottom-up flattening as well. Bottom-up flattening works in the opposite way, from the first to the last clustering level. If a cluster's members are less than MNM, and, considering its parent cluster, the following holds $|MNM - childmembers| > |MNM - (parentmembers + childmembers - 1)|$, then the cluster is collapsed.

Last but not least, a Clean-up step is also performed during Bottom-Up clustering, which collapses any leftover clusters with few members. Any cluster with members less than 20% of MNM will be collapsed. An example as to why Clean-up is necessary is the following. If a cluster was to end up with 40 single standard cell clusters, for an MNM of 20, only

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

20 of the 40 would be collapsed, ending up with 20 standard cells and 20 single standard cell clusters. The Clean-up step resolves such corner cases, but it can lead to unclustered standard cells or more clusters than had been requested. The unclustered standard cells or the greater number of final clusters can be filtered at a post-clustering step, such as partitioning. This work focuses only on the clustering step, while the experiments gives an insight on clustering QoR, that impact on partitioning and placement steps.

4.2.5 ”2nd” Version of the Algorithm

Even though, the described algorithm and the post-processing method appear sophisticated and well-designed, the findings were unexpected. The main problem was that the algorithm used to create substantially more levels than the expected, while the grouping at each level was of poor-quality, as very few objects were grouped together, leaving a significant amount of standard cells completely unclustered at each level. This feature could compromise the entire ML ASIC flow, as the objects' number at the final level was not notably reduced as it should be. To overcome this obstacle, we came up with a second version of the algorithm which is perfectly aligned with the basic pillars of the first one, introducing some new features. Due to the fact that this project was truly extensive, minor details of the second version should be omitted to keep the length of this report reasonable.

The basic new feature could be located in line 10 of Algorithm 4. As it can be recalled, the previous algorithm skipped the covered seed nets, while in this version, it looks for new candidates in the connections of the seed net fanout, giving a sense of depth first search in the algorithm. The idea behind this approach was that the previous version used to come to an abrupt halt during the growing of the clusters as the seed nets were overlapping. Thus, there were two available options to tackle this issue. The first one was to spread the seed nets towards the logic levels of the circuit, and the second to prevent the precipitously exiting by inserting more candidates to process. The first idea is substantial more complex to be implemented, as there are numerous heuristics and assumptions that should be included. Still, it is one of our future goals to include it into our algorithm, as it could yield better quality results. The second strategy, on the other hand, has been implemented, showing substantial gains in results. Delving into the details of the new feature, the same metrics and methods are introduced for the additional gate candidates in order to be sorted and used accordingly. The user has the ability to select different order and criticality metric to assign into the initial

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

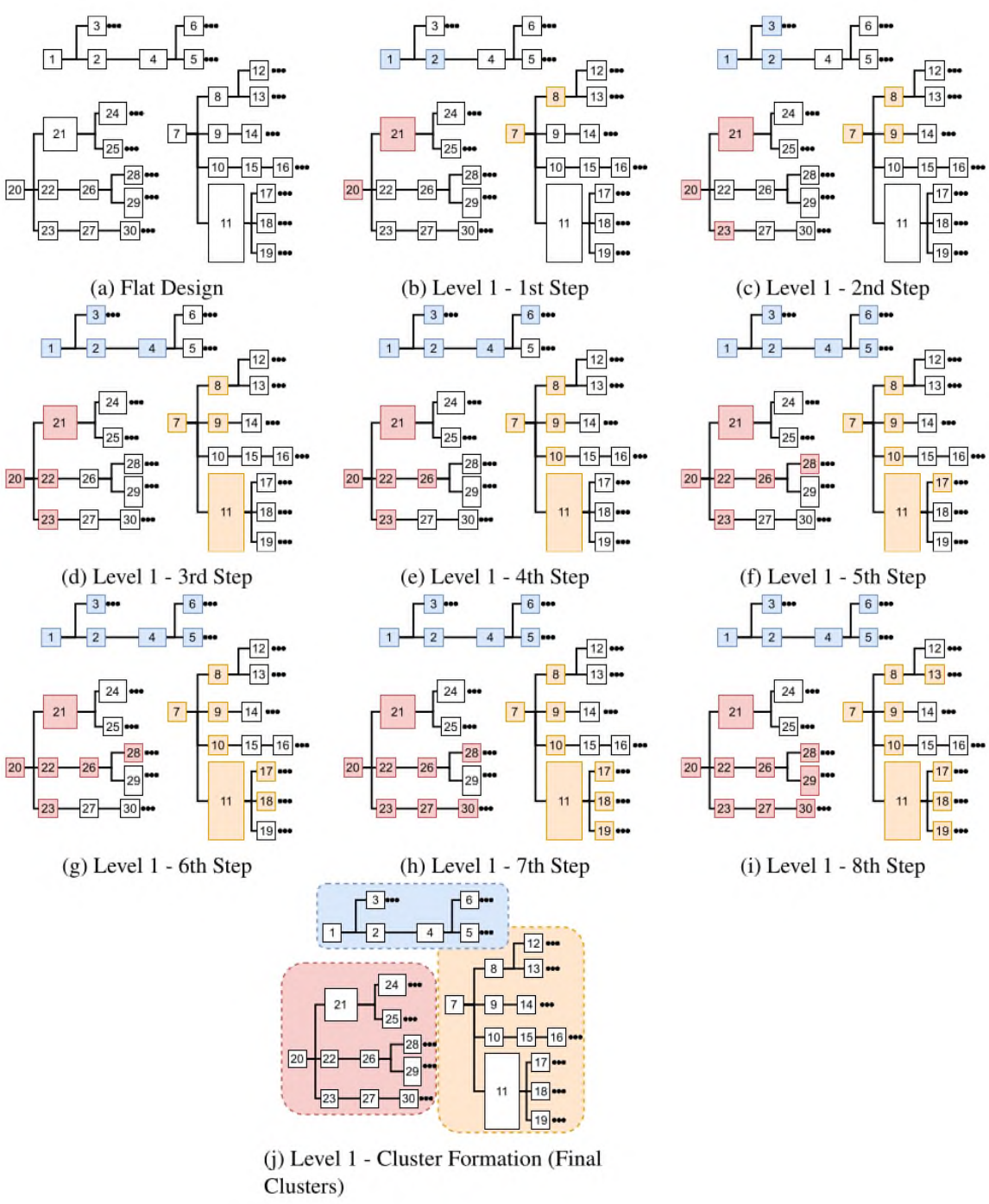


Figure 4.2: Second version of Multi-Level Clustering algorithm flow overview [36].

and supplementary candidates. This detail is critical, as this way the grouping of the circuit instances obtains a perception of weights adjusted to the circuit level. This is why the algorithm produces groups suitable to be used in timing driven oriented operations, `i.e.` the critical delay paths distribution remains low, while at the same time the fanout distribution of the critical nets follows the same trend. The flow presented in Figure 4.2 stands as a proof of concept of the newly introduced feature, showing better results, considering the number

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Design	Levels	Level Area Ratio	Through Levels Area Ratio	Unclusterd Components
Industrial 1	14	104.49	1.59	2011
Industrial 2	18	802.89	1.56	246
vga_lcd 2	18	802.89	1.56	246
b19	57	62.20	1.11	1
jpeg	26	1870.80	1.38	0
leon3	20	25.94	1.22	2
netcard	25	27.78	1.27	4250

Table 4.2: First algorithm version Clustering QORs results using the open-source designs.

of levels as long as the required number of steps to complete.

The previous Table 4.2 and Table 4.3 proves the substantial improvement achieved by this small modification of the algorithm as regarding the required levels number, the level area ratio balance of the clusters and the final level unclustered components number. These results highlight the notable reduction in these metrics without worsening the fourth metric, considering the area ratio of the average cluster across two consecutive levels. This could be translated as that the clusters are able to grow larger inside the level absorbing more objects while at the same time they respect the area balance constraints regarding the inner level and through hierarchy. The execution time overhead and the memory consumption was purposely excluded from the results, as both versions complete their operations in a few seconds in all considered benchmarks.

4.2.6 Macro aware Clustering technique

In order to facilitate a generic algorithm able to work with all kinds of circuits, it is not possible to exclude the handling of circuit macros. Macros are usually large objects specified as standard cells, such as SRAMs or other pre-characterised subcircuits. The problem with macros during the clustering phase is that, because of their vast area and high number of connections, they tend to have high criticality and gain, but bringing together such an object, even with the smallest object, may cause irreparable area imbalance to the clusters. In such circumstances, the pull force towards other objects may be so strong that this group could

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Design	Levels	Level Area Ratio	Through Levels Area Ratio	Unclusterd Components
Industrial 1	9	10.43	1.01	1253
Industrial 2	8	10.13	1.18	231
vga_lcd	8	10.13	1.18	231
b19	5	8.16	1.28	0
jpeg	5	10.72	1.27	0
leon3	5	15.09	1.33	0
netcard	25	6.67	0.97	1

Table 4.3: Second algorithm version Clustering QORs results using the open-source designs.

continue to absorb standard cells through levels even though it violates the area balance criteria. A set of groups including such a case can not be used afterwards into a partitioning or placement algorithm, as both of them utilise objects area into their cost functions.

Algorithm 5 Objects Areas Outliers Detection Algorithm

Input: List of objects areas sorted in ascending order $sorted\_objects\_list[]$

Output: Lower area bound $lower\_area\_bound$, Upper area bound $upper\_area\_bound$.

```
1: lower_quartile_area = sorted_objects_list[0.25 * size]
2: upper_quartile_area = sorted_objects_list[0.75 * size]
3: IQR = upper_quartile - lower_quartile
4: lower_area_bound = lower_quartile_area - [1.5 * IQR]
5: upper_area_bound = upper_quartile_area + [1.5 * IQR]
```

To avoid such cases, our algorithm encompasses a method to detect and exclude such objects which their area exceeds a dynamic upper bound limit. It is important to clarify that our approach do not detect macros as they are a specific type of objects, instead it generally detects large objects. However, most of the time they include the majority of macros. To detect these objects, our algorithm exploits the concept of **statistical outliers**, translating them into an upper and lower dynamic area bound limit respectively. In our case, only the upper limit is useful because it can broadly classify the objects areas into regular and large. By the end of objects separation, performed by the Algorithm 5, the algorithm marks the violating objects in order to be excluded from all operations in the next level run. Thus, it is ensured that an object large enough, such as a RAM, will not be grouped with any other

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

object during this level. However, it would be a mistake to permanently exclude an object from this phase, but considering that the clusters are growing through levels, eventually the outliers will be vanished, and all the objects of the level will be considered to be grouped, maintaining the area balance.

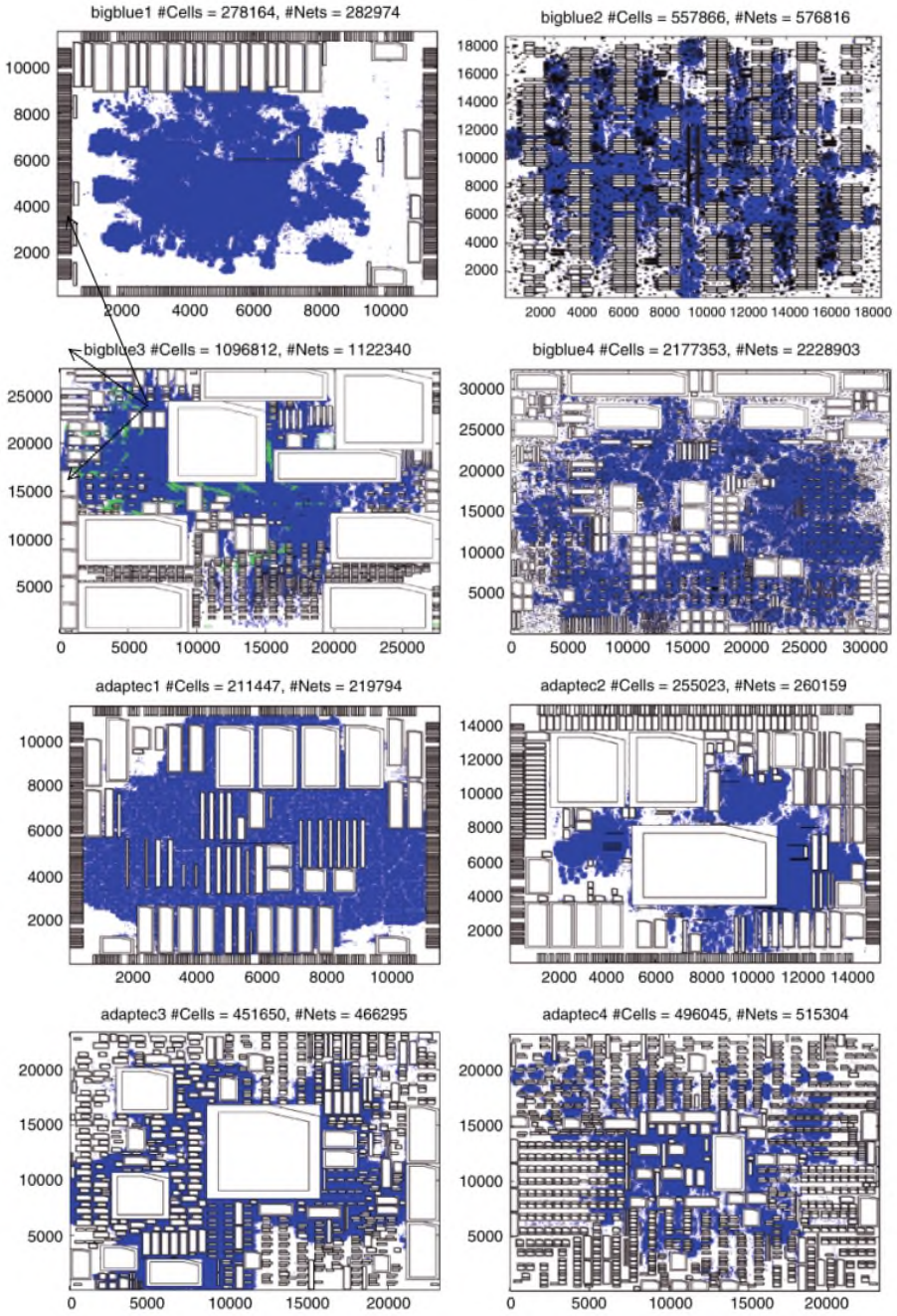


Figure 4.3: Present the placement result of four of the under review benchmarks containing large objects [12].

Using a small but representative set of benchmarks, we evaluated this method, regarding the novel clustering metrics, against the second version of the algorithm. The Table 4.4

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

presents the obtained results. To further support the intuitive understanding of this topic, we exhibit in Figure 4.3 the placement of some of the benchmarks to understand the difference between the included large objects and the standard cells of the designs. The presented results show an important reduction of level area ratio in some cases, such as case four and five, while at the same time the execution time is reduced. On the other hand, we can observe cases such as the first, the second, the sixth and the seventh where the area ratio remained the same or slightly increased. However, this behaviour is observed on designs with relatively small number of instances, rendering the method suboptimal for these kinds of benchmarks.

It is important to mention at this point that the achieved area balance is not suitable for almost any kind of applications. Yet, due to the fact that the clusters of the final level, across benchmarks, are a few tens of thousands in number the final partitions most are able to respect the user defined constraints or at least produce a result close enough to the requested.

4.3 Multi-Level Partitioning

Continuing towards the main contribution of this thesis, the heuristics and methodologies considering the Multi-Level partitioning phase of the ASIC flow are presented in this section. The top-level partitioning technique is explained briefly in this section, followed by a set of optimizations that improve the quality and increase the operational spectrum of both the novel FM and the Vcycle flow. The input of this methodology is the hierarchy of levels created by the clustering phase, as long as the physical and timing characteristics of the circuit components. The output contains a set of partitions, including only gate-level instances.

