Spring 5-31-2022

Integrated Circuits Parasitic Capacitance Extraction Using Machine Learning and its Application to Layout Optimization

Mohamed Saleh Abouelyazid Saleh
mohsaleh@aucegypt.edu

Graduate Studies

Integrated Circuits Parasitic Capacitance Extraction Using Machine Learning and its Application to Layout Optimization

A THESIS SUBMITTED BY

Mohamed Saleh Abouelyazid Saleh

TO THE

Department of Electronics and Communications Engineering

UNDER THE SUPERVISION OF

Prof. Yehea Ismail

Dr. Sherif Hammouda

May, 2022

in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electronics and Communications Engineering

Declaration of Authorship

I, Mohamed Saleh, declare that this thesis titled, "Integrated Circuits Parasitic Capacitance Extraction Using Machine Learning and its Application to Layout Optimization" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:
Mohamed Saleh

Date:
13-April-2022

To my beloved family

Abstract

The impact of parasitic elements on the overall circuit performance keeps increasing from one technology generation to the next. In advanced process nodes, the parasitic effects dominate the overall circuit performance. As a result, the accuracy requirements of parasitic extraction processes significantly increased, especially for parasitic capacitance extraction. Existing parasitic capacitance extraction tools face many challenges to cope with such new accuracy requirements that are set by semiconductor foundries (< 5% error). Although field-solver methods can meet such requirements, they are very slow and have a limited capacity. The other alternative is the rule-based parasitic capacitance extraction methods, which are faster and have a high capacity; however, they cannot consistently provide good accuracy as they use a pre-characterized library of capacitance formulas that cover a limited number of layout patterns. On the other hand, the new parasitic extraction accuracy requirements also added more challenges on existing parasitic-aware routing optimization methods, where simplified parasitic models are used to optimize layouts.

This dissertation provides new solutions for interconnect parasitic capacitance extraction and parasitic-aware routing optimization methodologies in order to cope with the new accuracy requirements of advanced process nodes as follows.

First, machine learning compact models are developed in rule-based extractors to predict parasitic capacitances of cross-section layout patterns efficiently. The developed models mitigate the problems of the pre-characterized library approach, where each compact model is designed to extract parasitic capacitances of cross-sections of arbitrary distributed metal polygons that belong to a specific set of metal layers (i.e., layer combination) efficiently. Therefore, the number of covered layout patterns significantly increased.

Second, machine learning compact models are developed to predict parasitic capacitances of middle-end-of-line (MEOL) layers around FINFETs and MOSFETs. Each compact model extracts parasitic capacitances of 3D MEOL patterns of a specific device

type regardless of its metal polygons distribution. Therefore, the developed MEOL models can replace field-solvers in extracting MEOL patterns.

Third, a novel accuracy-based hybrid parasitic capacitance extraction method is developed. The proposed hybrid flow divides a layout into windows and extracts the parasitic capacitances of each window using one of three parasitic capacitance extraction methods that include: 1) rule-based; 2) novel deep-neural-networks-based; and 3) field-solver methods. This hybrid methodology uses neural-networks classifiers to determine an appropriate extraction method for each window. Moreover, as an intermediate parasitic capacitance extraction method between rule-based and field-solver methods, a novel deep-neural-networks-based extraction method is developed. This intermediate level of accuracy and speed is needed since using only rule-based and field-solver methods (for hybrid extraction) results in using field-solver most of the time for any required high accuracy extraction.

Eventually, a parasitic-aware layout routing optimization and analysis methodology is implemented based on an incremental parasitic extraction and a fast optimization methodology. Unlike existing flows that do not provide a mechanism to analyze the impact of modifying layout geometries on a circuit performance, the proposed methodology provides novel sensitivity circuit models to analyze the integrity of signals in layout routes. Such circuit models are based on an accurate matrix circuit representation, a cost function, and an accurate parasitic sensitivity extraction. The circuit models identify critical parasitic elements along with the corresponding layout geometries in a certain route, where they measure the sensitivity of a route's performance to corresponding layout geometries very fast. Moreover, the proposed methodology uses a nonlinear programming technique to optimize problematic routes with pre-determined degrees of freedom using the proposed circuit models. Furthermore, it uses a novel incremental parasitic extraction method to extract parasitic elements of modified geometries efficiently, where the incremental extraction is used as a part of the routing optimization process to improve the optimization runtime and increase the optimization accuracy.

Acknowledgements

I would like to thank my supervisor, Prof. Yehea Ismail, who made this work possible. His great guidance, understanding, and support carried me through all the phases of writing this dissertation. I am grateful to him for teaching me many scientific research methods that will benefit me in either the academic or professional domains.

I would like to thank my manager, Dr. Sherif Hammouda, for his supervision, continuous support, guidance, and understanding. With his great guidance and help, I managed to pursue and finish my studies. I am grateful to him.

I would like to give a special thanks to my entire family, father, mother, sisters, wife, and boys for their continuous support, understanding, and encouragement. With your prayers, I made it this far.

Finally, I thank Allah for everything.

Contents

List of Figures

List of Tables

List of Abbreviations

IC	**Integrated Circuit**
NN	**Neural Network**
MEOL	**Middle-End-of Line**
SVR	**Support Vector Regression**
3DIC	**Three-Dimensional Integrated Circuit**
TSV	**Through-Silicon-Via**
ILD	**Inter Layer Dielectric**
ADAM	**Adaptive Moment Estimation**
SGD	**Stochastic Gradient Descent**
RELU	**Rectified Linear Activation Unit**
RBF	**Radial Basis Function**
MSE	**Mean Square Error**
STDEV	**Standard Deviation**
RSD	**Raised Source Drain**
RO	**Ring Oscillator**
SRAM	**Static Read Access Memory**
VCO	**Voltage-Controlled Oscillator**
CM	**Cache Memory**
DAC	**Digital to Analog Converter**
SPR	**Sum of Positive Ranks**
SNP	**Sum of Negative Ranks**
RC	**Resistance and Capacitance**
RLCK	**Resistance, Capacitance, and Self and Mutual Inductance**
PDK	**Process Design Kit**
CF	**Cost Function**
RCF	**Relative Cost Function**
DCF	**Delay Cost Function**
MR	**Maximum coupling capacitance interaction range**
BW	**Band Width**
PM	**Phase Margin**
GM	**Gain Margin**
CMFB	**Common Mode Feedback**

List of Symbols

ε_r	Dielectric constant
θ	Sidewall edge angle
G	Admittance Matrix
C	Capacitance Matrix
Pi	A parasitic element
R_i	A parasitic resistive element
Cc_i	A parasitic capacitive element
m_k	A circuit moment at k^{th} order
r_t	A ratio of a threshold voltage to a max voltage
S_i	A circuit (i.e., network) response
Ge	Route Geometries
Π	RC PI model

Chapter 1

Introduction

During the past decades, the semiconductor industry has developed considerably. There is a continuous increase in the market demand to integrate more functionalities together on a single chip at a much lower cost and higher speed. Such an increasing demand motivated process technology nodes to scale down in a continuous manner. Therefore, the density of integrated circuits keeps increasing, and the dimensions of metal wires (i.e., interconnect) keep decreasing from one technology generation to the next. This resulted in an increase in the impact of interconnect parasitic elements on chips performance, which is one of the major problems in advanced process technology nodes [1]–[4]. Fig. 1.1 shows the main contributors on the total circuit delay across different process technology nodes highlighting that the RC parasitic effects became the main contributor in smaller process technology nodes. Such an increasing trend of the RC parasitic impact on the total delay has continued in more advanced nodes. As a result, more accurate parasitic extraction methods are required in order to 1) provide accurate post-layout circuit simulation results; 2) reduce the design turn-around time; and 3) improve the yield, especially for analog and mixed-signal designs.

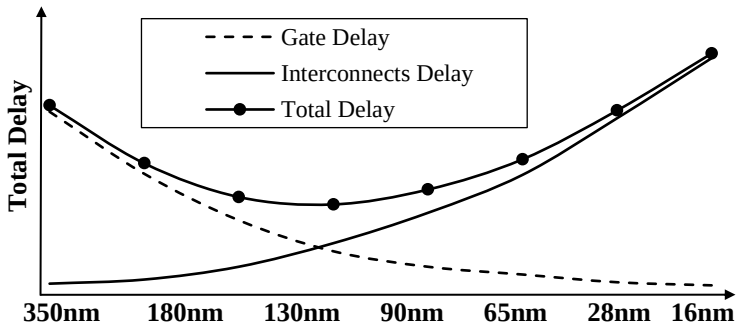


Fig. 1.1. An illustrative graph that shows the increasing impact of interconnects delay on total circuit delay across different process technology nodes (based on [1]–[3], [5]).

1.1. Interconnect Parasitic Elements

Interconnect parasitic elements represent the unintended passive circuit elements, such as resistors, capacitors, and inductors, that are not included in original circuit designs but exist in final chips. Such parasitic elements are associated with circuit routes (i.e., interconnect) that connect circuit devices together [6]–[8].

The impact of parasitic inductances on a circuit performance is usually negligible, except for parasitic inductances that are associated with global interconnect in some very high frequency applications. Therefore, the extraction of parasitic inductances is not required for most of current applications.

As for parasitic resistances, the number of resistance elements significantly increased in advanced nodes adding more challenges to circuit simulators to handle such a huge network. However, existing parasitic resistance extraction tools still can handle layout structures of advanced nodes as parasitic resistances are mainly correlated with interconnect's dimensions, and foundries provide required resistance parameters such as resistivity and sheet resistance values for each metal layer in a process technology node throughout an interconnect technology format (ITF) specifications file. Therefore, parasitic extraction tools use the provided resistance parameters to calculate interconnect parasitic resistances. Once parasitic resistances are extracted, they are easily combined with parasitic capacitances in a single network [6]–[8].

As for parasitic capacitances, the increasing complexities of layout designs and process stacks, in advanced process technology nodes, have a significant impact on the accuracy of current parasitic capacitance extraction tools, where layouts became denser, and the number of interactions and fringing coupling capacitances among interconnects significantly increased.

1.2. Layout Parasitic Extraction

The layout parasitic extraction is an essential step in integrated circuit (IC) design flows as shown in Fig. 1.2. It is used to extract the parasitic elements of a given layout and associate them with the corresponding circuit network (i.e., netlist). The netlist is later used by a circuit simulator to perform a post-layout simulation in order to verify the performance of the corresponding layout. In case of any violation in post-layout simulation results, the layout designer would adjust his layout, re-extract its parasitic elements, and re-simulate it. Such a process is repeated until the simulation results meet the required circuit specifications [9].

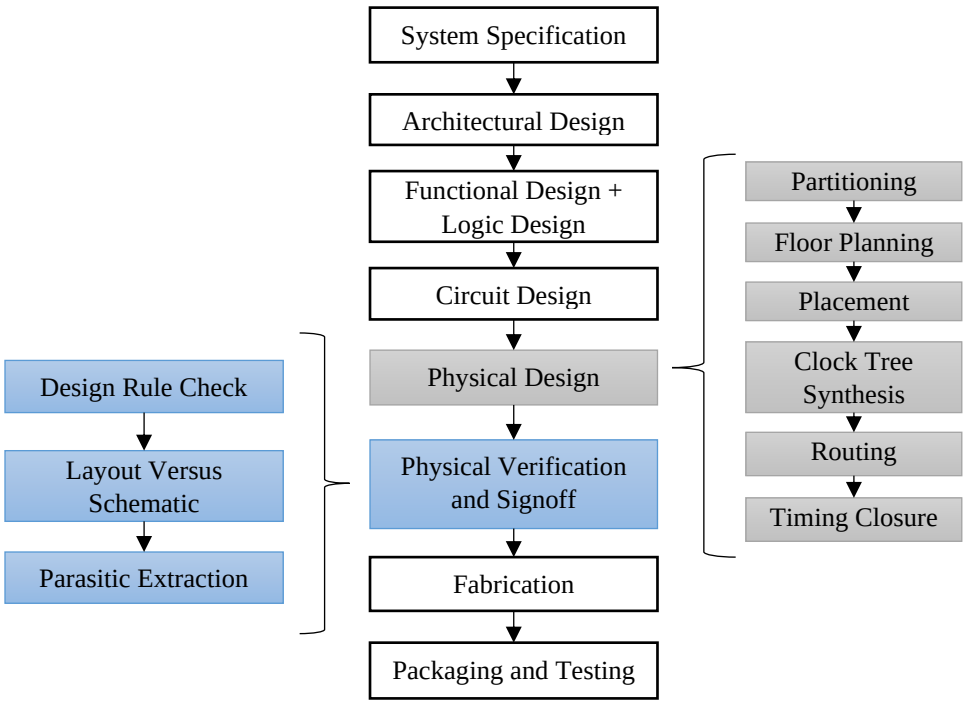


Fig. 1.2. VLSI design flow.

In other words, the current IC design flow requires multiple parasitic extraction and simulation runs until the given layout meets the required circuit specifications. Any inaccuracy in the extracted parasitic elements would generate misleading post-layout simulation results. Such misleading results would degrade the yield and increase the

turn-around time of a circuit design. Moreover, in advanced process technology nodes, the accuracy requirements of the parasitic capacitance extraction significantly increased by semiconductor foundries (< 5% error) due to the increasing impact of parasitic elements on a circuit performance. This increase added more challenges on parasitic capacitance extraction tools in order to meet such new requirements [10]–[13].

There are two main parasitic capacitance extraction methods including field-solver and rule-based extraction methods. Field-solvers provide very accurate parasitic capacitance results relative to measurements; however, they are very slow and have a limited capacity [14]. Field-solvers mainly use numerical methods to perform electrostatic (or electromagnetic) simulations over a given layout. This is done by solving Maxwell equations across the entire layout domain using any of computational methods such as finite difference (FDM), finite element (FEM), and boundary element (BEM) methods. On the other hand, rule-based extraction methods, also known as 2.5D extraction methods, are way faster than field-solvers, and they can handle full chips with a reasonable accuracy. Rule-based extraction methods use pattern matching operations to match every layout pattern with corresponding pre-characterized analytical or empirical parasitic capacitance formulas that are stored in a database (i.e., library) of pre-characterized formulas [11], [15], [16]. To improve the accuracy of rule-based extraction methods, one solution is to extract the parasitic capacitances of complicated and problematic layout structures using a field-solver; however, this is not a sustainable solution because the efficiency of existing rule-based methods is decreasing from one technology generation to the next, and the size of layout designs keeps increasing. Therefore, more layout patterns would be extracted by field-solvers impacting the performance and the capacity of parasitic capacitance extraction processes. As a result, there is a strong need to either improve the accuracy of rule-based parasitic capacitance extraction models or create new accurate parasitic extraction methods that can cope with the new accuracy requirements and handle the complicated and denser layout designs in advanced process nodes [6], [11], [12], [17].

There are two main uses of parasitic extraction tools that include: 1) a parasitic extraction for full chip verification (i.e., for sign-off) and 2) a parasitic extraction in optimization loops (i.e., implementation-level parasitic extraction tools) [18].

1.2.1. Parasitic Extraction for Verification

The parasitic extraction for full chip verification uses sign-off parasitic extraction tools, such as Calibre PEX [19]. Such tools are mainly used to verify the performance of a final chip design before being manufactured by a foundry. They must be accurate and have high capacity in order to handle full chips efficiently. Moreover, they should provide an early and accurate understanding of the impact of parasitic elements on the overall circuit behavior.

1.2.2. Parasitic Extraction in Optimization Loops

The parasitic extraction methods that are used in layout generation and parasitic-aware layout optimization flows are called implementation-level parasitic extraction methods. Such methods must provide fast and accurate parasitic extraction results in order to obtain (or generate) an optimized layout in a reasonable runtime. There are two extraction approaches for the implementation-level parasitic extraction that include 1) simplified models and 2) incremental parasitic extraction approaches.

As for the simplified models, each route, in a given layout, is represented by a simple lumped RC Π (i.e., PI) model. The RC values of the Π model are evaluated using simplified resistance and capacitance formulas. Such an approach provides a very fast optimization; however, its accuracy is poor as compared to the accuracy requirements of advanced process nodes. [20]. As for the incremental parasitic extraction approach, it may use a sign-off layout parasitic extraction tool to extract the parasitic elements of a whole layout design before the optimization process starts. After that, once the optimization process starts, the incremental extraction approach identifies the modified layout polygons, re-extracts the corresponding parasitic elements, and update the circuit and parasitic network (i.e., netlist) accordingly. Such an approach provides more

accurate layout optimization results (in case of using a sign-off extraction tool) with a minor impact on the optimization runtime [18].

1.3. Parasitic-Aware Layout Optimization

Layouts are usually generated by using automatic layout generation and optimization tools. Such tools help layout designers in generating a layout that meets the required circuit specifications. The layout generation tools are commonly used for digital circuit designs, where cell-based tools are employed to cover circuit synthesis, mapping, and physical design steps [21]. However, layout generation tools do not provide a full automation environment for analog designs, where analog circuit designers still need to do many manual analysis and layout modifications until their analog design meets the required circuit specifications.

In analog designs, the layout optimization tools are usually used to determine device sizes, circuit topologies, and routing paths. However, they still deal with the effects of interconnect (i.e., route) parasitic elements as second order effects ignoring that the interconnect parasitic effects became one of the dominant factors on a circuit performance in advanced process nodes. In order to control the effects of interconnect parasitic elements, the corresponding routes need to be routed in a way that reduces the associated parasitic elements [22], [23]. This step is called parasitic-aware routing optimization, and it is part of the routing process.

The routing is the process of creating connections between devices. It is mainly divided into two stages that include global and detailed routing. The global routing is responsible for identifying general paths of each connection. It usually divides the routing region into windows and identifies the general window-to-window paths for all connections (i.e., routes) [24]. After that, the detailed routing is performed as it identifies exact paths, metal layers, and vias for each net in a given layout. The routing processes usually consider multiple constraints, such as maintaining net symmetry, minimizing wire lengths, having a maximum number of vias, complying with corresponding design rules, and minimizing parasitic elements [23], [25].

1.3.1. Net Symmetry Constraints

In net symmetry constraints, the layout geometrical matching is no longer enough to achieve a net symmetry as it does not necessarily provide a performance matching across the required nets. This problem significantly increases in advanced process technology nodes because layouts became more complicated and the parasitic coupling interactions with the surrounding polygons significantly increased increasing the impact of parasitic elements on circuits performance. In order to achieve the performance matching, the parasitic elements of the target nets need to be considered while applying the net symmetry constraint.

1.3.2. Parasitic Constraints

Parasitic-aware routing processes aim to reduce the parasitic elements that are mainly associated with critical routes in order to meet the required circuit specifications. This is done by modifying layout geometries of critical routes in a way that reduces the effects of associated parasitic elements. However, modifying layout geometries will not only impact the associated parasitic elements, but it will also impact the parasitic interactions among surrounding and nearby metals. Therefore, a full layout parasitic extraction is required with every geometrical change in order to accurately measure the impact of modifying routes [23]. However, the use of a full layout parasitic extraction during the optimization process would significantly increase the optimization runtime. Therefore, an incremental parasitic extraction may be considered to reduce the optimization runtime.

1.4. Problem Definition

This work tackles three main problems that include: 1) the problems of existing rule-based extraction methods; 2) the limitations of existing rule-based 2.5D extraction methods; and 3) the problems of existing parasitic-aware routing optimization methods.

1.4.1. Problems of Rule-Based Extraction Methods

The current rule-based extraction methods have three main problems that include 1) a limited pattern coverage; 2) potential pattern mismatches; and 3) a limited handling

of systematic process variations. With regards to the limited pattern coverage, the current rule-based extraction tools rely on limited pre-characterized layout patterns. Such patterns are generated using a limited number of geometrical parameters, such as widths and spacings, that are used to create corresponding parasitic capacitance formulas. Such formulas cannot handle the complicated layout patterns in recent layout designs as they do not have enough geometrical parameters to accurately represent such patterns. Therefore, detailed and multi-dimensional models are required to capture all required geometrical parameters that impact parasitic capacitances in a certain layout pattern. Regarding the potential pattern mismatch, it means that parasitic capacitances of a certain layout pattern are extracted using inappropriate capacitance formulas. This results in extracting wrong parasitic capacitance values. There is a tradeoff between pattern coverages and pattern mismatches, where increasing the number of pre-characterized patterns increases the probability of pattern mismatches. As for systematic process variations, they represent physical variations in layout interconnects and devices. Such variations are layout-dependent, and they mainly occur during layout manufacturing processes. The most common systematic variations of interconnects include metal thickness variations, loading effects (i.e., inter layer dielectric thickness variations), metal width variations (e.g., etching), and trapezoidal variations (i.e., sidewall slope of metals). The impact of such variations on parasitic capacitances significantly increased in advanced process nodes, where the dimensions of metal wires are smaller, and systematic variations started to represent considerable portions of metal dimensions. Therefore, layout parasitic capacitance extraction processes must consider systematic process variations in order to provide accurate parasitic netlists [10], [26].

The current rule-based extraction tools handle the impact of systematic process variations on parasitic capacitances independently using sensitivity formulas that represent the sensitivity of a certain capacitance component to a certain variation parameter. Such formulas are pre-characterized with limited geometrical parameters [27]–[29]. Therefore, they also suffer from potential pattern mismatch and limited

pattern coverage problems. To consider systematic process variations during the parasitic capacitance extraction, each capacitance component is calculated using a single capacitance formula and multiple sensitivity formulas. This way of handling systematic variations neglects the cross-dependency impact of different variation parameters on parasitic capacitances, where the capacitance sensitivity to each variation parameter is calculated independently while keeping other parameters fixed. Moreover, the computational runtime of capacitance calculations significantly increased due to using multiple pre-characterized formulas to calculate a single capacitance component.

1.4.2. Limitations of Rule-Based 2.5D Extraction Methods

The current extraction techniques cannot cope with the continuous scaling down and complexity of advanced technologies. While field-solvers can provide the required high accuracy levels, they are very slow and have a limited capacity. On the other hand, the errors of rule-based 2.5D extraction methods started to explode with the increase of accuracy requirements at newer nodes [11], [17], [30]. The main reason behind such inaccuracy is that the rule-based methods use simplified 2D vertical cross-section models to calculate parasitic capacitances. Such models have an incomplete view of the surrounding environment (e.g., metal wires). This results in errors in the extracted parasitic capacitances and negligence of many 3D fringing capacitance components. Such a problem is a limitation of the 2.5D extraction method. Most of the recent efforts only focused on improving the prediction accuracy of rule-based models, as in [31], and they did not provide a solution to overcome the limitations of the 2.5D extraction method. Fig. 1.3 shows the error margin of rule-based 2.5D parasitic capacitance extraction methods, as compared to field-solvers, after performing many experiments across different real designs using 28nm process node. The results show that the errors are exceeding 15% at many data points.

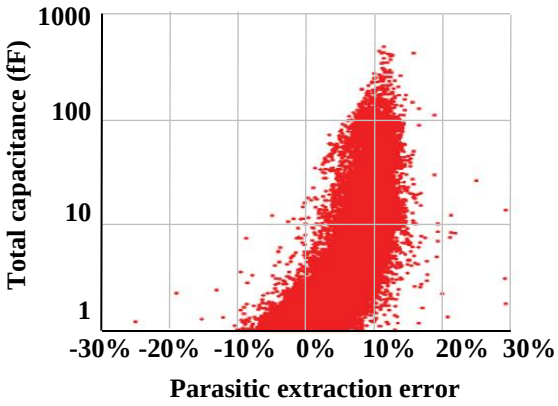


Fig. 1.3. Errors of parasitic capacitance extraction using typical rule-based 2.5D capacitance extraction tools as compared to a 3D field-solver, Calibre xACT3D [32], over many 28nm designs.

Hybrid parasitic capacitance extraction methods that combine rule-based and field-solvers are considered in advanced nodes. Despite the accuracy improvements introduced by those methods, they suffer from two main problems. First, the proportion of patterns that have to go to field-solvers increases in advanced nodes (as technology scales down). With this proportion approaching 50%, using these hybrid methods does not save much time as compared to field-solvers. Second, these hybrid methods do not eliminate all outliers because the layout patterns are assigned to extraction methods based on a pre-characterized library. To improve the accuracy and runtime of the hybrid parasitic capacitance extraction, a new adaptive, accurate, and faster intermediate extraction method is required to replace field-solvers in extracting most of layout patterns. Furthermore, a smarter way to direct each layout pattern to an appropriate extraction method based on the required accuracy is needed.

1.4.3. Problems of Existing Parasitic-Aware Layout Optimization Flows

The current optimization flows do not deal with the parasitic effects as dominant factors. They are still dealing with the parasitic effects as second order effects. Moreover, current flows do not provide proper layout analysis and debugging methodologies to help circuit designers in identifying the problematic parasitic elements and the corresponding layout geometries. As a result, circuit designers need to manually analyze the impact of interconnect parasitic elements on a circuit

performance, which is a very time-consuming and error-prone operation. Nowadays, the time consumed in analyzing the post-layout simulation results is more critical than post-layout simulation runtime itself. Therefore, there is an increasing demand to provide algorithms that help circuit designers in understanding the impact of parasitic elements on post-layout simulation results and identifying the most problematic parasitic elements along with the corresponding layout geometries in a given layout. The problems of existing layout optimization (i.e., routing optimization) methods can be summarized as below:

a. They do not provide debugging and analysis methodologies that help circuit designers in understanding the impact of parasitic elements on a system's (i.e., route) performance, such as identifying the problematic parasitic elements along with the associated geometries.
b. Most of existing efforts are customized to simple parasitic formulas that cannot cope with the accuracy requirements by semiconductor foundries in advanced process technology nodes (< 5% error). Also, they are not designed to use complicated parasitic networks that are generated from sign-off parasitic extraction tools, such as Calibre PEX [19] and STARC [33]. On the other hand, few methods perform a full layout extraction at each design iteration in order to improve the optimization accuracy; however, such methods consume a lot of time and are not suitable for large designs.
c. Many of existing layout optimization flows rely on multiple circuit simulations in order to identify the parasitic bounds for each parasitic element; however, such methods consume a lot of time in circuit simulations, especially for large designs.
d. Existing efforts do not help circuit designers in understanding the impact of parasitic elements on a system's (e.g., route) performance.

1.5. Contributions

This work mainly focuses on providing innovative solutions to improve the accuracy of 1) the interconnect parasitic capacitance extraction and 2) the parasitic-aware layout

routing optimization method. The contributions of this work cover several parasitic capacitance extraction methods that include rule-based 2.5D, MEOL, and hybrid parasitic capacitance extraction methods. Moreover, the contributions include developing a new parasitic-aware routing optimization methodology that can cope with the increasing impact of parasitic elements in advanced process nodes. The contributions are as follows.

1.5.1. Rule-Based 2.5D Capacitance Extraction Models

The contributions of the rule-based 2.5D extraction focuses on implementing a new interconnect parasitic capacitance compact models for 2D cross-section layout patterns in rule-based 2.5D extraction tools. The new models use novel input pattern representations that considers systematic process variations efficiently. The new models are compact, have high pattern coverage, mitigate pattern mismatches, and provide a faster layout parasitic capacitance extraction process. Also, the proposed compact models can replace thousands of existing capacitance and sensitivity formulas, where each model can calculate a coupling capacitance between two certain polygons using a single computation instead of multiple computations (using multiple capacitance and sensitivity formulas) in traditional rule-based methods. The contributions of rule-based 2.5D extraction method are summarized below:

a. Machine learning compact models that predict interconnect parasitic capacitances in layouts of a certain process node are proposed. Each model predicts parasitic coupling capacitances of cross-section patterns covering a certain set of metal layers with arbitrary distributed polygons considering systematic process variations. Unlike existing models that require multiple computations to calculate a capacitance component, the compact models can calculate a certain capacitance component using a single computation. Therefore, there is no need to invoke multiple capacitance formulas to calculate a certain capacitance component anymore.
b. Three different representations of 2D cross-section layout patterns are proposed. The proposed representations include ratio-based, dimensions-based, and vertex-

based 2D cross-section representations. The ratio-based and dimensions-based representations consider the impact of metal width variations, whereas the vertex-based representation accounts for systematic process variations including metal thickness variations, loading effects, metal width variations, and trapezoidal variations.

c. The compact models are generated using two different machine learning methods including Neural Networks (NN) and Support Vector Regression (SVR) methods.
d. The proposed methodology is tested over test chips of 28nm, 14nm, and 7nm process nodes with more than 6.7M interconnect cross-section patterns. The results show that the proposed methodology provided outstanding accuracy as compared to field-solvers and rule-based models with an average error < 0.15% and a standard deviation < 3.3%, whereas the average errors and standard deviations of rule-based models exceed 6%, for the same test chips. Also, the computational runtimes of the proposed compact models are almost 2.5X faster than rule-based models.

1.5.2. MEOL Rule-Based Capacitance Extraction Models

The contributions of MEOL parasitic capacitance extraction focuses on developing a new modeling methodology that can generate machine learning compact models to predict parasitic capacitances of MEOL patterns as below:

a. A new modeling methodology based on machine learning methods is developed to predict parasitic coupling capacitances for MEOL structures in rule-based extractors. The methodology provides accurate and compact parasitic capacitance models for MEOL around the devices for a certain process technology node. The generated models significantly improve the parasitic capacitance extraction accuracy of MEOL patterns by increasing the pattern coverage, considering metal connectivity, and mitigating pattern mismatches. The generated models are compact, where each model can handle many MEOL patterns with different

arrangements. Each compact model can replace thousands of pre-characterized patterns and provide more accurate results.

b. A novel geometry-based pattern representation is proposed to represent the MEOL input patterns.
c. Two machine learning approaches are used and tested to implement the MEOL parasitic capacitance models including Neural Networks (NN) and Support Vector Regressions (SVR).
d. The testing covered more than 40M devices of several different real designs that belong to 28nm and 7nm process technology nodes. The proposed methodology managed to provide outstanding results as compared to field-solvers with an average error < 0.2%, a standard deviation < 3%, and a speedup of 100X.

1.5.3. Accuracy-Based Hybrid Parasitic Capacitance Extraction Method

As for the hybrid parasitic capacitance extraction, A novel hybrid parasitic capacitance extraction flow is developed. The proposed hybrid flow can alternate between three extraction methods: field-solver, rule-based, and novel Deep Neural-Networks (DNN) based extraction methods. The novel DNN-based extraction is considered to be an intermediate method that is order of magnitudes faster than field-solvers. Also, it provides high accuracy values as the hybrid extraction flow can find a faster alternative to field-solvers to extract most patterns. Moreover, an accuracy-based neural-networks classifier is introduced to efficiently assign each layout pattern to the fastest extraction method that meets the user predetermined accuracy requirements. The contributions are summarized as below:

a. A novel DNN-based parasitic capacitance extraction method intended as an intermediate method between field-solver and rule-based methods in terms of performance and accuracy. This intermediate level of accuracy and speed is needed since using only rule-based and field-solver methods (for hybrid extraction) results in using field-solver most of the time for any required high accuracy extraction.

b. A novel accuracy-based hybrid parasitic capacitance extraction flow that allows the user to determine the required accuracy level. This flow divides the layouts into windows and extracts the parasitic capacitances of each window using one of three parasitic capacitance extraction methods that include: 1) rule-based; 2) novel neural-networks-based; and 3) field-solver methods.
c. A smart layout patterns classifier that assigns layout patterns to the fastest extraction method (field-solver, new DNN-based method, or rule-based) that meets the required accuracy requirement.
d. The testing covered four designs of 28nm and 7nm process nodes. The results show that the proposed DNN-based extraction method extracts capacitances of complicated structures with high accuracy (< 3% average error) and 100X faster than field-solvers. However, few outliers have an error exceeding 5% in extracted capacitances. Furthermore, the proposed hybrid flow managed to meet the required accuracy (< 5% error) with 99% accuracy and 70X faster than field-solvers.

1.5.4. Parasitic-Aware Layout Analysis and Routing Optimization Methodology

A new parasitic-aware re-routing optimization method using circuit moments is proposed. The proposed methodology enables circuit designers to debug and analyze the impact of parasitic elements on a circuit performance. Also, the proposed method provides a mechanism to identify the problematic parasitic elements and correlate them with specific layout geometries. Moreover, it uses nonlinear programming to re-route the problematic paths (i.e., routes) in order to achieve the required specifications with full consideration of the surrounding environment. The proposed methodology is very efficient with net symmetry constraints and parasitic-aware re-routing. The contributions can be summarized as below:

a. Circuit models to measure and analyze the impact of parasitic elements and corresponding layout geometries on a pre-defined cost function, such as net

symmetry and maximum delay cost functions. In other words, they measure the sensitivity of system's performance cost function to layout geometries.

b. The proposed models are used in an algorithm that identifies the geometries and parasitic elements that the system's performance is most sensitive to without any circuit simulations.
c. A parasitic-aware re-routing optimization algorithm that uses nonlinear programming to automatically modify the most critical routes in order to meet the required performance cost function. The proposed algorithm accepts pre-determined degrees of freedom (e.g., route's corners) and dynamic constraints.
d. A novel incremental parasitic extraction methodology that considers second order parasitic capacitance effects efficiently. The proposed incremental methodology is applied on top of a full layout parasitic extraction tool, Calibre PEX [19]. It provides very accurate parasitic extraction results with a maximum error < 1% and a speedup of up to 40X as compared to a full layout extraction.
e. The proposed methodology is tested on different designs of 7nm and 65nm process nodes.

1.6. Organization

This dissertation is organized as follows. Chapter 2 provides a background on different parasitic capacitance extraction methods, the impact of systematic process variations on parasitic capacitances, and the parasitic capacitance extraction in advanced process technology nodes, and the parasitic-aware template-based layout optimization method. Chapter 3 provides the related works. Chapter 4 describes the proposed 2D cross-section interconnect parasitic capacitance models. Chapter 5 describes the proposed MEOL parasitic capacitance models. Chapter 6 describes the proposed hybrid parasitic capacitance extraction method. Chapter 7 describes the proposed parasitic-aware layout routing optimization method. Chapter 8 provides the conclusion. Chapter 9 presents the future work.

Chapter 2

Background

This chapter provides background on relevant topics to parasitic capacitance extraction and parasitic-aware routing optimization methods. It is organized as follows. Section 2.1 explains the rule-based 2.5D parasitic capacitance extraction method. Section 2.2 discusses the systematic process variations. Section 2.3 illustrates the different field-solver methods. Section 2.4 provides the main challenges of the parasitic capacitance extraction in advanced process nodes. Section 2.5 discusses the template-based layout optimization method and provides a brief discussion on system moments.

2.1. Rule-Based 2.5D Capacitance Extraction

Rule-based parasitic capacitance extraction methods are used in several commercial extraction tools, such as Calibre PEX [19] and StarRC [33], because they can handle full chips efficiently. Rule-based methods employ 2.5D extraction approaches in order to extract interconnect parasitic capacitances of a given layout. In 2.5D approaches, parasitic extraction tools scan a given layout in the x and y directions to obtain all corresponding 2D cross-section layout patterns. For each cross-section pattern, plate and fringing coupling capacitances (per unit length) are calculated using pre-characterized capacitance formulas [11]. The mapping between cross-section patterns and the corresponding capacitance formulas is performed using pattern matching operations. Once all capacitances are calculated, they are multiplied by the corresponding projection length to get the total capacitance values.

Fig. 2.1 provides an illustrative example of extracting a certain metal polygon using the 2.5D extraction approach. The figure shows a layout structure of three metal layers including metal1, metal2, and metal3 layers. The target metal polygon is the middle metal2 polygon. There are four cross-sections for the target metal polygon. Three cross-sections are in the z-y plane, and one cross-section is in the z-x plane. Cross-section1 ($C1$)

and crosssection2 (*C*2) are identical, and each contains five capacitance components. Cross-section3 (*C*3) contains eight capacitance components. Cross-section4 (*C*4) contains six capacitance components. The fringing and lateral capacitances are calculated for each cross-section using corresponding capacitance formulas. Then, each capacitance component is multiplied by the corresponding projection length (i.e., *L*1 to *L*4). As for plate capacitances, they are calculated in one cross-section, either *z*-*x* or *z*-*y* cross-sections, and multiplied by the corresponding projection length. This is done to avoid duplicate calculations of the same plate capacitance.

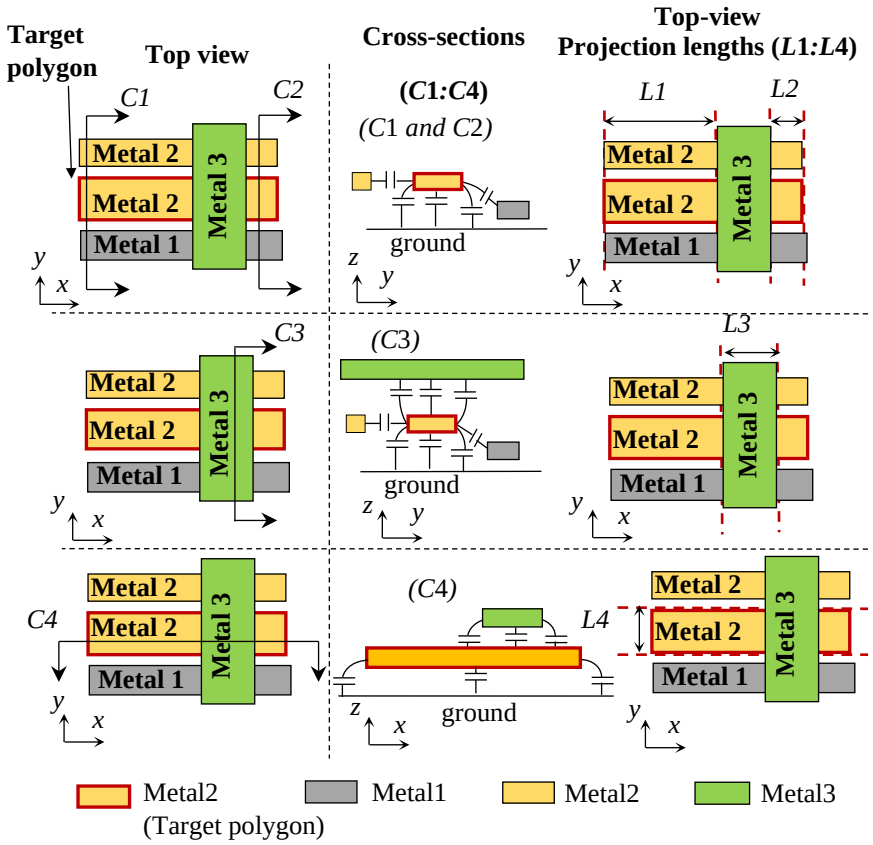


Fig. 2.1. An illustrative example of extracting a metal polygon using the 2.5D parasitic capacitance extraction method.

The rule-based extraction method has two main steps: 1) a pre-characterization (i.e., calibration) step, as shown in Fig. 2.2 (a); and 2) a layout parasitic capacitance extraction step, as shown in Fig. 2.2 (b). The pre-characterization (i.e., calibration) step is responsible for generating a pre-characterized library of capacitance and sensitivity formulas, where each process technology node has a different pre-characterized library. On the other

hand, the layout parasitic extraction step is responsible for analyzing layouts and calculating corresponding parasitic elements using the corresponding pre-characterized library.

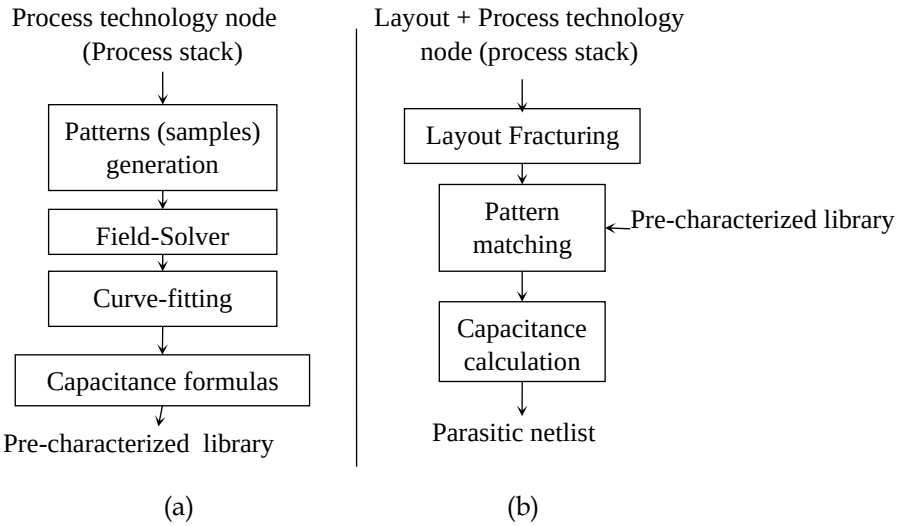


Fig. 2.2. Rule-based parasitic capacitance extraction steps including (a) pre-characterization and (b) layout parasitic extraction steps.

2.1.1. The Pre-Characterization Step

In this step, a pre-characterized library of capacitance and sensitivity formulas are generated for a certain process technology node. The pre-characterization process starts with generating many 2D and 3D layout patterns based on the corresponding technology specifications. The structures of those patterns are pre-characterized. Then, a field-solver tool is used to extract reference parasitic capacitance values for each layout pattern. The reference capacitance numbers are either formatted in lookup tables or passed to a curve fitting tool. The curve fitting tool generates a capacitance formula for each capacitance component as below:

$$C = f(p1, p2, \dots), \tag{2.1}$$

where C represents a certain capacitance component, f ($p1$,$p2$,..) represents the curve fitted capacitance formula, whereas p represents a certain geometrical parameter (e.g., width or spacing). Moreover, sensitivity formulas are generated to measure the impact of systematic process variations on each capacitance component, where each

capacitance component is calculated using a single capacitance formula and multiple sensitivity formulas as below:

$$C = f(p1, p2, \dots) + \sum_{i=0}^{n} \Delta S_i \cdot \frac{\partial C}{\partial S_i}, \quad (2.2)$$

where S represents a certain variation parameter (e.g., a metal thickness variation), $\partial C/\partial S$ represents a sensitivity formula that measures a capacitance sensitivity to a certain variation parameter, whereas n represents the number of systematic process variation parameters. Eventually, the generated capacitance and sensitivity formulas are stored in a pre-characterized library in order to be later used by parasitic capacitance extraction tools [34].

2.1.2. Layout Parasitic Capacitance Extraction Step

The layout parasitic capacitance extraction step is responsible for extracting parasitic capacitances of a given layout and writing the extracted parasitic elements into a parasitic netlist. The extraction flow starts with analyzing and measuring the geometries of a layout. After that, layout geometries are fractured into 2D cross-section patterns as shown in Fig. 2.1. Then, a pattern matching operation is performed to match each 2D cross-section pattern with corresponding pre-characterized capacitance and sensitivity formulas. Eventually, the measured geometries are passed to the obtained pre-characterized formulas to calculate the corresponding capacitance values. Once all parasitic capacitances are extracted, a parasitic netlist is generated to be later used by circuit simulators to perform post-layout simulations [34].

2.2. Systematic Process Variations

As process technology nodes scale down, the dimensions of metal wires continue to shrink, and the difficulty of controlling the variations of interconnect geometries and device parameters significantly increased [35]. There are two types of variations including random and systematic variations. Random variations represent the unpredictable and stochastic variations that cannot be associated with specific conditions

or layout patterns. They might change from time to time and from location to another. The random variations are usually modeled using statistical models as in [36]–[39].

On the other hand, systematic variations represent the predictable and deterministic variations that are associated with specific process conditions (e.g., chemical mechanical polishing) and layout patterns. In advanced process technology nodes, the impact of systematic variations on parasitic capacitances increases because systematic variations represent higher percentages of interconnect and device dimensions [26], [27], [40]. The main systematic process variations include metal thickness variations, inter layer dielectric thickness variations (i.e., loading effects), metal width variations (e.g., etching), and trapezoidal variations of metal layers as shown in Fig. 2.3. Fig. 2.3 (a) shows examples of metal thickness variations and inter layer dielectric thickness variations (i.e., loading effects). The loading effects mainly impact the thickness of inter layer dielectrics and the elevation of the corresponding upper metal layers, whereas the metal thickness variations mainly impact the top thickness of the corresponding metal layer. Fig. 2.3 (b) shows an example of trapezoidal variations in metal layers, where the sidewall slope of a certain metal layer changes. Fig. 2.3 (c) shows an example of a metal width variation that impacts the width of metals and the separation between them. Since systematic variations are pattern dependent, parasitic capacitance extraction tools usually model their effects using sensitivity formulas as in [27]–[29]. Such a modeling approach has three main problems that impact the extraction accuracy: 1) it neglects the cross-dependency impact among different variation parameters on parasitic capacitances; 2) it uses a limited number of patterns and parameters to model the impact of systematic variations on parasitic capacitances; and 3) it has a high potential of pattern mismatches similar to the case of capacitance calculations (i.e., formulas). Moreover, the current handling of systematic variations introduces extra computational runtime, where each capacitance component is calculated using a single capacitance formula and multiple sensitivity formulas as shown in (2.2). Fig. 2.4 shows an example of the cross-dependency between the sidewall slope (θ°) of metal1 layer and metal1 lateral capacitance sensitivity to metal1 thickness ($\partial C_l/\partial t$). The experiment used metal1 layer of 28nm process node.

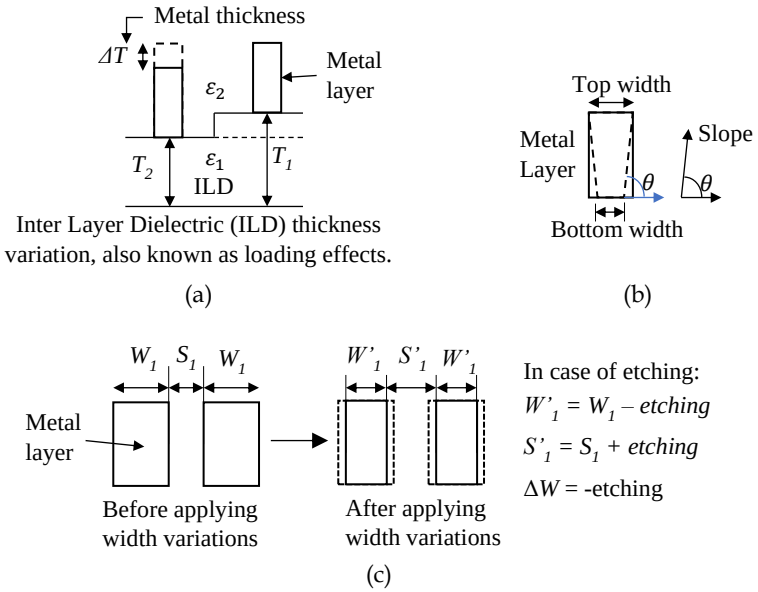


Fig. 2.3. Examples of systematic process variations showing (a) metal thickness variation and loading effects, (b) trapezoidal variations, and (c) metal width variations.

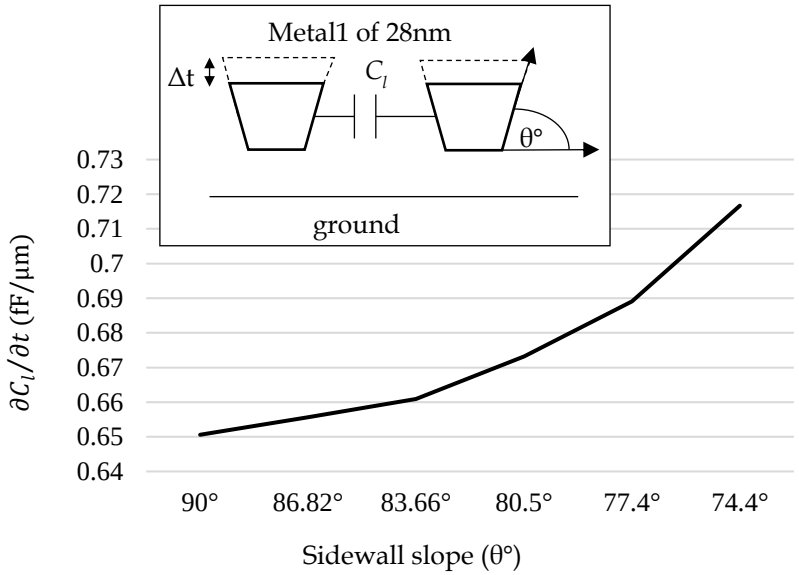


Fig. 2.4. An example of the cross-dependency between a sidewall slope (θ°) of a metal layer and a lateral capacitance sensitivity to a metal thickness ($\partial C_l / \partial t$). The experiment used metal1 layer of 28nm process node.

2.3. Field-Solvers

Numerical methods (field-solvers) are the most accurate techniques to calculate the parasitic capacitances. They simulate electrostatic fields among different metal polygons in integrated circuits to capture the coupling capacitances among different metal

polygons. Assume M metal polygons arbitrarily distributed. The parasitic coupling capacitances among different polygons can be calculated by:

$$Q_i = \sum_{j=1}^{M} C_{ij} * V_i \,, \tag{2.3}$$

where Q_i is the electric charge on a certain metal polygon, C_{ij} is the coupling capacitance between two different metal polygons, and V_i is the electric potential on a certain polygon; however, solving this formula is not trivial, especially with complicated structures and multiple dielectrics. Laplace equation helps in calculating the parasitic capacitances in multi-dimensional structures. For each homogenous dielectric region, the electric potential V is governed by Laplace equation as below:

$$\nabla^2 V = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = 0, \tag{2.4}$$

There are several numerical methods that is used to solve Laplace equation. They can be classified into three main classes: discretization methods, integral methods, and stochastic methods.

2.3.1. Discretization Methods

The discretization methods include finite difference methods (FDM), finite element methods (FEM), and finite volume methods (FVM) [14]. The discretization methods are used to solve Laplace equation by discretizing the whole domain and generating a high order linear system. This provides very accurate results but with a very limited computational runtime.

2.3.2. Integral Methods

The integral methods include methods of moments (MOM), direct boundary element methods (DBEM), and indirect boundary element methods (IBEM) [14]. In the integral methods, only the boundaries of the domain are discretized.

2.3.3. Stochastic Methods

The stochastic methods include floating random walk algorithms (FRW). Such methods are useful for very large size applications. The main idea behind the stochastic methods is to calculate conductor charges by Monte Carlo (MC) integrations using random walks [14], [41], [42].

2.4. Parasitic Capacitance Extraction in Advanced Process Technology Nodes

With the continuous down scaling in advanced nodes, metal interconnects are shrunk, and more metal layers are introduced to provide more routing paths. This introduced three main challenges to the parasitic capacitance extraction accuracy. First, the accuracy requirements increased significantly due to the increasing impact of interconnect parasitic effects on circuit performances. Second, the mask misalignments (due to double and multi-patterning) and lithography effects cause more geometrical variations that impact parasitic capacitances. Third, the impact of surrounding metals on parasitic capacitances significantly increased due to the increased density and shrunk dimensions of interconnects (e.g., 3D fringing coupling capacitances [3], [5], [6], [10], [30].

Traditional 2.5D extraction methods cannot cope with the new accuracy requirements by semiconductor foundries (< 5% error) in advanced process nodes because they suffer from three main problems. First, the 2.5D extraction method only considers the capacitance interactions and arrangements of metal polygons in *z-x* and *z-y* planes. Therefore, many 3D fringing capacitance interactions are neglected as shown in Fig. 2.5. Moreover, many surrounding metal polygons that impact parasitic capacitances are not considered because they are not detected in either *z-x* or *z-y* planes. Second, the pattern mismatches, which occur when parasitic capacitances of a layout pattern are extracted using inappropriate capacitance formulas, increase due to the increased complexity of layout patterns. Third, the pattern coverage is limited, where the pre-characterized layout patterns and capacitance formulas are not enough to efficiently handle the more diverse layout patterns in advanced nodes.

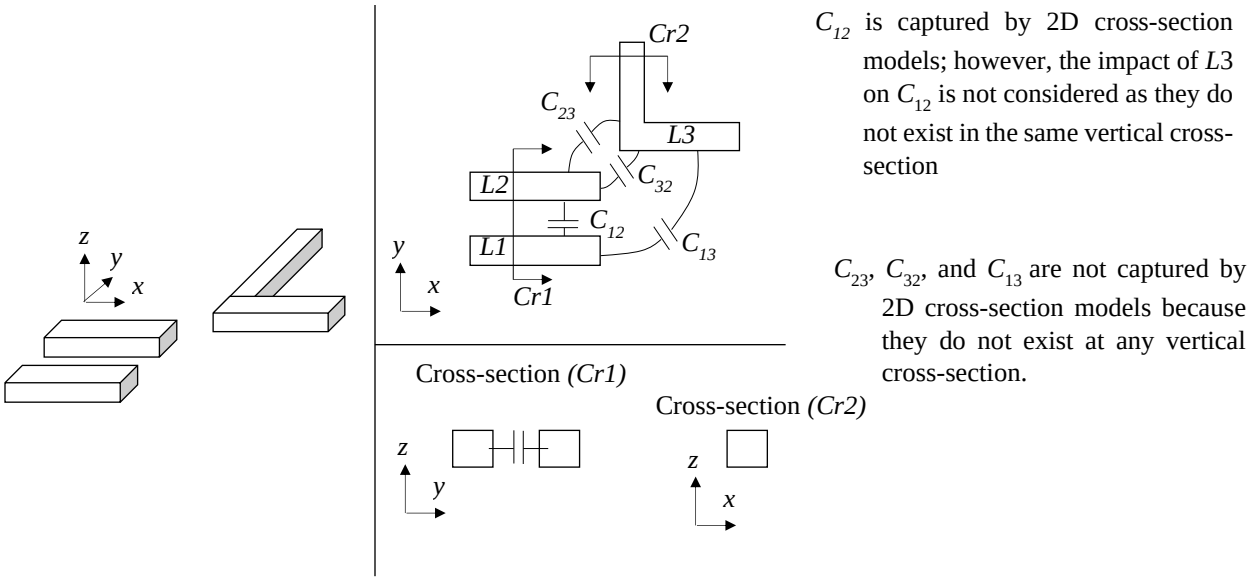


Fig. 2.5. An illustrative example of the limitations of existing 2D cross-section models of 2.5D extraction method.

The proposed DNN-based extraction method, in Chapter 6, can handle layout patterns of advanced process nodes efficiently because of four main reasons. First, the proposed method provides compact parasitic capacitance models that extract the 3D fringing capacitances efficiently. Second, the proposed method improves pattern coverage because each compact model handles many diverse layout patterns with arbitrary distributed polygons. Third, the proposed method mitigates pattern mismatches as patterns of a certain layer combination are extracted using a single compact model. Fourth, it considers different lithography effects, such as double and multi-patterning, as the inputs to DNN models use layout polygons of effective actual dimensions (not drawn dimensions) as shown in Fig. 2.6. However, some foundries prefer to model mask misalignments by changing dielectric constants of surrounding dielectrics [6]. In such a case, the proposed method handles that by representing each mask in a separate layer.

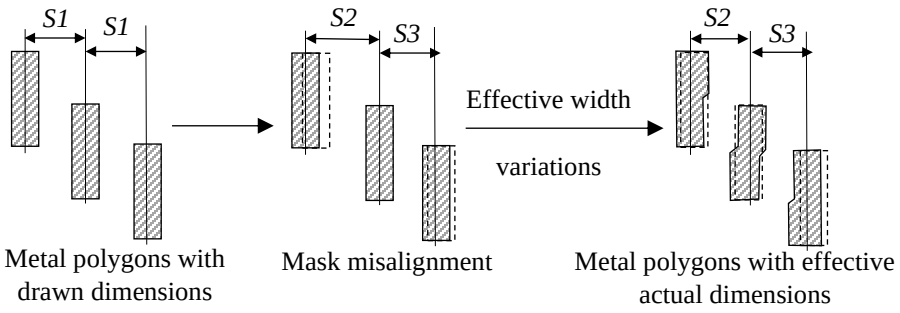


Fig. 2.6. An example of three metal polygons showing the impact of applying mask misalignments and effective line width variations on drawn dimensions.

2.5. Layout Optimization and System Moments

This section explains the template-based parasitic-aware layout optimization highlighting the importance of parasitic extraction during the optimization loops. Moreover, a brief discussion on system moments is introduced.

2.5.1. Template-Based Parasitic-Aware Layout Optimization

A layout optimization is the process of modifying and optimizing layout designs in order to meet the required circuit specifications. One of the most efficient layout optimization methods is the template-based method. The template-based method is used to either migrate a layout design from one process node to another or optimize an existing layout to meet the required constraints and specifications. It consists of two main steps that include symbolic template extraction and layout generation steps as shown in Fig. 2.7. The symbolic template extraction step is responsible for extracting a symbolic template of an existing layout design, whereas the layout generation step is responsible for generating and optimizing an existing layout by using the obtained symbolic template.

2.5.1.1. Symbolic Template Extraction:

This step is responsible for generating a set of geometrical and electrical constraints (i.e., symbolic template) of an existing layout considering the required circuit specifications. It starts with extracting transistors, devices, and nets from an existing (i.e., original) layout. Then, the constraints of an existing layout are extracted, translated into equations or inequalities, and included in the corresponding symbolic

template, for example, the connectivity of interconnects, corresponding design rules, proximity, and symmetry constraints [20], [22], [43], [44]. On the other hand, a corner stitching is used, as a data structuring technique [45], in order to help in performing the different geometry operations. It is worth mentioning that the symbolic template is usually represented by mathematical formulas (e.g., compaction formulas) such as in Fig. 2.8.

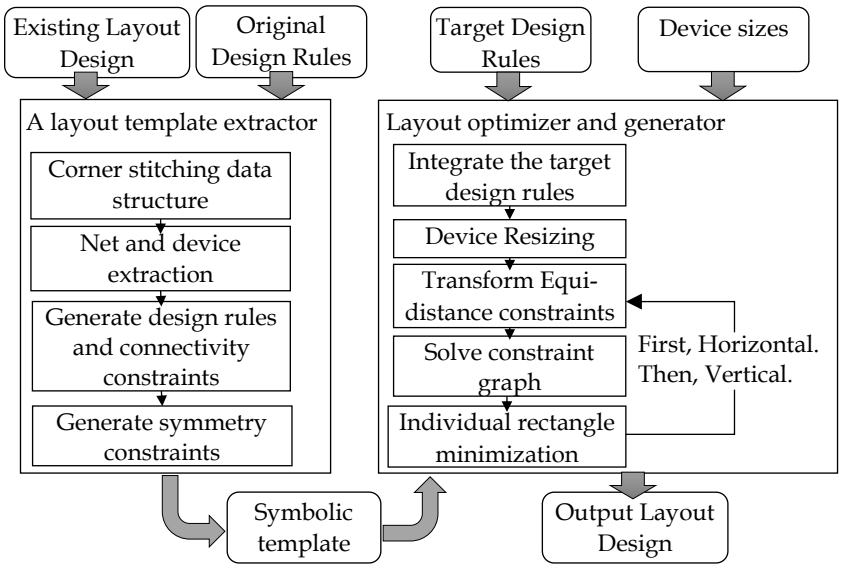


Fig. 2.7. Template-based layout optimization flow [20], [44].

2.5.1.2. Layout Generation and Optimization

In this step, an existing layout is optimized using the obtained symbolic template constraints and the new design requirements in order to achieve the required specifications. The optimization constraints include geometrical (e.g., geometrical symmetry) and electrical (e.g., parasitic constraints) constraints [20], [22], [43], [44]. The layout optimization step starts with a device sizing followed by a routing optimization, where the routing optimization is performed in the horizontal and vertical directions separately.

2.5.1.3. Geometrical Constraints

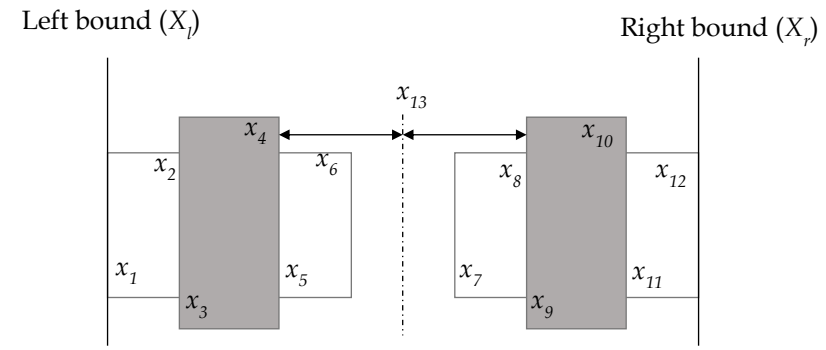
The constraints that are used in the optimization processes, excluding the parasitic effects, can be expressed by compaction formulas and solved in the vertical

and horizontal directions separately [44], [46]. For example, let X_r represents the right most end of a target layout, whereas X_l represents the left most end of the same target layout. Therefore, the compaction formulas in the x-direction are expressed by:

$$\begin{aligned} &\min(X_r - X_l) \\ \text{subject to} \quad &(x_i - x_j) \geq \text{LOB}, \\ &(x_i - x_j) = \text{EXB}, \\ &(x_i - x_j) = (x_k - x_l), \end{aligned} \tag{2.5}$$

where LOB is a constraint that represents geometrical lower bounds (e.g., polygons minimum dimensions), EXB is a constraint that represents exact bounds, whereas x variables represent either the symmetry axes or the right and left edges of layout polygons. The constraints should include and describe the properties that are needed to meet the required specifications, such as design rules, new device size specifications, and crosstalk minimization [44].

Fig. 2.8 shows an example of a simple layout with 13 horizontal coordinates (x_1 to x_{13}), given that x_{13} represents the symmetry axis. The formulas that are shown in Fig. 2.8 represent the geometrical constraints that can be used in the corresponding routing optimization process. The highlighted constraints represent the corresponding design rules, a layout symmetry, and minimum dimensions. For example, $(x_{12} - x_{11}) \geq 2$ aims to set the minimum width of a certain polygon to 2µm, whereas $(x_4 - x_3) = (x_{10} - x_9)$ aims to maintain the geometrical symmetry of the given two structures.



$\min(X_r - X_l)$, subject to:

$(x_1 - X_l) \geq 0, (x_3 - X_l) \geq 0, \ldots$

$(X_r - x_{12}) \geq 0, (X_r - x_{10}) \geq 0, \ldots$

$(x_2 - x_1) \geq 2, (x_{12} - x_{11}) \geq 2, \ldots$

$(x_3 - x_2) = 0, (x_{11} - x_{10}) = 0, \ldots$

$(x_4 - x_3) = (x_{10} - x_9), (x_{13} - x_4) = (x_9 - x_{13}), \ldots$

Fig. 2.8. An Example of template geometrical constraints for a simply layout in the x-direction [44].

2.5.1.4. Parasitic Constraints

The layout optimization processes must consider the impact of parasitic elements on a circuit performance in order to achieve more accurate optimization results. Therefore, layout parasitic elements must be extracted and reduced in order to meet the required circuit specifications efficiently. Since layout parasitic elements are highly correlated with the corresponding layout geometries, parasitic constraints can be converted into geometrical constraints. In other words, the parasitic constraints can be represented by functions of x and y coordinates of the corresponding metal polygons. This is usually performed in two steps. First, the bounds of parasitic elements that ensure the circuit performance are identified using circuit simulations. Therefore, the parasitic constraints as functions of the corresponding geometrical coordinates are given by:

$$f(x, y) \leq \text{UPB}, \tag{2.6}$$

where UPB is an upper bound of a certain parasitic element. Second, the corresponding geometrical constraints are defined in both x and y directions as below:

$$(x_i - x_j) \geq LB, \qquad (y_i - y_j) \geq LB, \tag{2.7}$$

$$(x_i - x_j) \leq UB, \qquad (y_i - y_j) \leq UB, \tag{2.8}$$

$$(x_i - x_j) = (x_k - x_l), (y_i - y_j) = (y_k - y_l), \tag{2.9}$$

where LB represents a geometrical lower bound, and UB represents a geometrical upper bound. Such geometrical bounds are obtained from the corresponding parasitic bounds as shown in (2.6). Therefore, (2.7) and (2.8) represent the lower and upper geometrical bounds in x and y directions, whereas (2.9) represents the matched interconnect that are required to achieve identical parasitic elements. Once all required geometrical and parasitic constraints are obtained, the optimization (i.e., searching) process is performed to achieve a layout that meets the required specifications.

The main challenges of existing parasitic-aware layout optimization processes are:

a. The calculations of parasitic bounds require multiple circuit simulations that consume a lot of time. Moreover, the optimization process requires the calculations of a circuit performance sensitivity to each parasitic element, which consume a lot of time as the sensitivities are usually calculated using multiple circuit simulations.
b. The relationship between parasitic elements and layout geometries is nonlinear, which complicates the optimization process [22], [44].
c. Usually, the compaction formula is solved in the x and y directions separately (one after another). Such a way of handling the compaction formula cannot provide efficient results when it comes to the nonlinearity of parasitic constraints. Moreover, it does not consistently provide the required solution, and it may require manual modifications afterwards. On the other hand, the efforts that solve

the compaction formulas in the x and y dimensions simultaneously did not handle the situations of non-Manhattan shapes such as in [22], [44].

d. Most of existing efforts rely on simple and lumped parasitic formulas that cannot cope with the accuracy requirements of advanced process technology nodes.
e. Existing efforts do not help circuit designers in understanding the impact of parasitic elements on system (e.g., route) performance.

The impact of such problems in parasitic-aware optimization flows significantly increases in advanced process nodes, where the interconnect parasitic effects dominate the overall circuit performance.

2.5.2. System Moments

Assuming an RC linear circuit, the corresponding general nodal analysis equations are given by:

$$G\,\underline{V} + C\,\dot{\underline{V}} = \underline{b}, \tag{2.10}$$

where G is an $n \times n$ admittance matrix that is obtained from the interconnections among the resistive elements, C is an $n \times n$ capacitance matrix that is obtained from the interconnections among the capacitive elements, $\underline{b}$ is a vector of size n that represent the inputs at each node, $\underline{V}$ is a vector with n state variables that represent the capacitor voltages (i.e., voltage response at each node), whereas n represent the number of nodes (or capacitor voltages) for a linear system with RC elements. The response, $V(s)$, at any node in a given linear circuit can be expressed by a Taylor series expansion as below [47]:

$$V(s) = m_0 + m_1 s + m_2 s^2 + m_3 s^3 + \cdots, \tag{2.11}$$

where m_i represents the i^{th} moment of a given linear system at a given node.

Substitute (2.11) in (2.10), we get:

$$G[m_0 + m_1 s + m_2 s^2 + \cdots] + C\,s[m_0 + m_1 s + m_2 s^2 + \cdots] = \underline{b}\,. \tag{2.12}$$

Equating the coefficients of s^n in both sides of (2.12), we get [47]:

$$G\,\underline{m_0} = \underline{b}$$

$$\underline{m_0} = G^{-1}\,\underline{b}$$

$$\underline{m_1} = G^{-1}\,C\,\underline{m_0}$$

$$\underline{m_2} = G^{-1}\,C\,\underline{m_1}$$

$$\vdots$$

$$\underline{m_n} = G^{-1}\,C\,\underline{m_{n-1}}. \qquad (2.13)$$

Therefore, the moments of a linear system provide a detailed representation of its response (i.e., a system response) as shown in (2.11), and system moments can be obtained by (2.13) [47].

2.6. Summary

This chapter provides backgrounds on 1) rule-based 2.5D parasitic capacitance extraction highlighting the main limitations of such an extraction method; 2) different field-solver methods; 3) the main systematic process variations; 4) the challenges of parasitic capacitance extraction in advanced process nodes; and 5) the template-based parasitic-aware layout optimization method.

Chapter 3

Review of and Feature Comparison to Related Previous Work

This chapter discusses state-of-the-art related work of parasitic capacitance extraction and parasitic-ware layout optimization methods. Moreover, it summarizes the main differences between our work and state-of-the-art related work. This chapter is organized as follows. Section 3.1 discusses related work of rule-based 2.5D capacitance extraction highlighting the main differences between our work and several state-of-the-art related work. Section 3.2 provides related work of MEOL parasitic capacitance extraction. Section 3.3 explains the different extraction methods and compares them with the proposed hybrid extraction method. Section 3.4 presents several previous efforts of parasitic-aware routing optimization methods, and it provides a comparison among state-of-the-art related work including our work.

3.1. Rule-Based 2.5D Capacitance Extraction

Many efforts were done to improve the accuracy of rule-based 2.5D parasitic capacitance extraction methods [9], [15], [16], [27], [28], [31], [48]–[50]; however, most of them either use simplified models to improve pattern matching as in [9], [31], or tackle specific interconnect structures by using analytical models as in [15], [16], [50]. As for systematic process variations, all previous efforts of modeling the impact of systematic variations on parasitic capacitances, in rule-based methods, focused only on the accuracy of the capacitance and sensitivity formulas. They completely ignored other sources of inaccuracies such as pattern mismatches and pattern coverages. Also, they did not consider the impact on the extraction runtime after incorporating their formulas [27], [48].