The objective of this algorithm is to create as many partitions as the user requested, respecting as much as possible the predetermined area balance in reasonable execution time utilising minimum memory resources. This challenge was particularly difficult as the core algorithm of our partitioning approach is the Fiduccia–Mattheyses (FM), which as described before is an exhaustive method having significant performance issues whenever it is used for large scale circuits. Despite all that, we managed to achieve our goal, introducing a set of heuristics which will be thoroughly analysed in the following sections.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Design	Novel Algoirthm			
	Levels	Unclustered	AreaRatio	Exec Time
adaptec1	9	664	93.44	35.695
adaptec3	16	2012	403.30	74.35
adaptec4	16	3228	458.44	105.663
adaptec5	18	3975	1643.95	271.471
bigblue1	10	1060	216.59	45.695
newblue1	8	2201	146.79	40.95
newblue3	15	9837	306.20	65.92
newblue6	18	4845	834.34	412.61
Design	**Large Objects Awareness**			
	Levels	Unclustered	AreaRatio	Exec Time
adaptec1	9	863	94.28	44.916
adaptec3	14	3325	457.09	84.59
adaptec4	15	4896	311.63	72.10
adaptec5	17	5751	1351.18	180.74
bigblue1	10	1426	157.02	34.99
newblue1	8	2260	145.14	30.383
newblue3	14	10997	265.42	62.231
newblue6	17	6953	601.53	330.33

Table 4.4: The first part of the table present the novel algorithm version Clustering QORs results. The second part present the large objects aware algorithm version Clustering QORs results. Both parts use the same designs with macros.

4.3.1 Top-Level Partitioning Algorithm

Proceeding to present our partitioning approach, it is wise to present the outline of our proposed partitioning algorithm, Algorithm 6, as it includes the FM in various steps of the process. Our partitioning approach consists of two phases. Much like the clustering methodology, the first phase creates the required partitions while the second refines them, producing the final result. As noted in previous sections, the key difference with the clustering is that the number of partitions is non-negotiable and must be achieved. To do that, the initial

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

partitioning phase utilises the recursive bipartitioning and the second the kway-partitioning approaches respectively, as they described in the respective sections accordingly.

The first method is used to optimally separate the circuit into partitions, avoiding the random distribution of the cells. This way, the exact number of partitions will be created unless the objects are not enough. In previous section was mentioned that a binary tree is created and at each tree-node an FM algorithm is performed to assign objects into the children nodes until the leaf nodes are reached. The main drawback of this initiative is that if the bisection at each node is performed producing perfectly balanced partitions, the final result will be imbalanced, unless if the requested number of groups is a power of two. To avoid this problematic situation, have to implement a routine to assign partitioning area ratios to each intermediate node of the binary tree, which the bisection algorithm must respect.

Our routine ensures perfectly area balanced leaf nodes. This challenge is addressed by computing the required percentage of the circuit which must be included in each partition, and assigning it into every leaf node. Afterwards, the two children partitions will compute the ratio of their assigned percentage as:

$$\frac{LHS\_percentage}{RHS\_percentage}$$

and the produced number will be assigned into the parent node. This procedure will continue until the root node is reached. Then, starting from the root node, the bisection algorithm will start to operate, assigning the produced objects lists to each children node respectively. In this routine there are many corner cases which are covered in our implementation, but it will not be mentioned in this report owing to space restrictions.

It is obvious that after the assignment of the ratios the forward traversal which recursively bisects the nodes is 100% vectorised, which means that it is completely parallelisable. We also implement a multithread version of recursive bipartitioning methodology, yielding notable execution time improvement. By the end of this procedure, a post-processing area balancing algorithm is performed if needed, else the next phase begins. This optimisation method will be comprehensively described in the following sections, as it is used in more than one time during the entire partitioning phase.

The next phase, the so-called `refinement` phase, utilises the kway-partitioning approach, as many groups are created and have to be optimised concurrently. As highlighted before, the novel FM is designed to bisect the circuit instead of splitting it into multiple groups, thus the initial FM had to be extended in order to address this challenge also. As was

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Algorithm 6 Top-Level Partitioning Algoirthm

Input: Netlist ($Standard\ Cells$, $Nets$), Clusters Hierarchy, Partitions Number PN, Area Balance Factor ABF, Gain Type GT, Post Processing Optimisation Phase $PPOP$, FM type FMT, Level Unfolding Strategy LUF.

Output: Set of Partitions, satisfying input parameters.

```
1: Hierarchy_Level = get_mlclusters_maxlevel();
2: initial_phase = 1;
3: repeat
4:    if (initial_phase == 1) then
5:       objects = get_level_objects(Hierarchy_Level);
6:       initialise_recursive_bipartitioning_binary_tree(Netlist, PN);
7:       status = recursive_bipartitioning_MT(objects, ABF, GT, FMT);
8:       if (status == -1) then
9:          /* error status, the algorithm must exit */
10:         break;
11:      else if (status == -2) then
12:         /* warning status, not enough objects, to fill in all partitions, in this level */
13:         Hierarchy_Level = Hierarchy_Level - 1;
14:         continue;
15:      else
16:         Hierarchy_Level = Hierarchy_Level - 1;
17:         initial_phase = 0
18:      end if
19:      optimise_area();
20:   end if
21:   unfold_partitions_level(Hierarchy_Level, Clusters Hierarchy, LUF);
22:   objects = get_level_objects(Hierarchy_Level);
23:   if (check_level_constraints() == TRUE) then
24:      status = kway_partitioning(objects, ABF, GT, FMT)
25:      if (status == -1) then
26:         /* error status, the algorithm must exit */
27:         break;
28:      else
29:         optimise_area();
30:         Hierarchy_Level = Hierarchy_Level - 1;
31:      end if
32:   end if
33: until (Hierarchy_Level == 0)
34: optimise_cutsize(PPOP)
```

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Algorithm 7 Recursive Bipartitioning Binary Tree Initialisation

Input: Netlist ($Standard\ Cells$, $Nets$), Partitions Number PN.

Output: Set of Clusters per Level, up to a computed Maximum Level, satisfying input parameters.

```
1: tree_nodes_number = (2 * PN) - 1;
2: tree_nodes = initialise_tree(tree_nodes_number);
3: for each node in tree_nodes backwards do
4:     if node == TRUE then
5:         /* compute circuit percentage which must be included in this partition */
6:         /* this value will be used only to compute the parent note bisection area ratio */
7:         area_percentage = 100 / PN;
8:         assign_area_percentage(node, area_percentage)
9:     else
10:        /* compute ratio based on children nodes area percentage */
11:        ratio = node.LHS_child_area_percentage / node.RHS_child_area_percentage;
12:        assign_ratio(node, ratio)
13:        /* compute node area percentage as the combination of its children */
14:        area_percentage = node.LHS_child_area_percentage + node.RHS_child_area_percentage;
15:        assign_area_percentage(node, area_percentage)
16:    end if
17: end for
```

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

presented, the FM utilises two sorted heaps, one for each partition, to obtain the gains of the objects towards this partition. To extend its operation to multiple partitions, we modified it to maintain as many sorted heaps as the number of partitions storing the gains of the object.

It is not that far-fetched, that this heuristic will explode the memory consumption, creating a relatively low upper bound on the number of partitions and instances that this approach can handle within reasonable time and resources. That prospect disproves our initial claims of efficiency in large scale circuits, leaving us with no other option rather than to address this issue. This way, we came up with four policies which trim the heap lengths into a manageable size to ensure the quality and effectiveness of our method. These polices are completely tunable by the user, but also can be combined into a framework to automatically trim the heap lengths based on the initial size and the contents of each one of them specifically.

Apart from the cutsize optimisation step included in the second phase, it must gradually unfold the inserted hierarchy of clusters, mapping the assigned objects of previous level partitions into the next one. Even though this stage seems as a straight-forward technique, it holds as a key aspect of the over all performance, as it depends vastly on the provided clustering hierarchy, affecting the number of instances which will be taken under consideration by the optimisation step. Thus, keeping in mind the necessity of efficiency in our work, we devised five partition unfolding algorithms to better automatically adapt to the specified clustering hierarchy.

The `V-cycle` approach predicts that the kway algorithm is performed at each level after the mapping and the unfolding of current level objects into the next one. However, to further reduce the computational costs, we alter the novel `Vcycle` flow by skipping or repeating levels with certain characteristics. Thus, unnecessary initialisations are avoided on levels with few dozens of objects and on the other hand levels overloading by objects are treated accordingly to avoid endless runs compromising the efficiency of the algorithm. All the briefly presented optimisation features of this section are assiduously discussed in the following sections.

4.3.2 FM algorithm optimisations

Until now, we have mentioned several times that we altered the novel FM algorithm to better match the requirements of our goal. Before we further dive deeper into case specific optimisations, we believe that it is the best opportunity to present the overview of our mod-

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

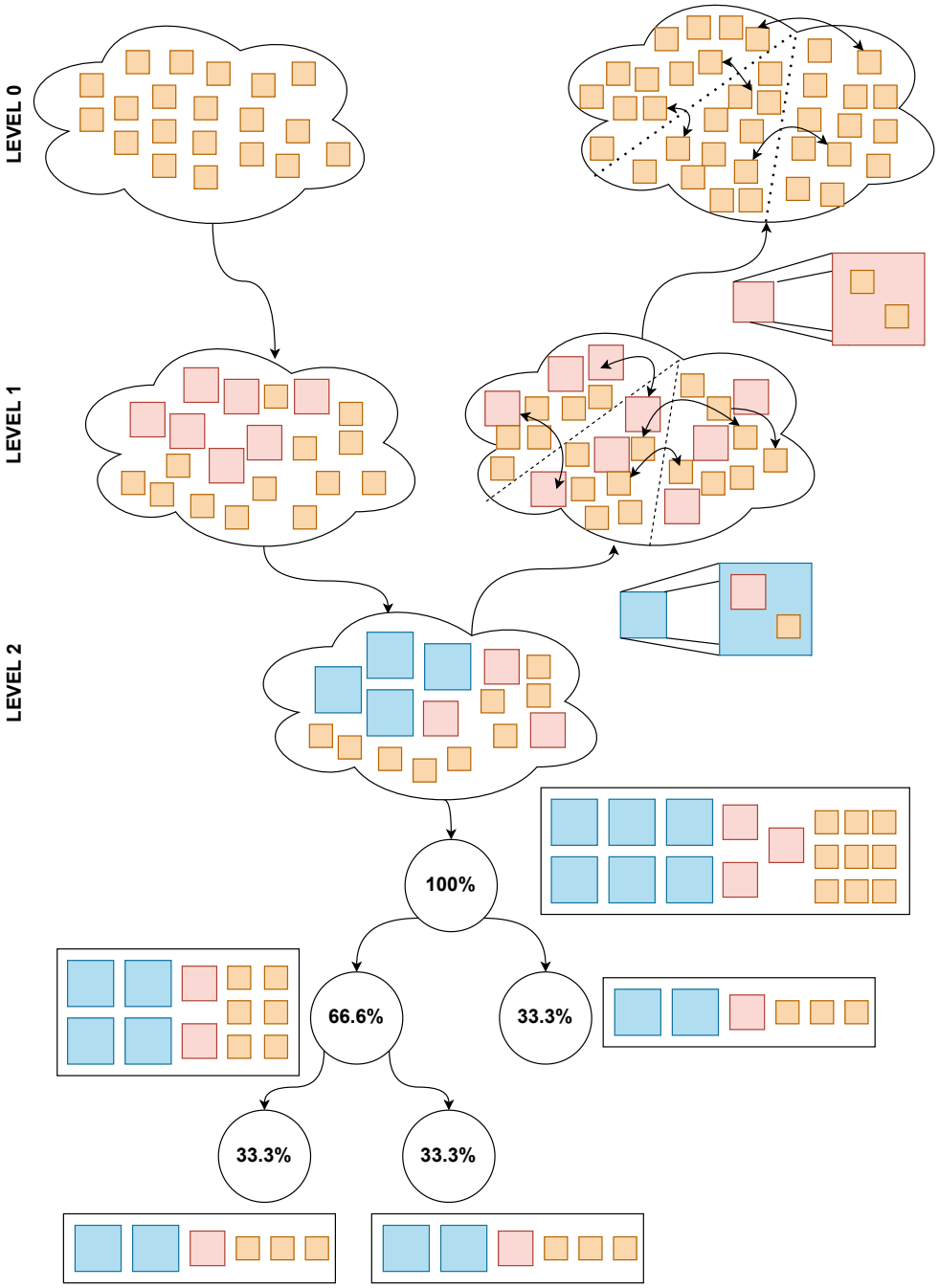


Figure 4.4: Complete V-Cycle flow followed in order to extract K-Way partitions

ified FM algorithm. In previous section was mentioned that the algorithm consists of one loop iterating through all objects in sorted order, locking their positions to eventually find the minimum cutsize value. Our proposal, as presented in Algorithm 8, utilises two nested loops iterating through all objects in sorted order, temporarily locking them to detect a local minimum of cutsize at each inner loop. This method yields a significantly better outcome,

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

since the testing order of the objects has a big impact on the final results.

Algorithm 8 Proposed FM algorithm

Input: Level Objects, Partitions Number PN.

Output: Objects list representing partitions.

```
1: initialise_heaps(PN);
2: extract_partition_cutsize_and_gains_mlobjects();
3: repeat
4:    sort_heaps();
5:    store_partitions_characteristics(&gain, &arearatio, &object);
6:    repeat
7:       pop_larger_gain_heap_node();
8:       get_object_characteristics(&area, &destination_partition);
9:       if (check_object_movement_for_area_violations(arearatio) == TRUE) then
10:         store_violating_object(object);
11:         continue;
12:      else
13:         pop_all_other_identical_instances(object);
14:      end if
15:      reinsert_all_violating_objects_into_heaps();
16:      move_object(destination_partition);
17:      update_partitions_characteristics();
18:      store_movement_logistics();
19:   until (FM_tentative_iterations_evaluation() == TRUE)
20:   detect_minimum_cutsize();
21: until (FM_iterations_evaluation() == TRUE)
22: return(create_objects_lists());
```

Even though this algorithm covers only a few lines of code, each one of them hides underneath a sophisticated method to perform its task at the best performance. If we start from the beginning, the very first line of the algorithm holds the majority of the execution time overhead. Like mentioned before, the data structures, storing all this information, are sorted binary heaps to reduce the computational time required to preserve them sorted. A binary heap considered sorted when the following rule holds.

$$((heap\_node[i] > heap\_node[2 * i])AND(heap\_node[i] > heap\_node[(2 * i) + 1]))$$

Thus, the removal of the larger heap node or the insertion of a new one requires maximum $\log nodes$ base 2 operations to resort the heap. This way, a significant amount of execution

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

time was saved, as this operation is performed every time a new FM iteration begins, without affecting the final product quality.

The second line will be skipped for now and will be discussed in the next section. Right after that, the enclosed loop begins to pop one by one the nodes of the heaps to test their tentative movements. Each insertion of the heaps include an object, a destination partition and a gain of the object towards this partition. The algorithm has to evaluate if the area balance ratio that will occur in case that the object move into that partition violates the specified bounds. If the object movement is marked as invalid, then the object has to be stored with the rest of the area violating objects in order to be reinserted into the heaps later in the process. In other case, regarding if the algorithm works in a bisection or kway mode has to pop out all the other identical nodes, specifically the other possible movements of this object, and then reinsert all the previously declared violating objects into the heaps. The intuition behind this decision is the after this movement, the area ratio might change just enough in order for a previous invalid verdict to change into an acceptable action.

The following three lines are the core procedure of this algorithm, transferring the objects between partitions and logging and updating their movement statistics. The details of these procedures will remain hidden for this report, to preserve a reasonable length. However, the outline is that each movement modifies the cutsize and area ratio of the partitions and based on this information the evaluation checkpoints in lines nineteen and twenty-one respectively determine if the algorithm will continue or not. Also, based on the logging of the movements, the algorithm in line twenty locates the minimum achieved cutsize so far and reverts the state of the partitions into that log state. The next FM iteration will start from that state, considering it as the initial distribution of the objects into the partitions.