In [27], a modeling methodology was developed to improve the accuracy of the 2.5D parasitic capacitance extraction method by considering reactive ion etching (RIE)

variations using sensitivity formulas. This effort used traditional sensitivity methods to handle interconnect thickness variations that are caused by RIE. Such an approach has three main problems. First, it has a limited pattern coverage as it only considers basic three wires patterns. Therefore, it cannot be generalized on complicated layout patterns. Second, it adds more computational runtime on parasitic extraction tools as it introduces additional sensitivity formulas to be computed on top of existing formulas. Third, it only considers RIE effects and completely ignores the cross-dependency impact of different variation parameters on parasitic capacitances. As a result, it is not suitable for advanced process nodes.

In [48], a modeling methodology for interconnect parasitic capacitances considering lithography effects was developed. The methodology uses a lithography simulator across many 3D layout patterns to incorporate lithography effects, such as metal width variations, into the generated 3D layout patterns. Then, it passes the modified layout patterns to a 3D field-solver to extract their parasitic capacitances. After that, the modified patterns and their parasitic capacitances are stored in a pre-characterized library to be later used by parasitic extraction tools. This effort has several problems. First, it is not applicable for 2.5D extraction methods since it only considers 3D layout patterns. Second, the lithography simulator would generate a lot of curvilinear layout shapes that add a lot of complications on layout parasitic extraction processes. The complications include more pattern mismatches, more parasitic extraction runtime due to running a lithography simulator on layouts, and a huge pre-characterization runtime due to running a 3D field-solver over many curvilinear shapes. Third, such a methodology has a very limited pattern coverage, and it cannot provide good accuracy on full chips. Moreover, the authors did not introduce any solution for the pattern coverage and pattern mismatch problems.

In [15] and [16], field-based parasitic capacitance formulas for metal wires were developed. Such formulas consider the different 3D parasitic effects of a metal wire including fringing and corner coupling capacitances; however, those formulas are only

valid for isolated wires. They do not consider the impact of surrounding metals and systematic process variations. Hence, they are not efficient for full chip interconnect parasitic extraction in advanced process nodes.

In [51], a neural-network model was developed for several 3DIC interconnect structures around through-silicon-vias (TSVs). This model uses a single dielectric structure, and it is only limited to certain interconnect structures around TSVs. Hence, such a model is not efficient for multi-dielectric environment and full chip extraction.

In [31], a pattern matching classifier was developed using neural networks in order to assign each layout pattern to a corresponding capacitance model. Also, interconnect parasitic capacitance models using neural networks were developed for 2D cross-section layout patterns. Such an approach managed to reduce pattern mismatches and improve the accuracy of parasitic capacitance extraction results; however, it has three main problems. First, the proposed models use layout patterns with limited geometrical parameters. Second, the models completely ignore systematic process variations. Third, the proposed models were only verified on simple 2D cross-section patterns, and they were not verified on real layout patterns.

Table 3.1 summarizes the contributions and limitations of our work and related works. Table 3.2 provides a comprehensive comparison among related works including our work. The comparison includes ten factors as below:

a. The considered systematic process variations.
b. The modeling methodology of systematic process variations, where the impact of systematic variations on parasitic capacitances can be implicitly considered while predicting parasitic capacitances by using the capacitance models (i.e., embedded inside the model), or it can be modeled using sensitivity formulas or lithography simulators.
c. The pattern coverage, where some models may only cover a limited number of layout patterns.
d. Type of input layout patterns (2D cross-section or 3D layout patterns).

e. The pattern matching mechanism, which is an essential step in 2.5D extraction flows. The pattern matching is used to match layout patterns with corresponding pre-characterized capacitance formulas (or models) in order to calculate the corresponding parasitic capacitance numbers. There are two types of pattern matching that include geometry-based and layer-based. In the geometry based, the pattern matching is performed based on the geometrical structures of layout patterns, whereas in the layer-based, the pattern matching is performed based on the layer names, where each set of metal layers is handled by a specific model (regardless of its geometrical structure).
f. The possibility of pattern mismatches that occur when parasitic capacitances of a layout pattern are calculated using inappropriate capacitance formulas.
g. The support of multi-dielectric environment.
h. The modeling method, which represents the method that is used to implement the models, such as analytical formulas, curve fitted formulas, lookup tables, neural networks, and support vector regressions.
i. The testing and validation methodologies, which indicate whether the related work was verified on real process technology nodes or not.
j. The overhead on the actual parasitic capacitance extraction runtime, where some models require additional computations in order to predict the impact of systematic process variations on parasitic capacitances.

Table 3.1. Contributions and limitations of the proposed 2D cross-section models and related works in rule-based capacitance extraction.

Research	Contribution	Limitation
Karsilayan *et al.* [27]	• Sensitivity formulas were developed to model the impact of a reactive ion etching (RIE) on interconnect parasitic capacitances. • This effort used traditional sensitivity methods to handle interconnect thickness variations that are caused by RIE.	• It has a limited pattern coverage as it only considers basic three wires patterns. • It adds more computational runtime on parasitic extraction tools. • It only considered reactive ion etching (RIE) effects and completely ignored the cross-dependency impact of different variation parameters on parasitic capacitances.
Tsai *et al.* [48]	• A modeling methodology for interconnect parasitic capacitances considering lithography effects was developed. • A lithography simulator was used to incorporate lithography effects into pre-characterized 3D layout patterns. Then, the parasitic capacitances of the modified patterns were extracted using a field-solver and stored in a pre-characterized library. • In the extraction phase, a lithography simulator was used to incorporate lithography effects into a given layout. Then, a pattern matching operation was performed to match layout patterns with pre-characterized patterns.	• Limited pattern coverage as the method only supports 3D layout patterns. • The lithography simulators would generate a lot of curvilinear layout shapes that add a lot of complications on layout parasitic extraction processes that include more pattern mismatches, more parasitic extraction runtime, and a huge pre-characterization runtime. • It only considered metal width variations. • It is neither suitable for advanced process nodes nor three-dimensional integrated circuits (3DIC) technologies.
Zhang *et al.* [15], [16]	• Analytical capacitance formulas were developed to calculate the different parasitic capacitance components of an isolated metal wire. • Such formulas consider the different 3D parasitic effects of a metal wire such as fringing and corner coupling capacitances.	• The proposed formulas are only valid for isolated metal wires. • They are not efficient for full chip interconnect parasitic extraction in advanced process nodes. • The formulas do not consider systematic process variations. • They are neither suitable for advanced process nodes nor 3DIC technologies.
Li and Shi [31]	• A pattern matching classifier was developed using neural networks in order to assign each layout pattern to a corresponding capacitance model. • Interconnect parasitic capacitance models using neural networks were developed for cross-section layout patterns. • This approach managed to reduce the pattern mismatches.	• Limited pattern coverage as it used limited number of pre-characterized patterns. • The method did not mitigate pattern mismatches. • The method does not consider systematic process variations. • It is not suitable for 3DIC technologies.
The proposed ratio-based and dimensions-based models	• Neural network models were developed for 28nm process node to predict interconnect parasitic capacitances of 2D cross-section layout patterns efficiently. • The proposed models use two different pattern representations that include: ratio-based and dimensions-based representations. • The proposed models managed to mitigate the pattern mismatches and improve the extraction accuracy.	• The models only consider metal width variations, and they do not consider the different systematic process variations. • The models are slower than Calibre PEX, rule-based tool, by 1.3X. • The models were only verified on 28nm process technology node. • The models are not suitable for 3DIC technologies.
The proposed vertex-based models	• Machine learning compact models were developed to predict interconnect parasitic capacitances of 2D cross-section layout patterns efficiently. • The models can handle layout patterns of arbitrary distributed polygons and mitigate pattern mismatches. • The models use a new vertex-based layout representation to help in handling systematic process variations. • The models are almost 2.5X faster than existing rule-based models. • The models consider the cross-dependency impact of different variation parameters on parasitic capacitances.	• The generated models are only valid for 2DIC technologies, where the die is mounted on a single plane inside the package. As for 3DIC technologies, multiple dies of different (or same) process nodes would be connected together by using either TSVs or interposers. This requires special modeling to predict the parasitic coupling interactions among TSVs and interconnects of different dies.

Table 3.2. A comparison among different parasitic rule-based capacitance extraction methods including the proposed 2D cross-section parasitic capacitance models.

		Karsilayan *et al.*	**Tsai *et al.***	**Zhang *et al.***	**Li and Shi**	**The proposed ratio-based and dimensions-based models**	**The proposed vertex-based models**
Systematic process variations	**Metal width variations**	No	Yes (Supported)	No	No	Yes (Supported)	Yes (Supported)
	Metal Thickness Variations	Yes (Supported)	No	No	No	No	Yes (Supported)
	Loading effects	No	No	No	No	No	Yes (Supported)
	Trapezoidal variations	No	No	No	No	No	Yes (Supported)
Modeling Method of variations		Sensitivity models	lithography simulator	Not applicable	Not applicable	Embedded inside the model	Embedded inside the model
Pattern coverage		It has a limited pattern coverage because their models cover a limited number of layout patterns.				High pattern coverage as the proposed models can handle patterns with arbitrary distributed polygons. However, they cannot handle metal polygons that are vertically overlapped.	High pattern coverage as the proposed models can handle patterns with arbitrary distributed polygons.
Type of supported patterns		2D cross-section patterns of three parallel wires.	3D patterns of parallel wires.	3D patterns of isolated wires.	2D cross-section patterns of parallel wires.	2D cross-section patterns of arbitrary distributed metal polygons (i.e., wires).	
Pattern matching mechanism		Geometry-based	Geometry-based	Geometry -based	Geometry-based	Layer-based	Layer-based
Pattern mismatches		High potential pattern mismatches as the pattern matching operations are performed based on the geometrical structures.				No pattern mismatches as the pattern matching operations are performed based on the layer names. Therefore, each set of metal layers are handled using a single compact model.	
Multi-dielectric environment		Yes	Yes	No	Yes	Yes	Yes
Modeling methods		Curve fitted formulas.	Lookup tables	Analytical formulas	Neural networks	Neural networks	Neural networks Support vector regressions
Validation and testing		Not Tested on real designs.				Only tested on 28nm process node.	Tested on 28nm, 14nm, and 7nm process nodes.
Overhead on the actual parasitic capacitance extraction runtime		It doubles the number of computations as sensitivity formulas are computed separately.	Running a lithography simulator adds a lot of overhead on runtimes.	It is fast as it uses analytical formulas with small number of variables.	It adds extra runtime due to running a classifier to assign each layout pattern to a capacitance model.	It is slower than existing commercial rule-based tool (i.e., Calibre PEX [19]) by 1.3X when being tested on 28nm process node.	It is 2.5X faster than existing commercial rule-based tool (i.e., Calibre PEX [19]) when being tested on 28nm, 14nm, and 7nm process nodes.

3.2. MEOL Parasitic Capacitance Extraction

Several efforts were done to implement MEOL parasitic capacitance models [53]–[59]. However, most of these efforts did not provide a general methodology that can predict parasitic capacitances of MEOL patterns regardless of their geometrical structure. There are four main methods to extract MEOL patterns that include: 1) field-solvers; 2) analytical formulas [55], [56], [59], [60]; 3) pre-layout parasitic extraction models that do not consider layout geometries [53], [54]; and 4) a pre-characterized library of MEOL patterns [57], [58], where the parasitic capacitances of those patterns are extracted by field- solvers and stored in a library to be later used by parasitic extraction tools.

The methods that rely on field-solvers to extract MEOL patterns suffer from three main problems including: 1) capacity limitations, 2) excessive runtime, and 3) the consumption of many computational resources [11], [56], [59]. The methods that rely on analytical models are limited to specific MEOL patterns. On the other hand, the pattern matching methods that use a pre-characterized library of MEOL patterns suffer from three problems including: 1) potential pattern mismatches that impact the accuracy of extracted capacitances; 2) insufficient pattern coverage; and 3) the consumption of large diskspace. As a result, the pre-characterization approach results in a poor parasitic capacitance extraction accuracy with error percentages exceeding 10% [57].

In [53], a neural-network model is developed to predict parasitic elements and device parameters (e.g., geometrical parameters) of devices. This is done by converting a circuit schematic into graphs that include: 1) the number of fanouts of each transistor terminal and 2) the device sizes. After that, the graphs as an input to the models in order to predict the parasitic elements and device parameters. In terms of MEOL parasitic capacitance extraction, this effort has three main problems. First, it predicts the parasitic elements as lumped capacitances. Second, it does not consider the interconnect layout geometries around the devices, which have a major impact on parasitic capacitances. Third, it can only handle the regular device structures of FINFETs with Manhattan geometries.

In [54], machine learning-based models were developed to predict parasitic elements around devices. They are pre-layout models that provide a very fast extraction runtime with reasonable accuracy. The inputs of those models are device parameters such as channel length and channel width. This effort has two main problems. First, it does not consider any layout geometries. Therefore, the post-layout simulation results might not be accurate especially for analog designs. Second, it only considers regular device structures with Manhattan geometries. Therefore, it cannot handle irregular MEOL structures.

In [55], parasitic capacitance models that predict gate-around parasitic capacitances were developed based on SPICE models. Such models were mainly developed for MOSFET structures in 40nm process technology node. They can efficiently predict the parasitic capacitances around the gate; however, they have two main problems. First, this approach only considers MOSFET devices with regular structures. Second, it is only validated on 40nm process node.

In [57], a parasitic capacitance extraction methodology for MEOL capacitances was developed. This methodology aims to identify the MEOL structures and extract their parasitic capacitances using a 3D field-solver tool. Such a methodology provides very accurate MEOL parasitic capacitance results; however, it is very slow and has a limited capacity as it uses a 3D field-solver tool. Therefore, it is not suitable for large designs.

In this work, machine learning based MEOL parasitic capacitance models were developed. The developed models are suitable for MEOL around MOSFETs and FINFETs. The models can handle regular and irregular device structures with either Manhattan or non-Manhattan geometries. They are very fast and have high capacity as compared to field-solvers.

3.3. Hybrid and Intermediate Parasitic Capacitance Extraction Methods

Hybrid parasitic capacitance extraction methods that combine rule-based and field-solvers are considered in advanced nodes. Despite the accuracy improvements introduced by those methods, they suffer from two main problems. First, the proportion of patterns that have to go to field-solvers increases in advanced nodes (as technology scales down). With this proportion approaching 50%, using theses hybrid methods does not save much time as compared to field-solvers. Second, these hybrid methods do not eliminate all outliers because the layout patterns are assigned to extraction methods based on a pre-characterized library. Sometimes, in case of encountering a pattern that does not exist in the pre-characterized library, a field-solver is used, which results in more overhead to the runtime and capacity [8]. Table 3.3 summarizes the main differences among field-solver, rule-based, existing hybrid, the proposed intermediate extraction, and the proposed accuracy-based hybrid extraction methods.

Table 3.3. Advantages and disadvantages of the different parasitic capacitance extraction methods.

Method	Advantages	disadvantages
Field-solvers	• They are very accurate as compared to measurements.	• They are slow with a limited capacity. • They cannot handle large designs
Rule-based methods	• They are very fast with reasonable accuracy. • They can handle full chips. • They are used for sign-off for many legacy nodes (>20nm)	• They cannot cope with the increasing accuracy requirements in advanced process nodes. • They neglect many 3D fringing capacitances impacting the accuracy in advanced process nodes.
Existing Hybrid method	• It can provide very accurate results. • It can handle large designs.	• It cannot cope with the accuracy requirements in advanced nodes as more patterns would be extracted using field-solvers.
Proposed intermediate	• It provides accurate results in a reasonable runtime. • It can predict all 3D fringing capacitances withing a pre-defined window. • It is suitable for advanced process nodes.	• It is 2-3X slower than existing rule-based method. • It requires a large characterization runtime.
Proposed Hybrid	• It provides very accurate results. • It can predict all 3D fringing capacitances withing a pre-defined window. • It is suitable for advanced process nodes. • It is faster than existing hybrid method.	• It requires a large characterization runtime.

3.4. Parasitic-Aware Routing Optimization

Most of existing parasitic-aware routing methods suffer from two problems. First, they use either simplified parasitic formulas or a full layout parasitic extraction in order to measure the parasitic elements for each layout modification in the design loop. The simplified parasitic models are not accurate and cannot cope with the increasing parasitic extraction accuracy requirements in advanced nodes leading to inaccurate layout optimization. On the other hand, the use of a full layout extraction is very time-consuming and not suitable for optimizing large layout designs. Second, the existing routing optimization methods do not provide a systematic way to help circuit designers in understanding the impact of parasitic elements and the corresponding layout geometries on a system's (i.e., route) performance.

In [61], a parasitic-aware routing method was developed based on simplified parasitic formulas. This approach aims to reduce the delay and routing area considering the interconnect parasitic elements of a given layout. This method identifies multiple candidate routes for each connection. Then, it evaluates the performance of each candidate until the candidates that meet the required performance are achieved. This method has three main problems. First, it uses simplified parasitic formulas that cannot cope with the new accuracy requirements of advanced process nodes. Second, this method does not deal with the parasitic effects as dominant factors on a circuit's performance. Third, this method relies on a pre-determined set of candidate routes that do not necessarily achieve the required performance.

In [62], an automatic optimization-based sizing and routing methodology was developed for analog circuits. This methodology uses a layout generator that computes the optimal electrical current correct wire topology and global routing in loop for each different sizing solution. Such a methodology relies on simplified parasitic models in order to achieve reasonable optimization runtime as it requires many optimization loops (i.e., iterations). This methodology has three main problems. First, it requires many iterations to achieve good results. Second, it uses simplified parasitic formulas that

cannot cope with the new accuracy requirements of advanced process nodes. Third, it does not deal with the parasitic effects as dominant factors on circuit's performance.

In [63]–[65], parasitic-aware routing methodologies based on circuit moments were developed. The proposed methodologies aim to optimize layout routes by minimizing a cost function. The cost function considers parasitic resistance, capacitance, self-inductance, and mutual coupling inductance effects (RLCK), and it provides a representation of the delay and ringing of the signals. Therefore, the minimization of the developed cost function helps in achieving a good balance between route's delay and ringing. These efforts have five problems. First, they require a full layout parasitic extraction in order to evaluate the corresponding cost function with every optimization iteration. Second, the cost function is only valid for delay and ringing effects. Third, they are not suitable for both net symmetry constraints and analog designs. Fourth, they do not provide good understanding to the impact of parasitic effects on a route's performance. Fifth, they do not correlate parasitic elements to certain geometries.

In [20], a template-based parasitic-aware layout optimization method was developed. Traditional template-based methods optimize layout routes in x and y directions separately. This method aims to overcome this problem by optimizing layout routes in x and y directions simultaneously. Such a method uses a hybrid algorithm that consists of nonlinear programming and graph-based algorithms in order to achieve more accurate layout optimization. However, this method has three problems. First, it does not deal with the parasitic effects as dominant factors on a circuit's performance as it uses very simple parasitic formulas to extract the parasitic elements of a given layout. Such formulas cannot cope with the new accuracy requirements of advanced process nodes. Second, it does not provide a mechanism to help circuit designers in understanding the impact of parasitic effects on a system's (i.e., route) performance. Third, it only considers rectilinear and Manhattan geometries, and it cannot handle non-Manhattan geometries.

In [22], [43], [44], [66], template-based parasitic-aware routing optimization methodologies were proposed. They are used for either retargeting or layout

optimizations. They aim to create a symbolic template with a set of constraints such as net symmetry, connectivity, parasitic bounds, and corresponding design rules. The calculations of parasitic bounds rely on multiple circuit simulations in order to identify a parasitic bound for each parasitic element. The parasitic model for each route is represented by a simple RC Π (i.e., pi) model in order to speed up the calculations of parasitic bounds. Such methodologies are fast; however, they are suffering from three problems. First, they use simplified parasitic formulas that cannot cope with the new accuracy requirements of advanced nodes. Second, they do not provide a mechanism to help circuit designers in understanding the impact of parasitic effects on a system's (i.e., route) performance. Third, most of them cannot handle non-Manhattan geometries.

In [67], analog layout design tool called LAYGEN II was developed. It uses a symbolic template (i.e., template-based) approach in order to perform placement and routing. This approach is very efficient in achieving a good initial layout for a given circuit design; however, it requires a lot of computational resources in order to handle large layouts.

In [68], an analog layout design tool was developed. It uses a combination of symbolic template (i.e., template-based) and optimization approaches in order to generate layouts. This method uses a template approach in order to reduce the search (i.e., solution) space. This method is efficient in achieving a good initial layout for a given circuit design; however, it requires a lot of computational resources in order to handle large layouts. Moreover, it is not designed to handle non-Manhattan geometries.

In [69], a routing algorithm was developed using a discrete particle swarm optimization and multi-stage transformation methods. The proposed algorithm optimizes layout routes using two types of Steiner minimal tree models that include Manhattan and non-Manhattan Steiner minimal trees. Therefore, the selected route structure can contain Manhattan and non-Manhattan geometries. This flow has two problems. First, it does not consider the impact of parasitic elements except for a route's delay. Second, it does not have a mechanism to help circuit designers in understanding the impact of parasitic elements on system's performance.

The problems of existing layout routing optimization methods can be summarized as below:

a. They do not provide a mechanism to help circuit designers in understanding the impact of parasitic elements on a system's (i.e., route) performance, such as identifying the problematic parasitic elements along with the corresponding layout geometries.
b. Most of existing efforts use either simplified parasitic formulas, such as in [9], [20], [22], [43], [66], [70], [71], or a full layout extraction, such as in [63]–[65], [72], in order to extract the parasitic elements of a given layout. The methods that use simplified parasitic formulas suffer from an accuracy problem as the accuracy of such parasitic formulas cannot cope with the increasing accuracy requirements in advanced process nodes, whereas the methods that use a full layout extraction suffer from a long runtime problem as they require a full layout extraction with every optimization iteration.
c. Many of existing layout optimization flows rely on circuit simulations with every optimization iteration as in [73], [74].

This work focuses on overcoming these problems. First, it provides a routing optimization method that can be applied either after or within the detailed routing. Second, it provides sensitivity circuit models that help circuit designers in understanding the impact of parasitic elements and the corresponding layout geometries on a route's performance. Third, it uses a novel incremental parasitic extraction method to extract the parasitic elements of modified layouts during the optimization process. Such an incremental method provides very accurate results (<1% error) with a speedup of up to 40X as compared to a full layout extraction. Fourth, it does not require multiple circuit simulations. Table 3.4 summarizes the contributions and limitations of related works including our work. Table 3.5 provides a functional comparison among related works and our work.

Table 3.4. Contributions and limitations of state-of-the-art layout routing optimization works including our work.

	Contributions	Limitations
Smey *et al.*, [61]	• A routing optimization method that aims to reduce the area and delay was developed. • This method optimizes layout routes by identifying multiple candidate routes for each connection. Then, it selects the candidate route with minimum area and minimum delay.	• It uses simplified RC parasitic formulas that cannot handle complicated layout structures in advanced nodes. Moreover, the accuracy of such formulas cannot cope with the increasing accuracy requirements in advanced nodes. • It does not consider inductance effects. • It relies on pre-determined candidate routes that do not necessarily achieve the required performance. • It does not handle non-Manhattan geometries.
Lourenco *et al.*, [62]	• An automatic optimization-based sizing and routing methodology was developed for analog circuits. • It uses a layout generator that computes the optimal electrical current correct wire topology and global routing in-loop with each different sizing solution. It relies on simplified parasitic models in order to achieve reasonable runtime as it requires many optimization loops (i.e., iterations).	• It requires many iterations to achieve good results. • It uses simplified RC parasitic formulas that cannot handle complicated layout structures in advanced nodes. Moreover, the accuracy of such formulas cannot cope with the increasing accuracy requirements in advanced nodes. • It does not consider inductance effects. • It does not handle non-Manhattan geometries.
Bhaduri and Vemuri [63]–[65]	• Parasitic-aware routing optimization methodologies based on circuit moments were developed. • The proposed optimization methodologies use a cost function that consider the different RLCK parasitic elements in a candidate route. The minimization of the cost function helps in achieving a good balance between route's delay and ringing. • The proposed methodologies identify multiple candidate routes and select the route with a minimum cost value.	• They use a full layout parasitic extraction to evaluate the cost function, which consumes a lot of time. • The developed cost function has limited applications as it only considers the delay and ringing effects. • The cost function is not suitable for net symmetry constraints and analog designs. These efforts do not provide a mechanism to help circuit designers in understanding the impact of parasitic effects on a route's performance.
Zhang *et al.*, [44]. Liu and Zhang [22], [66]. Bhattacharya *et al.*, [43]. Jangkrajarng *et al.*, [20].	• Template-based parasitic-aware routing optimization methodologies were proposed. • They aim to create a symbolic template with a set of constraints such as design rules, connectivity, and net symmetry constraints. Moreover, they identify the parasitic bounds of each parasitic element using circuit simulations. The symbolic template constraints and parasitic bounds are used as inputs to the routing optimization flow. • The calculations of parasitic elements use simple resistance and capacitance parasitic formulas. • They are efficient in achieving a good initial layout for a given circuit design in a reasonable time.	• They use simple RC parasitic formulas that cannot handle complicated layout structures in advanced nodes. Moreover, the accuracy of such formulas cannot cope with the increasing accuracy requirements in advanced nodes • They do not deal with the parasitic effects as dominant factors on a circuit's performance. • They do not consider inductance effects. • Most of them do not handle non-Manhattan geometries. They do not have a mechanism to help circuit designers in understanding the impact of parasitic elements on a system's performance
Naguib *et al.*, [68]	• An analog layout design tool was developed. This tool contains an automatic routing algorithm that uses symbolic template approaches. • The developed tool uses a template algorithm approach in order to reduce the routing search (i.e., solution) space. This approach is efficient in achieving a good initial layout for a given circuit design.	• It requires a lot of computational resources in order to handle large layout designs. • It does not consider inductance effects. • It is not designed to handle non-Manhattan geometries. • It does not have a mechanism to help circuit designers in understanding the impact of parasitic elements on a system's performance.

Table 3.4. Contributions and limitations of state-of-the-art layout routing optimization works including our work.-continued.

	Contributions	Limitations
Liu *et al*., [69]	• A routing algorithm was developed using a discrete particle swarm optimization and multi-stage transformation methods. • The proposed flow optimizes the routes using two types of Steiner minimal tree models that include Manhattan and non-Manhattan Steiner minimal trees. Therefore, the selected route structure can contain Manhattan and non-Manhattan geometries.	• It does not consider the impact of parasitic elements except for a route's delay. • It does not have a mechanism to help circuit designers in understanding the impact of parasitic elements on system's performance.
This work	• Sensitivity circuit models were developed to measure and analyze the impact of parasitic elements and corresponding layout geometries on a circuit performance cost function. • A parasitic-aware re-routing optimization methodology that uses nonlinear programming was developed. The developed methodology automatically modifies the most critical routes to meet the required performance cost function without circuit simulations. Moreover, it handles Manhattan and non-Manhattan geometries. • The routing optimization methodology uses a novel incremental parasitic extraction method in order to provide an accurate parasitic extraction results very fast. The proposed incremental extraction method considers second order parasitic capacitance effects efficiently.	• It only considers the RC parasitic elements. Hence, this model is appropriate for local interconnect at any frequency and global interconnect at a lower frequency. For high frequency global interconnect, inductance and more complex models need to be included.

Table 3.5. A comparison among several state-of-the-art layout routing optimization works including our work.

	Routing methodology	**Parasitic extraction**	**Considered parasitic elements**	**Requires circuit simulations**	**Handling of Non-Manhattan geometries**	**Models to analyze the impact of layout geometries on a system's performance**
Smey *et al.*, [61]	Minimize route's area and delay.	Simplified 2D cross-section models.	RC	Yes	No	No
Lourenco *et al.*, [62]	Automatic electromigration-aware wire topology and global routing in-loop for each different sizing solution.	Simplified 2D cross-section models.	RC	Yes	No	No
Bhaduri and Vemuri [63]–[65]	Minimize a cost function that provides a balance between the delay and ringing effects. It uses template-based approach to generate routing candidates.	Full layout extraction	RLCK	No	No	No
Zhang *et al.*, [44]. Liu and Zhang [22], [66]. Bhattacharya *et al.*, [43]. Jangkrajarng *et al.*, [20].	A symbolic template approach that is used to minimize parasitic effects and a route's area.	Simplified 2D cross-section models.	RC	Yes	No	No
Naguib *et al.*, [68]	A symbolic template approach that is used to minimize parasitic effects and a route's area.	Simplified 2D cross-section models.	RC	Yes	No	No
Liu *et al.*, [69]	Swarm optimization algorithms that are used to minimize parasitic effects and a route's area	Full layout extraction	RC	No	Yes	No
This work	Nonlinear programming to minimize a performance cost function based on circuit moments and sensitivity models.	Incremental layout parasitic extraction.	RC	No	Yes	Yes

3.5. Summary

This chapter discusses previous related works of 1) rule-based 2.5D parasitic capacitance extraction methods; 2) MEOL parasitic capacitance extraction methods; 3) hybrid parasitic capacitance extraction methods; and 4) parasitic-aware routing optimization methods. Moreover, it provides a comprehensive comparison among state-of-the-art related work including our work.

Chapter 4

Machine Learning Compact Models for Rule-Based 2.5D Capacitance Extraction

A novel modeling methodology of interconnect parasitic capacitance extraction is developed in rule-based extraction methods. The proposed methodology uses machine learning methods to create compact models that predict parasitic coupling capacitances between metal polygons in 2D cross-section layout patterns. Unlike existing models, the compact models handle patterns with arbitrary distributed polygons, consider connected polygons (i.e., polygons that hold the same potential), reduce pattern mismatches, increase pattern coverage, and consider systematic process variations. The compact models are technology-dependent, where each process technology node has a pre-characterized set of compact models. The proposed compact models enabled the extraction of more complicated and multi-dimensional layout patterns. Moreover, each compact model can replace hundreds to thousands of existing capacitance and sensitivity formulas. Therefore, the compact models managed to provide a lower computational runtime, significant reduction in pattern mismatches, and significant accuracy improvements.

The implementation process of the parasitic capacitance compact models consists of five main steps as follows: 1) identify the main characteristics of input patterns; 2) obtain training patterns; 3) generate reference parasitic capacitance numbers of training patterns; 4) extract features of cross-section patterns; and 5) train machine learning models. Fig. 4.1 shows the implementation process of interconnect parasitic capacitance compact models.

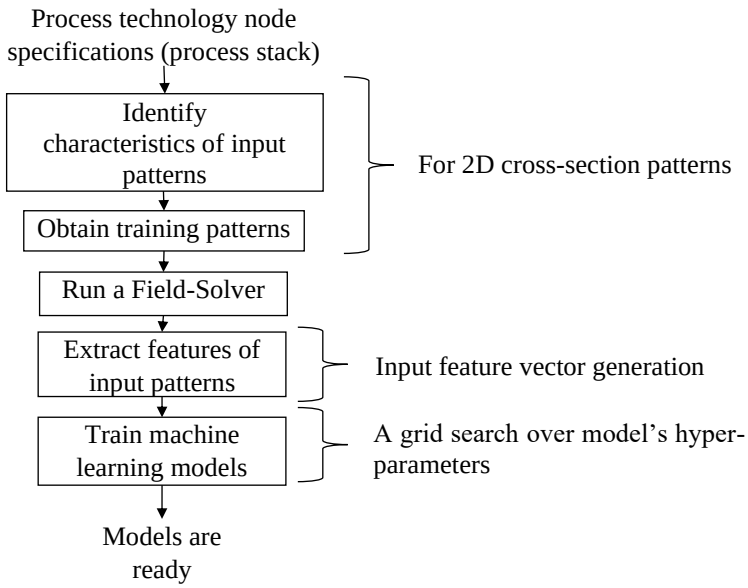


Fig. 4.1. The process of implementing 2D cross-section machine learning compact models for rule-based extraction methods.

4.1. Identify Input Patterns Characteristics

To create a compact model, we need to study several factors that identify the main characteristics of input patterns. The factors include: the surrounding multi-dielectrics, the window size of a cross-section pattern, the number of metal layers in a pattern window, the number of metal polygons in each layer, and systematic process variations.

4.1.1. Surrounding Multi-Dielectrics

Each process technology node (i.e., process stack) consists of multiple metal layers that are placed vertically and surrounded by dielectrics. Each metal layer has its own geometrical specifications such as minimum width, minimum spacing, thickness, elevation, and corresponding systematic process variation parameters. The metal layers are separated by dielectric structures. The dielectrics can be planar or conformal. Each dielectric has certain specifications such as a dielectric constant and thickness. Fig. 4.2 shows an example of a typical process technology node stack (i.e., process stack) with multi-dielectric environment. The surrounding dielectrics have a direct impact on coupling capacitances between metal layers. So, they must be considered during parasitic capacitance extraction processes. However, including the surrounding

dielectrics into the input parameters to our parasitic models would complicate the models, require more training patterns, increase pattern mismatches, and add more overhead on training and prediction runtimes. Therefore, to avoid such complications and generate effective parasitic models, each process technology node (i.e., process stack) must have its own set of parasitic capacitance compact models. Also, each pre-defined set of metal layers (i.e., metal collection), in a certain process technology node, must have a certain parasitic capacitance compact model as shown in Fig. 4.3, for example, metal1-metal2-metal3 collection has a compact model, whereas metal3-metal4-metal5 collection has another compact model. In other words, each process technology node would have a separate pre-characterized library of machine learning compact models.

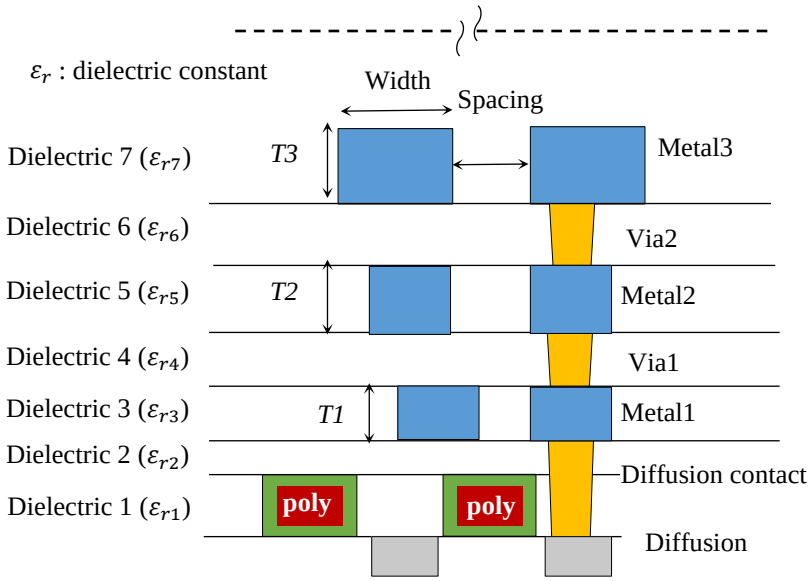


Fig. 4.2. An example of a process technology node (i.e., process stack) with multi-dielectric environment.

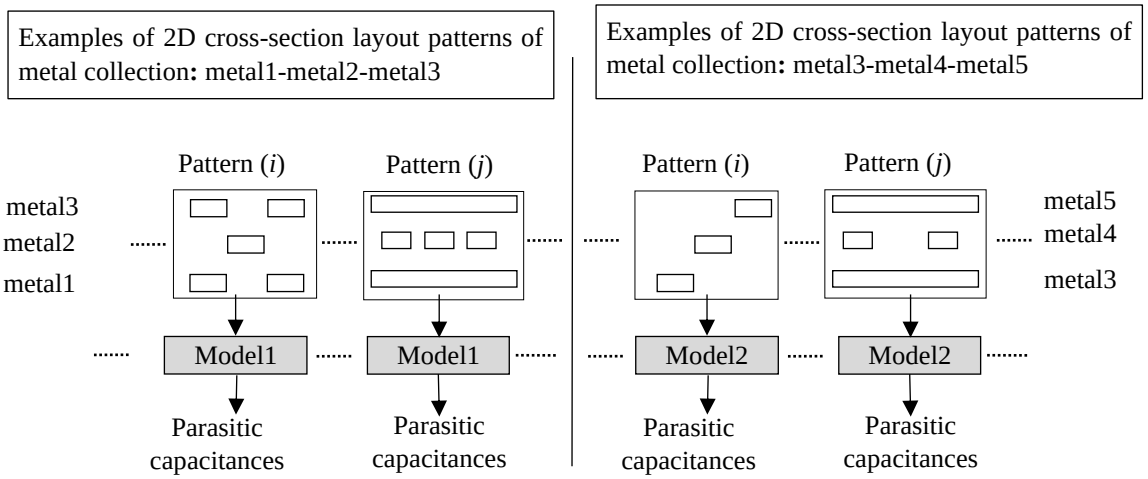


Fig. 4.3. An example of machine learning compact models for cross-section patterns of two different metal collections.

4.1.2. Window Size of Cross-Section Patterns

The window size of a 2D cross-section pattern represents the width of the pattern in the horizontal direction as shown in Fig. 4.4. When the size of a pattern window increases, the number of polygons that overlap with the window increases. Hence, more coupling capacitance components are extracted. However, this would trigger the extraction of minor capacitance components that do not have any observable impact on the extraction accuracy. Moreover, extracting such minor capacitance components would significantly increase the extraction runtime without any considerable gain. As a result, the pattern window should only consider the coupling capacitances that impact the extraction accuracy.

As the separation between any two metal polygons increases, the coupling capacitance between them decreases as shown in Fig. 4.5. Hence, any metal polygon would have an effective coupling distance (i.e., range), where any coupling capacitance to a polygon that is outside of this range is negligible.

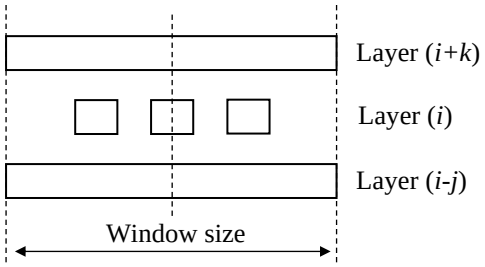


Fig. 4.4. An example of a 2D cross-section pattern of a certain metal collection showing the corresponding window size.

A pattern window size is identified by using the maximum coupling range of a target metal layer. The maximum coupling range is the maximum distance where the lateral coupling capacitance between two polygons, which belong to the same target metal layer, represents 1% of their total capacitances. Therefore, all coupling capacitances to polygons that are outside of this range are ignored. For each metal layer, the maximum coupling range is calculated by constructing a 2D cross-section pattern of two adjacent polygons using minimum dimensions. The total and lateral coupling capacitances are calculated by Raphael2D, a 2D field-solver tool [75]. The separation (i.e., spacing) between the two polygons is increased until the lateral coupling capacitance between the two polygons is less than or equal to 1% of the total capacitance on one polygon. Fig. 4.5 shows an example of calculating the maximum interaction range for metal3 layer in 28nm process node. The capacitance unit is in femtofarad (fF), whereas the separation unit is in micrometer (µm).

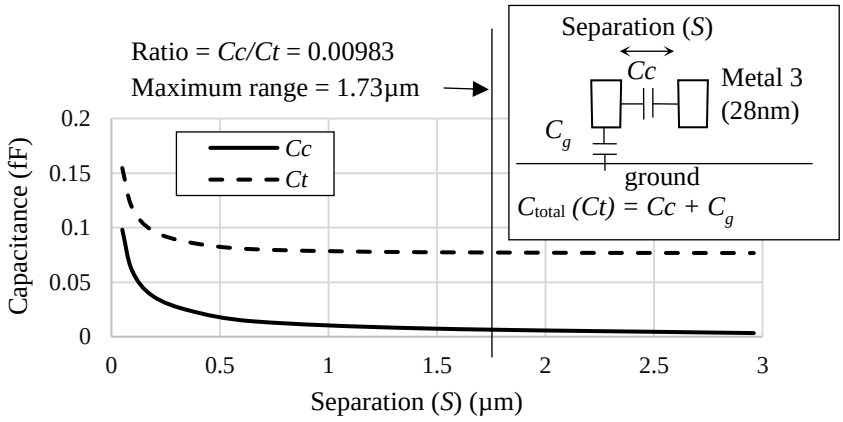


Fig. 4.5. An example of calculating the maximum coupling range using metal3 layer with minimum dimensions in 28nm process technology node. The capacitance unit is in femtofarad (fF), whereas the separation unit is in micrometer (µm).

4.1.3. The Number of Metal Layers in a Pattern

Each cross-section layout pattern consists of arbitrary distributed metal polygons that belong to the same or different metal layers. Most of existing rule-based models handle cross-section layout patterns with one, two, and three metal layers [76]–[79]. This might be enough for high density layout designs; however, for low density designs, the capacitance models should consider more than three metal layers to provide a higher extraction accuracy.

The maximum number of layers in a pattern is identified by measuring the impact of adding multiple upper and lower metal layers on total and lateral capacitance of a target metal layer. The maximum number of upper metal layers (or lower layers) is identified by constructing multiple 2D cross-section patterns of two adjacent metal polygons. Each 2D cross-section pattern has a different numbers of upper metal layers (or lower) as shown in Fig. 4.6 (a). The lateral capacitance, of a target metal polygon, is measured using Raphael2D, a 2D field-solver tool [75], while adding more upper metal layers, until the impact of adding more upper metal layers on the lateral capacitance is negligible (< 1% difference in the lateral capacitance). It is worth mentioning that the patterns are constructed on a way that minimize the impact of intermediate upper metal layers and maximize the impact of the most upper metal layer on the lateral capacitance, where all intermediate upper metal layers are represented by a single polygon with minimum dimensions, whereas the most upper metal layer is represented by a plane. This process is applied on all metal layers on a process stack. Also, the same process is applied to the maximum number of lower metal layers.

Fig. 4.6 shows an example of identifying the maximum number of upper metal layers using metal1 as a target layer in 28nm process node. Fig. 4.6 (a) shows the constructed patterns, whereas Fig. 4.6 (b) shows the lateral coupling capacitance values with increasing the number of upper metal layers. The results show that adding more than two upper layers has a minor impact (< 1% difference in the lateral capacitance) on the lateral capacitances. This process is tested on different process nodes including

28nm, 14nm, and 7nm nodes to identify the maximum number of upper and lower metal layers. The experiments show that adding more than two upper or lower metal layers has a minor impact on the lateral capacitance of a target layer. As a result, the maximum number of metal layers in a pattern is five, i.e., two upper layers, two lower layers, and one target layer.

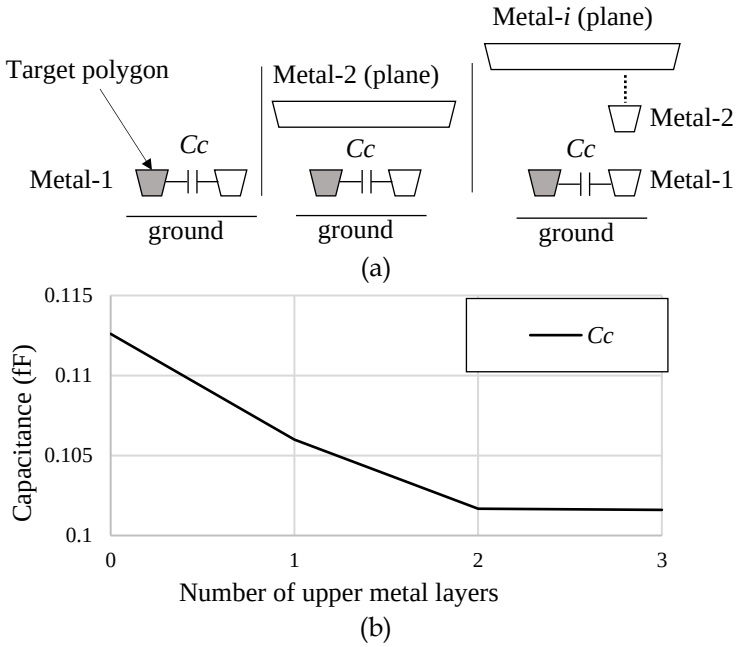


Fig. 4.6. An example showing (a) 2D cross-section patterns that are created to identify the maximum number of upper metal layers for a target metal layer, and (b) the impact of adding upper metal layers on the lateral capacitance of a target metal layer. The results are generated using metal1 as a target layer in 28nm process node.

4.1.4. Maximum Number of Polygons in a Pattern

Each pattern may contain multiple polygons across different metal layers. It is not necessarily for all polygons to have considerable coupling capacitances to target polygons, where some of the capacitances are considerable and impact the extraction accuracy, whereas other capacitances may be minor and do not impact the extraction accuracy. As a result, surrounding polygons that only impact the parasitic extraction accuracy, of target metal polygons, should be considered by the corresponding model. The maximum possible number of polygons in a pattern is identified for each metal layer separately, where each metal layer in a pattern may have a different maximum number of polygons. In other words, the maximum number of polygons is identified

for a target metal layer and surrounding (i.e., secondary) metal layers in a cross-section pattern. As for a target metal layer, the maximum number of polygons is identified by constructing 2D cross-section patterns of 3, 5, and 7 adjacent polygons as shown in Fig. 4.7 (a). The lateral capacitance between the middle and right polygons is measured in each case, by using Raphael2D, a 2D field-solver tool [75], until the impact of adding more adjacent polygons on the lateral capacitance is negligible (< 1% difference in the lateral capacitance). This process is applied on all metal layers in a process stack.

Fig. 4.7 shows an example of identifying the maximum number of target metal polygons using metal1 as a target layer in 28nm process node. Fig. 4.7 (a) shows the constructed patterns, whereas Fig. 4.7 (b) shows the lateral coupling capacitance values with increasing the number of adjacent polygons. This process is tested on different process nodes including 28nm, 14nm, and 7nm nodes. The experiments show that the appropriate maximum number of polygons for a target metal layer is 5.

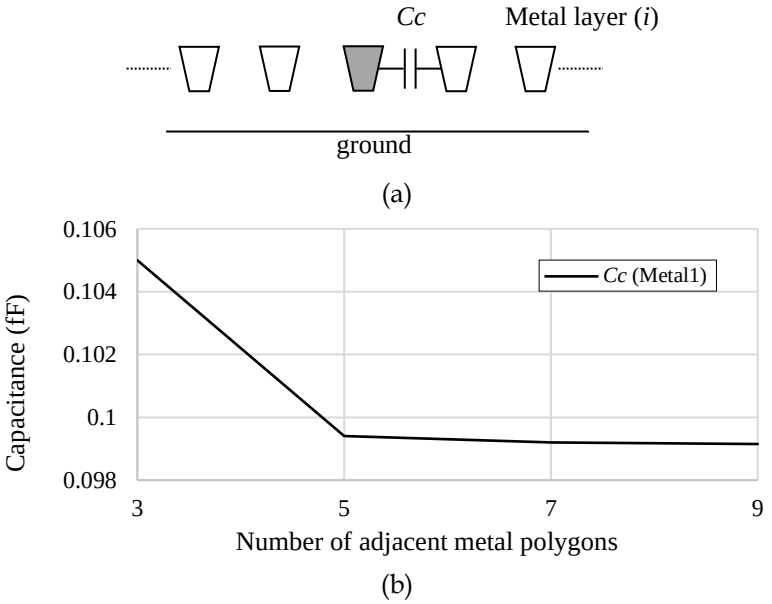


Fig. 4.7. An example of (a) 2D cross-section patterns that are used to identify the maximum number of target metal polygons in an input pattern, and (b) the impact of adding more adjacent polygons on a same layer lateral capacitance. The results are generated using metal1 as a target layer in 28nm process node.

As for upper and lower (i.e., secondary) metal layers, the maximum numbers of polygons are calculated by constructing 2D cross-section patterns of two metal layers (i.e., the target and secondary layers) as shown in Fig. 4.8 (a). The target metal layer contains one polygon at the middle, whereas the secondary metal layer has a varying number of polygons (from 2 to 7). All polygons are constructed using the corresponding minimum dimensions. The total capacitance on the middle target polygon is measured, using Raphael2D, a 2D field-solver tool [75], in each case until the impact of adding more secondary layer polygons on the total capacitance is negligible (< 1% difference in the total capacitance). This process is applied on all metal layers in a process stack.

Fig. 4.8 shows an example of identifying the maximum number of secondary metal layer polygons using metal1 as a target layer and metal2 as a secondary layer in 28nm process node. Fig. 4.8 (a) shows the constructed pattern, Fig. 4.8 (b) shows the total capacitance values with increasing the number of secondary metal layer polygons. This process is tested on different process nodes including 28nm, 14nm, and 7nm nodes. The experiments show that the appropriate maximum number of polygons for a secondary metal layer is 4.

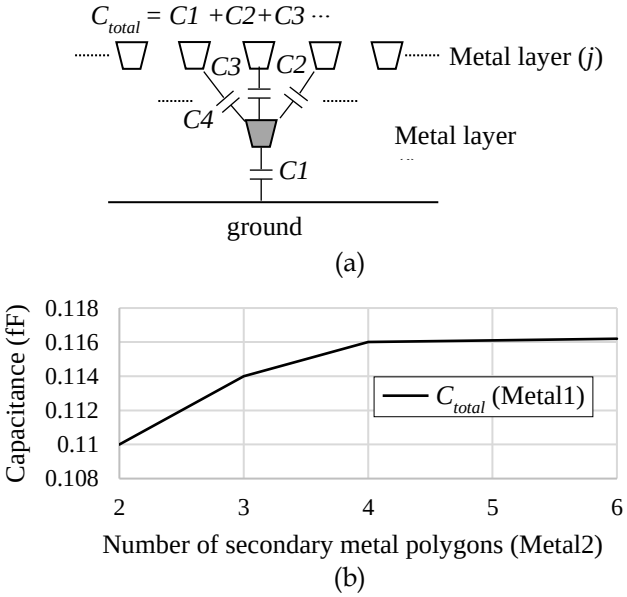


Fig. 4.8. An example of (a) 2D cross-section patterns that are used to identify the maximum number of secondary metal polygons in an input pattern, and (b) the impact of adding more secondary metal polygons on a target layer total capacitance. The results are generated using metal1 as a target layer and metal2 as a secondary layer in 28nm process node.

Eventually, the maximum number of polygons in a target metal layer is 5, whereas the maximum number of polygons in each secondary metal layer is 4. For example, the maximum number of polygons in metal1-metal2-metal3 cross-section pattern is 13, where metal1 may contain up to 4 polygons, metal2 may contain up to 5 polygons, and metal3 may contain up to 4 polygons.

4.1.5. Systematic Process Variations

Systematic process variations may have a major impact on parasitic capacitances in advanced process technology nodes. They do not only impact parasitic capacitances of associated polygons, but they also may impact parasitic capacitances of surrounding polygons [10], [26], [80]. Therefore, parasitic models must consider systematic process variations along with input patterns in order to improve the accuracy of parasitic capacitance extraction processes. In other words, the inputs to a parasitic model should be a 2D cross-section layout pattern along with the corresponding systematic process variations.

Systematic process variations are pattern dependent. They are provided by foundries in the form of lookup tables through a technology specifications file such as interconnect technology file (ITF) [26]. Therefore, systematic variations can be processed by parasitic extraction tools. Fig. 4.9 shows an example of metal width variations using metal1 layer with minimum dimensions in 28nm process node. Fig. 4.9 (a) show the impact of metal width variations on metal dimensions. The width variations impact both the width of metal wires and the separation between them, where increasing the width of metal wires decreases the separation between them. Fig. 4.9 (b) shows the impact of width variations on lateral and total capacitances using metal1 layer with minimum dimensions in 28nm process node. The width variations may cause the lateral and total capacitances to change by more than 50%.

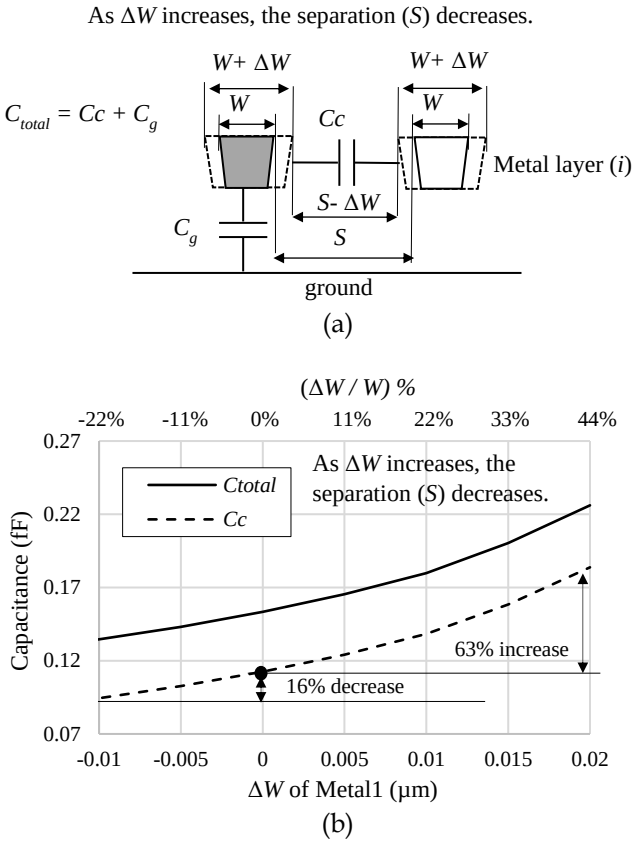


Fig. 4.9. An example of (a) width variations in cross-section interconnect patterns, and (b) the impact of metal width variations on same layer lateral and total capacitances. The results are generated using metal1 in 28nm process node.

Fig. 4.10 (a) shows an example of metal thickness variations using metal1 layer with minimum dimensions in 28nm process node. Fig. 4.10 (b) shows the impact of metal thickness variations on lateral and total capacitances. The results show that the metal thickness variations may cause the lateral and total capacitances to change by more than 20%.

Fig. 4.11 (a) shows an example of inter layer dielectric (ILD) thickness variations below metal1 layer with minimum dimensions in 28nm process node. Fig. 4.11 (b) shows the impact of ILD thickness variations on the total capacitance. The results show that the ILD thickness variations may cause the total capacitances to change by more than 10%.

Fig. 4.12 (a) shows an example of trapezoidal variations using metal1 layer with minimum dimensions in 28nm process node. Fig. 4.12 (b) shows the impact of

trapezoidal variations (i.e., sidewall slope) on the lateral and total capacitances. The results show that the trapezoidal variations may cause the total and lateral capacitances to change by more than 9%. Table 4.1 summarizes all required characteristics of input patterns.

Eventually, the maximum number of models for a process stack with N metal layers is given by:

$$\text{Number of models} = ({}^{N}C_{k} + {}^{N}C_{k-1} \ldots + {}^{N}C_{1}), \tag{4.1}$$

where C is the combination function, k is the maximum number of layers in a pattern of a certain layer collection. Usually, the number of models in a process stack ranges from tens to few hundreds, whereas the corresponding number of traditional rule-based formulas is in the range of many thousands.

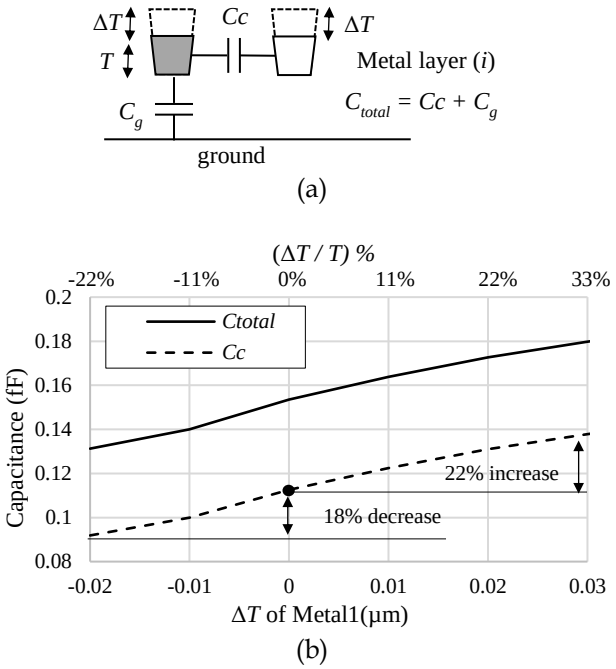


Fig. 4.10. An example of (a) metal thickness variations in cross-section interconnect patterns, and (b) the impact of metal thickness variations on same layer lateral and total capacitances using metal1 with minimum dimensions in 28nm process technology node.

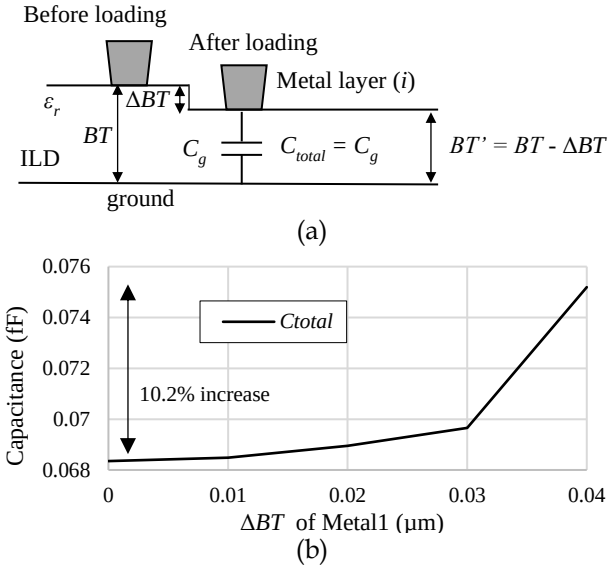


Fig. 4.11. An example of (a) ILD thickness variations in cross-section interconnect patterns, and (b) the impact of ILD thickness variations on total capacitances using metal1 with minimum dimensions in 28nm process technology node.

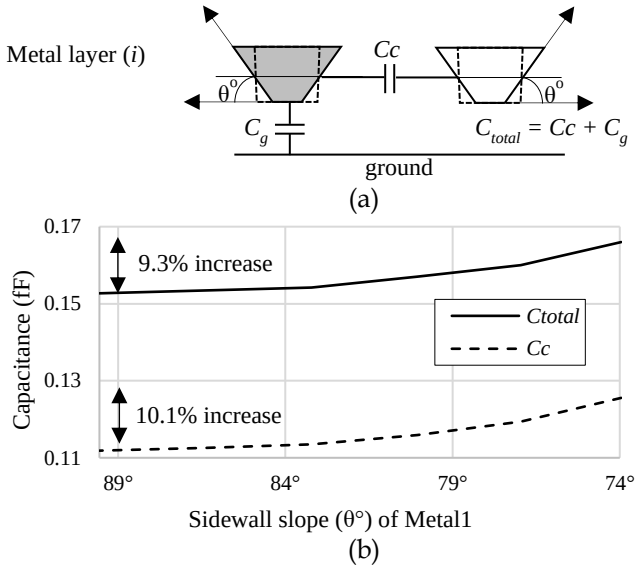


Fig. 4.12. An example of (a) trapezoidal variations in cross-section interconnect patterns, and (b) the impact of trapezoidal variations on total capacitances. The results are generated using metal1 with minimum dimensions in 28nm process technology node.

Table 4.1. A summary of input pattern characteristics.

Characteristic	The way of handling a certain characteristic
Multi-dielectric stacks	Multi-dielectric environments are handled by using a different pre-characterized library for each process stack.
The size of cross-section pattern window (in the horizontal *x*-direction)	The pattern size, in *x*-direction, equals to the maximum coupling range of the corresponding target metal layer.
The maximum number of metal layers in a pattern	5 metal layers
The maximum number of metal polygons	For a target metal layer: 5 polygons For a secondary metal layer: 4 polygons
Systematic process variations	Inputs patterns of parasitic models should include systematic variations.

4.2. Generate 2D Cross-Section Patterns

Once all input pattern characteristics are identified, they are used to generate input and training patterns for parasitic models. The training patterns are obtained from several real designs in order to increase the pattern coverage and make sure that training patterns reflect real design topologies. The generation process of training patterns starts with selecting several real designs, for example, ring oscillator (RO), static read access memory (SRAM), and digital to analog converter (DAC) layout designs. Then, the geometries and dimensions of all selected designs are modified by applying the corresponding systematic process variations. After that, the modified layouts are fractured into 2D cross-section patterns taking into considerations the corresponding characteristics of input patterns. In addition, more patterns are generated randomly for each metal collections covering the ranges from 1X to 10X of minimum dimensions. Eventually, the obtained 2D cross-section patterns are used as training patterns to machine learning models. The total number of obtained cross-section patterns for each model is 30K patterns, where each model handles patterns of a certain metal layer collection (e.g., metal1-metal2-metal3).

4.3. Field-Solver Execution

Once all training patterns are obtained, their parasitic capacitances are extracted using Raphael2D, a 2D field-solver tool [75]. The extracted parasitic capacitances are used as reference numbers to train our machine learning models.

4.4. Input Pattern Representation

Layout patterns are represented by a set of parameters that are used as inputs to NN models to predict the corresponding parasitic capacitances. These parameters should represent the main features of each cross-section pattern that are required to predict a certain capacitance component. The required features include the pattern's geometrical characteristics, polygons connectivity, and the required capacitance component. The proposed features consist of three feature vectors. The first vector provides geometrical characteristics of the whole pattern, the second vector provides geometrical characteristics of aggressor polygons, while the third vector provides geometrical characteristics of victim polygons. The three vectors are combined and used as an input to a NN model.

Providing geometrical characteristics of aggressor and victim polygons helps in identifying the connectivity and the required coupling capacitance component, which is the coupling capacitance between the aggressor and victim polygons. Since the three vectors belong to the same layout pattern, they have the same size. Each vector consists of a number of metal layers (from 1 to 5) that varies based on the corresponding layers collection (i.e., layer combination), and each metal layer is represented by a vector of features (i.e., parameters). In other words, each pattern's vector is represented by an array of vectors, where each vector in the array represents a metal layer's feature vector, and the size of the array is the number of metal layers in the corresponding pattern. In order to represent the geometrical characteristics of each cross-section patterns, three different feature representations are proposed. The proposed representations include ratio-based and dimensions-based representations.

4.4.1. Ratio-Based Representation

In this representation, each metal layer, in a certain cross-section pattern, is represented by an array of segments, and each segment contains a value that represents the ratio between the overlapping polygon width and the segment width (i.e., density). So, each metal layer is represented by a vector of densities as shown in Fig. 4.13. The experiments showed that the segment width should be less than half the minimum spacing of the corresponding metal layer to avoid having multiple polygons on the same segment. The number of segments is calculated based on the pattern's size and the corresponding technology specifications as below:

$$\text{Number of segments per layer} = \frac{\text{Pattern size}}{\text{segment width}}. \tag{4.2}$$

Since

$$\text{segment width} \leq (0.5 \times \text{min spacing}), \tag{4.3}$$

then

$$\text{number of segments per layer} \geq \frac{\text{Pattern size}}{0.5 \times \text{min spacing}}. \tag{4.4}$$

Therefore, the input vector size for ratio-based representation is given by:

$$\text{Ratio-based input vector size} = 3 \times (\text{number of layers} \times \text{number of segments per layer}), \tag{4.5}$$

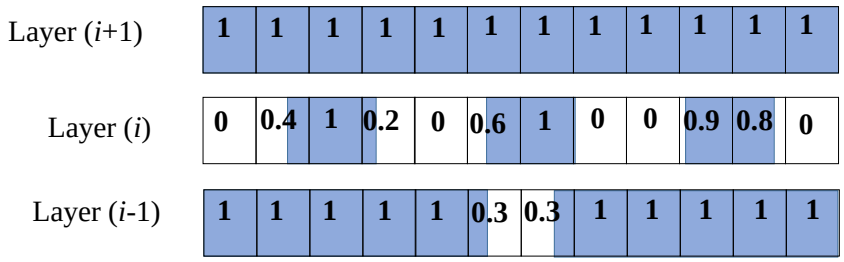


Fig. 4.13. An example of the proposed width ratio-based pattern representation.

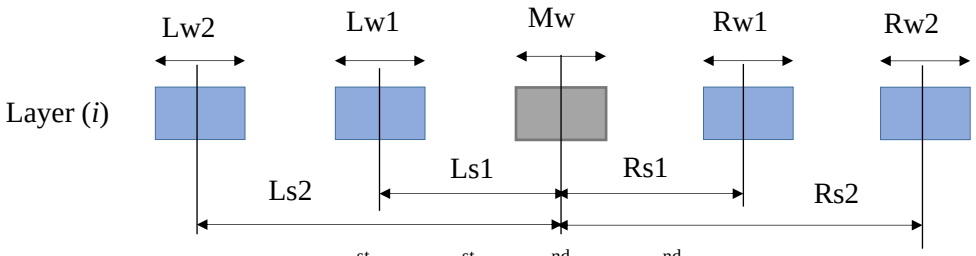
This representation does not consider the different systematic process variations, except for width variations. The rest of systematic process variations are handled using the traditional sensitivity formulas as in [27]–[29].

4.4.2. Dimensions-Based Representation

In this representation, each metal layer is represented by a vector of widths and spacings that are measured from the middle of the corresponding pattern. The vector length is twice the maximum number of polygons in the corresponding pattern's size as shown in Fig. 4.14. The number of parameters is different from a metal layer to another and from layers collection (i.e., layer combination) to another, given that the maximum number of target metal polygons is 5, and the maximum number of secondary metal polygons is 4. Therefore, the maximum number of target layer parameters is 10, and the maximum number of secondary layer parameters is 8. The maximum input vector size for dimensions-based representation is given by:

$$\text{Dimensions-based input vector size} = 3 \times (8 \times \text{number of secondary layers} + 10). \quad (4.6)$$

This representation does not consider the different systematic process variations, except for width variations. The rest of systematic process variations are handled using the traditional sensitivity formulas as in [27]–[29].



Layer's feature vector = [middle; 1st right; 1st left; 2nd right; 2nd left; …]

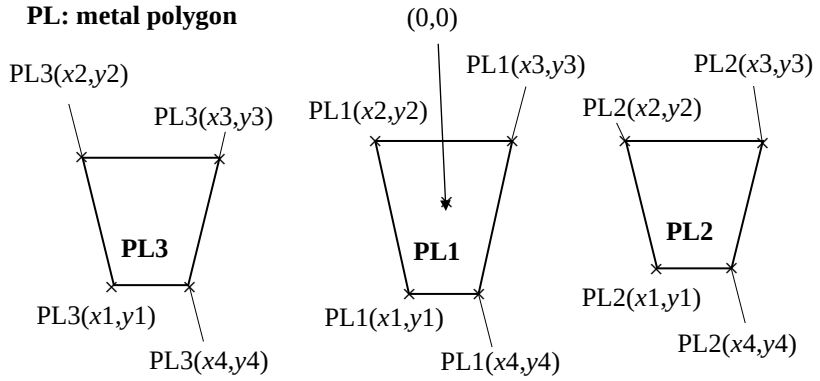
= [Mw 0; Rw1 Rs1; Lw1 Ls1; Rw2 Rs2; Lw2 Ls2;…]

Fig. 4.14. An example of the proposed dimensions-based representation.

4.4.3. Vertex-Based Representation

In this representation, the pattern's geometries and systematic process variations are represented together by using a novel vertex-based feature representation. As

shown in Fig. 4.15, each metal layer in a pattern is represented by a vector of polygons. The number of polygons of each metal layer in a pattern is shown in Table 4.1, where the maximum number of polygons of a target metal layer is 5, whereas the maximum number of polygons of a secondary metal layer is 4. Each metal polygon in a vector is represented by the polygon's vertices, where each vertex is measured from the center of the corresponding pattern. In other words, each polygon is represented by 8 displacement parameters including (x_1, y_1), (x_2, y_2), (x_3, y_3), and (x_4, y_4) as shown in Fig. 4.15. As a result, each polygon is represented by 8 values (vertices), and the vector size of each layer is estimated by (8 × maximum number of polygons in a metal layer). It is worth mentioning that the vertices of empty polygons are represented by zeros as shown in Fig. 4.15.



Layer(i) vector = [middle polygon, 1st right, 1st left, 2nd right, 2nd left]
= [PL1($x1$, $y1$, $x2$, $y2$, $x3$, $y3$, $x4$, $y4$), // middle polygon
PL2($x1$, $y1$, $x2$, $y2$, $x3$, $y3$, $x4$, $y4$), // 1st right polygon
PL3($x1$, $y1$, $x2$, $y2$, $x3$, $y3$, $x4$, $y4$), // 1st left polygon
0,0,0,0,0,0,0,0, // 2nd right polygon is empty
0,0,0,0,0,0,0,0 // 2nd left polygon is empty
]

Fig. 4.15. An example showing the novel vertex-based pattern representation using three polygons of the same metal layer in a cross-section pattern.