To fully understand the proposed algorithm, it is mandatory to analyse the exit conditions previously referred, as they notably can alter its operation. The first algoirthm, named `FM_tentative_iterations_evaluation()` determines if the tentative moves should stop or not. The has three modes based on which decides the exit condition of the inner loop. The first and more exhaustive mode signals the loop to exit whenever the heaps are empty, while the second raises the exit flag whenever a negative value of gain is reached. This is a greedy approach and because the negative gain value means that the current move will probably increase the partitions cutsize we terminate the inner loop to start over using positive values. The last of the three methods is called `Early Exit` FM and for each movement

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

compute the slew between the minimum cutsize so far and the current value to decide if the exit condition is met or not. The boundary value, above which the exit flag is raise, is determined by the user. The next checkpoint controls the outer loop of the algoirithm and is called `FM_iterations_evaluation()`. This one also integrates the previous three modes but in a different way. The exhaustive mode allows the achieved cutsize value to be lower or equal to the previous iteration result, while the greedy mode breaks the loop if the current value is lower than the previous. On the other hand, the third one, `Early Exit` mode, exits if the reduction of cutsize value is not greater than a specified boundary chosen by the user. All of these heuristics were discovered through extensive experiments based on modern industrial and academic designs.

The charts in Figure 4.5 present an industrial circuit behaviour using the exhaustive methods. The conclusions that can be extracted by the first chart are that the progression of the cutsize through the tentative moves presents hills and sinks as the novel FM algorithm predicts, making necessary to maintain the hill climbing nature of the algorithm. However, on the bigger picture, it is quite recognisable from the created V-Cycle that the hill climbing tolerance effort must be tolerated for a narrow range of values before the algorithm give up. From the second chart, we can observe that the after the fifth FM iteration, the minimum cutsize has been very slowly reduced. On top of that, after the first iteration the reduction of cutsize is performed on the first few tentative moves while the rest of them perform only negative moves. This information in addition to the previous conclusions extracted from the first chart renders the third approach as the most efficient considering the performance-quality trade off. The second one, addresses cases where a quick and dirty result is enough, while the exhaustive approach, scenarios that even the slightest reduction in cutsize value is helpful.

4.3.3 Gain Value Calculation

During the previous section, the second line of the Algorithm 8 was skipped, as it is an excellent opportunity to define the gain value and to analyse the ways we compute it in our methodology. Thus, it would be a same if this analysis was underestimated by the other equally important information provided in the previous section. So to start, the gain value could be compared with the pulling force applied into an object by other partitions to move it from its current one. The most well-known approach is to assign a value, as gain, proportional to the connectivity of the object with the rest of the partitions. In the background chapter, we

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

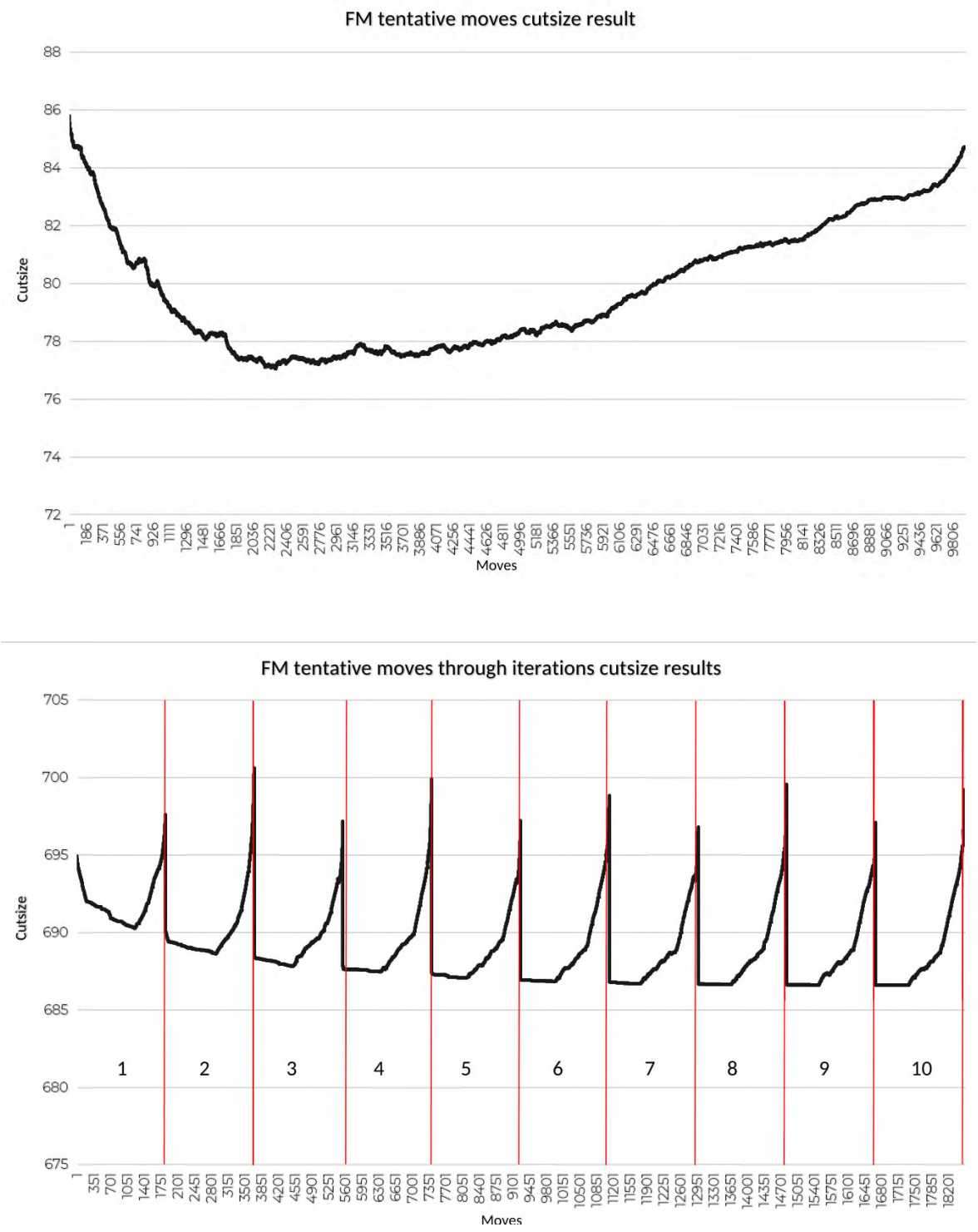


Figure 4.5: The top side chart presents the progression of cutsize with respect to the tentative moves, while the bottom side chart presents the progression of cutsize with respect to the tentative mooves collectively with all FM iterations.

mentioned two types of circuit connectivity representations, the first one was the directed graph and the second was the hypergraph. In our approach, both of these representations are

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

used where they best correspond to the current phase of the algorithm.

Using the directed graph representation, the gain of an object can be computed as the number of inter-partition connections of the object minus the number of intra-partition connections. In a more simplified manner, $External - Internal$ connections number. The first approach does not integrate any circuit oriented characteristic, while the other one, which is a bit more complicated, considers the gain of an object based on its nets. In this way, an object increases its gain by 1 towards a partition if it is the only object of the net laying in the current partition and a portion or the entire net exists on the other partition in which the gain is referred to. In continuation of this, the object reduces it gain by 1 in case that the entire net is located into one partition and preserves the gain intact in case that the net is evenly distributed into partitions. Due to the simplicity of the first approach, the computation of the value and update of data structures are of very low-cost while the second one because of the nets' consideration require significantly more processing power and memory consumption to evaluate the gain and update the respective structures.

Even if the second approach seems more well-suited for our goal, the first one yields the best results. If we consider the complexity of nets into a densely connected design, it is most probably that initially each partition will maintain a portion of many nets and only a few will be completely grouped together or at only one object will be excluded. This can cause degenerate cases where the heaps contain almost exclusively zero gains nodes, restraining the algorithm to perform really movements without significantly improving the cutsize but instead spending execution time and memory resources. On the other hand, the first representation at the early levels of the algorithm creates a better assignment of gains, performing many more high-gain movements and as a result notably improving the cutsize. However, as the levels of the algorithm proceed this representation correlates, and we can observe a significant reduction into the quality and amount of objects swaps. In this situation, the nets based approach will be performed, because the majority of the nets are either totally grouped or have missed one or two objects, and as more detailed and circuit oriented to approach will refine all these spots and will provide and high quality result.

The Table 4.5 presents the results of eight benchmarks evaluating the gain value calculation methods introduced before. The results are obvious rendering the per_flyline method as the best choice for the coarsening phase of the algorithm. Starting by the first case where we can observe 100% increment in execution time overhead and almost 50% increment in

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Design	per_flylines			per_nets		
	Exec Time	Cutsize	Area Ratio	Exec Time	Cutsize	Area Ratio
Industrial 1	17.405	22023	3.498	36.835	33570	3.500
Industrial 2	25.282	58243	3.500	475.696	65878	3.500
b19	31.015	38188	3.500	327.222	65027	3.500
jpeg	43.348	73463	3.500	500.872	95612	3.500
leon3	1050.561	138738	3.500	5023.114	203939	3.500
netcard	2168.491	288981	3.500	-	-	-
adaptec1	60.979	65678	3.500	-	-	-
adaptec2	835.725	83585	12.475	-	-	-

Table 4.5: Presents the evaluation of gain value calculation strategies as regarding the standard partitioning metrics.

cutsize value towards the largest ones where the execution time increment reaches 500% increment in execution time, it is self-explanatory that the second method is suitable for local optimisation steps. The dashes in the lower right part of the table indicate the high execution time of these cases, which led us to skip the completion of these experiments.

4.3.4 Heap Strategies

In previous paragraphs, it was mentioned many times the concern regarding the memory consumption and execution time overhead required to store and maintain the modified FM binary heaps. Of course, this issue could not be overlooked, and thus we propose four strategies able to address this issue without sacrificing much of the partitioning QOR. All the methodologies are targeted to reduce the length of the heaps at each iteration of the algorithm as they are recreated every time, by avoiding inserting objects with small or negative gain. The methods are organised on an ascending scaled basis, from the most strict towards the exhaustive approach, gradually limiting the insertion of objects into them.

The first strategy. which is the strictest method effects both the heaps' length and the allowed number FM iterations, disabling the heel climbing nature of the algorithm. In detail, this strategy for each object detects the movement with the greater gain and allows only this one to be inserted into the heaps, while at the same time permits only one FM iteration. These characteristics render it the most time efficient method, having a small quality penalty. The second approach, aiming to balance the time quality trade off, enables the heel climbing

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Design	low effort			normal effort			high effort		
	Exec Time	Cutsize	Area Ratio	Exec Time	Cutsize	Area Ratio	Exec Time	Cutsize	Area Ratio
Industrial 1	13.882	33,802	1.762	16.128	21,538	3.500	16.353	21,534	3.499
Industrial 2	91.144	112,482	3.500	23.452	56,902	3.500	22.213	56,426	3.500
b19	277.179	94,149	1.158	29.934	38,188	3.500	30.308	38,297	3.500
jpeg	1,432.354	202,030	3.500	39.444	75,956	3.500	36.008	75,700	3.500
leon3	489.591	287,143	3.500	1,102.813	138,738	3.500	610.281	136,292	3.500
netcard	271.872	432,575	3.500	2,526.849	288,981	3.500	1,865.148	287,285	3.500
adaptec1	132.423	115,708	1.232	52.637	65,678	3.500	49.190	61,396	3.500
adaptec2	123.594	150,659	14.840	698.191	83,585	938.914	12.573	83,199	12.475

Table 4.6: Presents the evaluation of heap size strategies as regarding the standard partitioning metrics.

feature of the FM, while instead of allowing only one movement per object this approach inserts the top ten gain movements. The number ten is determined experimentally, aiming to replace it by an automated algorithm in the near future.

Moving on to the two last modes, which are the most detailed and exhaustive, the execution time increases dramatically while the cutsize outcome improves marginally. The third mode retains the same notions as the FM iterations, but loosens the object movement insertion limitations even further by allowing all movements towards partitions that comprise a portion of the object connections to be entered. As a result, the FM algorithm has greater flexibility to make a wrong decision regarding the gain value in order to improve the area ratio of the partitions. The last mode is the exhaustive mode, encompassing every movement, and is not advised for use except in circumstances when exceedingly thorough circuit separation is required regardless of execution time.

The Table 4.6 presents the results of heap size strategies. The numbers indicate the low effort method as the fastest for the majority of the cases, while on the other hand it produces the worst cutsize results. In general the results endorse the hypothesises based on which these methods were introduced. However, there are cases where these do not hold, such as in the fourth design, where the low effort method is substantially more time-consuming rather than the others. This could happen due to the inappropriate combination of the unfolding strategy, which will be explained below, and heap size strategy, leading to substantially increase the amount of objects required to be handled in the lower levels.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

4.3.5 Unfolding Strategies

Unfortunately, there are cases where even the strictest heap strategy can lead to be inserted millions of objects into the heaps, compromising the algorithm and reducing its effectiveness. An auxiliary measure to ensure the effective and gradually handling of the objects at each level is to exploit the imported clustering hierarchy characteristics. During the unfolding of the clusters from one level to another, their connections with other gates or smaller level clusters are already examined in the current level. This way, to further reduce the size of the heaps instead of inserting bidirectionally these connections, it is preferable to insert only the one direction starting from the currently unfolded objects.

Also, in large circuits there are cases where objects and their connections are completely included into the same partition, and they have no interaction with the outer world. As a result, it is ensured that they will enter the heaps having a large negative gain value. These simple ideas inspired us to create five distinct techniques for managing the number of current level movable objects based on the outcome of the unfolding process and the contribution of object connections into cutsize.

For the better understanding of these techniques, they can be described by a Venn Diagram of two intersecting circles. The first mode, depicted by the first circle, allows only the current level unfolded objects to be deemed moveable and enter the heaps, preventing this way the reevaluation of the previous level objects contributing to the cutsize. The second approach, represented by the second circle, allows all the current level objects having inter-partition connections to enter the heaps, rendering a more detailed mode as the number of these objects will be considerably larger than the previous technique. The third scheme stands as the intersection of these modes, allowing only the current level unfolded objects which have cross partition connections to be considered as movable for this level. It is obvious that this is the most rigorous of all methods, significantly reducing the heaps entries. The next mode is the union of these sets approaching the fifth and final method, which is the universe of the Venn diagram, including all level objects.The last two methods are used in relatively smaller circuits to improve the quality of results, as the savings in objects with the first three methods are limited to a few hundred objects.

The previous Table 4.7 and Table 4.8 includes a set of results regarding the presented unfolding strategies. The blue coloured benchmarks include large objects, and they are the larger in terms of instances number. From this table, it is observed that for small designs such

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Design	On Cut			On Cut and Unfolded		
	Exec Time	Cutsize	Area Ratio	Exec Time	Cutsize	Area Ratio
Industrial 1	20.765	20,010	3.498	22.857	18,703	3.500
Industrial 2	40.791	52,015	3.500	47.230	50,277	3.500
b19	42.080	29,794	3.500	63.471	28,228	3.500
jpeg	66.545	54,194	3.500	111.184	44,983	3.500
leon3	2272.985	111,116	3.500	2305.853	101,585	3.500
netcard	6376.440	249,056	3.500	5059.119	249,376	3.500
adaptec1	-	-	-	969.061	40,116	3.500
adaptec2	-	-	-	971.795	59,365	13.335
adaptec3	-	-	-	2,346.700	74,249	10.668
adaptec4	-	-	-	1,902.323	43,881	3.500
adaptec5	-	-	-	9,776.844	129,713	5.887

Table 4.7: This table presents the results of two of the unfolding strategies for a set of benchmarks, which the one coloured blue include large objects while the other one not.