Such a vertex-based representation considers metal thickness variations, loading effects, wire width variations, and trapezoidal variations of all polygons in a pattern simultaneously. In other words, it includes systematic process variations during capacitance calculations. Therefore, there is no need to invoke traditional sensitivity

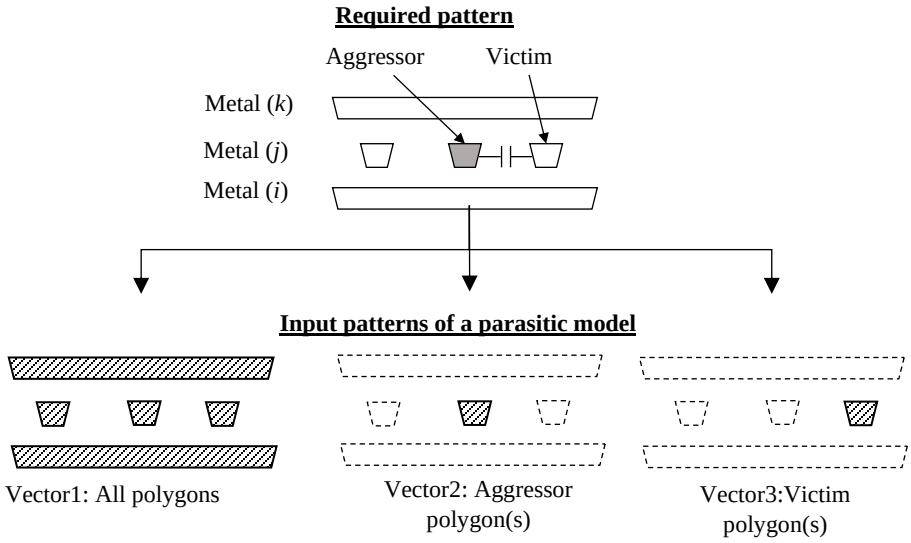
formulas or any special modeling to handle systematic process variations. Also, such a representation considers the cross-dependency impact of different variation parameters on parasitic capacitances. This resulted in fewer computations, better performance, and more accurate parasitic extraction results. The next required input parameter by parasitic models is the required capacitance component, which informs the model about the capacitance components to be extracted. The required capacitance components are identified by including the geometries of aggressor and victim polygons to the input vector of parasitic models. Therefore, the input feature vector is represented by three internal vectors. The first vector contains geometries of all polygons, the second vector contains geometries of aggressor polygons, whereas the third vector contains geometries of victim polygons as shown in Fig. 4.16. The three vectors have the same size. The novel vertex-based pattern representation is used to represent the polygons in the three vectors. The size of an internal vector is estimated by:

$$\text{Vertex-based internal vector size} = 8 \times (4 \times \text{number of secondary layers} + 5 \times \text{number of target layers}), \quad (4.7)$$

where the number of target layers is usually 1, whereas the input feature vector size of is estimated by:

$$\text{Vertex-based input feature vector size} = 3 \times \text{Vertex-based internal vector size}, \quad (4.8)$$

for example, the input vector size of a pattern with one target metal layer is 120, where the maximum number of polygons of a target metal layer is 5, each polygon is represented by 8 parameters (i.e., vertices), and there are three internal vectors with the same size (i.e., all polygons, aggressor polygons, and victim polygons). Table 4.2 shows the input vector sizes of different metal collections (i.e., models) using the proposed vertex-based, ratio-based, and dimensions-based pattern representations.



Vector1 = all polygons of [Metal(*i*); Metal(*j*); Metal(*k*)]

Vector2 = aggressor polygons in [Metal(*i*); Metal(*j*); Metal(*k*)]

Vector3 = victim polygons in [Metal(*i*); Metal(*j*); Metal(*k*)]

Input feature vector = [Vector1; Vector2; Vector3]

Fig. 4.16. An example showing the input vector of a parasitic model.

Table 4.2. Input vector sizes of several metal collections and models.

Collection	Input vector sizes		
	vertex-based	**Ratio-based**	**Dimensions-based**
Metal1-Metal2-Metal3	Target Metal: Metal2 Secondary Metals: Metal1, Metal3 Target Metal polygons: 5 Secondary Metal polygons: 4 + 4 = 8 Number of Polygons: 8 + 5 = 13 Vertex-based representation: 13 × 8 = 104 Input vector size: 104 × 3 = 312	Assuming each metal is represented by 50 segments. Whole pattern: 3×50 Aggressor: 3×50 Victim: 3×50 Input vector size: 3×3×50 = 450	Target and secondary metals are represented by 10 and 8 parameters, respectively. Whole pattern: 10 + 8 + 8 = 26 Aggressor: 10 + 8 + 8 = 26 Victim: 10 + 8 + 8 = 26 Input vector size: 3×26 = 78
Metal2-Metal3-Metal5-Metal6	Target Metal: Metal3 Secondary Metals: Metal2, Metal5, Metal6 Target Metal polygons: 5 Secondary Metal polygons: 4 + 4 + 4 = 12 Number of Polygons: 12 + 5 = 17 Vertex-based representation: 17 × 8 = 136 Input vector size: 136 × 3 = 408	Assuming each metal is represented by 50 segments. Whole pattern: 4×50 Aggressor: 4×50 Victim: 4×50 Input vector size: 3×4×50 = 600	Target and secondary metals are represented by 10 and 8 parameters, respectively. Whole pattern: 10+3×8 = 34 Aggressor: 10+3×8 =34 Victim: 10+3×8 =34 Input vector size: 3×34 = 102

Table 4.2. Input vector sizes of several metal collections and models. continued.

Collection	Input vector sizes		
	vertex-based	**Ratio-based**	**Dimensions-based**
Metal3	Target Metal: Metal3 Target Metal polygons: 5 Number of Polygons: 5 Vertex-based representation: 5 × 8 = 40 Input vector size: 40 × 3 = 120	Assuming each metal is represented by 50 segments. Whole pattern: 1×50 Aggressor: 1×50 Victim: 1×50 Input vector size: 3×1×50 = 150	Target metal is represented by 10 parameters. Whole pattern: 10 Aggressor: 10 Victim: 10 Input vector size: 3×10 = 30
Metal1-Metal2	Target Metal: Metal1 Secondary Metals: Metal2 Target Metal polygons: 5 Secondary Metal polygons: 4 Number of Polygons: 4 + 5 = 9 Vertex-based representation: 9 × 8 = 72 Input vector size: 72 × 3 = 216	Assuming each metal is represented by 50 segments. Whole pattern: 2×50 Aggressor: 2×50 Victim: 2×50 Input vector size: 3×2×50 = 300	Target and secondary metals are represented by 10 and 8 parameters, respectively. Whole pattern: 10 + 8 = 18 Aggressor: 10 + 8 = 18 Victim: 10 + 8 = 18 Input vector size: 3×18 = 54

4.5. Training Parasitic Models

Two different machine learning methods are used to create parasitic capacitance models including Neural Networks (NN) and Support Vector Regressions (SVR). The models are used to predict parasitic coupling capacitances between metal polygons in 2D cross-section patterns. For a certain process technology node, there is a model for each metal collection, where metal1-metal2-metal3 has a model, whereas metal2-metal4-metal5 has another model. The inputs of the models are the pattern representation of all polygons followed by aggressor and victim polygons as shown in Fig. 4.17.

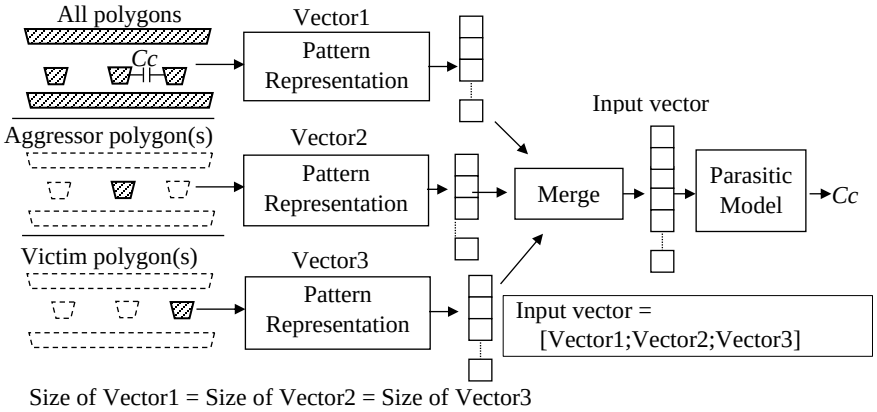


Fig. 4.17. An example showing the flow of generating an input feature vector of a parasitic capacitance model.

4.5.1. Neural-Networks Models

A Neural Network (NN) model is implemented to predict parasitic capacitances in 2D cross-section patterns. There is a NN model for each metal collection in a certain process technology node. The architecture and hyper-parameters of NN models are obtained using a grid search algorithm. The purpose of applying a grid search algorithm is to obtain appropriate NN architectures. The NN architectures are obtained based on the number of metal layers in the corresponding metal collection. For example, a metal collection with five metal layers has a NN architecture, whereas a metal collection with four metal layers has another NN architecture.

The grid search algorithm is applied on fully connected neural networks. The search range of the grid search covers several parameters including the number of layers, number of neurons in each layer, activation functions, optimizer, batch size, learning rate, and initializations. Table 4.3 summarizes the search ranges of each parameter. The evaluation criteria of selecting a NN architecture are set based on the test set accuracy, where the grid search observes the accuracy of test sets across all architectures until a mean square error of 0.01% is achieved. Such a process is applied on 28nm, 16nm, and 7nm process nodes in order to obtain unified NN architectures for each metal collection model. Table 4.4 shows the obtained NN architectures for the different proposed pattern representations.

As for hyper-parameters, the dataset is divided into 70% training data and 30% test data, validation set is 10%, the number of epochs is 1K, adaptive moment estimation (ADAM) optimizer is used, the learning rate is set to 1e-3, the batch size is set to 500, the cost function is set to a mean square error, and the batch normalization is applied. These parameters are obtained using a grid search.

Table 4.3. Search ranges of neural network architectures.

Parameter	Search range
Number of layers	From 1 to 7 with a step size of 1
Number of neurons in each layer	$n/7$, $n/6$, $n/5$, $n/4$, $n/3$, $n/2$, and n, where n is the input vector size.
Activation function	The rectified linear activation unit (RELU) and hyperbolic tangent function (tanh)
Optimizer	The adaptive moment estimation (ADAM) and stochastic gradient descent (SGD)
Batch size	500, 1000, and 1500
Learning rate	1e-2, 1e-3, and 1e-4

Table 4.4. Neural network architectures of parasitic capacitance models.

Input vector	NN Architecture		
	vertex-based	**Ratio-based**	**Dimensions-based**
One metal layer	**Input vector size:** 120 **Number of hidden layers:** 2. **Number of neurons in hidden layers:** 40 and 60, respectively. **Activation functions:** RELU and tanh, respectively. **Cost function:** mean square error. **Initializations:** He's normal [81] and Glorot's normal [82] initializations, respectively.	**For input vector size:** 50. **Number of hidden layers:** 2. **Number of neurons in hidden layers:** 50 and 50, respectively. **Activation functions:** RELU and tanh, respectively. **Cost function:** mean square error. **Initializations:** He's normal [81] and Glorot's normal [82] initializations, respectively.	**Input vector size:** 30. **Number of hidden layers:** 2. **Number of neurons in hidden layers:** 30 and 30, respectively. **Activation functions:** RELU and tanh, respectively. **Cost function:** mean square error. **Initializations:** He's normal [81] and Glorot's normal [82] initializations, respectively.
Two metal layers	**Input vector size:** 216 **Number of hidden layers:** 2. **Number of neurons in hidden layers:** 72 and 108, respectively. **Activation functions:** RELU and tanh. **Cost function:** mean square error. **Initializations:** He's normal and Glorot's normal initializations, respectively.	**For input vector size:** 100. **Number of hidden layers:** 2. **Number of neurons in hidden layers:** 50 and 100, respectively. **Activation functions:** RELU and tanh. **Cost function:** mean square error. **Initializations:** He's normal and Glorot's normal initializations, respectively.	**Input vector size:** 54. **Number of neurons in hidden layers:** 54 and 54, respectively. **Activation functions:** RELU and tanh. **Cost function:** mean square error. **Initializations:** He's normal and Glorot's normal initializations, respectively.

Table 4.4. Neural network architectures of parasitic capacitance models,-continued.

Input vector	NN Architecture		
	vertex-based	**Ratio-based**	**Dimensions-based**
Three metal layers	**Input vector size:** 312 **Number of hidden layers:** 2. **Number of neurons in hidden layers:** 78 and 156, respectively. **Activation functions:** RELU and tanh, respectively. **Cost function:** mean square error. **Initializations:** He's normal and Glorot's normal initializations, respectively.	**For input vector size:** 150. **Number of hidden layers:** 2. **Number of neurons in hidden layers:** 75 and 150, respectively. **Activation functions:** RELU and tanh, respectively. **Cost function:** mean square error. **Initializations:** He's normal and Glorot's normal initializations, respectively.	**Input vector size:** 78. **Number of neurons in hidden layers:** 78 and 78, respectively. **Activation functions:** RELU and tanh, respectively. **Cost function:** mean square error. **Initializations:** He's normal and Glorot's normal initializations, respectively.
Four metal layers	**Input vector size:** 408 **Number of hidden layers:** 3. **Number of neurons hidden layers:** 68, 102, and 204, respectively. **Activation functions:** RELU, tanh, and tanh, respectively. **Cost function:** mean square error. **Initializations:** He's normal, Glorot's normal, and Glorot's normal initializations, respectively.	**For input vector size:** 200. **Number of hidden layers:** 3. **Number of neurons hidden layers:** 67, 100, and 200, respectively. **Activation functions:** RELU, tanh, and tanh, respectively. **Cost function:** mean square error. **Initializations:** He's normal, Glorot's normal, and Glorot's normal initializations, respectively.	**Input vector size:** 102. **Number of hidden layers:** 3. **Number of neurons hidden layers:** 51, 102, and 102, respectively. **Activation functions:** RELU, tanh, and tanh, respectively. **Cost function:** mean square error. **Initializations:** He's normal, Glorot's normal, and Glorot's normal initializations, respectively.
Five metal layers	**Input vector size:** 504 **Number of hidden layers:** 3. **Number of neurons hidden layers:** 84, 126, and 252, respectively. **Activation functions:** RELU, tanh, and tanh, respectively. **Cost function:** mean square error. **Initializations:** He's normal, Glorot's normal, and Glorot's normal initializations, respectively.	**For input vector size:** 250. **Number of hidden layers:** 3. **Number of neurons hidden layers:** 84, 125, and 250, respectively. **Activation functions:** RELU, tanh, and tanh, respectively. **Cost function:** mean square error. **Initializations:** He's normal, Glorot's normal, and Glorot's normal initializations, respectively.	**Input vector size:** 126. **Number of hidden layers:** 3. **Number of neurons hidden layers:** 126, 63, and 126, respectively. **Activation functions:** RELU, tanh, and tanh, respectively. **Cost function:** mean square error. **Initializations:** He's normal, Glorot's normal, and Glorot's normal initializations, respectively.

4.5.2. Support Vector Regressions

Support vector regression (SVR) models are implemented to predict parasitic coupling capacitances of 2D cross-section patterns. There is a model for each metal collection in a certain process technology node. In order to obtain unified hyper parameters for all models, a grid search algorithm is applied across 28nm, 14nm, and 7nm process nodes. The search range of SVR models includes kernel, regularization parameter (*C*), gamma, and epsilon parameters. The search ranges of these parameters are listed in Table 4.5. The cost function is set to a mean square error. The evaluation criteria are set based on the test set accuracy, where the grid search observes the accuracy of test sets across different combinations of hyper-parameters until a mean square error of 0.01% is achieved. Table 4.6 shows the obtained SVR hyper-parameters for each input vector size for the three pattern representations.

Table 4.5. Search ranges of support vector regression hyper-parameters.

Parameter	The search range
Kernel	Radial Basis Function (RBF) and polynomial.
Regularization parameter (*C*)	From 1 to 20 with a step size of 1
Epsilon	From 0.05 to 0.5 with a step size of 0.05
Gamma	From 0.1 to 1 with a step size of 0.1

Table 4.6. SVR hyper-parameters of parasitic capacitance models for the proposed ratio-based, dimensions-based, and vertex-based pattern representations.

Input vector of a model	SVR Hyper-parameters
One metal layer	**Kernel:** Radial Basis Function (RBF) **Regularization parameter (*C*):** 8 **Epsilon:** 0.1 **Gamma:** 0.3
Two metal layers	**Kernel:** Radial Basis Function (RBF) **Regularization parameter (*C*):** 9 **Epsilon:** 0.1 **Gamma:** 0.4
Three metal layers	**Kernel:** Radial Basis Function (RBF) **Regularization parameter (*C*):** 9 **Epsilon:** 0.1 **Gamma:** 0.4
Four metal layers	**Kernel:** Radial Basis Function (RBF) **Regularization parameter (*C*):** 10 **Epsilon:** 0.1 **Gamma:** 0.3
Five metal layers	**Kernel:** Radial Basis Function (RBF) **Regularization parameter (*C*):** 10 **Epsilon:** 0.1 **Gamma:** 0.3

4.6. Experimental Results

The proposed modeling methodology was tested across three different process technology nodes including 28nm, 14nm, and 7nm process nodes. The testing covered several real designs for each node. The accuracy of the generated compact models was measured relative to Raphael, 2D field-solver. Also, the accuracy and runtime of the generated NN and SVR compact models were compared against Calibre PEX cross-section models [19]. The proposed ratio-based and dimensions-based used the sensitivity formulas of Calibre PEX to handle systematic process variations, whereas the proposed vertex-based can handle the impact of systematic process variations without using external sensitivity formulas. The relative error was measured for each capacitance component in a layout pattern using the below formula:

$$\text{Relative error} = (\text{predicted} - \text{reference}) / \text{predicted}, \tag{4.9}$$

where the predicted value represents the capacitance value that is obtained from the model, whereas the reference value represents the corresponding capacitance value that was obtained from Raphael, 2D field-solver. Moreover, nonparametric statistical tests were performed to test the significant difference in performance (i.e., accuracy) between each two models.

For each process technology node, the proposed modeling methodology was used to generate NN and SVR models. The training data were obtained from real layouts including static read access memory (SRAM), digital to analog converter (DAC), and ring oscillator (RO) designs. Also, more training patterns were randomly generated covering the ranges from 1X to 10X of the minimum dimensions. The training and model's generation used Tensor flow libraries, [83], and the implementation is done using Python [84]. The training used Intel Xeon(R) E5-2680, 4CPU, 2.50GHz, and 16G of RAM. The errors of extracted capacitances were measured relative to Raphael, 2D field-solver. There are six types of developed models that include NN using ratio-based, SVR using ratio-based, NN using dimensions-based, SVR using dimensions-based, NN using vertex-

based, and SVR using vertex-based. The six types were tested and compared to each other.

4.6.1. Testing Designs of 28nm Process Nodes

The total number of generated models for each modeling method is 130. The generated models cover 130 different metal collections each includes 1 to 5 different metal layers. Each model (i.e., NN or SVR model) was trained over 30K cross-section patterns, where 21K patterns (70%) were used for the training set, and 9K patterns (30%) were used for the test set. Table 4.7 shows the training runtime of NN and SVR models using the three different proposed pattern representations.

Table 4.7. Training runtimes of the 2D cross-section parasitic models for 28nm process node.

	Ratio-based		Dimensions-based		Vertex-based	
	NN	**SVR**	**NN**	**SVR**	**NN**	**SVR**
Training runtime	18.9 hours	11.2 hours	18.2 hours	10.3 hours	19.3 hours	12.7 hours

As for NN models, the total training runtime of all models for the ratio-based, dimensions-based, and vertex-based representations are 18.9, 18.2, and 19.3 hours, respectively. As for SVR models, the total training runtime of all models for the ratio-based, dimensions-based, and vertex-based representations are 11.2, 10.3, and 12.7 hours, respectively. The training (i.e., models generation) runtimes can significantly improve by multi-processing. It is worth mentioning that the models were generated only once for each process node. After that, the generated models are used numerous times by parasitic extraction tools. Table 4.8 shows the test sets accuracy the proposed models for the three different representations.

Table 4.8. The accuracy and relative errors of test sets for all developed cross-section models of 28nm process node.

	Ratio-based		Dimensions-based		Vertex-based	
	NN models	**SVR models**	**NN models**	**SVR models**	**NN models**	**SVR models**
Mean of relative errors	0.051%	0.062%	0.049%	0.058%	0.031%	0.052%
Standard deviation of relative errors	2.61%	2.95%	2.93%	3.17%	2.13%	2.35%
Outliers with relative error > 5%	2.25%	2.37%	2.45%	2.71%	1.45%	1.67%
Mean square error	0.00314	0.00335	0.00416	0.00422	0.00173	0.00216

The training and test sets accuracy comparison used four main criteria including 1) the mean of all relative errors, 2) the standard deviation of all relative errors, 3) the percentage of outliers that exceeds 5% relative error (i.e., the number of outliers to the total number of extracted capacitance components), and 4) the mean square error across all models. The accuracy results of test sets show that the proposed models provide a high accuracy, where almost 98% of the extracted capacitances have relative errors below 5%.

As for testing the generated models on real design patterns of 28nm process node, the generated models were tested over cross-section patterns of three different test chips including dynamic read access memory (DRAM), static read access memory (SRAM), and voltage-controlled oscillator (VCO) designs that were not included during the training processes. The total numbers of extracted cross-section patterns in DRAM, SRAM, and VCO designs are 790K, 327K, and 953K patterns, respectively. The corresponding total number of capacitance components are 2.76M, 1.3M, and 4.2M capacitances, respectively. Therefore, the total number of extracted cross-section patterns across all designs is 2.07M patterns, and the total number of extracted capacitances across all designs is 8.26M.

Fig. 4.18 shows histograms of relative errors covering all extracted capacitances across all tested 28nm designs using the ratio-based NN, ratio-based SVR, dimensions-based NN, dimensions-based SVR, vertex-based NN, vertex-based SVR, and rule-based extraction cross-section models. The accuracy comparisons show that the proposed vertex-based NN and SVR models provide high accuracy results as compared to existing rule-based cross-section models and other proposed models. Table 4.9 shows the percentages of extracted capacitance components with relative errors above 5% using the different extraction methods.

Table 4.9. Percentages of extracted capacitances with relative errors above 5% for 28nm designs.

	Ratio-based		**Dimensions-based**		**Vertex-based**		**Rule-based**
	NN	**SVR**	**NN**	**SVR**	**NN**	**SVR**	
Outliers > 5% relative error	7.71%	7.84%	9.37%	10.13%	1.5%	1.9%	24.76%

It is worth mentioning that most of the outliers, with more than 5% relative error, that were generated using the proposed vertex-based NN and SVR models have very small capacitance values (<1e-4 fF).

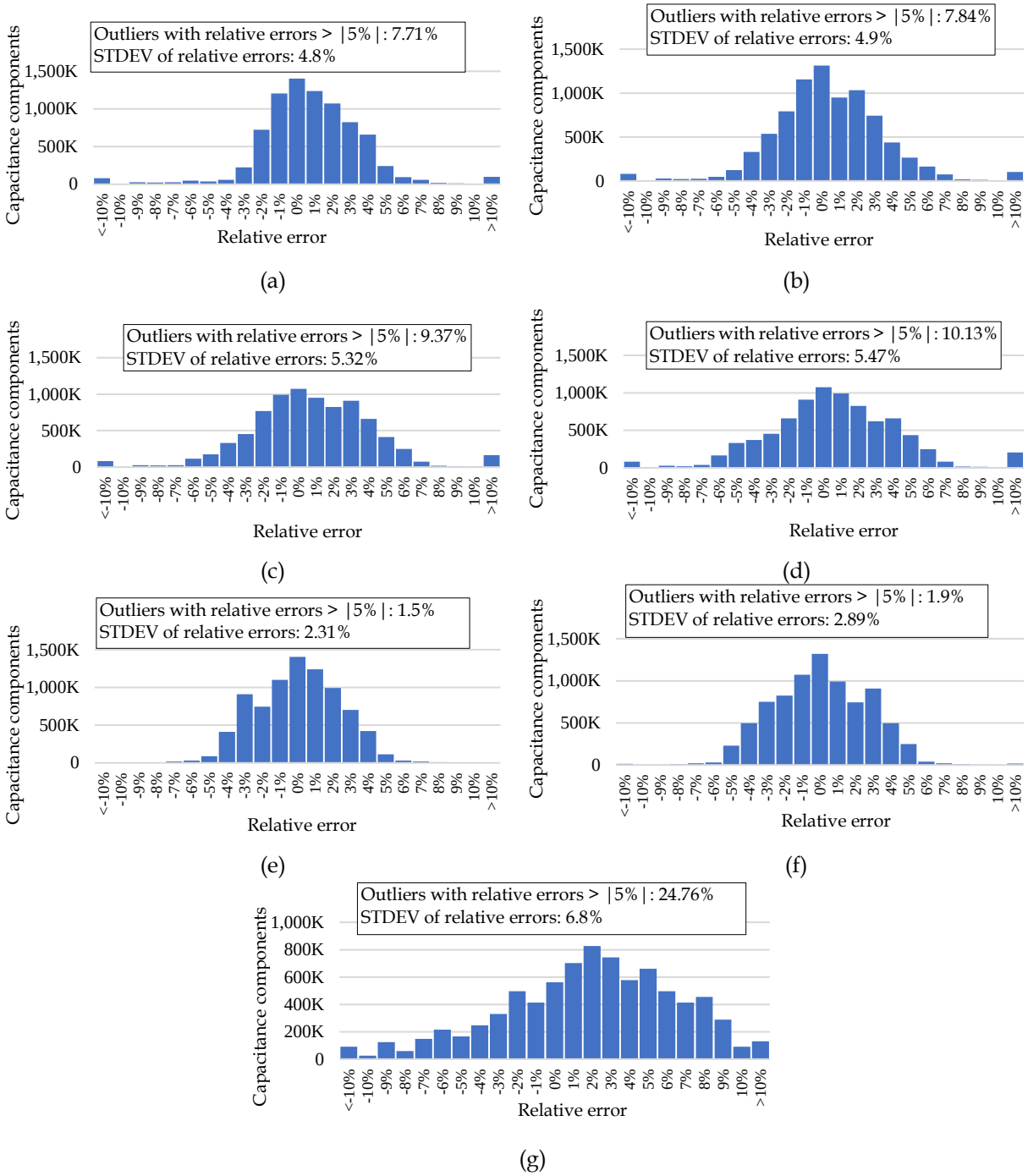


Fig. 4.18. Relative error histograms, as compared to Raphael 2D [75], of extracted capacitances for 2D cross-patterns of 28nm designs using the (a) ratio-based NN, (b) ratio-based SVR, (c) dimensions-based NN, (d) dimensions-based SVR, (e) vertex-based NN, (f) vertex-based SVR, and (g) Calibre rule-based cross-section models.

As for runtime comparisons, the total runtimes of extracting (i.e., computing) all cross-sections (i.e., 2.07M patterns) including the sensitivity formulas are shown in Table 4.10. The capacitance computations were done on a single CPU using Intel Xeon(R) E5-2680, 2.50GHz, and 16G of RAM. The results shows that the runtimes of the ratio-based NN, ratio-based SVR, dimensions-based NN, dimensions-based SVR, vertex-based NN, and vertex-based SVR models relative to rule-based models are 1.2, 1.09, 1.13, 1.06, 0.417, and 0.38, respectively. Therefore, the corresponding speeding ups as relative to rule-based models are 0.833, 0.917, 0.88, 0.943, 2.398, and 2.634, respectively. As a result, proposed vertex-based models are almost 2.5X faster than existing rule-based and other proposed models. Such a speeding up is achieved because the impact of systematic process variations is incorporated inside the proposed vertex-based models. Therefore, there is no need to apply additional computations in order to measure the impact of systematic process variations as in the other methods. Eventually, the vertex-based models (either NN or SVR models) managed to achieve high accuracy results as compared to existing rule-based cross-section models and other proposed models with an average speed up of 2.5X.

Table 4.10. The computations runtime of the proposed extraction models and existing rule-based models when executed over several designs of 28nm process nodes.

	Ratio-based		Dimensions-based		Vertex-based		Rule-based
	NN	**SVR**	**NN**	**SVR**	**NN**	**SVR**	
Computations runtime	19.27 hours	17.52 hours	18.23 hours	17.04 hours	6.7 hours	6.1 hours	16.07 hours
Relative runtime to rule-based models	1.2	1.09	1.13	1.06	0.417	0.38	1

4.6.2. Testing Designs of 14nm Process Nodes

The total number of generated models for each modeling method is 175. The generated models cover 175 different metal collections each includes 1 to 5 different metal layers. Each model (i.e., NN or SVR model) was trained over 30K cross-section patterns, where 21K patterns (70%) were used for the training set, and 9K patterns (30%) were used for the test set. Table 4.11 shows the training runtime of NN and SVR models using the three different proposed pattern representations.

Table 4.11. Training runtimes of the 2D cross-section parasitic models of 14nm process node.

	Ratio-based		Dimensions-based		Vertex-based	
	NN	**SVR**	**NN**	**SVR**	**NN**	**SVR**
Training runtime	19.4 hours	12.9 hours	18.7 hours	12.1 hours	21.7 hours	13.9 hours

As for NN models, the total training runtime of all models for the ratio-based, dimensions-based, and vertex-based representations are 19.4, 18.7, and 21.7 hours, respectively. As for SVR models, the total training runtime of all models for the ratio-based, dimensions-based, and vertex-based representations are 12.9, 12.1, and 13.9 hours, respectively. The training (i.e., models generation) runtimes can significantly improve by multi-processing.

Table 4.12 shows the test sets accuracy of the proposed models. The results show that the proposed models provide high accuracy values, where almost 98% of the extracted capacitances have relative errors below 5%.

Table 4.12. The accuracy and relative errors of test sets for the proposed models of 14nm process node.

	Ratio-based		Dimensions-based		Vertex-based	
	NN models	**SVR models**	**NN models**	**SVR models**	**NN models**	**SVR models**
Mean of relative errors	0.072%	0.074%	0.077%	0.081%	0.042%	0.061%
Standard deviation of relative errors	3.12%	3.55%	3.41%	3.65%	2.21%	2.75%
Outliers with relative error > 5%	2.07%	2.27%	2.21%	2.37%	1.77%	1.97%
Mean square error	0.00351	0.00406	0.00397	0.00496	0.00237	0.00316

As for testing the generated models on real design patterns of 14nm process node, the generated models were tested over cross-section patterns of three test chips including cache memory, DRAM, and VCO designs that were not included during the training processes. The total numbers of extracted cross-section patterns in cache memory, DRAM, and VCO designs are 630K, 915K, and 1.03M patterns, respectively. The corresponding total number of capacitance components are 2.8M, 4M, and 4.4M capacitances, respectively. Therefore, the total number of extracted cross-section patterns is 2.575M patterns, and the total number of extracted capacitances is 11.2M.

Fig. 4.19 shows histograms of relative errors covering all extracted capacitances across all tested 14nm designs using the ratio-based NN, ratio-based SVR, dimensions-based NN, dimensions-based SVR, vertex-based NN, vertex-based SVR, and rule-based extraction cross-section models. The accuracy comparisons show that the proposed vertex-based models (either NN or SVR models) provide high accuracy results as compared to existing rule-based cross-section models and other proposed models. Table 4.13 shows the percentages of extracted capacitance components with relative errors above 5% using the different extraction methods. It is worth mentioning that most of the outliers, with more than 5% relative error, that were generated from the proposed vertex-based models have very small capacitance values (<1e-4 fF).

Table 4.13. Percentages of extracted capacitance components with relative errors above 5% in 14nm designs.

	Ratio-based		**Dimensions-based**		**Vertex-based**		**Rule-based**
	NN	**SVR**	**NN**	**SVR**	**NN**	**SVR**	
Outliers > 5% relative error	12.6%	14.3%	12.4%	14.7%	1.97%	2.19%	25.7%

As for runtime comparisons, the total runtimes of extracting (i.e., computing) all cross-section patterns (i.e., 2.575M patterns) including the sensitivity formulas are shown in Table 4.14. The capacitance computations were done on a single CPU using Intel Xeon(R) E5-2680, 2.50GHz, and 16G of RAM. The results shows that the runtimes of the ratio-based NN, ratio-based SVR, dimensions-based NN, dimensions-based SVR, vertex-based NN, and vertex-based SVR models relative to rule-based models are 1.2, 1.093, 1.19, 1.068, 0.415, and 0.3914, respectively. Therefore, the corresponding speeding ups as relative to rule-based models are 0.83, 0.91, 0.84, 0.936, 2.41, and 2.55, respectively. As a result, proposed vertex-based models are almost 2.45X faster than existing rule-based and other proposed models. Such a speeding up is achieved because the impact of systematic process variations is incorporated inside the proposed vertex-based models. Therefore, there is no need to apply additional computations in order to measure the impact of systematic process variations as in the other methods.

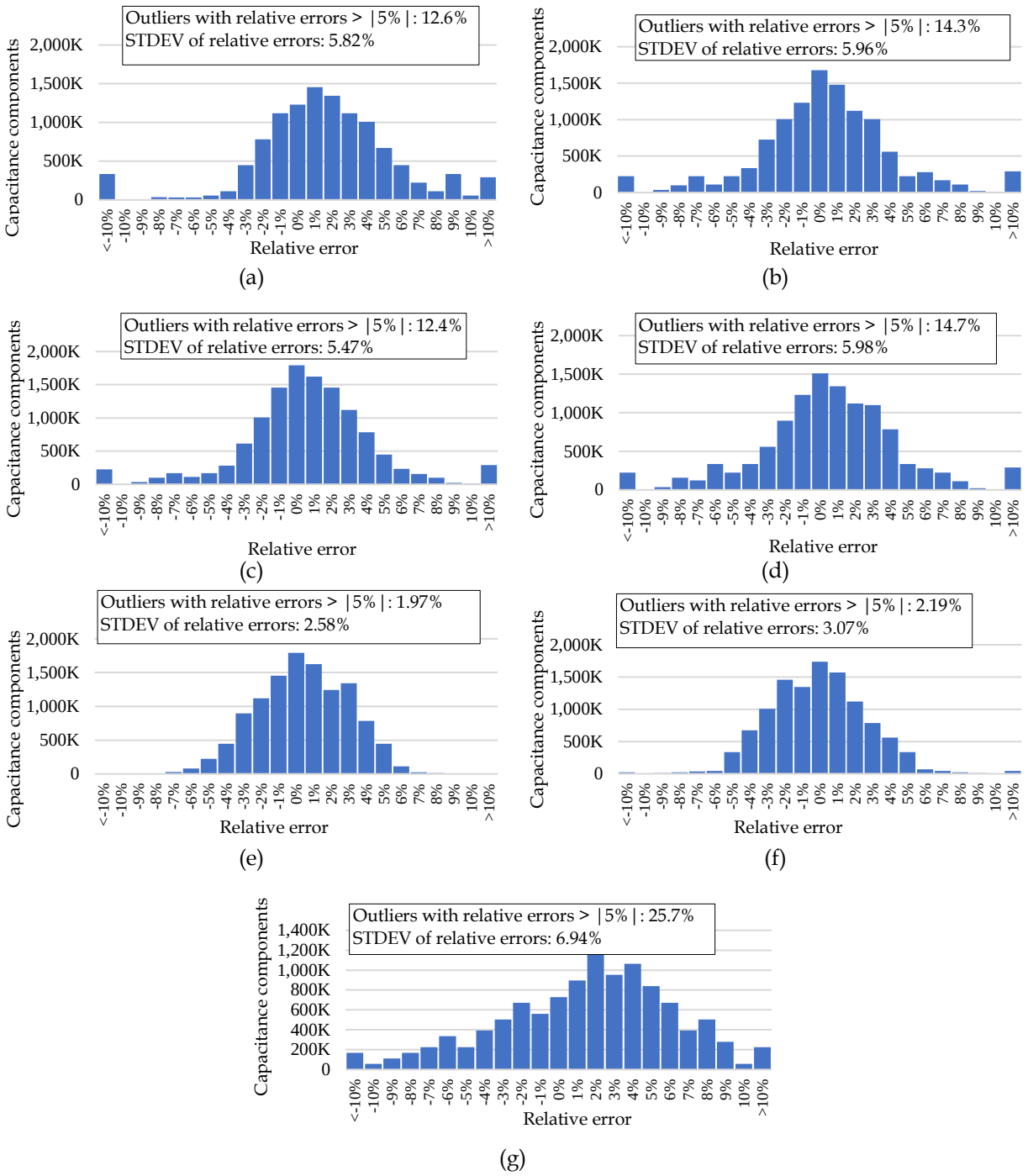


Fig. 4.19. Relative error histograms, as compared to Raphael 2D [75], of extracted capacitances for 2D cross-patterns of 14nm designs using the (a) ratio-based NN, (b) ratio-based SVR, (c) dimensions-based NN, (d) dimensions-based SVR, (e) vertex-based NN, (f) vertex-based SVR, and (g) Calibre rule-based cross-section models.

Table 4.14. The computations runtime of the proposed extraction models and existing rule-based models when executed over several designs of 14nm process nodes.

	Ratio-based		Dimensions-based		Vertex-based		Rule-based
	NN	**SVR**	**NN**	**SVR**	**NN**	**SVR**	
Computations runtime	24.1 hours	21.9 hours	23.8 hours	21.4 hours	8.32 hours	7.84 hours	20.03 hours
Relative runtime to rule-based models	1.2	1.093	1.19	1.068	0.415	0.3914	1

4.6.3. Testing Designs of 7nm Process Nodes

The total number of generated models for each modeling method is 231. The generated models cover 231 different metal collections each includes 1 to 5 different metal layers. Each model (i.e., NN or SVR model) was trained over 30K cross-section patterns, where 21K patterns (70%) were used for the training set, and 9K patterns (30%) were used for the test set. Table 4.15 shows the training runtime of NN and SVR models using the three different proposed pattern representations.

Table 4.15. Training runtimes of the 2D cross-section parasitic models for 7nm process node.

	Ratio-based		Dimensions-based		Vertex-based	
	NN	**SVR**	**NN**	**SVR**	**NN**	**SVR**
Training runtime	22.7 hours	13.2 hours	21.5 hours	12.9 hours	23.01 hours	15.03 hours

As for NN models, the total training runtime of all models for the ratio-based, dimensions-based, and vertex-based representations are 22.7, 21.5, and 23.01 hours, respectively. As for SVR models, the total training runtime of all models for the ratio-based, dimensions-based, and vertex-based representations are 13.2, 12.9, and 15.03 hours, respectively. The training (i.e., models generation) runtimes can significantly improve by multi-processing. It is worth mentioning that the models were generated only once for each process node. After that, the generated models are used numerous times by parasitic extraction tools.

Table 4.16 shows the test sets accuracy of NN and SVR models. The results show that the proposed models provide high accuracy values, where almost 97% of the extracted capacitances have relative errors below 5%.

Table 4.16. The accuracy and relative errors of test sets for the proposed models of 7nm process node.

	Ratio-based		Dimensions-based		Vertex-based	
	NN models	**SVR models**	**NN models**	**SVR models**	**NN models**	**SVR models**
Mean of relative errors	0.067%	0.091%	0.081%	0.089%	0.065%	0.083%
Standard deviation of relative errors	3.52%	3.67%	3.76%	3.9%	3.02%	3.17%
Outliers with relative error > 5%	2.71%	3.1%	3.01%	3.2%	2.31%	2.6%
Mean square error	0.0059	0.0068	0.0062	0.0072	0.0052	0.0063

As for testing the generated models on real design patterns of 7nm process node, the generated models were tested over cross-section patterns of two test chips including cache memory (CM) and VCO designs that were not included during the training processes. The total numbers of cross-section patterns of cache memory and VCO designs are 920K and 1.17M patterns, respectively. The corresponding total number of capacitance components are 4.1M and 5M capacitances, respectively. Therefore, the total number of extracted cross-section patterns is 2.09M patterns, and the total number of extracted capacitances is 9.1M.

Fig. 4.20 shows histograms of relative errors covering all extracted capacitances across all tested 14nm designs using the ratio-based NN, ratio-based SVR, dimensions-based NN, dimensions-based SVR, vertex-based NN, vertex-based SVR, and rule-based extraction cross-section models. The accuracy comparisons show that the proposed vertex-based models (either NN or SVR models) provide high accuracy results as compared to existing rule-based cross-section models and other proposed models. Table 4.17 shows the percentages of extracted capacitance components with relative errors above 5% using the different extraction methods. It is worth mentioning that most of the outliers, with more than 5% relative error, that were generated from the proposed vertex-based models have very small capacitance values (<1e-4 fF).

Table 4.17. Percentages of extracted capacitance components with relative errors above 5% in 7nm designs.

	Ratio-based		Dimensions-based		Vertex-based		Rule-based
	NN	**SVR**	**NN**	**SVR**	**NN**	**SVR**	
Outliers > 5% relative error	23.3%	23.9%	24.1%	24.7%	2.53%	3.2%	28.9%

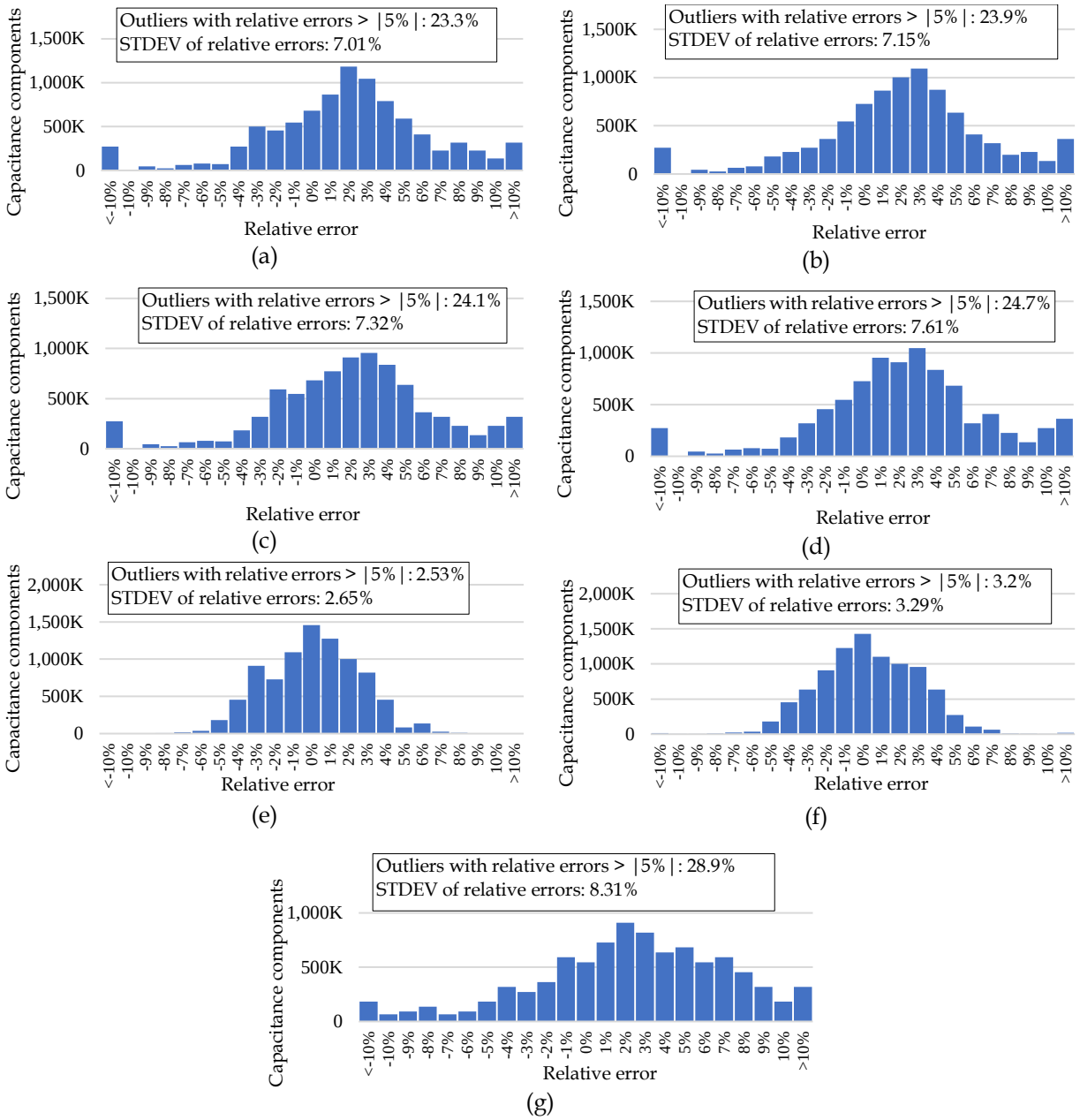


Fig. 4.20. Relative error histograms, as compared to Raphael 2D [75], of extracted capacitances for 2D cross-patterns of 7nm designs using the (a) ratio-based NN, (b) ratio-based SVR, (c) dimensions-based NN, (d) dimensions-based SVR, (e) vertex-based NN, (f) vertex-based SVR, and (g) Calibre rule-based cross-section models.

As for runtime comparisons, the total runtimes of extracting (i.e., computing) all cross-section patterns (i.e., 2.09M patterns) including the sensitivity formulas are shown in Table 4.18. The capacitance computations were done on a single CPU using Intel Xeon(R) E5-2680, 2.50GHz, and 16G of RAM. The results shows that the runtimes of the ratio-based NN, ratio-based SVR, dimensions-based NN, dimensions-based SVR,

vertex-based NN, and vertex-based SVR models relative to rule-based models are 1.196, 1.065, 1.162, 1.044, 0.426, and 0.419, respectively. Therefore, the corresponding speeding ups as relative to rule-based models are 0.836, 0.94, 0.86, 0.958, 2.346, and 2.39, respectively. As a result, proposed vertex-based models are almost 2.35X faster than existing rule-based and other proposed models. Such a speeding up is achieved because the impact of systematic process variations is incorporated inside the proposed vertex-based models. Therefore, there is no need to apply additional computations in order to measure the impact of systematic process variations as in the other methods.

Table 4.18. The computations runtime of the proposed extraction models and existing rule-based models when executed over several designs of 14nm process nodes.

	Ratio-based		Dimensions-based		Vertex-based		Rule-based
	NN	**SVR**	**NN**	**SVR**	**NN**	**SVR**	
Computations runtime	19.46 hours	17.31 hours	18.9 hours	16.98 hours	6.93 hours	6.81 hours	16.26 hours
Relative runtime to rule-based models	1.196	1.065	1.162	1.044	0.426	0.419	1

4.6.4. Statistical Tests

Nonparametric statistical tests were performed to test the significant difference in performance (i.e., accuracy) between each two models using Wilcoxon signed-ranks test [85]. Since the vertex-based NN models provided better accuracy results as compared to other models. The Wilcoxon signed-rank tests are performed to compared test the significant difference in accuracy between the vertex-based models and all other models. In our case, the null hypothesis indicates a lack of a significant difference between the two tested models. The null hypothesis will be rejected if the p-value is less than 0.05 (p-value < 0.05). The mean square error (MSE) was used as a performance metric to help in performing statistical tests. MSE values were obtained for the four extraction models over 13 datasets using Raphael, 2D field-solver, as a reference, as shown in Table 4.19. Fig. 4.21 shows the differences in mean square errors between the rule-based model and each proposed model across different test designs. The figure shows that the vertex-based models provide better results as compared to all other models. Moreover, the vertex-based models show outstanding results as compared to the other proposed

models when the impact of systematic process variations on parasitic capacitances increases, mainly in advanced nodes.

Table 4.19. Accuracy comparisons in terms of mean square errors for rule-based, ratio-based, the proposed SVR, and the proposed NN models.

Dataset	Mean square error (MSE)						
	Calibre rule-based [19]	Ratio-based NN	Ratio-based SVR	Dimensions-based NN	Dimensions-based SVR	Vertex-based NN	Vertex-based SVR
RO (28nm)	0.00710	0.00231	0.00320	0.00265	0.00337	0.00173	0.00141
DRAM(28nm)	0.00930	0.00213	0.00260	0.00192	0.00221	0.00201	0.00219
SRAM(28nm)	0.00120	0.00160	0.00115	0.00219	0.00223	0.00233	0.00173
DAC(28nm)	0.00760	0.00432	0.00471	0.00417	0.00471	0.0019	0.00201
VCO(28nm)	0.00314	0.00470	0.00530	0.00427	0.00527	0.00211	0.00379
RO(14nm)	0.00301	0.00217	0.00207	0.00218	0.00209	0.00237	0.00296
DAC(14nm)	0.00430	0.00523	0.00514	0.00596	0.00572	0.0019	0.00327
DRAM(14nm)	0.00810	0.00455	0.00473	0.00512	0.00538	0.00385	0.00316
VCO(14nm)	0.01300	0.00710	0.00722	0.00801	0.00891	0.0031	0.00411
RO(7nm)	0.05010	0.10810	0.09370	0.10783	0.09730	0.00387	0.00510
SRAM(7nm)	0.05230	0.05030	0.05410	0.0502	0.0532	0.0049	0.00471
CM(7nm)	0.08150	0.07711	0.08132	0.07984	0.08041	0.0059	0.00720
VCO(7nm)	0.10300	0.13800	0.13910	0.1337	0.1401	0.0061	0.00811

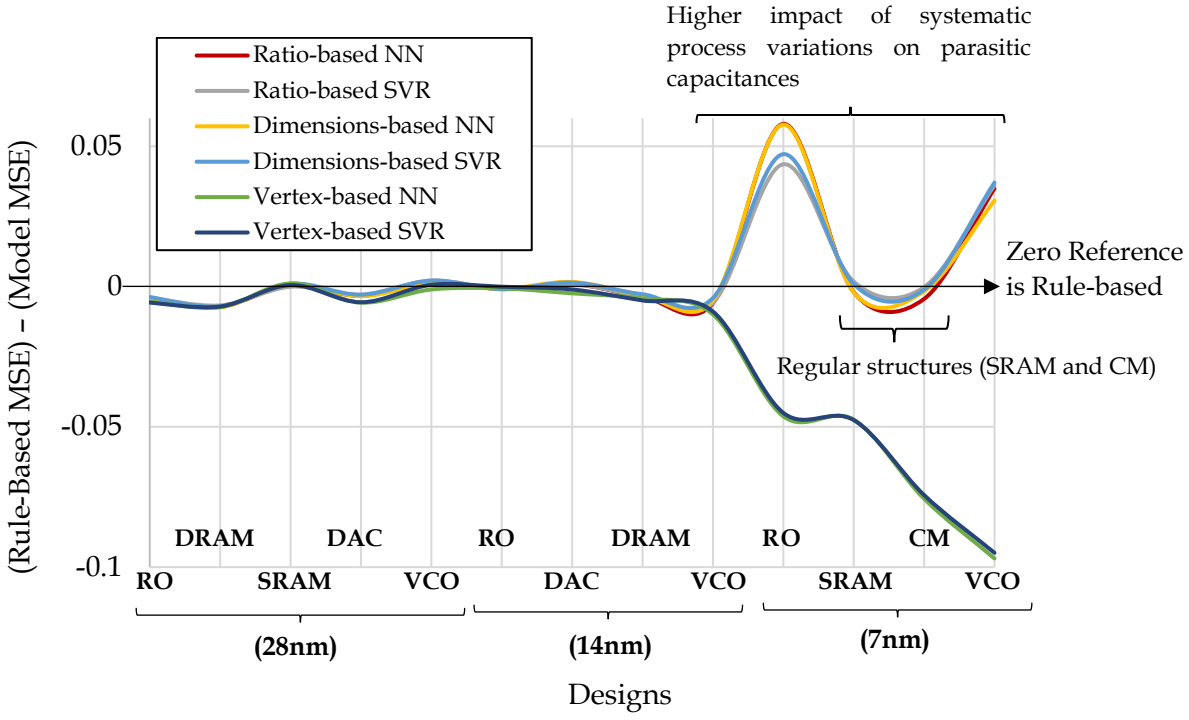


Fig. 4.21. A graph showing the mean square errors of all proposed cross-section parasitic capacitance models across different designs taking the mean square errors of rule-based models as references.

Table 4.20 shows statistical comparisons using Wilcoxon signed-rank tests. The table shows the *p*-value and *z*-value for each paired comparison test. Also, the table shows the sum of positive ranks (SPR) and sum of negative ranks (SNR) for each paired comparison test. The comparisons show that there is no significant difference between the proposed vertex-based NN and vertex-based SVR models as the *p*-value is greater than 0.05. However, the results show significant differences (i.e., rejecting the null hypothesis) between the proposed models and each compared extraction model as the *p*-values are less than 0.05.

Table 4.20. Paired comparisons using Wilcoxon signed-rank test (two-tailed) to test the significant difference between each two models, where the mean square error, against a field-solver, is used as a performance metric.

Pairwise comparison		SNR	SPR	*z*-value	*p*-value	Significance
Model1	**Model2**					
Vertex-based NN	**Vertex-based SVR**	71	20	-1.7821	0.07508	No
Vertex-based NN	**Calibre rule-based** [19]	88	3	-2.9701	0.00298	Yes
Vertex-based NN	**Ratio-based NN**	84	7	-2.6906	0.00714	Yes
Vertex-based NN	**Ratio-based SVR**	86	5	-2.8304	0.00466	Yes
Vertex-based NN	**Dim-based NN**	85	6	-2.7605	0.00578	Yes
Vertex-based NN	**Dim-based SVR**	87	4	-2.9003	0.00374	Yes

SNR: Sum of negative ranks; SPR: Sum of positive ranks.

4.7. Conclusion

A novel modeling methodology for interconnect parasitic capacitances is developed for rule-based extraction tools using machine learning methods. The proposed methodology managed to overcome several problems in rule-based extraction tools such as handling systematic process variations, high pattern mismatches, and limited pattern coverages. The proposed methodology creates cross-section compact models for a certain process technology node. Such compact models predict the parasitic coupling capacitances between metal polygons on a given 2D cross-section layout pattern considering the impact of systematic process variations. The modeling methodology process starts with processing process stack specifications to identify the main characteristics of layout input patterns, such as pattern's size, the maximum number of metal layers in a pattern, handling multi-dielectric stacks,

systematic process variations, and the maximum number of polygons in a pattern. The input of the compact models is a given cross-section pattern including the required capacitances and the corresponding systematic process variations. Three different pattern representations are introduced. First, a ratio-based representation. Second, a dimensions-based representation. Third, a novel vertex-based pattern representation that considers systematic process variations as a part of the geometrical characteristics of a given pattern. The compact models are implemented using two different machine learning methods: neural networks and support vector regression methods. The proposed methodology is tested over thirteen real designs of 28nm, 14nm, and 7nm process nodes with more than 6.7M interconnect patterns. The generated compact models are faster than traditional rule-based models by 2.5X. Also, they managed to achieve outstanding results as compared to field-solvers and rule-based cross-section models, where the average relative error of the generated models is < 0.15% and the standard deviation of relative errors is < 3.31%.

Chapter 5

Machine Learning Compact Models for Middle End of Line Parasitic Capacitances

The massive improvement in the semiconductor industry enabled the integration of more systems and functionalities on the same chip. Such integrations are empowered by the feature scaling and the introduction of FINFETs. The continuous scaling down of technologies resulted in parasitic effects such as interconnect resistances and capacitances to dominate circuit performances, increasing the importance of interconnect parasitic extraction. The parasitic effects that are associated with the Middle-End-Of-Lines (MEOL), which are the interconnects connecting devices to upper metal layers, have a major impact on circuit performances in advanced process nodes as shown in [53]–[56], [60]. Fig. 5.1 shows some MEOL parasitic capacitances in case of FINFETs and MOSFETs, respectively. Usually, commercial parasitic extraction tools use field-solvers to extract the MEOL parasitic elements to achieve high accuracy levels. However, field-solvers are slow, have a limited capacity, and consume a lot of computational resources [11].

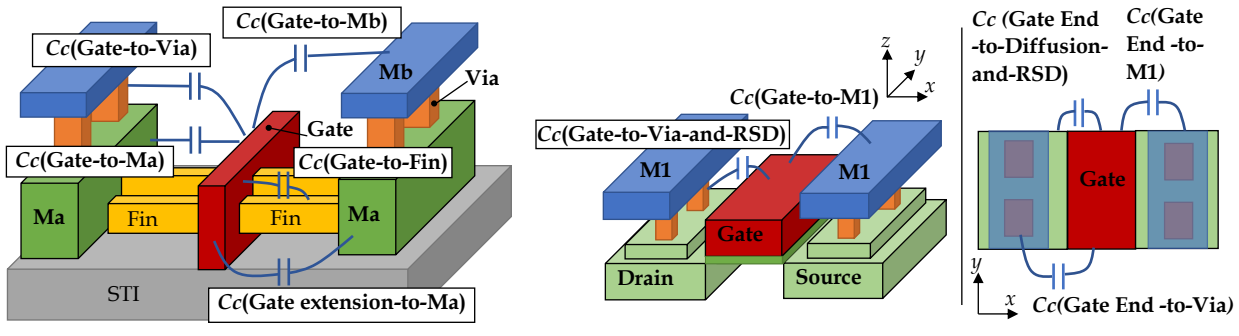


Fig. 5.1. Some MEOL parasitic capacitances around (a) typical FINFET (Sun *et al.*, 2015 [57]) and (b) MOSFET structures.

A novel MEOL parasitic capacitance modeling methodology using machine learning is introduced. The proposed modeling methodology aims to provide a compact multi-dimensional model for each device (i.e., transistor) type (e.g., N-type and P-type) for a certain process node. Each compact model would replace thousands of pre-characterized patterns. The inputs of the compact model are MEOL geometrical properties, whereas the output is a certain pre-identified coupling capacitance component. The proposed methodology uses a novel geometry-based representation to represent the required features of MEOL patterns. Therefore, the geometry-based representations of MEOL patterns are used as inputs to the compact models. Unlike existing pre-characterized models, the proposed methodology can efficiently decrease pattern mismatches and handle many varieties of MEOL patterns. Hence, the accuracy and runtime of MEOL parasitic extraction processes are significantly improved.

As shown in Fig. 5.2, the proposed modeling methodology consists of three main phases in order to create parasitic capacitance machine learning models for MEOL patterns. The first phase aims to prepare training data. This is done by generating many MEOL patterns, applying systematic process variations (e.g., etching), and extracting parasitic capacitances of MEOL patterns using a field-solver to obtain reference parasitic capacitance numbers. The second phase aims to extract features of MEOL patterns. This is done by representing each MEOL pattern using a novel geometry-based representation. Eventually, the third phase aims to train and create machine learning models for MEOL parasitic capacitances.

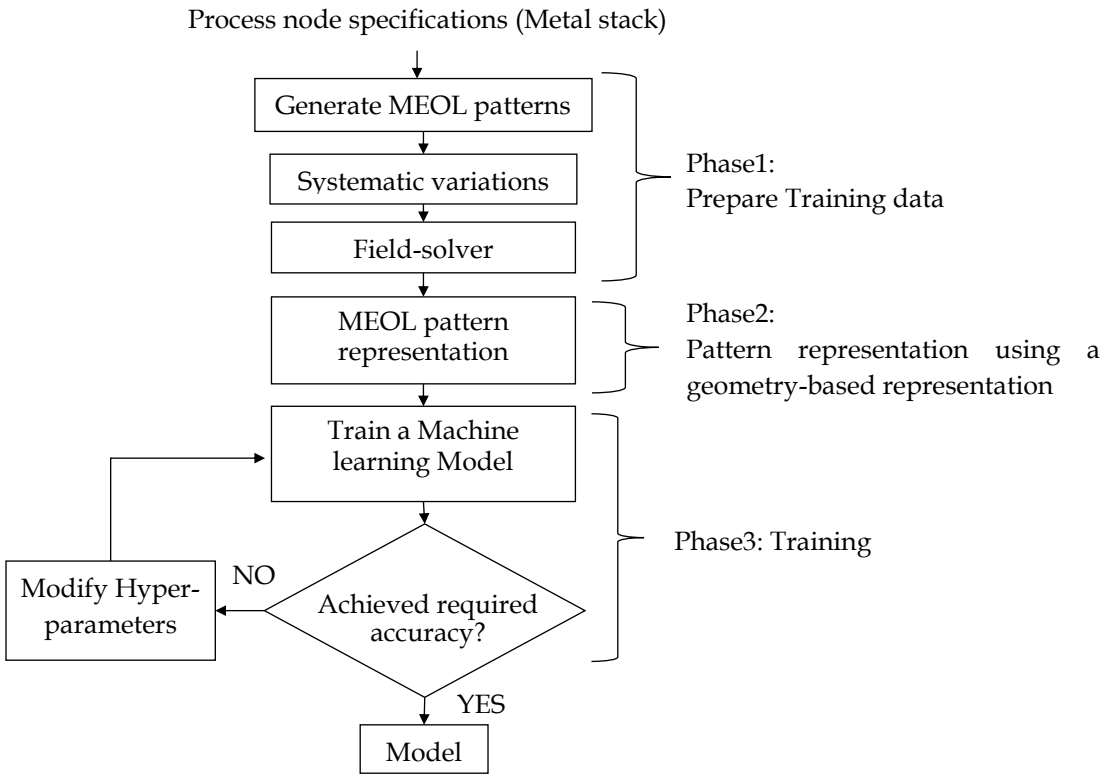


Fig. 5.2. The process of implementing MEOL parasitic capacitance models.

5.1. Generate Training MEOL Patterns

The training patterns consists of 25K MEOL patterns for each device (transistor) type in a certain process node. Each MEOL pattern consists of several metal layers that construct gate, source, and drain terminals of a certain device as shown in Fig. 5.1. They are created by scanning many real layout designs including cache memory, digital to analog converter, and voltage controlled oscillator layouts. Also, additional MEOL patterns were randomly generated covering different dimensions from 1X to 10X of the minimum technological dimensions. To obtain realistic and practical MEOL training patterns, two main factors need to be considered: the multi-dielectric environment and the multi finger devices.

5.1.1. Multi-Dielectric Environment

Multi-dielectric process stacks became very common in advanced process technology nodes, where each metal layer may overlap with multiple planar and conformal dielectrics each with a different dielectric constant (ε_r). The use of multi-

dielectric characteristics as inputs to the capacitance models would significantly complicate the modeling process and generate less accurate models with a slow computational runtime. In order to consider the multi-dielectric characteristics without using them as inputs to the parasitic capacitance models, each process node (i.e., metal stack) should have its own models. Moreover, since each device type, in a certain process node, may overlap with different dielectrics, each device type must also have its own parasitic capacitance model. As a result, each device type per process node has a capacitance model.

5.1.2. Multi-Finger Devices

The multi-finger devices contain multiple gates and MEOL patterns. Therefore, the MEOL training patterns should consider the position of each gate within the corresponding multi-finger device as the spatial position of each gate may impact the calculations of MEOL parasitic coupling capacitances. Fig. 5.3 shows three identical gates with a different spatial position within a multi-finger device. Fig. 5.3 (a) shows a single gate MOSFET, Fig. 5.3 (b) shows a gate MOSFET on the edge of a multi-finger device, whereas Fig. 5.3 (c) shows an intermediate gate MOSFET as a part of a multi-finger device. The results show that the coupling capacitance between the source and gate varies based on the location of the gate. Fig. 5.4 shows some examples of MEOL patterns for both FINFET and MOSFET technologies.

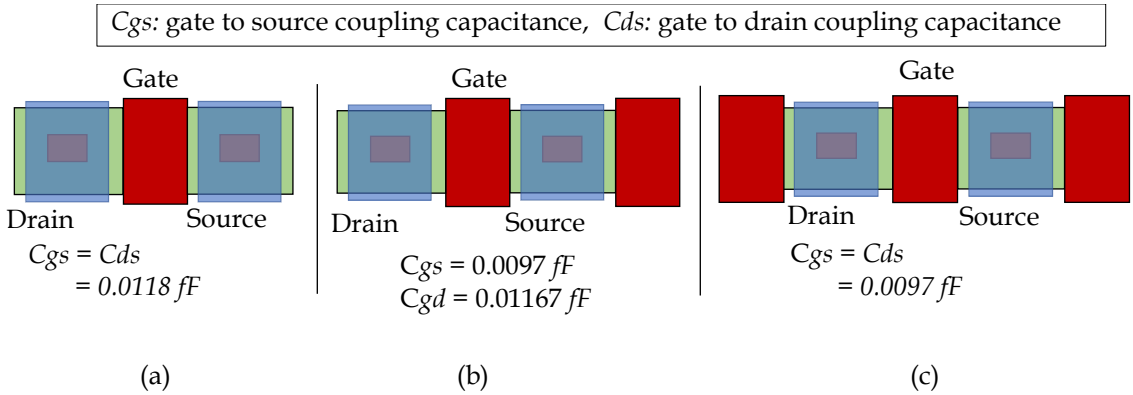


Fig. 5.3. Examples of MEOL coupling capacitances in case of (a) a single MOSFET, (b) a gate at the edge of a multi-finger device, and (c) a gate in the middle of a multi-finger device. The experiment used 28nm node with minimum dimensions. Calibre xACT3D is used to extract MEOL parasitic capacitances.

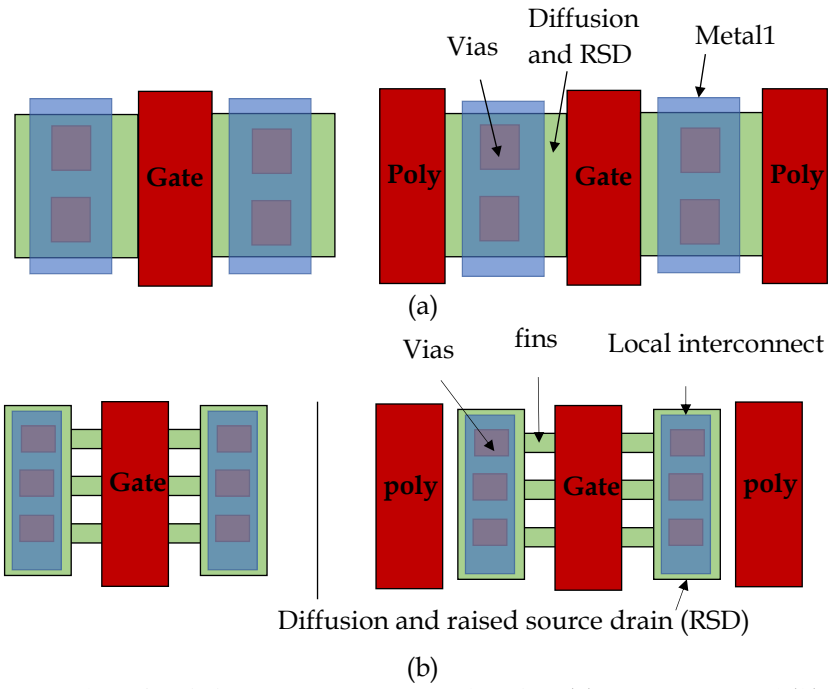


Fig. 5.4. Examples of training MEOL patterns showing (a) MOSFETs and (b) FINFETs.

5.2. Generate Reference Parasitic Capacitances

After generating MEOL training patterns, their parasitic capacitance reference numbers are extracted using Calibre xACT3D, a 3D field-solver [32]. The reference numbers are used to train MEOL parasitic capacitance machine learning models.

5.3. MEOL Pattern Representation

In order to create an efficient MEOL parasitic capacitance model for each device type, the inputs of each model must provide sufficient information about the input MEOL pattern that help in describing the geometries and identifying the required capacitance component. Therefore, the inputs must include geometrical properties of the whole MEOL pattern, aggressor polygons, and victim polygons. The geometrical properties of the whole pattern help in determining the pattern structures, whereas the geometrical properties of aggressor and victim polygons help in identifying the required parasitic coupling capacitance component, which is between the aggressor and victim polygons. The geometrical properties are represented by feature vectors. Hence, there is a feature vector for the whole pattern, a feature vector for aggressor polygons, and a feature vector for victim polygons. All feature vectors have the same size. Eventually, the three feature

vectors are combined together conforming a final feature vector that represents the given MEOL pattern and the required capacitance component. Fig. 5.5 shows an example of the proposed MEOL feature vector that is used to predict a coupling capacitance between gate and source nets.

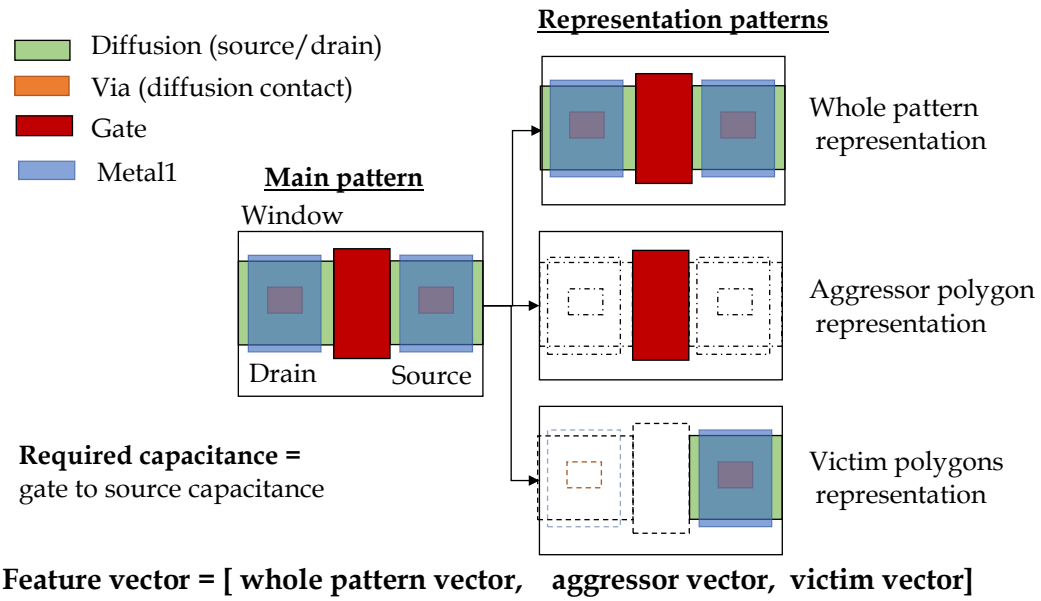


Fig. 5.5. An example of the proposed MEOL feature vector that is used to predict gate to source coupling capacitance in a MOSFET.