Design	Unfolded on Cut			Unfolded		
	Exec Time	Cutsize	Area Ratio	Exec Time	Cutsize	Area Ratio
Industrial 1	18.694	22,392	3.499	18.694	21,651	3.498
Industrial 2	34.676	58,396	3.500	32.603	57,662	3.500
b19	58.187	38,188	3.500	47.243	35,222	3.500
jpeg	39.958	74,017	3.500	65.469	50,127	3.500
leon3	916.663	138,738	3.500	397.786	125,847	3.500
netcard	1808.874	288,981	3.500	255.838	279,523	3.500
adaptec1	286.331	65,678	3.500	308.040	42,902	3.500
adaptec2	681.017	83,585	12.475	319.581	70,001	12.564
adaptec3	6,483.530	122,083	10.653	2393.446	94,107	10.599
adaptec4	95.865	93,081	3.500	360.759	52,157	3.500
adaptec5	29,994.876	211,733	5.903	10,735.438	203,455	5.951

Table 4.8: This table presents the results of two of the unfolding strategies for a set of benchmarks, which the one coloured blue include large objects while the other one not.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

as the first the different strategies have minor impact on the QORs while as the number of instances increase the effects are more profound. The next thing we have to mention is the partitions area ratio, which is stabilised near three point five, which were the user request, even for the larger objects with the macros in them. Continuing, we should justify the dashes in the upper part of the table, which are standing as no result due to large execution time overhead. In detail, the explanation is really simple and has to do with the fact that as the circuit size is increased, the number of objects having inter partition connection will increase. As a result, the number of objects that the core algorithm will have to handle will reach prohibiting values. However, in fourth case, we can observe a contradictory behaviour as the first strategy is more efficient than the others in terms of execution time. This is perfectly normal as the effectiveness of the strategies is not related only on the circuit size but also on the graph connectivity characteristics such as density.

4.3.6 Level Skipping and repeating flow

Until now, we discussed only point optimisations and methodologies regarding the FM algorithm or the construction of the necessary data structures. Only in the previous section, we briefly considered effects of the clusters' hierarchy into the partitioning level, introducing the unfolded objects. Still, even in that section, we took as granted that the clustering result is of high quality and is suitable to be used for our purpose. Unfortunately, this is not always the case, as during a clustering level the grouped number of objects varies from a few dozens of objects to a few millions of objects. This variation must be predicted from the partitioning methodology in order to appropriately adjust its internal algorithms.

Our approach, as mentioned in Algorithm 6 in line twenty-three, checks this corner case before the top level algorithm proceed to the refinement phase of the current level. In case that only a bunch of objects are marked as moveable, based on the unfolding techniques, for this level, the algorithm will skip it and will assign its objects into the next one. This way, a clustering hierarchy containing a lot of levels grouping only a small portion of objects in each one of them combined with an inappropriate selection of unfolding strategy, will not be a problem for the algorithm as it can quickly unfold enough levels as if they were one and proceed with the level refinement handling a sufficient amount of objects to notably improve the cutsize. Another advantage of using this heuristic is that the structures of the FM method that would have been constructed for each skipped level without improving the quality of

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

results are not initialized and destroyed, saving a large amount of execution time.

The second checkpoint ensures that the current level mapping did not mark as movable more objects than the algorithm can handle. However, if that is the case, our proposed algorithm has two alternatives to effectively reduce the number of objects in this level. The first one is to select a stricter unfolding strategy and reextract the movable objects of the level. This alternation of strategies has effect only on the current level, consuming only a small amount of execution time comparatively with the execution time that would have been spent by FM to perform all these movements. The second technique arises because the first quick hack might not solve the problem, and a more robust approach is required. the second method flow is called the W shape because in cases like that stops the refinement phase and re-enters the coarsening phase altering the sets of parameters to achieve better clustering results. Afterwards, it starts over from the initial phase to recreate the partitions. An intuitive model of this flow is presented in Figure 4.6 below.

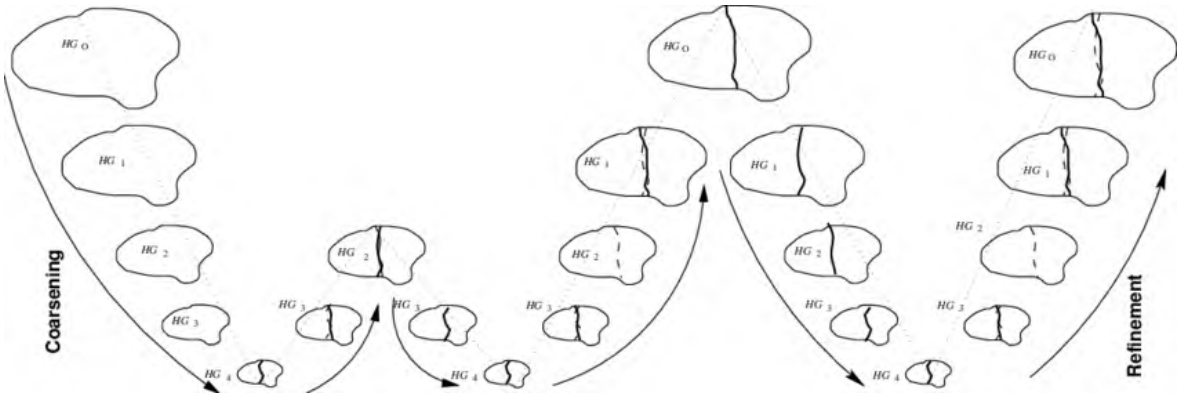


Figure 4.6: Presents the W shape flow alternative to the V shape flow which, in the situation of a poorly formed clustering level, reverts to the coarsening phase.. Following that, it comes back to the partitioning method from the beginning, reproducing the partitions. Depending on the clustering quality outcome, this back and forth might be repeated numerous times.

4.3.7 3D ASIC Flow Extention

The final part of our contribution includes the modifications which had to be performed in our methodologies [70], [71] to support multiple technologies oriented tier assignment algorithm. During this thesis we took part into the ICCAD 2023 3D macro aware placement contest in which we submitted a complete project. One of the contest main requirements was

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

to support placement with multiple technologies, one for each tier. Thus, the flow we followed was to place the cells into one tier, as if they were all assigned all into the bottom tier and then used our partitioning tool to bisect the circuit assigning the objects into the respective tier. There are two catches with this flow. The first one is that the contest requires specific utilisation percentage in each tier, and the second is that each object has different area in each technology. Thus, the arbitrary movement of objects into the upper tier is not an efficient option.

To tackle these obstacles, we developed an algorithm which detects and sorts the objects based on their ratio between the bottom and top technology in descending order. This way, in order to ensure that our result will not violate the tiers' utilization, we replaced the initial random partitions assignment algorithm. The new one assigns the objects with the maximum ratio into the top tier, which encompass the smaller nanonmeter technology, aiming to enclose as many objects as possible, while the remaining objects are assigned into the lower tier. Afterwards, we continue to the optimisation phase as described in previous section respecting the tiers area balance by assigning different area into the objects according to their tiers.

The Figure 4.7 presents a comparison of four of the well-known tools against our own in 3D design flow during the tier assignment phase. In this phase, the partitioning tools must separate the circuit into 2 partitions respecting the utilisation ratio requested by the user. The orange cells indicate that the tool failed to respect the required utilisation.

	Tier Utilisation Ratio					MAX Allowed	Vias				
	hmetis	kahypar	kahypar_MT	Patoh	Ours	Vias	hmetis	kahypar	kahypar_MT	Patoh	Ours
case1	2.716231665	1.465517241	1.14999925	1.241387872	0.873139541	9	4	5	4	3	3
case2	2.35501105	2.006218414	2.401602078	2.106656963	0.94563904	2091	1118	538	1145	503	538
case2_h	1.032087654	0.983663751	1.179962219	1.034515632	1.000988993	2091	1093	538	1145	503	682
case_3	1.235168936	0.942022745	1.792628071	1.03799689	0.998553331	36864	18819	7615	14959	6902	3899
case_3_h	1.223910041	1.470954185	2.782987179	1.611737486	1.139133124	36481	18956	7615	14959	6902	24592
case_4	1.688974648	0.089350216	1.635174746	26.57263262	1.033511799	184470	74042	2508	74034	43185	68769
case_4_h	1.166310336	0.070904978	21.08088554	1.297723869	1.100349014	179776	74814	2508	43185	74034	96191
case1	0.803425839	1.375	1.037796731	1.576630828	1.085020663	12	4	2	4	2	4
case2	1.843017443	0.941774585	14.9754448	1.035880504	1.004517148	12875	8940	2742	8940	2879	3562
case2_h	3.766070777	1.93531307	30.62251384	1.470805975	0.838924595	25432	9026	2574	8940	234	119
case3	2.394224022	2.152451593	2.01538816	1.361105769	9.95039E-05	133570	82316	84889	81560	82921.66667	13773

Figure 4.7: Comparative results of four partitioning tools against ours in 3D designs.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Chapter 5

Comparative Results

5.1 Introduction

In previous section, we thoroughly discussed our proposed algorithm regarding Multi-Level Partitioning phase, claiming that the heuristics and methodologies presented yield significantly better results than the other well-known approaches. To back up our statements, in this section we present a comprehensive set of experiments addressing each one of these optimisation steps presented both in clustering and in partitioning sections respectively. Also, to prove that our complete tool stands as an excellent complete alternative to the other established tools, we exhibit a comprehensive set of experiments proving the superiority of our methodology in the partitioning oriented metrics. Furthermore, as stated numerous times throughout the thesis, the partitions generated by this tool must be of sufficient quality to be utilized in the subsequent ASIC flow steps. This way we also present a set of experiments including 3D placement utilising partitions generated from all the previous compared tools.

The outline of this chapter continues as follows. The next section presents thoroughly the experimental methodology followed to compare our tool against the other well-established tools mentioned before, describing the framework in which the tools were had to be integrated and the tools parameters values investigated during the experimental phase. Following that, the points of the comparison are analysed for both the partitioning oriented part and the 3D part of the analysis. Also, includes both an overview and the analytical tables of the experiments conducted during the comparison with the other tools using the designs with the large objects in them. Last but not least, the final section includes the 3D application results regarding the scores achieved against the contest upper limits.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

5.2 Experimental Methodology

5.2.1 Experimental framework

In order to test all these algorithms and features, we had to integrate them into a greater framework supporting additional features necessary for the extraction and the analysis of the results. This framework is a proprietary suite of tools addressing the entire novel ASIC flow and some of its extensions, necessary for our work. One of the most useful feature of this tool for our project is its ability to parse and store efficiently industrial format files such as LIB, LEF, DEF and netlist files. For this reason, all the C/C++ language code development and the other tools' evaluation took place inside this framework, translating the imported circuit information into the appropriate format each time for each tool.

Another important feature of this tool is the integrated static timing analysis engine, which was used to evaluate the effectiveness of our algorithm for timing driven operation. Also, the integrated placement algorithm extended to support Multi-Level and 3D flow was used to measure the results related to half perimeter wire-length and design density. Beyond these standalone tools, it contains a set of auxiliary features such as python and TCL command line interface as long as a sophisticated Graphic User Interphase (GUI) combined with data analysis features such as histograms and scatter plots which significantly assist the analysis of the partitioning results.

Apart from the framework tool in which we developed the approach, a comprehensive set of designs must be utilised to thoroughly test the proposed algorithm and heuristics. Because of the method's size and complexity, a design suit large enough to cover as many scenarios as feasible must be formed in order to obtain a fair assessment of the algorithm against well-established tools. This way, we gathered almost seventy designs mainly academic to evaluate the various parts of the algorithm against other approaches and assess its effectiveness in following ASIC flow steps. Namely, the suit includes the cases from the following EDA contests DAC 2012[12], ISDP 2005/6[13], ISPD 2011[14] and ICCAD 2015[15], 2 industrial designs and 5 open source large scale designs namely b19 [17], Leon3mp [17], Netcard [17] and jpeg_ecoder [17]. All these designs accompanied by their characteristics, including their components, nets, macros and IOs numbers as long as their use case in the current analysis, are presented in Figure 5.1.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

5.2.2 Evaluation Metrics and Tools

The most significant step in evaluating an algorithm is determining the quality measures that will be used to evaluate the approach, as well as the other tools that will be utilized as reference points. In our case, the metrics which will be used are the following. As regarding the clustering phase, the number of hierarchy levels, the area ratio at each level and the area ratio through levels will be used as physical design quality metrics. Also, to check the timing driven operations eligibility of the clusters hierarchy, we will measure the delay and slack distribution of the top thousand critical nets. In order to be suitable for timing driven operations, the clusters of each level should prevent the creation of `snake-paths` entering and exiting multiple clusters. A clustering result separating the critical paths into multiple groups is not suitable for timing driven operations, as in case that the clusters are distributed into multiple servers the timing annotations of these paths will introduce significant error, making the associated operation, such as routing, to perform incorrect optimisations. Considering the partitioning phase, we will measure the cutsize, the partitions area balance and the skipped or repeated levels of each design. Of course, the timing driven metrics will also be evaluated for the partitioning result.

However, these measurements, are useless unless the reference points are not established. To do this, we employed four well-known external tools, namely the hMETIS [2] [3], KaHyPar[4] [5] [6] and PaToH [7], using their results as reference points. The other three tools were not tested as the MLpart is already thoroughly evaluated with all the other existing tools, the SpecPart is a post-processing optimisation tool already tested with hMetis and KaHyPar [8], [9] and lastly we could not set up and run the GAP framework due to lack of computational resources. These tools were run as stand-alone programs within the general framework, importing the circuit information in the format that they required. The retrieved output product was parsed from the wrapper framework in order to initialize the corresponding partitioning structures. All the tools used for this evaluation have a set of parameters that could be explored in order to detect the best case scenario for each one of them. however, this analysis would take substantially more time to complete, risking considering our algorithm outdated, as hundreds of millions of tests would be required for each tool. Instead of that, we used their proposed parameters as they presented them in their respective introduction papers, and so we did for our algorithm aiming into a fair comparison of the results.

The benchmarks are exhibited below, in Figure 5.1, accompanied by their important in-

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

formation. The information included in the tables is the number of the gate instances included in the circuit, the number of the macros included in the design obtained by the respective contest benchmark suite representation paper, the number of the IO ports of the circuit and the number of the nets. Last but not least, the tables include the PDK name used to for the respective benchmark. Most of them use the ASAP7 7nm PDK [72] while the others use the NANDGATE 90nm PDK. Both of them are open-source, and the reason we chose them was to be easy for everyone to reproduce and cross-check our results. It is important to highlight the range of experiments used to evaluate the algorithm. In our suite, we have cases from a few thousand gates all the way to a few million gates. Also, we used both cases with and without macros, as long as benchmarks both densely and loosely connected. This way, we tried to test all types of designs to safely conclude into the best choice as regarding the partitioning tools.