A geometry-based feature representation is proposed to extract geometrical properties of MEOL patterns including a whole pattern, aggressor polygons, and victim polygons. The proposed geometry-based representation has four main steps. The first step aims to scan MEOL patterns and fracture their polygons into quadrilateral or triangular polygons, e.g., rectangles and triangles. The second step aims to represent each polygon (i.e., fractured polygon) by the spatial position of its vertices, where the spatial position is measured from the center of the corresponding gate. In other words, the spatial position of a vertex is its displacement from the center of the corresponding gate. The third step aims to represent vias and fins. As for vias, each set of symmetrical vias is grouped into a cluster. The spatial position (i.e., displacements from the center of the corresponding gate) and the dimensions of via clusters are used as a representation of vias. On the other hand, the fins are represented by fin width, fin spacing, and fins count.

Eventually, the fourth step aims to concatenate all feature vectors together creating a final input vector that is used as an input to the required machine learning model. It is worth mentioning that such geometrical representation captures non-Manhattan geometries. The flow is described below.

5.3.1. Fracturing Polygons

In MEOL patterns, some polygons may have more than four vertices, e.g., T-shaped polygons. In such cases, the polygons are fractured into quadrilateral or triangular polygons. The fracturing is done by scanning the polygons in the x-direction (i.e., perpendicular to gate). In case of capturing any polygon with more than 4 vertices, the polygon is fractured vertically as shown in Fig. 5.6. Then, the polygons are scanned in the y-direction (i.e., parallel to gate). In case of capturing any polygon with more than 4 vertices, the polygon is fractured horizontally.

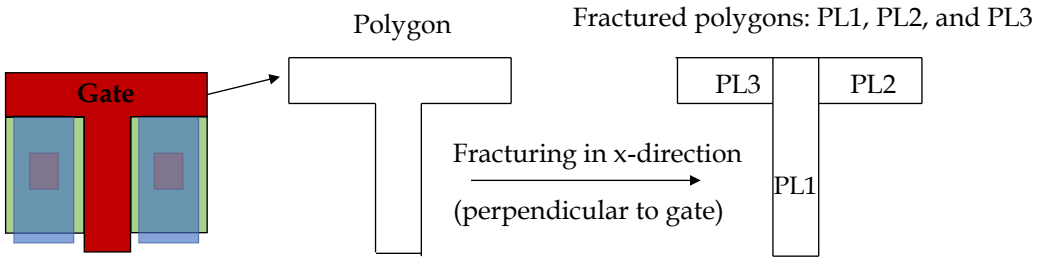
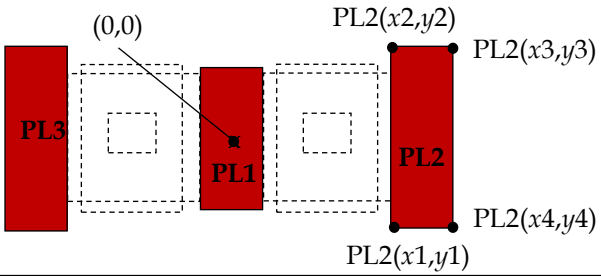


Fig. 5.6. An example of fracturing a polygon in x-direction.

5.3.2. Creating a Feature Vector for Each MEOL Layer

After fracturing all MEOL polygons for each layer in a certain MEOL pattern. Each polygon is represented by the spatial position of its four vertices, where the spatial position is measured from the center of the corresponding gate in a given MEOL pattern. Therefore, each polygon is represented by a vector of eight values that represent the x and y locations of each vertex. For example, an MEOL layer with 5 polygons is represented by a vector of 40 indices as shown in Fig. 5.7. In case of triangular polygons, they are represented by 4 vertices, but the last two vertices have the same position.



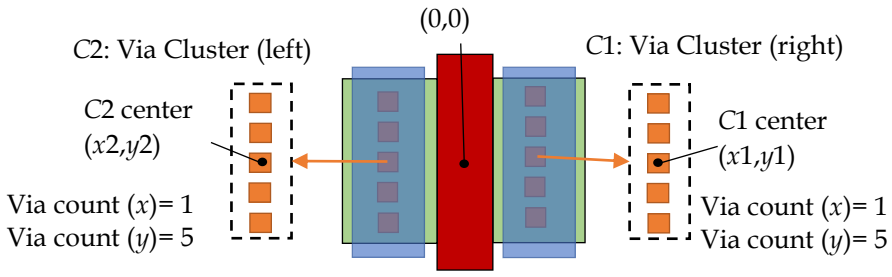
Gate/poly feature Vector =

[x1, y1, x2, y2, x3, y3, x4, y4	For PL1: Center polygon
, x1, y1, x2, y2, x3, y3, x4, y4	For PL2: 1st right polygon
, x1, y1, x2, y2, x3, y3, x4, y4]	For PL3: 1st left polygon

Fig. 5.7. An Example of representing MEOL layer polygons using a vector of vertices.

5.3.3. Representing Vias and Fins

MEOL patterns may contain many vias. Representing all vias with their vertices would significantly increase the feature vector size. Hence, vias are grouped into clusters, where each set of symmetrical vias are clustered together conforming a matrix (or a vector) of vias. Each cluster is represented by six parameters: the spatial position of its center (in *x* and *y* axes), the number of vias in x-direction, the number of vias in *y*-direction, the width of the corresponding vias, and the spacing between the corresponding vias as shown in Fig. 5.8. On the other hand, fins are represented by three parameters: fin width, fin spacing, and fins count.



Via feature vector =

[*x*1,*y*1, via count(x), via count (*y*), via width, via spacing	For *C*1
, *x*2,*y*2, via count(x), via count (*y*), via width, via spacing]	For *C*2

Fig. 5.8. An example showing the feature vector of MEOL vias.

5.3.4. Maximum Number of MEOL Polygons

Obtaining the maximum number of polygons for each layer (e.g., diffusion, poly, and metal1) in MEOL patterns helps in identifying the input vector size of MEOL models. The maximum number of polygons for each MEOL layer is identified by checking the corresponding layout design rules and by observing many fractured MEOL training patterns. In case of MOSFETs (28nm process node), the maximum numbers of fractured polygons for gate, field-poly, diffusion (including raised source drain (RSD)), and metal1 layers are 7, 7, 6, and 6, respectively. As for vias, the maximum number of via clusters is 4. On the other hand, in case of FINFETs (7nm process node), the maximum numbers of fractured polygons for gate, field-poly, diffusion, raised source drain (RSD), device local interconnect, and field local interconnect layers are 3, 6, 2, 2, 6, and 6, respectively. As for vias, the maximum number of via clusters is 4. All previous steps are performed for the whole MEOL pattern, aggressor polygons, and victim polygons. Eventually, the final input vector size is calculated by:

$$\text{input vector size of MEOL model} = 3\left(V + F + \sum_{i=1}^{n} vector\ size(i)\right), \tag{5.1}$$

where n is the number of MEOL layers, V is the via vector size, whereas F is the fin vector size. Fig. 5.9 shows an example of the final input feature vector for a certain MEOL pattern.

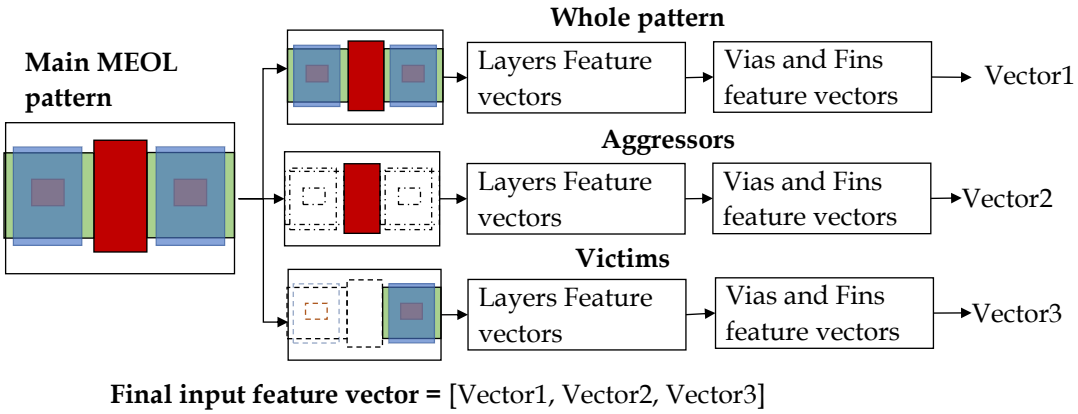


Fig. 5.9. An example showing the generating of the final input vector that is used for MEOL parasitic capacitance extraction.

5.4. MEOL Parasitic Capacitance Models

Two modeling approaches are used to create MEOL parasitic capacitance models. The first approach uses Neural-Networks (NN) models, whereas the second approach uses Support Vector Regressions (SVR). There is a model for each device type in a certain process node. The inputs of both methods are the concatenated feature vectors that include geometrical representations of an MEOL pattern, aggressor polygons, and victim polygons as shown in Fig. 5.9. The models are implemented based on Google Tensor flow libraries, [83], using Python [84].

5.4.1. Neural Networks Model

The NN architectures are obtained using neural architecture search [86]. As for the search space, the number of hidden layers varies from 1 to 4, the activation function of each layer alternates between RELU and tanh, the number of neurons per hidden layer varies from $n/6$ to n, where n is the input vector size, and the initialization parameter of each layer alternates between glorot_normal and he_normal. A fully connected NN is considered for MEOL models. A grid search is used as a search strategy. The lowest mean square error of test sets with smallest architecture is used as an evaluation method. Table 5.1 summarizes NN hyper-parameters. Fig. 5.10 shows the most common NN architecture that is obtained for MEOL patterns in 28nm and 7nm process nodes. It consists of three hidden layers with $n/4$, $n/4$, and $n/5$ neurons, respectively. The activation function of each layer is tanh, tanh, and RELU, respectively. The initialization parameters of the layers are golort_normal, golort_normal, and he_normal, respectively.

Table 5.1. Training hyper parameters of MEOL parasitic capacitances NN models.

Parameter	Value
Training set	80% (20K patterns)
Test set	20% (5K patterns)
Batch size	500
Validation set	10% (2K patterns)
Loss function	Mean square error
Learning rate	1e-3
Batch normalization	YES
Epochs	500
Optimizer	Adam

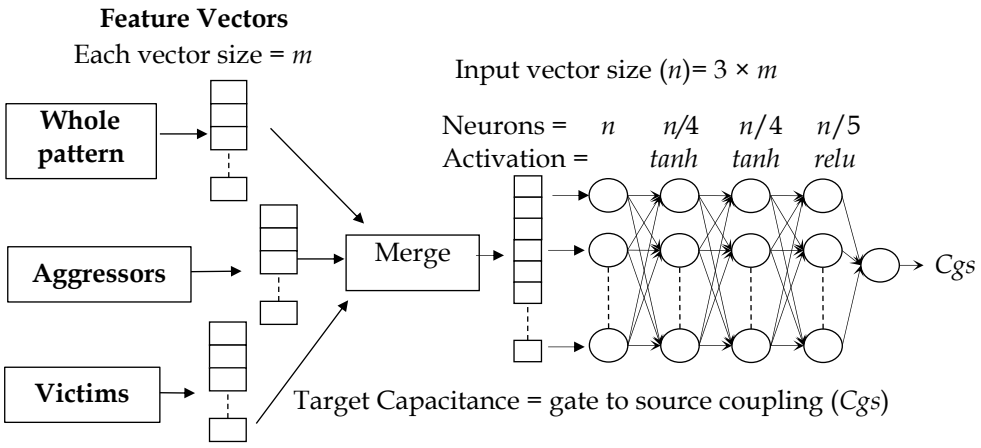


Fig. 5.10. An example of the most common NN architecture for MEOL parasitic capacitance extraction.

5.4.2. Support Vector Regressions

A grid search is used to obtain SVR hyper-parameters. The kernel is set to the radial basis function because MEOL parasitic capacitance extraction is a non-linear problem. As of the search space, the regularization parameter (*C*) varies from 1 to 20, gamma varies from 0.1 to 1, and epsilon varies from 0.05 to 0.5. The lowest mean square error of test sets is used as an evaluation method. The most common hyper parameters that are obtained for MEOL patterns in 28nm and 7nm process nodes are: 8 as a regularization parameter (*C*), 0.3 as a gamma, and 0.1 as an epsilon.

5.5. Experimental Results

The testing covered two process nodes that include 28nm and 7nm. The testing methodology aims to select MEOL patterns from several designs, which are not part of the training sets, and extract the selected devices using the proposed MEOL NN models, the proposed MEOL SVR models, Calibre xACT3D as a reference 3D field-solver [32], Calibre PEX as a rule-based extraction tool for 28nm process node [19], and Calibre xACT as a hybrid extraction tool [87], which is a hybrid extraction tool that extracts MEOL layers using a 3D field-solver, for 7nm process node. All errors are measured relative to Calibre xACT3D, 3D field-solver.

5.5.1. Testing Results on 28nm Process Node

The proposed MEOL modeling methodology was tested over 28nm process node, which is a MOSFET technology. The MEOL layers of 28nm process are diffusion, via (i.e., diffusion contact), gate, field poly, and metal1 layers. Two different MEOL models were implemented to cover two different device types (i.e., NMOS and PMOS). The total number of training MEOL patterns is 50K. They were obtained from different designs that include cache memory, digital to analog converters, and other MEOL patterns that were randomly generated. The parasitic capacitance reference numbers of the patterns were extracted using Calibre xACT3D, 3D field-solver, on Intel Xeon(R) E5-2680, 2.50GHz with 8 CPUs and 16G of RAM. The total runtime of extracting the patterns is 4.3 hours. The input vector size of the models is 696.

As for the fully connected NN models architecture, the two models have the same architecture. The NN architecture consists of three hidden layers with 174, 174, and 140 neurons, respectively. The activation functions are tanh, tanh, and RELU, respectively. The total training runtime of the two models is 3.6 hours using Intel Xeon(R) E5-2680, 2.50GHz, and 16G of RAM. As for SVR models, the two SVR models have the same hyper-parameters which are: 8 as a regularization parameter (C), 0.3 as a gamma, and 0.1 as an epsilon. The total training runtime of the two models is 1.7 hours using Intel Xeon(R) E5-2680, 2.50GHz, and 16G of RAM.

The proposed models were tested over more than 15M devices across several designs that include ring oscillators, voltage-controlled oscillator, and digital to analog converter designs. Fig. 5.11 shows the error histograms across all designs using the proposed NN models, SVR models, and Calibre PEX as a rule-based tool with a pre-characterized library. The results show that the proposed NN models have a superior accuracy as compared to Calibre PEX. Also, the prediction runtime of the NN and SVR models are faster than Calibre PEX by 1.057X and 1.23X, respectively. Moreover, the prediction runtime of the NN and SVR models are faster than Calibre xACT3D, 3D field-solver, by 97X and 112X, respectively. Table 5.2 summarizes the architectures of

the NN and SVR models, the accuracy results of training and test sets, and the training runtime. Table 5.3 summarizes the average prediction runtime of the different MEOL extraction models.

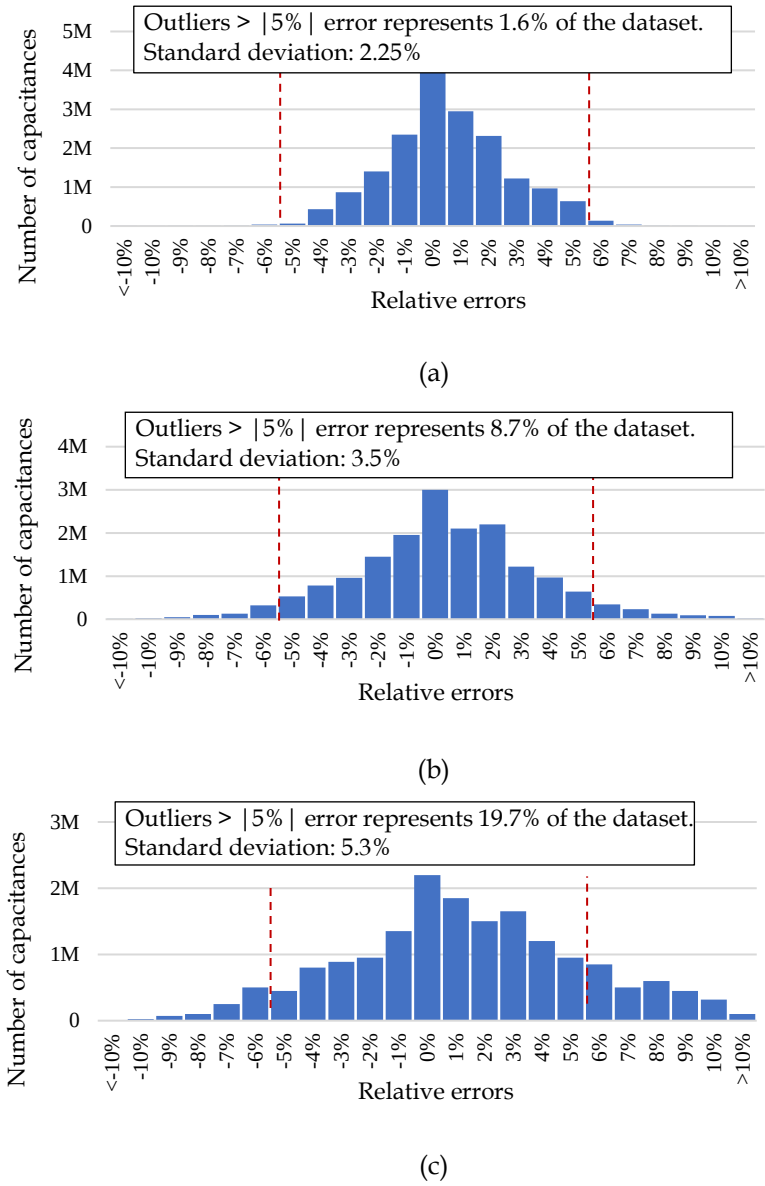


Fig. 5.11. Error histograms, as compared to Calibre xACT3D, of 28nm process node using (a) the proposed NN models, (b) the proposed SVR models, and (c) Calibre rule-based extraction tool.

Table 5.2. The architectures and parameters of 28nm MEOL patterns.

	NN models	**SVR models**
Input vector size (*n*)	696	
Test coverage	15M devices (RO, VCO, and DAC)	
Parameters	**Three hidden layers**: 174, 174, and 140 neurons, respectively. **Activations**: tanh, tanh, and RELU, respectively.	**Kernel**: RBF **Regularization** (C): 8 **Epsilon**: 0.1 **Gamma**: 0.3
Training set accuracy (MSE)	**3.7e-3**	**4.1e-3**
Test set accuracy (MSE)	**5.2e-3**	**6.7e-3**
Total training runtime for all models	**3.6 hours (2 models)**	**1.7 hours (2 models)**

Table 5.3. The average prediction runtime of the different 28nm MEOL parasitic capacitance models.

	MEOL NN	**MEOL SVR**	**Rule-based**	**Field-solver**
Average prediction runtime per MEOL pattern	2.43ms	2.08ms	2.57ms	235ms

5.5.2. Testing Results on 7nm Process Node

The proposed MEOL modeling methodology was tested on 7nm process node, which is a FINFET technology. The MEOL layers of 7nm process are diffusion, raised source drain (RSD), fins, gate, field poly, via, a local interconnect (e.g., Mb), and a field local interconnect. Six different MEOL models were implemented to cover six different device types, such as n-type high power (NHP), n-type low power (NLP), p-type high power (PHP), p-type low power (PLP), general n-type, and general p-type devices. The total number of training MEOL patterns are 150K patterns. They were obtained from different real designs including cache memory, digital to analog converters, and other MEOL patterns that were randomly generated. The parasitic capacitance reference numbers of the patterns were extracted using Calibre xACT3D, 3D field-solver, on Intel Xeon(R) E5-2680, 2.50GHz with 8 CPUs and 16G of RAM. The total runtime of extracting the patterns is 8.1 hours. The input vector size of the models is 681.

As for the fully connected NN models architecture, the six models have the same architecture. The NN architecture consists of three hidden layers with 170, 170, and 136 neurons, respectively. The activation functions are tanh, tanh, and RELU, respectively.

The total training runtime of the six models is 13.8 hours using Intel Xeon(R) E5-2680, 2.50GHz, and 16G of RAM. As for SVR models, the SVR models have the same hyper-parameters which are: 8 as a regularization parameter (C), 0.3 as a gamma, and 0.1 as an epsilon. The total training runtime of the six models is 6.18 hours using Intel Xeon(R) E5-2680, 2.50GHz, and 16G of RAM.

The proposed models are tested over more than 20M devices from ring oscillators, SRAM, and PLL clock generator designs. Fig. 5.12 shows the error histograms across all designs using the proposed NN models, SVR models, and Calibre xACT as a hybrid tool that uses a field-solver to extract MEOL. The results show that the proposed NN models has a good accuracy relative to Calibre xACT. Also, the prediction runtime of the NN and SVR models are faster than Calibre xACT by 90X and 98X, respectively. Moreover, the prediction runtime of the NN and SVR models are faster than Calibre xACT3D, 3D field-solver, by 101X and 110X, respectively.

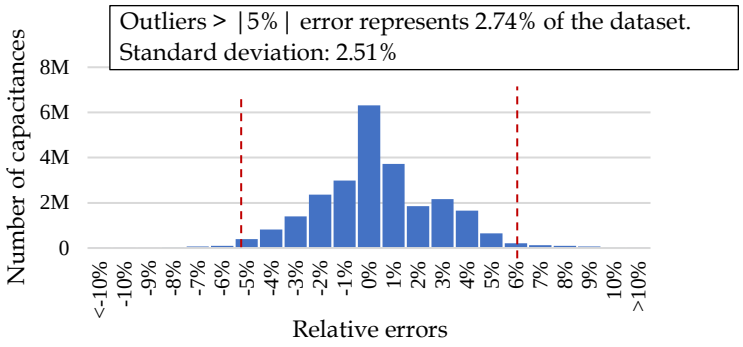
Table 5.4 summarizes the architectures of the NN and SVR models, the accuracy results of training and test sets, and the training runtime. Table 5.5 summarizes the average prediction runtime of the different MEOL extraction models.

Table 5.4. The architectures and parameters of 7nm MEOL patterns.

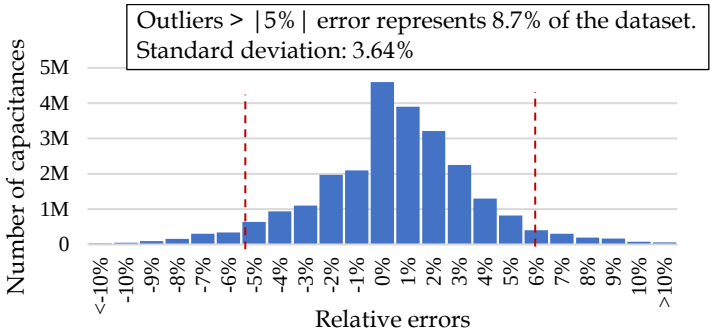
	NN models	**SVR models**
Input vector size (*n*)	681	
Test coverage	20M devices (RO, SRAM, and PLL clock generators)	
Parameters	**Three hidden layers**: 174, 174, and 140 neurons, respectively. **Activations**: tanh, tanh, and RELU, respectively.	**Kernel**: RBF **Regularization** (C): 8 **Epsilon**: 0.1 **Gamma**: 0.3
Training set accuracy (MSE)	**4.3e-3**	**5.7e-3**
Test set accuracy (MSE)	**5.8e-3**	**7.1e-3**
Total training runtime for all models	**13.8 hours (6 models)**	**6.18 hours (6 models)**

Table 5.5. The average prediction runtime of the different 7nm MEOL parasitic capacitance models.

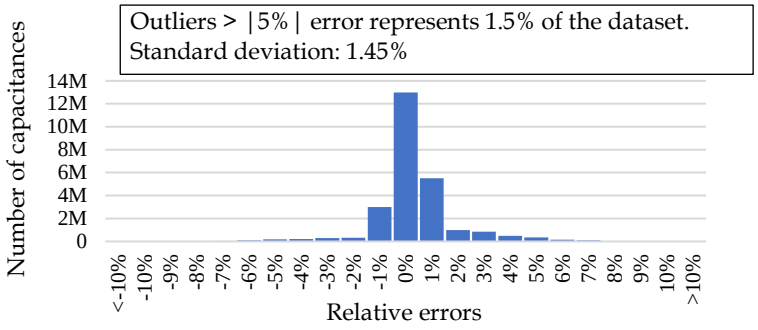
	MEOL NN	**MEOL SVR**	**Hybrid**	**Field-solver**
Average prediction runtime per MEOL pattern	2.61ms	2.4ms	235ms	264ms



(a)



(b)



(c)

Fig. 5.12. Error histograms, as compared to Calibre xACT3D, of 7nm process node using (a) the proposed NN models, (b) the proposed SVR models, and (c) a hybrid extraction tool.

5.6. Conclusion

A parasitic capacitance extraction modeling methodology is developed for middle end of line patterns around FINFETs and MOSFETs using machine learning methods. The current extraction tools either rely on field-solvers or pre-characterized libraries to extract MEOL patterns. The pre-characterized libraries suffer from several issues that impact the extraction accuracy including pattern mismatches and insufficient pattern coverage. On the other hand, the field-solver methods have a limited capacity and consume a lot of time. The proposed modeling methodology provides compact models that predict MEOL parasitic capacitances accurately. This is done by selecting all devices in a certain layout, identifying their MEOL patterns, and representing MEOL patterns using a novel geometry-based representation to be used as inputs to the required machine learning models. Two different machine learning methods are used to create MEOL parasitic capacitance models: support vector regressions and neural networks. The proposed methodology is tested over two process nodes including: 28nm and 7nm. The testing covered devices in several real designs with more than 40M devices. The proposed methodology provided outstanding results as compared to field-solvers with an average error < 0.2%, a standard deviation < 3%, and a speed up of 100X.

Chapter 6

Hybrid Parasitic Capacitance Extraction Using Machine Learning

The increasing parasitic capacitance extraction accuracy requirements in advanced process nodes by semiconductor foundries (< 5% error) added many challenges to the models of existing parasitic extraction tools. Hybrid parasitic capacitance extraction methods that combine rule-based and field-solvers are considered in advanced nodes. Despite the accuracy improvements introduced by those methods, they suffer from two main problems. First, the proportion of patterns that have to go to field-solvers increases in advanced nodes (as technology scales down). With this proportion approaching 50%, using these hybrid methods does not save much time as compared to field-solvers. Second, these hybrid methods do not eliminate all outliers because the layout patterns are assigned to extraction methods based on a pre-characterized library.

To improve the accuracy and runtime of the hybrid parasitic capacitance extraction, a new adaptive, accurate, and faster intermediate extraction method is required to replace field-solvers in extracting most of layout patterns. Furthermore, a smarter way to direct each layout pattern to an appropriate extraction method based on the required accuracy is needed. Therefore, an accuracy-based hybrid parasitic capacitance extraction method is proposed. The proposed hybrid flow divides a layout into windows (i.e., sliding window) and extract each window using one of three extraction methods, based on the required parasitic capacitance extraction accuracy level, that include field-solver, rule-based, and novel deep neural-networks-based (i.e., intermediate) extraction methods. Fig. 6.1 shows an illustrative example of extraction windows that are used in the proposed hybrid flow.

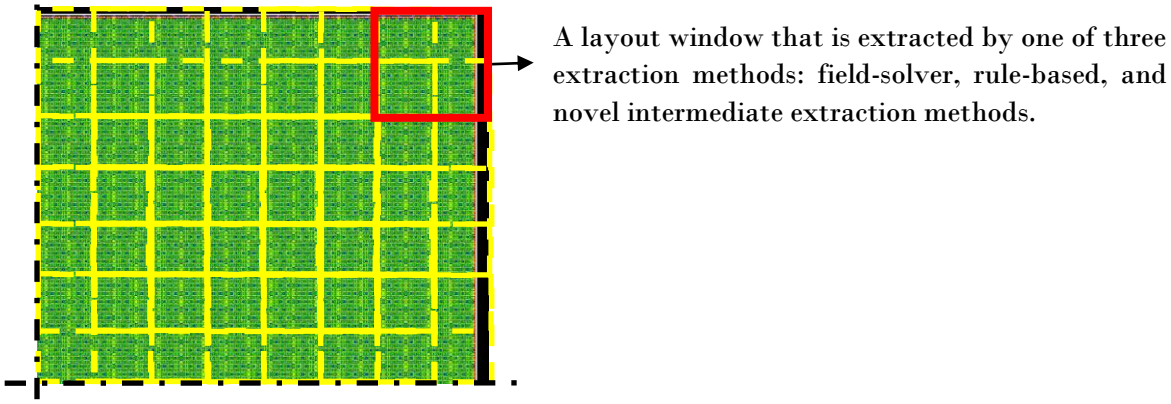


Fig. 6.1. An illustrative example of a partial layout showing an extraction window that is used in the proposed hybrid parasitic capacitance extraction flow.

6.1. Deep Neural-Networks Based Extraction

Deep Neural-Networks (DNN) models are implemented to predict parasitic capacitances effeciently in VLSI layout patterns based on Google Tensor flow libraries, [83], using Python [84]. The proposed models extract the different parasitic capacitance components of layout polygons that are enclosed inside a certain window. The proposed models use a novel hybrid density-voltage map feature representaion to represent the input layout patterns. The proposed models can handle the multi-dielectric environment of process nodes, where each process node has its own set of parasitic models. Two DNN models are created for each metal layers combination in a certain process node (i.e., metal stack): one to predict total capacitances on certain polygons and the other to predict coupling capacitances among polygons. Each metal layers combination contains a collection of one to three metal layers, where each combination has the same set of metals and dielectrics specifications, for example, metal1-metal2-metal3 is a combination, whereas metal1-metal3-metal4 is a different combination. As a result, the total number of combinations of a metal stack with M layers is ${}^{M}C_1 + {}^{M}C_2 + {}^{M}C_3$, where ${}^{M}C_n$ is the combinations function. It is worth mentioning that the proposed modeling methodology can handle combinations with more than three metal layers. However, three metal layers combinations are very common in parasitic capacitance extraction, and usually, they provide good parasitic extraction accuracy for real chips with high metal densities [50], [76], [88]–[90].

The process of implementing parasitic capacitance DNN is shown in Fig. 6.2. The process starts with generating real layout patterns based on drawn dimensions for a certain process node. After that, the drawn dimensions are converted into actual dimensions by applying the corresponding systematic process variations (e.g., etching). Then, a field-solver is used to extract reference parasitic capacitances for the patterns with actual dimensions. Eventually, the input patterns and their parasitic capacitances are used to train the required DNN models.

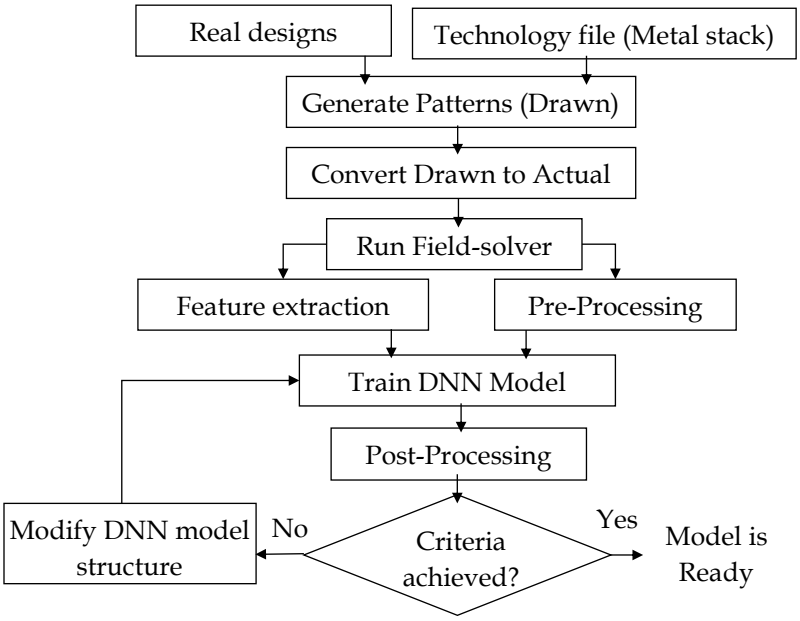


Fig. 6.2. The process of implementing a deep neural-networks model for parasitic capacitance extraction.

6.1.1. Input Patterns Generation

A dataset of layout patterns is obtained from several real designs that belong to a certain process technology node. The designs include Op-Amp, Ring Oscillator, DRAM, PLL, and sense-amplifier circuits. Also, additional patterns are generated using layout schema generator (LSG) in [91]. The LSG aims to generate random realistic patterns using Monte Carlo methods. The patterns are generated by creating a grid of square segments and then injecting polygons into the created grid in a way that complies with the corresponding design rules. The dataset consists of 250K layout patterns. Each pattern contains one to three metal layers. Fig. 6.3 shows an example of a layout pattern

with 1µm × 1µm window size using 28nm node. The size of each pattern is technology dependent, and it represents the maximum interaction distance of the target metal layer. The maximum interaction distance is calculated by simulating 2D cross-section patterns of two adjacent polygons that belong to the same target metal layer using a field-solver. The simulation starts with minimum technological dimensions and sweeps over the spacing between these two polygons. The maximum interaction distance is the spacing where the coupling capacitance between the two polygons is less than or equal to 1% of the total capacitance as shown in Fig. 6.4.

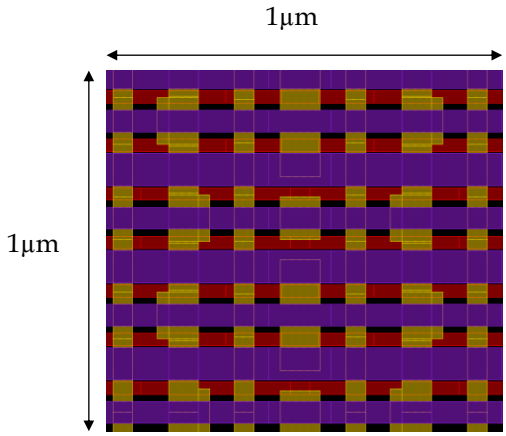


Fig. 6.3. An example of a layout pattern of three metal layers (28nm).