	Name	Components #	Macros #	IO Pins #	Nets #	Library
ISPD06	adaptec1	5.72E+05	542	1	583328	ASU7
	adaptec2	4.57E+05	543	1	469444	
	adaptec3	9.69E+05	723	1	992555	
	adaptec4	1.09E+06	1329	1	1125036	
	adaptec5	2.15E+06	646	1	2183992	
	bigblue1	5.98E+05	559	1	606381	
	bigblue2	8.30E+05	3313	1	882507	
	bigblue3	1.65E+06	675	1	1694238	
	newblue1	4.73E+05	390	1	486413	
	newblue2	6.61E+05	1171	1	711078	
	newblue3	8.32E+05	690	1	923452	
	newblue4	1.47E+06	569	1	1506429	
	newblue5	1.84E+06	1052	1	1927347	
	newblue6	2.71E+06	1376	1	2771776	
	newblue7	4.39E+06	6151	1	4624383	
ISPD11	superblue1	7.98E+05	432	6521	823024	ASU7
	superblue10	1.05E+06	1619	15141	1086013	
	superblue12	1.27E+06	89	1580	1293531	
	superblue15	1.07E+06	153	10556	1080519	
	superblue18	4.59E+05	207	3978	469076	
	superblue2	9.51E+05	654	8047	991109	
	superblue4	5.59E+05	306	6623	581127	
	superblue5	7.09E+05	784	4082	787292	

	Name	Components #	Macros #	IO Pins #	Nets #	Library
DAC12	superblue11	9.26E+05	1458	6872	959056	ASU7
	superblue12	1.27E+06	89	1580	1293531	
	superblue14	6.05E+05	340	5473	629772	
	superblue16	6.71E+05	419	4448	697660	
	superblue19	4.95E+05	286	3735	512053	
	superblue2	9.51E+05	654	8047	991109	
	superblue3	9.08E+05	575	6482	933398	
	superblue5	7.09E+05	784	4082	787292	
	superblue6	9.52E+05	565	5380	1006801	
	superblue7	1.32E+06	419	6499	1340566	
	superblue9	8.11E+05	272	4014	834024	
ICCCAD15	superblue1	1.21E+06	3787	3787	1215302	ICCAD Undesclosed LIB
	superblue10	1.88E+06	1696	1696	1897736	
	superblue16	9.82E+05	101	101	999559	
	superblue18	7.68E+05	653	653	771215	
	superblue3	1.21E+06	2074	2074	1224311	
	superblue4	7.96E+05	3471	3471	802245	
	superblue5	1.09E+06	1872	1872	1096924	
	superblue7	1.93E+06	4910	4910	1933334	
Open Source	Industrial_1	0.5E+05	0	2176	60883	IND
	Industrial_2	1.4E+05	0	1159	147960	
	b19	2.2E+05	0	47	225884	NANDGATE
	jpeg	6.7E+05	0	67	674353	
	leon3mp	6.5E+05	0	333	758278	
	netcard	9.6E+05	0	1846	1058447	

Figure 5.1: Benchmarks collections used for the evaluation of the algorithm features and the over all tool against other well-established tools.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

	DAC12				ICCAD 2015							
	50	100	300	500	50	100	300	500	50	100	300	500
	CutSize				CutSize				Top 1000 Fanout Distribution			
hMetis	3.95	3.23	3.07	3.21	4.79	3.28	2.35	2.25	1.03	0.77	0.63	0.70
Patoh	2.39	1.92	1.93	2.06	3.31	2.31	1.72	1.67	0.57	0.37	0.26	0.25
Kahypar	2.37	1.89	1.88	2.00	3.22	2.22	1.66	1.62	0.53	0.35	0.25	0.26
Kahypar_MT	3.70	2.79	2.55	2.63	4.73	3.23	2.30	2.20	1.06	0.72	0.57	0.59
	AreaRatio				AreaRatio				Top 1000 Delay Paths Distribution			
hMetis	9.40	14.45	10.53	1.78	5.72	8.07	9.75	4.63	7.58	4.78	4.59	4.26
Patoh	11065	6792	4467	1531	3396	22040	12409	10978	2.95	2.22	2.27	2.23
Kahypar	839	80	1300	533	19	370	8159	3942	3.02	2.07	2.09	2.05
Kahypar_MT	12.28	11.53	7.09	2.60	6.25	7.25	6.29	5.35	6.85	4.93	4.46	4.19
	Execution Time				Execution Time				Top 1000 Slack Paths Distribution			
hMetis	2.08	1.68	1.38	1.07	2.09	1.88	1.32	0.90	4.52	4.18	4.21	4.18
Patoh	0.90	1.31	0.88	0.80	1.02	1.15	0.81	0.54	2.97	3.11	3.62	3.77
Kahypar	2.48	3.53	4.52	5.34	4.52	8.25	21.15	28.76	3.02	3.05	3.31	3.34
Kahypar_MT	0.17	0.20	0.27	0.31	0.10	0.11	0.12	0.12	4.39	4.18	4.17	4.15

Table 5.1: ICCAD 2015 benchmarks results. The table includes the results of four different partitioning results, requesting 50, 100, 300 and 500 partitions each time, and the values represent the ratio of the other tools result over our proposed algoirthm.

5.3 Comparison Results

At this point, we have reached the core evaluation of our tool. In this section, we will present you the results against the four other tools mentioned before. In Table 5.1 we can observe the results of the ICCAD and DAC contests, while on the Table 5.2 we can observe the results of the ISPD contests. Due to the large amount of results, these tables contain a compressed form of the results, while the analytical tables can be found in the Appendices section. The tables presented in Appendices contain the ratio of the result produced by the tool specified in the header of the row over our tool result. The tables are populated by such a value for all benchmark, and each metric specified at the top of the columns for each one of the requested partitions number mentioned in the header of the columns.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

	ISPD06				ISPD11			
	50	100	300	500	50	100	300	500
	CutSize				CutSize			
hMetis	6.647	5.461	4.286	5.400	3.987	3.023	2.899	2.946
Patoh	5.773	4.427	3.505	4.428	2.378	1.794	1.867	1.956
Kahypar	5.790	4.401	3.486	4.404	2.381	1.749	1.795	1.881
Kahypar_MT	7.061	5.345	4.174	5.236	3.671	2.604	2.454	2.478
	AreaRatio				AreaRatio			
hMetis	3.26	6.24	10.14	3.69	8.37	12.82	11.33	4.38
Patoh	9.16	1530.30	1277.05	3261.35	4455.35	2157.42	420.45	174.37
Kahypar	11.12	76.64	187.23	268.06	1152.26	10.64	1916.99	673.49
Kahypar_MT	5.05	7.19	7.77	8.00	11.90	9.09	6.01	4.40
	Execution Time				Execution Time			
hMetis	3.987	3.363	2.463	1.506	2.397	2.157	1.504	1.158
Patoh	0.669	0.590	0.519	0.346	0.986	1.275	1.270	0.947
Kahypar	1.167	1.221	1.248	0.833	2.008	3.304	5.007	4.975
Kahypar_MT	0.237	0.266	0.297	0.241	0.213	0.255	0.348	0.351

Table 5.2: ISPD 2004/05/06/11 benchmarks results. The table includes the results of four different partitioning results, requesting 50, 100, 300 and 500 partitions each time, and the values represent the ratio of the other tools result over our proposed algoirthm.

The tables in this section contain the average value obtained by the analytical tables for each benchmark group, based on the contests they belong, for each metric and partitions number. For example, the first cell of the Table 5.1 reports the average cutsize ratio achieved between all DAC 2012 benchmarks against hMetis requesting 50 partitions. This result can be explained as that our algorithm achieves 3.95 times better cutsize on average for all DAC 2012 benchmarks against hMetis requesting 50 partitions.

Starting the comparison by the main metric which is cutsize we can observe that our approach produces substantially better results having a range of results starting by 1.62 all the way up to 5.4 times better average cutsize against all other tested tools. Continuing to the second metric, which is the area ratio, we can also detect that our approach outperforms all other tools with significant improvement. The extremely large numbers which can be found in this category inside the tables are cases where the other tools failed to create balanced partitions and the average value got skewed upwards, so they should be not taken under consideration

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

as absolute values, rather as indicators of incorrect result. However, regarding the third metric which is the runtime we can observe that KaHyPar MT and PaToH are significantly faster than our approach, approximating 0.8 and 0.2 times respectively.

Moving on, to the second set of metrics regarding timing characteristics, only the ICCAD contest benchmarks could be evaluated by our internal STA engine as the other academic designs had problems with dangling nets or undriven pins. Nevertheless, even from this small set of designs, we can extract some important indications about the suitability of the algorithm result for timing driven operations. For the first metric called fanout distribution, we evaluated the top thousand fanouts of the circuits. The results shown that our algorithm tend to break the large fanouts into many partitions, approximately five times over the other tools. On the other side, regarding the top delay ad slack paths distribution, the results prove that our algorithm separates three to seven times less the critical paths rather than the other frameworks. It is important to mention, that the number of separations for the first metric is less than 7, for the delay paths distribution metric is less than forty and for the slack path distribution metric less than thirteen.

Considering the placement application, the first metric indicates that the result is suitable as the partitions will be loosely connected, and the enclosed objects will be placed without being significantly affected by the other objects included in other partitions. That holds because the separation of the top fanout will reduce the cutsize, and as a results the partitions' connectivity, by separating of the forward logic cone into the first level which have the fewer connections compared with a deeper level. Furthermore, timing-wise, the second metric proves that our tool will produce partitions which are applicable to timing driven operations as the critical path will most likely be separated only a few times enabling the timing analysis on each one of the partitions without introducing as much error as the other approaches.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Chapter 6

Conclusions

6.1 Conclusions

Reaching the final sections of this thesis, it is important to summarise the obtained knowledge into a few paragraphs. First thing first, the Multi-Level flow will be necessary and thus established as the standard flow very soon. Because of that, the development of new partitioning and Multi-Level oriented tools in general is vital to meet the expectations of the rapidly-evolving semiconductor industry. However, these tools must be VLSI oriented in order to take under consideration the ASIC flow characteristics and produce high quality results.

Targeting more on the main topic of this thesis, there are many partitioning tools published in the literature, but yet there are unaddressed issues regarding modern applications such as 3D aware partitioning and Multi-Level placement aware partitioning. To address these issues is required great attention to detail and deep understanding of the VLSI theory to exploit every possible characteristic of the circuit, aiming to yield high quality results. These characteristics are physical-design oriented such as fanouts, gates areas and distances, timing analysis oriented such as paths distributions gates drives strengths and gate delays, as long as power aware such as gates switching activity and dynamic power consumption. Our proposed methodology, considering all these features, yield better results compared to the other tested approaches as regarding the ASIC flow application driven metrics presented in the previous sections. It is obvious that in order to exploit such characteristics, it is necessary to integrate the tool inside a closed loop optimisation framework with many other analyses tools such as power and timing analysers to extract these values and use them as quality metrics inside

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

an optimisation loop. Unfortunately, the necessary frameworks to support this attempt are limited, with the most known of them being the OpenRoad project [73]. Despite all that, I believe that the industry requirements will motivate more researchers to be involved with this never-ending research area.

6.2 Future Work

Having all these in our mind, we believe that our approach, even though it produces impressive results as regrading the novel partitioning metrics and the timing aware metrics it has the first weak point in the execution time and the second on the amount of parameters needed to be tuned for each design specifically to extract the optimal results. This way, the first and more important thing that we will address is the speed-up of the approach by further analysing step-by-step its sub-algorithms to detect and resolve its time consumption hotspots. The next major target will be to create an automated algorithm, deciding the values of the algorithm's tuning parameters at the beginning of the tool and during its operation if that is needed. Of course, by the description of the problem the first idea is to introduce a machine learning methodology which will analyse the design characteristics and the progress of the partitioning algorithm and based on those should modify the respective tool variables by a factor to produce the optimal result.

Also, three promising avenues for future research in the realm of ASIC design are the further exploration of 3D design flow, the investigation of partition-based Static Timing Analysis (STA) techniques, and the development of a distributed ASIC design flow. Further investigating 3D design flow entails adapting and optimizing current design methodologies for three-dimensional integration, considering emerging technologies like stacked memory, through-silicon vias and heterogenous chips. Enhancing partition-based STA involves breaking down complex designs into more manageable segments and developing efficient algorithms for the static timing analysis, removing the pessimism and error introduced by the critical path separation. The creation of a distributed ASIC design flow aims to facilitate collaborative work across dispersed agents, involving considerations such as data exchange, security, and the integration of cloud-based tools. These areas present opportunities for further advancing ASIC design in the face of evolving technologies and growing design complexities.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Bibliography

[1] C. J. Alpert, J.-H. Huang, and A. B. Kahng, "Multilevel circuit partitioning," in *Proceedings of the 34th annual Design Automation Conference*, pp. 530–533, 1997.

[2] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Application in vlsi domain," in *Proceedings of the 34th annual Design Automation Conference*, pp. 526–529, 1997.

[3] G. Karypis and V. Kumar, "A hypergraph partitioning package," *Army HPC Research Center, Department of Computer Science & Engineering, University of Minnesota*, 1998.

[4] L. Gottesbüren, M. Hamann, S. Schlag, and D. Wagner, "Advanced flow-based multilevel hypergraph partitioning," *arXiv preprint arXiv:2003.12110*, 2020.

[5] L. Gottesbüren, T. Heuer, and P. Sanders, "Parallel flow-based hypergraph partitioning," *arXiv preprint arXiv:2201.01556*, 2022.

[6] L. Gottesbüren, T. Heuer, P. Sanders, and S. Schlag, "Scalable shared-memory hypergraph partitioning□," in *2021 Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)*, pp. 16–30, SIAM, 2021.

[7] Ü. V. Çatalyürek and C. Aykanat, "Patoh (partitioning tool for hypergraphs)," in *Encyclopedia of parallel computing*, pp. 1479–1487, Springer, 2011.

[8] I. Bustany, A. B. Kahng, I. Koutis, B. Pramanik, and Z. Wang, "Specpart: A supervised spectral framework for hypergraph partitioning solution improvement," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, 2022.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

[9] I. Bustany, A. B. Kahng, I. Koutis, B. Pramanik, and Z. Wang, "K-specpart: A supervised spectral framework for multi-way hypergraph partitioning solution improvement," *arXiv preprint arXiv:2305.06167*, 2023.

[10] A. Nazi, W. Hang, A. Goldie, S. Ravi, and A. Mirhoseini, "Gap: Generalizable approximate graph partitioning framework," *arXiv preprint arXiv:1903.00614*, 2019.

[11] A. Gatti, Z. Hu, T. Smidt, E. G. Ng, and P. Ghysels, "Deep learning and spectral embedding for graph partitioning," in *Proceedings of the 2022 SIAM Conference on Parallel Processing for Scientific Computing*, pp. 25–36, SIAM, 2022.

[12] N. Viswanathan, C. Alpert, C. Sze, Z. Li, and Y. Wei, "The dac 2012 routability-driven placement contest and benchmark suite," in *Proceedings of the 49th Annual Design Automation Conference*, pp. 774–782, 2012.

[13] G.-J. Nam, C. J. Alpert, and P. G. Villarrubia, "Ispd 2005/2006 placement benchmarks," in *Modern Circuit Placement: Best Practices and Results*, pp. 3–12, Springer, 2007.

[14] N. Viswanathan, C. J. Alpert, C. Sze, Z. Li, G.-J. Nam, and J. A. Roy, "The ispd-2011 routability-driven placement contest and benchmark suite," in *Proceedings of the 2011 international symposium on Physical design*, pp. 141–146, 2011.

[15] M.-C. Kim, J. Hu, J. Li, and N. Viswanathan, "Iccad-2015 cad contest in incremental timing-driven placement and benchmark suite," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 921–926, IEEE, 2015.

[16] K.-S. Hu, I.-J. Lin, Y.-H. Huang, H.-Y. Chi, Y.-H. Wu, and C.-F. C. Shen, "2022 iccad cad contest problem b: 3d placement with d2d vertical connections," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–5, 2022.

[17] J. Jung, I. H.-R. Jiang, G.-J. Nam, V. N. Kravets, L. Behjat, and Y.-L. Li, "Opendesign flow database: The infrastructure for vlsi design and design automation research," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–6, IEEE, 2016.

[18] M. Anand, S. Ravi, K. Chouhan, and S. M. Ahmed, "Data self-healing technique using asic level security mechanisms,"

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

[19] D. S. ROY, “Placement and Routing for ASIC,” July 2020.

[20] “Ansys RedHawk-SC | SoC Power Integrity & Reliability Software.”

[21] “Intel will outpace Moore’s Law, CEO Pat Gelsinger says.”

[22] C. J, “A Semi-Persistent Clustering Technique for VLSI Circuit Placement - ppt download.”

[23] “10/25/ VLSI Physical Design Automation Prof. David Pan Office: ACES Lecture 3. Circuit Partitioning. - ppt download.”

[24] “Floorplan (microelectronics),” May 2023. Page Version ID: 1156157622.

[25] D. Medhat, “2.5/3D IC Reliability Verification Has Come A Long Way,” Aug. 2022.

[26] M. Xu, G. Gréwal, S. Areibi, C. Obimbo, and D. Banerji, “Near-linear wirelength estimation for fpga placement,” *Canadian Journal of Electrical and Computer Engineering*, vol. 34, no. 3, pp. 125–132, 2009.

[27] G. Jie and B. Jeremic, “Draft report on parallel, finite element method for inelastic problems,”

[28] D. Papa, N. Viswanathan, I. L. Markov, G. Nam, C. Sze, Z. Li, and C. Alpert, “Physical synthesis with clock-network optimization for large systems on chips,” *IEEE Micro*, vol. 31, pp. 51–62, jul 2011.

[29] F. A. Hussin, T. E. C. Yu, T. Yoneda, and H. Fujiwara, “Redsocs-3d: Thermal-safe test scheduling for 3d-stacked soc,” in *2010 IEEE Asia Pacific Conference on Circuits and Systems*, pp. 264–267, IEEE, 2010.

[30] “Dissolving The Barriers In Multi-Substrate 3D-IC Assembly Design.”

[31] J. Fan and C. S. Tan, “Low temperature wafer-level metal thermo-compression bonding technology for 3d integration,” *Metallurgy-Advances in Materials and Processes*, vol. 52, no. 2, pp. 302–311, 2012.

[32] “File:Graph-representation-of-the-modular-scale-free-network-The-nodes-are-colored-according.png - Wikipedia,” Oct. 2008.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

[33] V. Mehta, S. Bawa, and J. Singh, "Analytical review of clustering techniques and proximity measures," *Artificial Intelligence Review*, vol. 53, pp. 5995–6023, 2020.

[34] C. Alpert, A. Kahng, G.-J. Nam, S. Reda, and P. Villarrubia, "A semi-persistent clustering technique for vlsi circuit placement," in *Proceedings of the 2005 international symposium on Physical design*, pp. 200–207, 2005.

[35] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the national academy of sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.

[36] N. K. Σκετόπουλος, "3d ic cad placement flows and algorithms yielding improved ppa," 2021.

[37] A. Bretto, "Hypergraph theory," *An introduction. Mathematical Engineering. Cham: Springer*, vol. 1, 2013.

[38] D. A. Papa and I. L. Markov, "Hypergraph partitioning and clustering.," *Handbook of Approximation Algorithms and Metaheuristics*, vol. 20073547, pp. 61–1, 2007.

[39] K. Shahookar and P. Mazumder, "Vlsi cell placement techniques," *ACM Computing Surveys (CSUR)*, vol. 23, no. 2, pp. 143–220, 1991.

[40] A. Kennings and K. P. Vorwerk, "Force-directed methods for generic placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2076–2087, 2006.

[41] B. B. Ray, S. Das, K. Hazra, N. Patra, and S. K. Mohanty, "An optimized hpwl model for vlsi analytical placement," in *2015 International Conference on Information Technology (ICIT)*, pp. 7–12, IEEE, 2015.

[42] A. S. LaPaugh, "Vlsi layout algorithms," in *Algorithms and theory of computation handbook: Special topics and techniques*, pp. 8–8, 2010.

[43] D. Zaporozhets, D. V. Zaruba, and V. V. Kureichik, "Representation of solutions in genetic vlsi placement algorithms," in *Proceedings of IEEE East-West Design & Test Symposium (EWDTS 2014)*, pp. 1–4, IEEE, 2014.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

[44] K. Sakuma, *3D integration in VLSI Circuits: implementation technologies and applications*. CRC Press, 2018.

[45] E. Ozer, K. Flautner, S. Idgunji, A. Saidi, Y. Sazeides, B. Ahsan, N. Ladas, C. Nicopoulos, I. Sideris, B. Falsafi, *et al.*, "Eurocloud: energy-conscious 3d server-on-chip for green cloud services," in *Workshop on Architectural Concerns in Large Datacenters in conjunction with ISCA*, vol. 10, 2010.

[46] S. Oh, M. Cho, X. Wu, Y. Kim, L.-X. Chuo, W. Lim, P. Pannuto, S. Bang, K. Yang, H.-S. Kim, *et al.*, "Iot 2—the internet of tiny things: Realizing mm-scale sensors through 3d die stacking," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 686–691, IEEE, 2019.

[47] M. O. Agyeman, A. Ahmadinia, and A. Shahrabi, "Heterogeneous 3d network-on-chip architectures: area and power aware design techniques," *Journal of Circuits, Systems and Computers*, vol. 22, no. 04, p. 1350016, 2013.

[48] S. Wong, A. El-Gamal, P. Griffin, Y. Nishi, F. Pease, and J. Plummer, "Monolithic 3d integrated circuits," in *2007 International Symposium on VLSI Technology, Systems and Applications (VLSI-TSA)*, pp. 1–4, IEEE, 2007.

[49] H. Zhuang, J. Lu, K. Samadi, Y. Du, and C.-K. Cheng, "Performance-driven placement for design of rotation and right arithmetic shifters in monolithic 3d ics," in *2013 International Conference on Communications, Circuits and Systems (ICCCAS)*, vol. 2, pp. 509–513, IEEE, 2013.

[50] M. Koyanagi, T. Fukushima, and T. Tanaka, "High-density through silicon vias for 3-d lsis," *Proceedings of the IEEE*, vol. 97, no. 1, pp. 49–59, 2009.

[51] D. K. Nayak, S. Banna, S. K. Samal, and S. K. Lim, "Power, performance, and cost comparisons of monolithic 3d ics and tsv-based 3d ics," in *2015 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, pp. 1–2, IEEE, 2015.

[52] R. Wang, K. Chakrabarty, and S. Bhawmik, "Interconnect testing and test-path scheduling for interposer-based 2.5-d ics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 1, pp. 136–149, 2014.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

[53] X. Zhang, J. K. Lin, S. Wickramanayaka, S. Zhang, R. Weerasekera, R. Dutta, K. F. Chang, K.-J. Chui, H. Y. Li, D. S. Wee Ho, *et al.*, "Heterogeneous 2.5 d integration on through silicon interposer," *Applied physics reviews*, vol. 2, no. 2, 2015.

[54] J.-Q. Lu, S. Devarajan, A. Zeng, K. Rose, and R. Gutmann, "Die-on-wafer and wafer-level three-dimensional (3d) integration of heterogeneous ic technologies for rf-microwave-millimeter applications," *MRS Online Proceedings Library*, vol. 833, pp. 211–216, 2004.

[55] M. Hella, S. Devarajan, J.-Q. Lu, K. Rose, and R. Gutmann, "Die-on-wafer and wafer-level 3d integration for millimeter-wave smart antenna transceivers," in *The 2005 IEEE Annual Conference Wireless and Micrwave Technology, 2005.*, pp. 125–128, IEEE, 2005.

[56] Y. Du, K. Samadi, and K. Arabi, "Emerging 3dvlsi: Opportunities and challenges," in *2015 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, pp. 1–5, IEEE, 2015.

[57] S. K. Popat and M. Emmanuel, "Review and comparative study of clustering techniques," *International journal of computer science and information technologies*, vol. 5, no. 1, pp. 805–812, 2014.

[58] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," *arXiv preprint arXiv:1109.2378*, 2011.

[59] R. Manikandan, P. Swaminathan, and R. Sujitha, "Unimodular hypergraph based clustering approaches for vlsi circuit partitioning," *International Journal of Engineering and Technology*, vol. 5, no. 3, pp. 2755–2758, 2013.

[60] S. Na, L. Xumin, and G. Yong, "Research on k-means clustering algorithm: An improved k-means clustering algorithm," in *2010 Third International Symposium on intelligent information technology and security informatics*, pp. 63–67, Ieee, 2010.

[61] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady, "Dbscan: Past, present and future," in *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, pp. 232–238, IEEE, 2014.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

[62] S. White and P. Smyth, "A spectral clustering approach to finding communities in graphs," in *Proceedings of the 2005 SIAM international conference on data mining*, pp. 274–285, SIAM, 2005.

[63] G. Karypis and V. Kumar, "Multilevel k-way hypergraph partitioning," in *Proceedings of the 36th annual ACM/IEEE design automation conference*, pp. 343–348, 1999.

[64] Hagen and Kahng, "A new approach to effective circuit clustering," in *1992 IEEE/ACM International Conference on Computer-Aided Design*, pp. 422–427, IEEE, 1992.

[65] G. Guo, T.-W. Huang, and M. Wong, "Fast sta graph partitioning framework for multi-gpu acceleration," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2023.

[66] T. Verbelen, T. Stevens, F. De Turck, and B. Dhoedt, "Graph partitioning algorithms for optimizing software deployment in mobile cloud computing," *Future Generation Computer Systems*, vol. 29, no. 2, pp. 451–459, 2013.

[67] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Papers on Twenty-five years of electronic design automation*, pp. 241–247, 1988.

[68] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell system technical journal*, vol. 49, no. 2, pp. 291–307, 1970.

[69] S. K. Lim, D. Xu, *et al.*, "Large scale circuit partitioning with loose/stable net removal and signal flow based clustering," in *1997 Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)*, pp. 441–446, IEEE, 1997.

[70] N. Sketopoulos, C. Sotiriou, and S. Simoglou, "Abax: 2d/3d legaliser supporting look-ahead legalisation and blockage strategies," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1469–1472, IEEE, 2018.

[71] N. Sketopoulos, C. Sotiriou, and V. Pavlidis, "Metal stack and partitioning exploration for monolithic 3d ics," in *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 398–403, IEEE, 2020.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

[72] X. Xu, N. Shah, A. Evans, S. Sinha, B. Cline, and G. Yeric, "Standard cell library design and optimization methodology for asap7 pdk," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 999–1004, IEEE, 2017.

[73] T. Ajayi, V. A. Chhabria, M. Fogaça, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem, *et al.*, "Toward an open-source digital flow: First learnings from the openroad project," in *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–4, 2019.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

APPENDICES

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Appendix A

Benchmarks Suite Tables

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

	Name	Components #	Macros #	IO Pins #	Nets #	Library
ISPD06	adaptec1	5.72E+05	542	1	583328	ASU7
	adaptec2	4.57E+05	543	1	469444	
	adaptec3	9.69E+05	723	1	992555	
	adaptec4	1.09E+06	1329	1	1125036	
	adaptec5	2.15E+06	646	1	2183992	
	bigblue1	5.98E+05	559	1	606381	
	bigblue2	8.30E+05	3313	1	882507	
	bigblue3	1.65E+06	675	1	1694238	
	newblue1	4.73E+05	390	1	486413	
	newblue2	6.61E+05	1171	1	711078	
	newblue3	8.32E+05	690	1	923452	
	newblue4	1.47E+06	569	1	1506429	
	newblue5	1.84E+06	1052	1	1927347	
	newblue6	2.71E+06	1376	1	2771776	
	newblue7	4.39E+06	6151	1	4624383	
ISPD11	superblue1	7.98E+05	432	6521	823024	ASU7
	superblue10	1.05E+06	1619	15141	1086013	
	superblue12	1.27E+06	89	1580	1293531	
	superblue15	1.07E+06	153	10556	1080519	
	superblue18	4.59E+05	207	3978	469076	
	superblue2	9.51E+05	654	8047	991109	
	superblue4	5.59E+05	306	6623	581127	
	superblue5	7.09E+05	784	4082	787292	

Table A.1: ISPD 2005, 2006 and 2011 designs characteristics.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

	Name	Components #	Macros #	IO Pins #	Nets #	Library
DAC12	superblue11	9.26E+05	1458	6872	959056	ASU7
	superblue12	1.27E+06	89	1580	1293531	
	superblue14	6.05E+05	340	5473	629772	
	superblue16	6.71E+05	419	4448	697660	
	superblue19	4.95E+05	286	3735	512053	
	superblue2	9.51E+05	654	8047	991109	
	superblue3	9.08E+05	575	6482	933398	
	superblue5	7.09E+05	784	4082	787292	
	superblue6	9.52E+05	565	5380	1006801	
	superblue7	1.32E+06	419	6499	1340566	
	superblue9	8.11E+05	272	4014	834024	
ICCCAD15	superblue1	1.21E+06	3787	3787	1215302	ICCAD Undesclosed LIB
	superblue10	1.88E+06	1696	1696	1897736	
	superblue16	9.82E+05	101	101	999559	
	superblue18	7.68E+05	653	653	771215	
	superblue3	1.21E+06	2074	2074	1224311	
	superblue4	7.96E+05	3471	3471	802245	
	superblue5	1.09E+06	1872	1872	1096924	
	superblue7	1.93E+06	4910	4910	1933334	
Open Source	Industrial_1	0.5E+05	0	2176	60883	IND
	Industrial_2	1.4E+05	0	1159	147960	
	b19	2.2E+05	0	47	225884	NANDGATE
	jpeg	6.7E+05	0	67	674353	
	leon3mp	6.5E+05	0	333	758278	
	netcard	9.6E+05	0	1846	1058447	

Table A.2: DAC 2012 and ICCAD 2015 designs characteristics.

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Appendix A

Analytical Comparison Results Tables

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

		CutSize				AreaRatio				ExecutionTime			
		50	100	300	500	50	100	300	500	50	100	300	500
adaptec1	hmetis	485853	507432	532211	548022	4.684	12.283	57.363	37.838	143.938	146.662	163.252	172.439
	kahypar	417265	429316	454157	466736	4.775	5.725	10.076	19.355	63.680	84.813	116.558	127.755
	patoh	415764	431669	454398	465435	4.705	7.328	1139.565	1720.930	26.422	27.401	52.245	52.981
	kahypar_MT	484084	495494	514509	525214	7.792	12.207	39.135	69.329	12.830	16.283	30.523	41.350
	Ours	63056	76920	123760	153142	1.430	1.623	4.250	9.323	39.605	46.334	79.727	134.060
adaptec2	hmetis	378834	403606	429583	445486	19.856	60.697	317.026	126.269	157.490	152.555	162.673	174.039
	kahypar	295567	313141	346705	364778	13.574	1914.231	101.914	186.730	53.647	77.235	102.817	130.033
	patoh	299061	315455	344355	356873	20.058	21757.217	468769.136	468769.136	24.753	44.935	44.552	54.217
	kahypar_MT	379516	396948	418335	429509	36.364	79.403	245.442	380.441	13.370	17.750	26.528	35.141
	Ours	91899	125499	137635	131495	6.844	16.780	119.656	349.751	43.165	45.249	82.670	78.939
adaptec3	hmetis	800048	832923	875832	900518	21.866	54.635	150.587	117.980	581.726	530.594	554.502	440.314
	kahypar	667932	692724	730066	752439	11.902	27.621	7063.985	171.212	118.727	145.182	193.939	231.502
	patoh	671719	700191	734564	754487	11.727	91640.888	108037.419	120765.115	42.327	55.809	73.246	88.060
	kahypar_MT	799133	821233	855158	870752	41.900	102.341	182.263	318.802	23.304	35.308	48.946	62.003
	Ours	124946	163254	207222	157677	4.940	11.120	47.475	164.314	106.889	105.654	141.613	257.156
adaptec4	hmetis	898757	925437	961709	982556	7.247	17.654	62.617	42.937	331.383	337.060	343.043	369.265
	kahypar	758387	777896	810696	825130	5.656	6.602	424.560	28.773	91.409	115.475	160.647	185.892
	patoh	755635	780074	816428	830430	4.964	6.317	15.970	30.853	37.615	45.097	63.044	85.861
	kahypar_MT	886010	907455	935114	949815	14.035	25.080	57.628	107.491	21.959	26.212	42.401	55.250
	Ours	82190	99550	201224	207018	3.500	3.500	5.677	10.793	97.638	91.681	224.594	340.366

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

		CutSize				AreaRatio				ExecutionTime			
		50	100	300	500	50	100	300	500	50	100	300	500
adaptec5	hmetis	1769704	1830740	1891519	1928763	8.403	24.541	67.491	47.272	862.158	844.348	900.344	970.018
	kahypar	1543299	1585061	1644196	1672337	6.312	40.384	46.838	119.143	303.961	301.599	400.728	529.433
	patoh	1546547	1604277	1653907	1676917	7.239	16.269	38768.045	1001801.523	97.381	96.145	132.839	169.554
	kahypar_MT	1755868	1807555	1854195	1885087	17.275	29.614	97.364	175.927	51.105	63.465	87.009	118.277
	Ours	178468	290669	391365	470324	3.500	5.992	21.458	49.157	225.275	289.059	366.820	1700.849
bigblue1	hmetis	506670	541503	579868	601335	2.492	5.534	21.224	16.604	181.046	178.964	204.226	194.720
	kahypar	430066	445354	482970	503833	3.755	5.232	6.601	9.074	91.013	103.378	191.494	270.720
	patoh	425899	444922	479187	493428	4.213	5.220	7.450	12.522	35.083	49.967	67.878	78.212
	kahypar_MT	510371	526287	563550	578394	5.112	8.334	17.257	28.745	26.855	31.057	44.329	56.972
	Ours	92255	104061	147850	165926	3.087	3.500	3.500	3.500	61.550	67.760	76.385	152.099
bigblue2	hmetis	653503	678791	703403	720436	19.097	41.234	128.861	54.846	307.800	325.820	376.460	375.987
	kahypar	499166	513708	548805	567953	8.474	10.697	13.403	13.809	55.585	69.952	101.188	136.418
	patoh	502587	520693	557052	580050	8.662	10.633	12.771	14.432	30.580	40.728	66.702	74.169
	kahypar_MT	651882	660707	687409	698660	32.788	39.991	74.870	113.314	17.542	21.593	35.911	59.430
	Ours	58463	75456	138873	209312	3.500	3.500	3.500	3.570	68.740	77.792	88.209	190.853
bigblue3	hmetis	1267122	1314832	1367681	1397406	22.336	59.578	331.997	250.063	515.041	455.340	493.542	542.732
	kahypar	1011047	1047757	1109533	1140718	9.394	53.122	2112.695	45820.444	168.053	160.793	216.167	253.811
	patoh	974523	1054982	1121755	1150190	16.580	6034.123	120765.115	1001801.523	52.232	60.545	119.617	100.040
	kahypar_MT	1260233	1292589	1335060	1362572	36.627	70.094	276.760	477.868	25.876	32.619	50.850	64.268
	Ours	299600	254458	364326	379467	4.143	9.228	38.090	107.892	150.891	134.541	237.769	372.600

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

		CutSize				AreaRatio				ExecutionTime			
		50	100	300	500	50	100	300	500	50	100	300	500
newblue1	hmetis	373428	392175	415227	428078	13.663	42.892	246.481	155.403	165.860	143.769	157.941	183.817
	kahypar	250641	268721	298337	315931	15.495	123.248	146.796	1735.744	49.381	53.817	84.078	99.693
	patoh	245640	270534	300672	316961	486.677	14146.668	56509.156	56509.156	24.057	24.356	41.968	49.774
	kahypar_MT	371980	384702	406186	417077	17.602	35.575	143.418	222.881	8.196	12.297	22.597	31.721
	Ours	91868	95409	110646	137213	14.167	32.944	425.363	890.273	34.204	58.460	69.431	108.995
newblue2	hmetis	507134	537443	568285	587846	74.978	254.725	1224.814	965.086	307.980	287.236	319.547	341.455
	kahypar	379091	394957	417690	431511	25.907	1209.647	25611.112	77745.512	65.312	63.710	110.042	113.173
	patoh	381541	396294	422133	439941	715.722	24028.346	194363.781	388727.562	37.933	42.170	48.913	71.702
	kahypar_MT	511699	523615	548437	560697	81.987	218.075	877.384	1437.009	20.042	24.476	45.856	67.049
	Ours	132098	160119	139366	100183	10.069	11.374	59.232	145.384	53.640	56.682	106.221	162.132
newblue3	hmetis	707201	730648	750868	761866	44.138	127.671	393.530	159.498	253.534	217.748	249.458	263.493
	kahypar	550208	567297	593419	602629	1399.962	13314.078	152108.786	638157.552	80.727	80.648	111.752	131.371
	patoh	540180	564905	593772	609615	56.428	152376.732	158633.983	1261870.500	189.726	111.793	121.456	85.296
	kahypar_MT	698979	711438	727394	738562	68.582	195.567	386.897	864.082	8.759	11.850	21.725	32.782
	Ours	122514	164600	157303	121464	10.083	14.942	73.933	229.258	52.887	60.273	126.108	219.295
newblue4	hmetis	1285860	1319356	1362762	1384881	3.665	7.999	32.744	21.760	458.778	408.005	461.207	509.939
	kahypar	1057380	1086016	1131092	1151578	5.119	10.043	9.721	10.681	202.624	295.163	426.907	293.624
	patoh	1049881	1085544	1138171	1158530	3.902	8.738	10.361	13.112	45.823	52.188	80.201	93.113
	kahypar_MT	1266902	1288700	1323438	1345809	5.408	13.408	31.518	43.310	32.182	33.651	57.792	69.902
	Ours	152070	266218	300496	150416	3.500	3.500	3.500	3.500	109.805	147.965	117.986	197.153

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

		CutSize				AreaRatio				ExecutionTime			
		50	100	300	500	50	100	300	500	50	100	300	500
newblue5	hmetis	1466696	1527025	1609797	1652924	6.146	17.497	104.337	64.319	1412.600	1122.600	1260.101	1179.890
	kahypar	1075304	1129692	1237367	1287899	8.350	10.379	167.162	8552.340	180.129	250.923	359.475	438.067
	patoh	1086140	1144559	1245379	1301969	7.850	10.620	31138.781	62186.058	101.539	103.119	137.379	169.515
	kahypar_MT	1440813	1488235	1576729	1608670	10.586	24.715	66.605	108.450	38.057	47.120	76.336	103.104
	Ours	340798	509373	512835	181689	3.500	4.133	14.584	34.708	263.326	273.702	484.432	593.192
newblue6	hmetis	-	2275798	2358569	2403258	-	6.287	21.998	14.137	-	1035.244	1420.161	1624.519
	kahypar	1908589	1947276	2021170	2057102	5.895	6.133	8.540	9.935	317.230	347.199	479.578	583.244
	patoh	1898588	1950325	2033266	2070747	4.161	6.943	8.406	8.689	102.345	115.419	188.231	200.466
	kahypar_MT	2185883	2221011	2296360	2342975	4.482	8.437	26.473	35.214	44.293	53.539	131.048	163.977
	Ours	241230	393760	462281	236456	3.500	3.500	3.500	3.500	488.226	422.501	806.882	1767.314
newblue7	hmetis	-	3574852	3680750	3744430	-	27.791	90.088	28.592	-	2264.889	2218.654	2286.185
	kahypar	2867501	2928234	3047297	3118568	6.076	7.798	58.562	76.041	448.630	512.388	866.804	821.743
	patoh	2871632	2950234	3073386	3133395	6.734	7.858	41.057	62212.803	193.443	254.628	272.231	369.316
	kahypar_MT	3451201	3507743	3601474	3662874	19.731	25.872	59.817	108.260	76.093	97.040	157.545	227.548
	Ours	314907	558158	749549	749549	3.500	3.500	5.105	8.957	913.574	1386.512	4524.730	4265.235
superblue11	hmetis	765296	867050	937326	968109	43.093	102.897	417.106	298.013	345.421	312.181	357.727	348.091
	kahypar	426682	479491	553828	590218	8.161	2394.404	4927.463	16828.900	136.006	277.558	511.856	431.312
	patoh	423771	470630	577814	614797	19.676	121.772	53390.510	186501.096	83.778	166.971	232.840	217.757
	kahypar_MT	723689	761171	804544	822986	58.348	164.435	370.754	432.313	27.884	33.755	84.542	101.038
	Ours	147237	328659	328307	294670	4.404	9.559	51.783	182.540	219.258	287.462	388.639	590.096

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

		CutSize				AreaRatio				ExecutionTime			
		50	100	300	500	50	100	300	500	50	100	300	500
superblue12	hmetis	768741	868718	930325	964210	31.621	106.095	445.233	263.571	379.302	351.639	450.494	452.753
	kahypar	426682	479491	553828	590218	8.161	2394.404	4927.463	16828.900	148.853	313.970	553.063	469.224
	patoh	423771	470630	577814	614797	19.676	121.772	53390.510	186501.096	85.458	161.610	222.185	214.218
	kahypar_MT	723689	761171	804544	822986	58.348	164.435	370.754	432.313	18.358	25.097	54.310	76.776
	Ours	147237	328659	328307	294670	4.404	9.559	51.783	182.540	186.687	240.935	321.285	405.208
superblue14	hmetis	533051	566549	608719	630875	16.580	41.107	160.277	90.039	169.760	176.076	205.227	203.807
	kahypar	345455	368029	433121	462590	9.883	11.883	325.879	217.172	174.845	320.704	554.718	810.146
	patoh	351574	382185	440029	472477	10.924	14.627	74.604	100318.886	76.813	277.905	140.819	180.781
	kahypar_MT	512921	524439	560678	579525	21.435	30.652	111.335	107.551	20.440	30.357	64.323	90.618
	Ours	102706	134401	238386	275671	3.500	3.500	8.565	25.132	48.614	79.311	105.164	150.334
superblue16	hmetis	468392	497568	523335	539170	28.820	97.167	340.228	273.720	181.314	223.038	249.805	280.245
	kahypar	296830	313588	362865	375965	11.926	12.710	193.714	1260.404	120.499	234.666	307.283	456.143
	patoh	306845	330144	367122	389269	9.185	11.714	4149.761	25935.520	120.333	89.136	138.884	177.896
	kahypar_MT	456419	467665	489905	504435	43.190	69.820	196.526	363.573	12.751	16.153	34.685	43.042
	Ours	112857	143089	208505	222390	3.500	3.500	14.048	55.589	126.468	145.873	205.677	260.398
superblue19	hmetis	418313	460845	506407	534867	61.641	183.588	768.278	481.419	179.753	168.012	207.062	173.008
	kahypar	244961	273522	332626	357174	41.181	99.981	498.050	303097.260	643.161	1177.128	2798.619	3962.555
	patoh	268569	300584	351276	376243	252869.714	762968.966	378223.932	378223.932	130.620	141.980	187.848	155.879
	kahypar_MT	389476	410347	440624	462360	98.665	191.271	576.094	853.217	17.010	18.746	39.686	80.554
	Ours	140922	132898	165867	167477	22.467	48.747	539.385	1265.687	93.026	81.654	153.564	155.657

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

		CutSize				AreaRatio				ExecutionTime			
		50	100	300	500	50	100	300	500	50	100	300	500
superblue2	hmetis	746775	783774	830776	853026	310.052	1053.795	3331.614	2255.794	364.158	343.212	424.868	440.687
	kahypar	389435	433188	510549	549556	81461.135	983.016	1592194.909	1592194.909	195.164	214.336	371.585	449.770
	patoh	403769	446257	537027	576473	959683.712	959679.123	1592194.909	1592194.909	237.239	392.459	265.944	281.012
	kahypar_MT	706647	734314	772339	795076	395.931	450.880	2270.579	3768.181	23.734	26.582	51.730	76.805
	Ours	301196	234271	282285	266047	8.853	18.832	113.470	683.979	216.554	193.326	298.973	451.767
superblue3	hmetis	829016	1032365	1158056	1222839	28.935	76.489	327.669	212.387	306.991	272.155	320.056	345.695
	kahypar	541088	593644	662053	693198	15.182	35.794	1032.614	775024.023	627.691	706.850	1363.609	2388.251
	patoh	518650	562043	638894	686466	15.483	2225.337	106233.822	391966.172	124.435	193.296	275.296	307.799
	kahypar_MT	737423	771739	807593	825520	39.618	78.160	187.399	298.450	26.370	43.583	74.375	117.898
	Ours	219897	344025	309782	314951	5.038	12.385	66.689	287.758	165.877	231.232	286.859	407.884
superblue5	hmetis	612066	682061	735574	767718	75.964	171.435	847.148	553.883	241.654	199.446	256.894	257.669
	kahypar	390957	418499	467633	489105	18.783	47.888	319.037	88508.069	176.803	220.902	398.260	390.826
	patoh	393179	430734	472534	499081	14162.370	113889.059	265524.206	1056062.182	66.730	74.335	125.390	146.834
	kahypar_MT	588846	603562	627375	639085	90.217	164.830	477.740	888.466	16.631	22.776	32.433	44.518
	Ours	160538	244857	226813	223053	6.969	15.749	106.913	417.720	129.471	148.368	187.371	339.334
superblue6	hmetis	815157	918373	1001999	1045304	49.908	110.440	294.855	189.888	385.129	351.443	441.254	389.923
	kahypar	498683	555374	625051	661046	14.917	36.506	171.347	32142.356	418.756	592.783	1023.710	1455.893
	patoh	493292	551150	638077	675223	18.059	3229.179	517048.586	517048.586	204.850	239.792	211.827	317.595
	kahypar_MT	755035	783888	835123	849976	51.886	107.132	173.681	387.971	30.601	40.115	99.501	99.501
	Ours	222762	248894	301753	310695	4.446	10.449	53.784	219.726	182.165	147.367	212.179	248.588

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

		CutSize				AreaRatio				ExecutionTime			
		50	100	300	500	50	100	300	500	50	100	300	500
superblue7	hmetis	1071167	1273863	1436343	1499000	18.572	67.950	216.360	163.781	519.530	422.034	472.733	487.882
	kahypar	630513	743339	868989	934566	13.878	30.038	199.219	5277.102	424.490	631.680	1127.126	1389.610
	patoh	637176	722327	877284	962410	14.496	3386.343	839299.023	252939.432	168.811	272.737	381.934	505.014
	kahypar_MT	966925	1030541	1120187	1154369	18.721	51.757	157.405	211.812	34.929	52.463	107.045	140.189
	Ours	281930	409552	469087	448471	4.438	9.725	45.740	172.521	234.388	285.251	685.149	457.795
superblue9	hmetis	728876	927102	1056463	1125760	19.577	39.249	165.173	127.837	332.917	280.456	327.275	351.985
	kahypar	417329	467590	523103	557243	13.265	3604.508	91.451	397.259	529.964	647.332	1380.048	2078.353
	patoh	419626	463371	557084	589124	12.299	32.906	124.081	220.934	114.842	193.392	201.355	245.513
	kahypar_MT	654589	680709	726261	754340	22.876	41.705	113.209	200.827	25.861	39.255	62.089	119.085
	Ours	225093	297017	288634	310482	5.319	11.940	68.174	282.054	125.413	139.480	195.722	281.847
superblue1	hmetis	580395	632481	675723	702652	27.284	53.014	126.588	106.563	337.429	360.649	394.247	463.317
	kahypar	308953	350717	414593	450793	11.236	11.880	96.543	29409.252	206.825	536.180	1119.381	1336.500
	patoh	325133	371986	455821	499537	19615.637	9798.151	10638.222	10734.333	92.799	153.905	217.780	314.031
	kahypar_MT	559263	577246	633262	650338	21.036	39.031	100.307	150.668	30.151	36.880	81.582	125.657
	Ours	103281	235210	279083	260943	3.500	5.229	27.356	80.279	89.565	88.384	219.242	206.444
superblue10	hmetis	902073	1014592	1087893	1154139	29.181	34.500	192.030	106.731	396.560	367.899	388.414	390.183
	kahypar	508161	533997	641573	707024	11.711	183.477	21762.907	334791.027	115.969	238.361	362.747	477.814
	patoh	492345	549908	665581	716691	22908.743	24467.972	24439.380	24439.701	148.887	229.073	337.804	360.280
	kahypar_MT	795099	860264	931177	955125	62.648	101.778	103.501	151.968	31.444	38.865	79.036	151.258
	Ours	211127	341686	400391	380353	6.680	9.882	54.845	153.700	196.396	211.395	294.411	399.014

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

		CutSize				AreaRatio				ExecutionTime			
		50	100	300	500	50	100	300	500	50	100	300	500
superblue12	hmetis	833650	922137	979030	1019595	17.269	49.427	181.935	176.603	707.453	717.004	812.573	931.462
	kahypar	404410	469983	603121	681127	15.065	31.807	1344.543	995.764	821.206	1160.301	1207.761	1540.616
	patoh	415480	482889	629406	722463	27267.226	29382.784	29384.621	29384.417	222.247	225.873	501.293	469.966
	kahypar_MT	709778	765778	870372	927074	19.830	42.339	109.498	185.452	63.166	73.064	116.009	181.253
	Ours	249063	353946	346397	306692	7.395	17.101	141.698	401.401	132.553	148.253	220.766	348.131
superblue15	hmetis	930802	1051725	1171531	1229802	11.301	25.871	100.347	88.414	310.352	271.568	311.003	317.277
	kahypar	650424	705594	817901	874169	7.305	10.244	12.384	23.975	90.459	313.386	773.125	1072.397
	patoh	630251	691981	824383	873688	11389.877	7574.969	2842.852	2458.856	97.335	219.806	284.248	333.601
	kahypar_MT	816229	881115	946917	974551	13.065	23.261	35.883	74.957	31.467	53.498	110.936	148.704
	Ours	148461	244469	355853	496497	3.500	3.500	3.500	3.500	144.072	222.720	262.371	376.365
superblue18	hmetis	400719	556054	691896	753377	13.111	46.205	228.837	129.670	152.725	124.679	121.535	135.950
	kahypar	256812	281362	338143	358990	11.722	29.385	446.605	169222.126	601.174	1006.438	2676.277	3921.703
	patoh	265627	305480	363790	388411	13705.253	14355.959	14695.308	14719.044	255.102	259.958	358.717	299.892
	kahypar_MT	385314	397206	426149	445338	33.667	42.012	184.799	250.564	20.752	34.042	89.115	84.848
	Ours	173266	216337	224297	250367	5.752	15.714	120.482	482.905	181.693	164.955	123.728	178.517
superblue2	hmetis	748374	782592	827791	852537	247.279	1058.150	3854.915	2825.866	358.156	397.372	444.375	438.115
	kahypar	389435	433188	510549	549556	81461.135	983.016	159219.4.909	1592194.909	206.811	255.823	429.448	527.405
	patoh	400016	446257	524473	575108	70056.838	70056.503	70056.532	70056.532	224.893	386.274	273.556	329.311
	kahypar_MT	706647	734314	772339	795076	395.931	450.880	2270.579	3768.181	24.572	30.779	55.952	84.382
	Ours	301196	234271	279422	266047	8.853	18.832	112.064	683.979	210.213	204.246	316.642	586.713

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

		CutSize				AreaRatio				ExecutionTime			
		50	100	300	500	50	100	300	500	50	100	300	500
superblue4	hmetis	502333	540685	573789	591005	42.013	106.168	513.799	347.621	156.080	149.702	179.682	161.046
	kahypar	319978	343521	396591	429437	14.475	31.351	26395.768	3272.005	98.905	150.773	248.171	403.781
	patoh	313305	355061	408090	435328	16866.076	18081.276	18081.057	18081.014	79.182	108.786	114.829	140.203
	kahypar_MT	479454	493163	522736	536962	58.475	103.473	284.262	524.297	14.646	20.466	38.455	62.415
	Ours	157762	207816	209577	220220	4.868	11.622	55.257	231.800	100.911	124.140	246.806	393.094
superblue5	hmetis	612106	680920	732220	763294	81.686	170.574	739.415	557.912	263.398	239.106	249.724	260.876
	kahypar	394432	422248	465757	493551	20.066	44.408	24901.788	56944.529	156.435	233.530	349.472	412.154
	patoh	386984	426500	478790	497085	46462.231	46466.371	46466.371	46466.692	69.919	84.760	116.658	137.778
	kahypar_MT	587697	602899	627290	639267	84.756	169.034	483.433	872.080	19.300	30.301	38.115	54.735
	Ours	138940	219365	232189	242911	7.866	16.375	106.775	401.996	145.882	162.736	266.740	428.663
superblue1	hmetis	910929	969748	1024149	1050747	17.392	42.145	126.665	112.308	367.715	394.330	412.561	387.440
	kahypar	591313	628858	695052	716592	11.648	1776.730	460030.908	230103.955	682.965	1701.369	6667.250	8049.043
	patoh	621675	665892	715147	746527	8.800	460030.908	54135.980	184069	137.264	195.348	195.401	238.627
	kahypar_MT	907975	946956	994658	1023705	14.182	31.492	66.989	118.139	15.2610	20.357	34.346	54.777
	Ours	134844	299444	410016	467327	3.499	5.070	17.326	34.147	117.838	161.288	249.237	410.929
superblue10	hmetis	1590704	1639050	1693615	1720018	25.283	42.008	109.420	106.236	429.938	429.460	506.801	508.771
	kahypar	1086909	1125847	1196137	1234209	828.580	681.456	1122.377	116419.481	691.212	1733.719	10500.271	21749.373
	patoh	1121143	1180383	1247160	1277150	70577.066	349274.9164	419098.400	1047427.615	361.294	367.894	408.296	341.301
	kahypar_MT	1516400	1612863	1669421	1694002	20.487	36.791	117.336	115.662	19.3998	27.300	48.071	67.807
	Ours	239306	478413	626824	694175	6.0587	7.928	29.317	52.864	299.647	340.672	572.359	1119.295

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

		CutSize				AreaRatio				ExecutionTime			
		50	100	300	500	50	100	300	500	50	100	300	500
superblue12	hmetis	696441	739390	776477	793532	24.861	62.824	290.381	271.498	231.509	250.855	335.954	324.684
	kahypar	491473	504950	550237	565942	1.179	595.976	21621.406	4473.210	888.336	908.012	4426.728	11628.424
	patoh	497080	535331	581835	600357	3.094	4099.147	64858.673	194526.128	145.838	155.439	261.577	198.070
	kahypar_MT	697737	731348	759963	771470	25.704	55.445	183.781	289.900	12.241	13.585	23.100	31.117
	Ours	148442	214294	294582	309776	3.500	3.500	10.576	19.828	133.082	131.309	255.406	430.182
superblue16	hmetis	647896	676828	716780	739957	19.102	58.635	210.445	206.909	237.845	251.186	284.622	281.303
	kahypar	443670	472128	521593	547068	14.177	3586.312	5954.624	7469.741	420.246	1026.412	2992.179	4117.647
	patoh	435129	475849	525760	556479	16.826	1426.423	180324.714	97111.060	80.700	153.191	183.268	183.424
	kahypar_MT	650826	670882	709724	723481	19.652	39.918	125.571	246.242	13.390	19.301	34.210	44.015
	Ours	177418	248459	425347	378016	5.472	14.005	48.206	95.926	134.364	133.675	221.552	258.309
superblue18	hmetis	1005456	1055194	1117861	1139846	42.017	86.328	459.269	356.587	386.292	442.463	461.513	456.794
	kahypar	647108	704507	789306	825250	6.106	20168.690	1461780.083	584892.000	763.574	1820.090	5683.037	11676.707
	patoh	654544	706736	791625	834655	22324.184	112477.899	731171.261	731171.261	183.754	204.398	226.489	249.743
	kahypar_MT	1003188	1041525	1088359	1114434	48.527	104.878	256.663	506.216	19.443	26.059	43.635	63.373
	Ours	193544	295877	455567	540318	5.875	12.798	50.205	98.377	149.087	202.711	337.726	395.083
superblue2	hmetis	657653	689974	727608	743339	23.588	66.533	243.352	242.505	206.594	203.655	276.732	277.252
	kahypar	432486	464947	513444	539116	24.956	158.388	12402.400	22798.736	668.415	1482.739	6045.656	13367.338
	patoh	468678	497815	542013	556397	34451.111	155030.000	310060.000	774911.492	105.683	189.683	194.834	198.817
	kahypar_MT	648226	671709	708694	727909	22.758	71.133	172.106	239.934	10.905	13.628	24.969	38.991
	Ours	216505	277231	347368	372106	3.970	8.779	32.427	66.923	119.747	137.867	194.708	260.921

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

		CutSize				AreaRatio				ExecutionTime			
		50	100	300	500	50	100	300	500	50	100	300	500
superblue4	hmetis	923428	961468	1000038	1018105	61.820	189.701	672.442	663.745	285.397	251.053	326.555	321.612
	kahypar	635918	659734	697745	725443	12.334	108.399	221339.792	1991377.080	234.986	545.119	1899.132	3312.556
	patoh	650732	680073	728788	757224	13020.033	221339.792	1991377.080	1991377.080	104.417	195.294	196.317	162.957
	kahypar_MT	912726	944645	978178	996443	89.035	174.623	423.043	872.322	11.939	14.794	31.153	39.812
	Ours	198696	213100	428961	429354	6.770	21.077	58.630	128.811	132.549	133.692	208.178	328.530
superblue5	hmetis	1500890	1576690	1688512	1740263	17.038	52.523	228.877	210.409	666.322	701.511	761.815	849.094
	kahypar	1006951	1078921	1234653	1291089	6.631	4718.448	105536.000	33327.158	1752.652	4148.495	25279.412	63610.952
	patoh	1029124	1118425	1267402	1308443	4978.202	31660.970	452257.372	1055576.595	356.356	255.824	385.569	414.228
	kahypar_MT	1468115	1551589	1663464	1711840	20.168	36.649	157.594	271.455	28.364	37.310	72.679	110.919
	Ours	404842	557608	706266	756347	4.472	9.513	32.564	60.930	301.748	346.142	697.399	1111.844

		Top 1000 Fanout Distribution				top 1000 Delay Paths Distribution				Top 1000 Slack Paths Distribution			
		50	100	300	500	50	100	300	500	50	100	300	500
superblue1	hmetis	1.943	2.272	2.808	3.411	26.274	27.922	35.272	37.762	9.734	10.286	11.689	12.021
	kahypar	1.009	1.038	1.13	1.19	9.87	11.417	15.85	18.596	5.184	6.167	7.147	7.379
	patoh	1.072	1.113	1.166	1.206	10.105	11.234	17.262	18.98	5.761	6.232	7.892	8.222
	kahypar_MT	2.021	2.095	2.486	2.799	21.587	28.576	34.231	36.241	9.124	9.916	11.116	11.613
	Ours	1.446	3.011	3.956	5.069	2.988	5.921	8.057	8.023	1.517	2.065	2.473	2.691
superblue10	hmetis	1.942	2.11	2.428	2.749	26.441	28.246	37.594	38.783	10.484	10.81	11.403	11.513
	kahypar	1.001	1.003	1.046	1.117	9.158	11.977	17.11	17.731	6.725	8.119	9.757	10.039
	patoh	1.042	1.069	1.126	1.167	9.672	12.757	18.98	19.782	6.409	8.189	10.429	10.724
	kahypar_MT	1.931	1.956	2.219	2.485	24.168	31.312	35.997	38.178	10.258	10.933	11.403	11.484
	Ours	1.612	3.564	4.99	5.567	3.14	6.138	6.736	8.178	1.674	1.761	1.867	1.84
superblue12	hmetis	1.67	2.149	2.849	3.321	17.062	18.48	20.959	22.56	6.73	6.879	7.148	7.201
	kahypar	1.026	1.014	1.071	1.13	8.356	8.028	9.972	10.35	5.129	5.024	5.761	5.968
	patoh	1.087	1.166	1.252	1.261	7.453	8.523	11.188	12.165	3.775	4.972	6.35	6.703
	kahypar_MT	1.919	1.945	2.307	2.419	15.957	18.969	20.759	21.372	6.649	6.939	7.078	7.158
	Ours	1.777	2.605	4.142	4.412	2.473	3.611	5.375	6.248	1.491	1.592	1.671	1.706
superblue16	hmetis	2.1	2.625	3.561	4.508	22.681	24.697	32.346	34.162	4.907	4.955	5.077	5.094
	kahypar	1.025	1.074	1.342	1.672	12.297	13.129	17.243	19.064	3.745	3.868	3.986	4.066
	patoh	1.052	1.114	1.207	1.289	9.501	11.838	17.643	20.277	3.412	3.907	4.604	4.935
	kahypar_MT	2.192	2.503	3.51	3.984	22.916	26.068	31.793	34.676	4.823	4.952	5.041	5.087
	Ours	1.979	2.933	6.187	5.132	4.289	8.223	8.281	9.699	1.277	1.382	1.345	1.349

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148

		Top 1000 Fanout Distribution				top 1000 Delay Paths Distribution				Top 1000 Slack Paths Distribution			
		50	100	300	500	50	100	300	500	50	100	300	500
superblue18	hmetis	2.051	2.561	3.69	4.508	26.309	29.365	37.035	38.573	6.005	6.093	6.274	6.339
	kahypar	1.026	1.066	1.322	1.484	8.203	10.675	16.078	17.928	3.864	4.532	5.061	5.186
	patoh	1.069	1.099	1.199	1.261	8.467	12.928	15	18.318	3.836	4.556	5.402	5.792
	kahypar_MT	2.085	2.335	3.163	3.651	23.139	28.72	35.011	37.644	5.875	6.04	6.274	6.321
	Ours	1.701	2.484	3.955	5.118	1.884	3.853	7.311	7.652	1.563	1.638	1.601	1.607
superblue2	hmetis	1.961	2.181	2.712	3.271	17.14	17.853	21.2	22.495	6.456	6.494	6.719	6.768
	kahypar	1.018	1.021	1.188	1.251	7.035	8.831	10.48	11.325	4.184	4.876	5.495	5.624
	patoh	1.096	1.14	1.217	1.274	8.228	9.227	12.692	13.205	4.829	5.39	6.08	6.312
	kahypar_MT	1.975	2.081	2.467	2.765	16.658	19.086	21.181	21.963	6.336	6.567	6.7	6.757
	Ours	3.128	4.185	5.985	6.608	4.856	5.873	6.505	6.109	1.439	1.514	1.654	1.619
superblue4	hmetis	1.993	2.326	3.01	3.619	25.289	27.908	35.509	36.392	4.985	5.04	5.162	5.193
	kahypar	1.004	1.024	1.039	1.066	11.743	12.719	13.579	16.316	3.719	3.748	3.957	4.063
	patoh	1.112	1.133	1.161	1.212	11.184	14.526	16.213	18.912	3.793	4.005	4.238	4.796
	kahypar_MT	1.964	2.155	2.659	3.172	23.328	28.203	35.14	37.4	4.909	5.061	5.156	5.176
	Ours	2.284	2.733	5.777	5.876	2.866	4.406	5.426	7.352	1.582	1.676	1.603	1.634
superblue5	hmetis	1.985	2.174	2.702	3.16	23.728	25.558	35.799	38.066	5.875	5.863	6.127	6.176
	kahypar	1.012	1.056	1.299	1.509	8.544	11.551	17.068	18.652	4.191	4.459	5.069	5.127
	patoh	1.121	1.156	1.248	1.347	8.908	11.653	17.435	18.837	4.341	4.421	5.49	5.895
	kahypar_MT	2.014	2.093	2.557	2.888	20.25	26.652	34.199	37.443	5.662	5.919	6.116	6.166
	Ours	2.094	3.174	3.818	4.063	4.819	6.749	8.774	9.934	1.612	1.723	1.689	1.822

Institutional Repository - Library & Information Centre - University of Thessaly
13/07/2024 00:33:54 EEST - 3.142.172.148