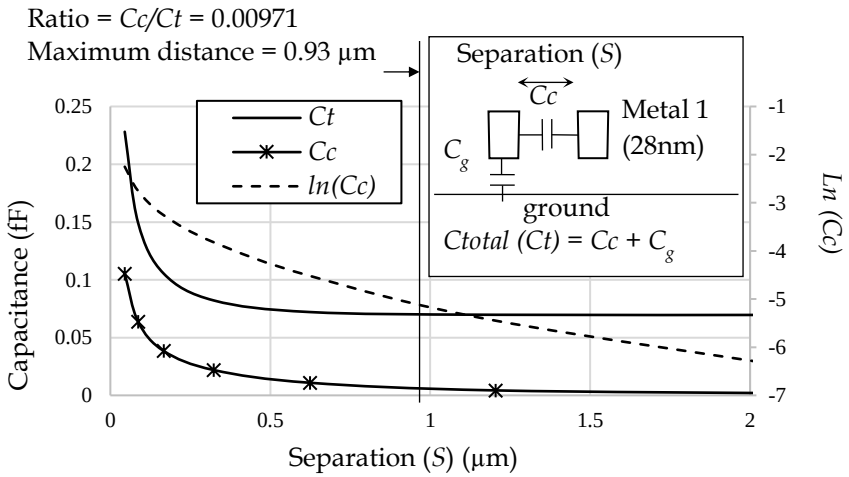


Fig. 6.4. An example showing the impact of increasing the separation between two adjacent metal polygons on the lateral coupling capacitance. The experiment used metal1 in 28nm process node.

6.1.2. Running Field-solver

Once all layout patterns are ready, they are passed to our field-solver, Calibre xACT3D [32], to extract all parasitic capacitance components that should be used as reference capacitance numbers. The field-solver generates a capacitance matrix of $m \times m$ dimensions for each layout pattern, where m represents the number of metal polygons in a layout pattern. In a typical pattern, there are around 10 to 15 metal polygons. Each off-diagonal entry in the matrix represents a coupling capacitance between two certain polygons, whereas each diagonal entry represents a total capacitance on each polygon.

6.1.3. Dataset Pre-Processing

To achieve high accuracy levels, two DNNs are created: one to predict the total capacitances on each polygon, and the other one to predict the coupling capacitances among polygons in a certain layout pattern. The total and coupling capacitances are pre-processed differently. The total capacitances are normalized to have a zero mean and unit standard deviation:

$$C_{normalized} = \frac{C - \text{mean}(\sum C)}{\text{stdev}(\sum C)}, \tag{6.1}$$

where $C_{normalized}$ is the normalized capacitance, C represents a certain capacitance component, mean$(\sum C)$ is the mean of capacitances, whereas stdev$(\sum C)$ is the standard deviation of capacitances. The normalized capacitances are used as reference numbers to the total capacitance DNN model. On the other hand, the coupling capacitances are pre-processed using:

$$C_{ln} = ln(C_{coupling}), \tag{6.2}$$

where C_{ln} is the pre-processed (linearized) capacitance and (ln) is the natural logarithm. The natural logarithm provides some sort of linearization as coupling capacitances decrease exponentially with the separation (i.e., spacing), as shown in Fig. 6.4. By such a linearization, the linearized capacitance margins are increased, and the non-linearity of coupling capacitances against the separation is decreased as shown in Fig. 6.4. Therefore, the training of parasitic models would converge faster on better coefficients

(or neuron weights). As a result, the overall coupling capacitance extraction accuracy is improved. The linearized capacitances are used as reference numbers to the coupling capacitance DNN model.

It is worth mentioning that the small coupling capacitance components that are less than 0.1% of the total capacitances are excluded as they do not impact the capacitance extraction accuracy and might disturb the training process. In the extraction (i.e., prediction) phase, the subtractive approach is followed, where for each polygon, the extracted coupling capacitances are subtracted from the total capacitance until the subtraction reaches zero. In case of any remaining residuals, they are distributed across the nearby polygons.

6.1.4. Hybrid Density-Voltage Map Feature Representation

A novel hybrid density-voltage map feature representation is proposed to extract features of layout patterns. Those features are used as inputs to parasitic capacitance DNN models. The proposed feature representation is a combination of density- map and voltage-map feature representations. The density-map feature representation is one of the most effective layout feature representations, where each layer in a given layout pattern is divided into segments, and the density of polygons is calculated in each segment creating a density matrix for each layer in a given layout pattern [92]. Fig. 6.5 shows an example of the density-map feature representation of two polygons that belong to the same metal layer creating 6×6 density matrix [92]. To have a meaningful density matrix, each segment must include information of a single polygon at maximum. The expirements showed that the segment's size must be less than half the minimum spacing of the corresponding metal layer.

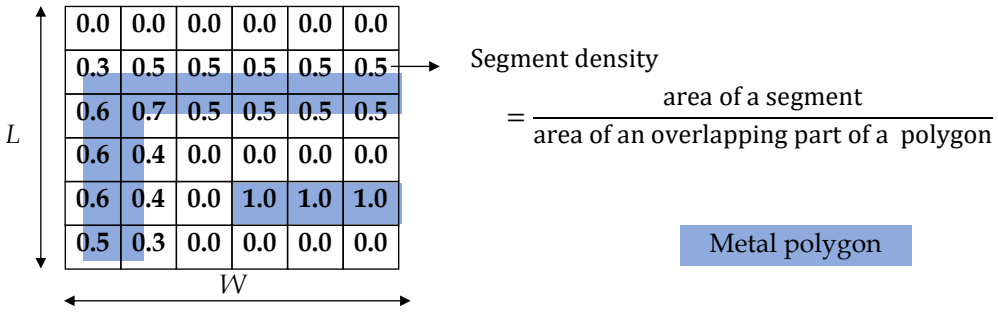


Fig. 6.5. An example of the density-map feature representation in a layout pattern using two polygons that belong to the same metal layer.

On the other hand, in the voltage-map representation, 1V is assigned to aggressor polygons, whereas 0V is assigned everywhere else. To extract parasitic capacitances, the density-map feature representation is not enough to predict different parasitic capacitance components efficiently because each pattern contains several and different numbers of capacitance components. In order to overcome this problem, each capacitance component must be associated with a different input (i.e., a layout pattern matrix).

A novel hybrid density-voltage map feature representation is proposed to provide a different input pattern (matrix) for each capacitance component as shown in Fig. 6.6. In the proposed representation, total capacitances are addressed by assigning (1V+density) on aggressor polygons and (0V+density) everywhere else, whereas coupling capacitances are addressed by assigning (1V+density) on aggressor polygons, (-1 × (1V+density)) on victim polygons, and (0V+density) everywhere else. Fig. 6.6 shows an example of the proposed hybrid density-voltage map feature representation highlighting patterns of total and coupling capacitances.

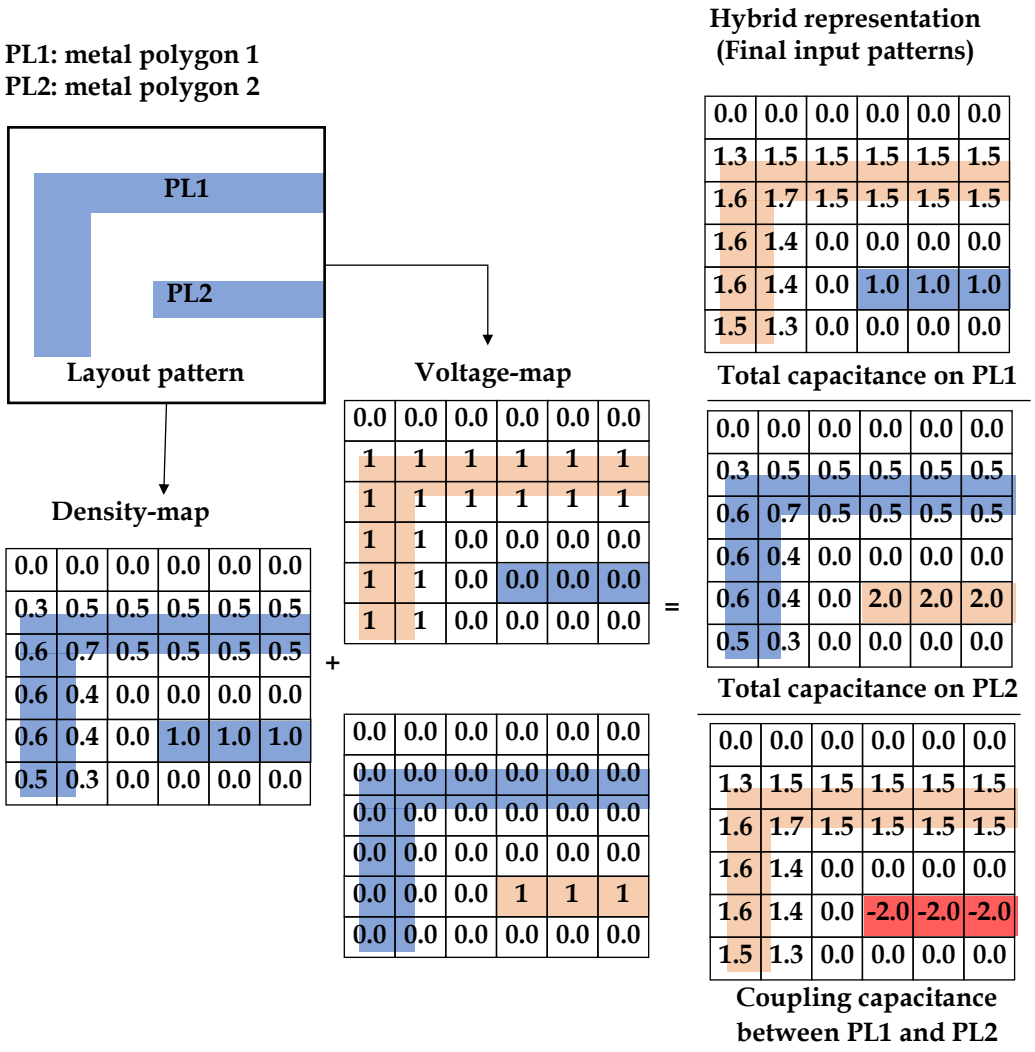


Fig. 6.6. The proposed hybrid density-voltage map feature representation for total and coupling parasitic capacitance extraction.

The size of the map is obtained for each metal layer using the pattern window size and the number of segments (i.e., pixels) in each window. First, each layout pattern window has a square shape, where the width and length of each window equal the maximum interaction distance of the target metal layer. Second, each window is segmented into *ns*×*ns* segments, where the length and width of each segment equal half the minimum spacing of the corresponding metal layer. Therefore, *ns* is estimated by:

$$ns = \text{window width} / (0.5 \times \text{minimum spacing}). \quad (6.3)$$

As for the window width (i.e., size), it is set to the maximum interaction distance to ensure that only significant coupling capacitances are considered, and minor coupling

capacitances that do not impact the accuracy of capacitance extraction are neglected. This would help in avoiding unnecessarily capacitance computations. As for the segment width, it must be less than or equal to half the minimum spacing to ensure that every layout pattern has a unique density representation, where all polygons should be separated by empty (i.e., zero) segments. Fig. 6.7 shows three examples with different segment sizes, where the same density-map matrix may represent multiple layout patterns as long as the corresponding segment width is greater than half the minimum spacing, whereas the patterns that have segment widths less than or equal to half the minimum spacing would have unique density-map matrices.

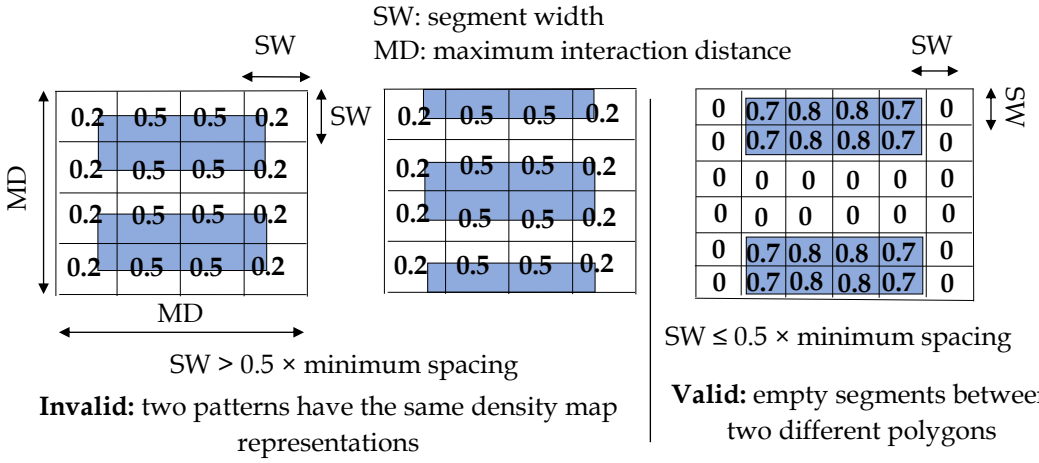
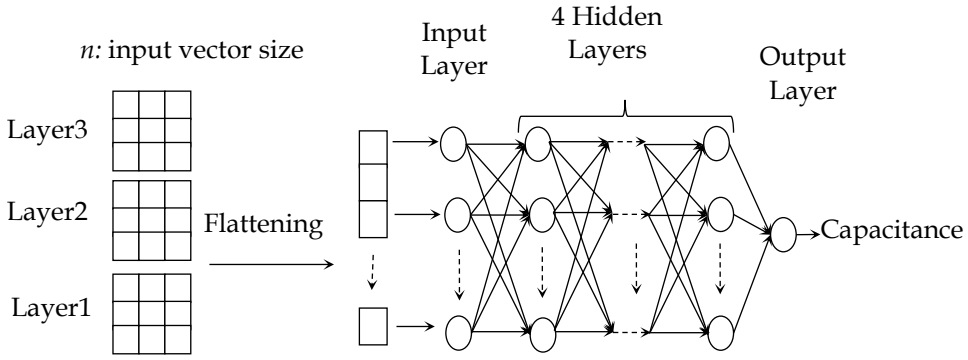


Fig. 6.7. An example of three layout patterns with different segment sizes.

6.1.5. DNN Construction

Two DNN models are implemented. One to predict total capacitances and the other one to predict coupling capacitances. The inputs of the DNNs are the flattened hybrid density-voltage matrices of a layout pattern as shown in Fig. 6.8. Since each layout pattern consists of three different metal layers, the input size is $(M^2 + N^2 + Y^2)$, where M^2 is the density-voltage matrix size of the first metal layer, N^2 is the density-voltage matrix size of the second metal layer, whereas Y^2 is the density-voltage matrix size of the third metal layer. The output of the first DNN is the total capacitance on a certain polygon, whereas the output of the second DNN is the coupling capacitance between two selected polygons in a given layout pattern.

As for DNN architectures, a fully connected NN architecture is selected. A grid search algorithm is used to obtain NN architectures. The search range includes the number of NN layers that varies from 2 to 7, the number of neurons per layer that varies from $n/9$ to n, where n represents the input vector size, and the activation function of each layer that alternates between RELU and tanh. As for the evaluation method, the mean square error (MSE) of testsets is used where the smallest DNN architecture that reaches MSE of 0.05% is selected. Fig. 6.8 shows the most common fully connected DNN architecture that is used to predict total and coupling parasitic capacitances in layout patterns. For each hidden layer, the number of neurons depends on the input vector size. For example, layout patterns with an input vector size of 6075 segments use a DNN architecture that has 4 hidden layers with 675, 868, 760, and 868 neurons, respectively.



Layer	1 (input)	2	3	4	5	6(output)
Neurons =	n	$n/9$	$n/7$	$n/8$	$n/7$	1
Activation =	RELU	RELU	tanh	RELU	tanh	Uniform

Fig. 6.8. A fully connected NN architecture to calculate total and coupling parasitic capacitances for a certain layout pattern.

6.1.6. DNN Training

In the training phase, the dataset is divided into training and test sets. They are randomly selected, where 70% of the dataset are used for training, whereas 30% of the dataset are used for testing. The training used Adam optimizer, 5K as a maximum number of epochs, 10% as a validation set, mean square error as a cost function, 1e-3 as a learning rate, and 1000 as a batch size. The input and four hidden layers initializations are: he_normal, he_normal, glorot_normal, he_normal, and glorot_normal, respectivily

[93]. These parameters are obtained using a grid search algorithm. An early stopping is used to stop the training epochs once the training converges. The convergence is achieved when the accuracy of the validation set starts to degrade, or after 50 epochs of not observing any further accuracy improvements in both the validation and training sets. The average training runtime of a parasitic capacitance DNN model is 4.9 hours using Intel(R) Xeon(R) E5-2680, 2.70GHz with 4 CPUs and 16G of RAM.

6.1.7. Comparison to Other Layout Representations

The most common layout representation methods include spectrum-based, concentric circle area sampling (CCAS), and density-based methods [92], [94]–[96]. In spectrum-based representation, a frequency domain transformation, such as Discrete Transform (DCT) and Discrete Fourier Transform (DFT), is applied on each layout pattern creating feature vectors of transformation coefficients (e.g., Fourier coefficients). After that, each feature vector is passed to a convolution step to be used as an input to the required models [92].

On the other hand, the CCAS representation method creates a set of concentric circles for each layout pattern with radiuses of: 0, 4, 8, 12, …, r_{in}, r_{in}+8, r_{in}+16, …, $L/2$ pixels, where r_{in} is a user controlled sampling density parameter, L is the pattern's length, and the unit of a pixel is nm, as shown in Fig. 6.9. Each circle is sampled uniformly with x samples conforming x digits binary numbers, where 1 means that the sample point overlaps with a polygon and 0 means that the sample point does not overlap with a polygon. After that, the binary number of each circle is encoded into a more compact representation (e.g., decimal number). The encoded numbers are combined into a feature vector and used as an input to the required models.

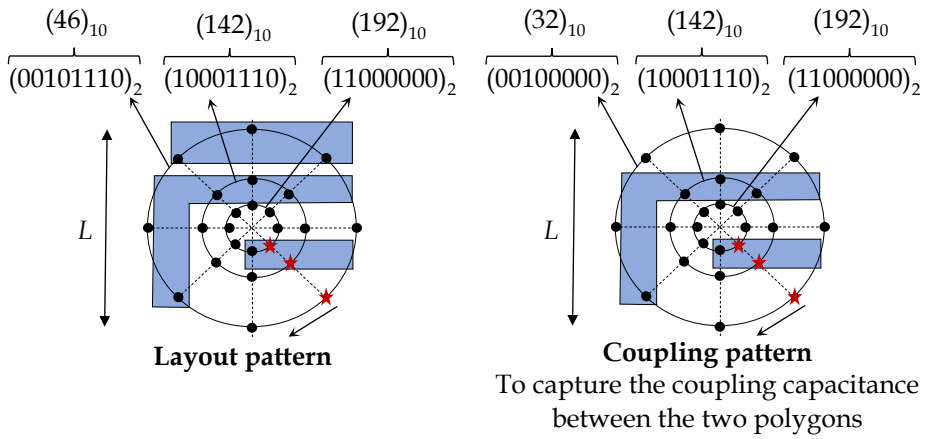


Fig. 6.9. An example of CCAS layout representation for capacitance extraction.

In capacitance extraction, any layout modification may impact many coupling capacitances. So, its layout representation must consider many layout details. The convolutional methods, as in spectrum-based, may lose a lot of layout details as they mainly target limited layout features. Other methods like CCAS representation can provide detailed information about layout patterns by reducing the separation between concentric circles and increasing the number of samples per circle; however, this will result in a lot of redundant data that may disturb the training process and cause overfitting [92]. On the other hand, the density-based representation, which uses density-map features, is a rasterization method that can capture many layout details. Also, it is less complicated as compared to other methods and can easily be reverse engineered for any debugging purposes.

The accuracy of the proposed hybrid density-voltage map is evaluated against the CCAS representation by comparing their results against Calibre xACT3D, field-solver. To extract the coupling capacitances using the CCAS representation, an additional feature vector that represents the target polygons is added to the input vector. Fig. 6.9 shows an example of a coupling pattern that is used to extract the coupling capacitance between two certain polygons.

The testing used 28nm process node and included 11M capacitance components across different 92 layers combinations that cover metal1 to metal8 layers. The training

process shown in Fig. 6.2 is followed to train CCAS NN models using the same hyper-parameters, same hardware, and same settings. The CCAS training used 0.5µm as a sampling density parameter, which approximately represents 0.5 × maximum interaction distance of most of the training patterns, and 8 sample points for each circle. This increased the number of concentric circles in the range of maximum interaction distance. Therefore, the sampling fidelity of layout patterns is increased providing more detailed representations. The NN architecture of the CCAS contains two hidden layers, each with 185 neurons and tanh activations. The testing included two main factors: the mean of relative errors and the standard deviation (STDEV) of relative errors, where the relative error is calculated as below:

$$\text{Relative error} = \frac{C_{model} - C_{reference}}{C_{reference}}, \tag{6.4}$$

where C_{model} represents a capacitance value that is calculated by a model, whereas $C_{reference}$ represents a reference capacitance value that is calculated by a field-solver. Table 6.1 shows the testing accuracy results of the proposed hybrid density-voltage map and CCAS layout representations. The results show that the hybrid density-voltage map representation provides better accuracy numbers as compared to the CCAS representation.

Table 6.1. Accuracy comparison between CCAS and the Proposed hybrid density-voltage map layout representations using 28nm process node.

	Mean of errors	STDEV of errors
CCAS	1.2%	13.7%
Hybrid density-voltage	0.021%	1.61%

6.1.8. Comparison Against Other Machine Learning Methods

The proposed DNN models are compared against Support Vector Regression (SVR), random forest regression (RFR), and Convolutional Neural Networks (CNN) models [97], [98]. The models use the proposed hybrid density-voltage map feature representation as inputs and the pre-processed capacitance values as outputs. Table 6.2 shows the used hyper-parameters of SVR, RFR, and CNN models. These hyper parameters are obtained using a grid search algorithm. The testing used 28nm process

node and included 11M capacitance components across different 92 layers combinations that cover metal1 to metal8 layers. The testing included two main factors: the mean of relative errors and the standard deviation (STDEV) of relative errors. Table 6.3 shows the test set accuracy results of the DNN-based, SVR, RFR, and CNN models. The results show that the DNN-based and CNN models provide outstanding accuracy results as compared to the other models. The accuracy of DNN-based models is better than CNN (LeNet5) models and very close to CNN (ResNet18) models. The training and prediction times of the DNN-based models are way faster than CNN models, where the average training times of DNN-based, CNN (LeNet5), and CNN (ResNet18) models are 4.9, 10.3, and 32.8 hours, respectively, whereas the corresponding average prediction times are 1.3, 4.7, and 15.8 ms, respectively. The training used Intel(R) Xeon(R) E5-2680, 2.70GHz with 4 CPUs and 16G of RAM, whereas the prediction is done on the same machine using a single CPU.

Table 6.2. A list of selected hyper-parameters of SVR, RFR, and CNN models for 28nm process node.

Method	**Hyper-parameters**
Support Vector Regression (SVR)	Kernel = polynomial Cost = 5 order (d) = 6
Support Vector Regression (SVR)	Kernel = radial basis function (RBF) Cost = 9 Gamma = 0.2
Random Forest for Regression (RFR)	Number of trees=2000
Convolutional Neural-Network (CNN) (architecture = LeNet5) [97]	Optimizer: Adam Learning rate: 1e-3 Batch normalization: Yes Validation set: 10% Batch size: 1000 Epochs: 500 or early stopping
Convolutional Neural-Network (CNN) (architecture = ResNet18) [98]	

Table 6.3. Test sets accuracy results of DNN-based, SVR(Polynomial), SVR (RBF), RFR, and CNN Modeling methods using 28nm process node.

Method	**Mean of errors**	**STDEV of errors**
DNN-based	0.02%	1.61%
SVR (kernel = polynomial)	1.6%	9.9%
SVR (kernel = RBF)	0.9%	6.8%
RFR	1.8%	8.5%
CNN (LeNet5) [97]	0.141%	3.11%
CNN (ResNet18) [98]	0.036%	1.63%

6.2. Hybrid Parasitic Extraction

A hybrid parasitic capacitance extraction flow is proposed to achieve high accuracy levels with a very good performance, regardless of the process technology node or the layout complexity. Unlike existing hybrid flows that are pattern-based, the proposed flow is accuracy and performance based. Existing hybrid flows rely on pattern structures, if a given layout pattern is covered by rule-based models then it is extracted using a rule-based extraction method, otherwise, it is extracted using a field-solver. Such flows do not necessarily provide good parasitic capacitance extraction accuracy because rule-based models do not consistently provide good accuracy levels on all covered and extracted patterns. There are several reasons for inaccuracy in rule-based extraction methods such as a curve fitting inaccuracy, insufficient samples, and pattern matching issues [11], [14], [34]. The proposed hybrid flow overcomes these issues by evaluating each layout pattern from the accuracy perspective not from the pattern perspective, and then assigning each layout pattern to the appropriate extraction method based on the pattern's accuracy characteristics with each method. Also, the proposed hybrid flow gives the user the ability to determine the required accuracy level based on design and design phase requirements. As a result, the proposed flow managed to meet the required accuracy level with more than 99% accuracy when tested on several real designs.

The proposed hybrid flow uses three extraction methods: field-solver, rule-based, and novel DNN-based extraction methods. To mitigate accuracy outliers of the used extraction methods, the proposed flow identifies the accuracy limitations of each extraction method for each layout pattern and extracts parasitic capacitances using the fastest extraction method that meets the user pre-determined accuracy level. Fig. 6.10 provides an illustration of accuracy limitations of the three extraction methods and their relative runtimes. Also, it shows the distribution of parasitic capacitance errors for each extraction method where the mean of parasitic capacitance errors of the proposed DNN-based extraction method is below 5%, the mean of rule-based extraction methods is above 5%, and the mean of field-solvers is close to zero. Moreover, the rule-based and DNN-based extraction methods have outliers that exceed 10%.

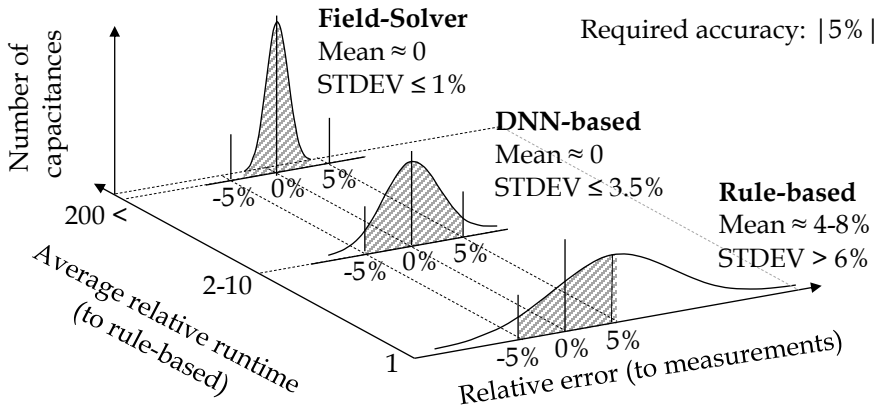


Fig. 6.10. Accuracy (i.e., relative error) and runtime distributions of rule-based extraction, DNN-based extraction, and field-solver methods.

The proposed hybrid flow starts with a sliding window that scans a whole layout design and divides it into patterns (i.e., windows). All patterns are passed to a classifier (i.e., extraction selector) that assigns each pattern to an appropriate extraction method. Moreover, the extraction selector enables the user to pre-determine the accuracy level based on his preferences.

In this work, Calibre xACT3D, [32], is used as a field-solver method, Calibre PEX, [19], is used as a rule-based method, and the novel DNN-based extraction is used as an intermediate extraction method.

6.2.1. Multi-Class Extraction Selector

A multi-class extraction selector is implemented to select an appropriate extraction method that meets the user pre-determined accuracy for each given layout pattern. As shown in Fig. 6.11, the extraction selector consists of a classifier's switch and five different classifiers each operates at a different accuracy level. Each classifier has three possible outputs (i.e., extraction method decision): rule-based, field-solver, and DNN-based extraction methods. In this flow, each layout pattern is passed to a classifier's switch that assigns each layout pattern to the appropriate extraction multi-class classifier based on the required accuracy level. After that, the chosen classifier predicts the appropriate extraction method for each given layout pattern. The multi-class classifiers are implemented using NNs. The layout density-map feature extraction,

shown in Fig. 6.5, is used to represent layout input patterns. Moreover, each combination of layers has a different extraction selector.

The process of implementing a NN classifier is shown in Fig. 6.12. The process starts with generating real layout patterns based on drawn dimensions, converting them into actual dimensions by applying process variations (e.g., etching), running the three extraction methods over all patterns, labeling each pattern with an appropriate extraction method, extracting features for each pattern, and train NNs classifiers with different architectures until a high accuracy classifier is obtained.

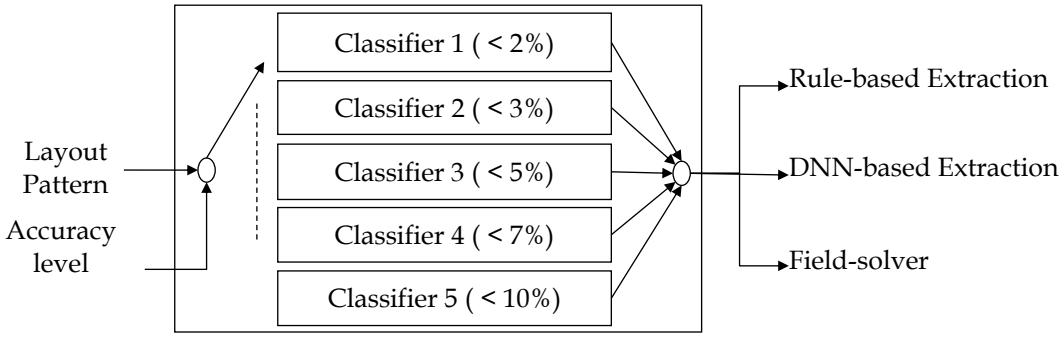


Fig. 6.11. The proposed multi-class parasitic capacitance extraction selector.

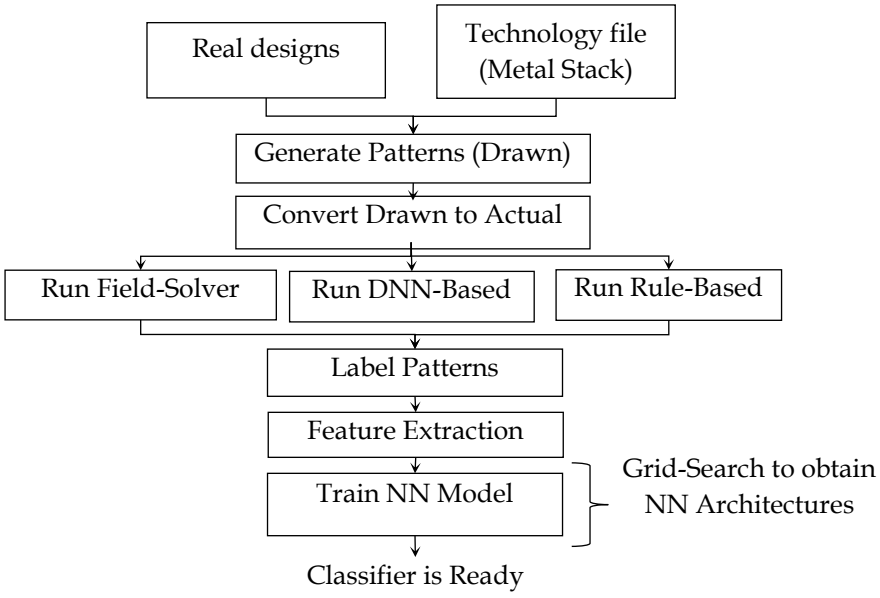


Fig. 6.12. The implementation process of NN Classifiers.

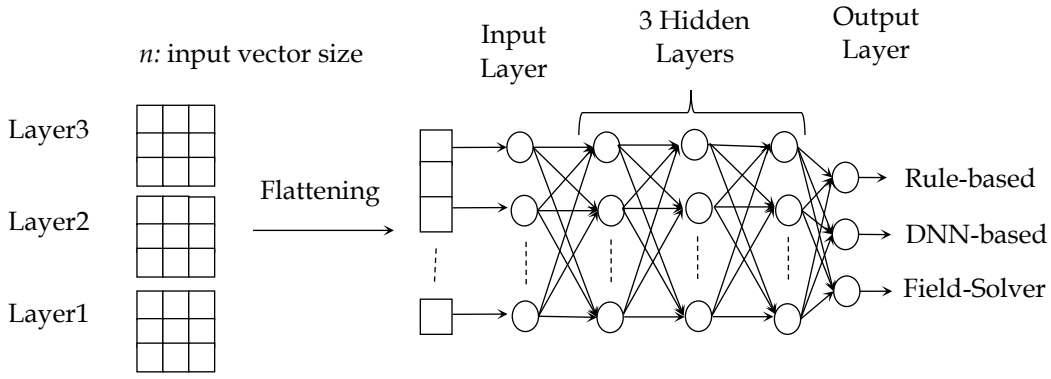
6.2.2. Training Patterns of Classifiers

The dataset of training layout patterns are obtained from several real designs that belong to a certain process node. the layout designs include Op-Amp, Ring Oscillator, DRAM, PLL, and sense-amplifier circuits. Also, to further increase the coverage of training patterns, additional patterns are generated using a layout schema generator (LSG) in [91]. The LSG aims to generate random realistic patterns that comply with the corresponding design rules. The dataset consists of 250K layout patterns. Each pattern contains one to three metal layers. The size of each pattern is technology dependent and it represents the maximum interaction distance of the target metal layer. To label layout patterns, each pattern is extracted by the three extraction methods using: Calibre xACT3D, Calibre PEX, and the proposed DNN-based extraction. After that, the worst parasitic capacitance error is recorded for each extraction method, using Calibre xACT3D as a reference. Then, each pattern is marked with all extraction methods that meet the required accuracy. However, since each layout pattern must be labeled only with one extraction method, each pattern is labeled with the fastest extraction method that meets the required accuracy, given that the rule-based method is faster than the DNN-based method, and both of them are faster than the used field-solver. Moreover, rule-based patterns must also be solvable using the DNN-based method, otherwise they will be labeled as field-solver patterns. Therefore, if a rule-based pattern is extracted using any of the three methods, it will still meet the required accuracy level.

6.2.3. NNs Construction

As for NN architectures, a fully connected NN architecture is selected. A grid search algorithm is used to obtain NN architectures. The search range includes the number of NN layers that varies from 2 to 7, the number of neurons per layer that varies from $n/9$ to n, where n represents the input vector size, and the activation function of each layer that alternates between RELU and tanh. Moreover, the search range includes the weight and decision threshold of each class in order to handle the unbalanced training data. As for the evaluation method, the F1-score of testsets are used, where the architecture with the largest F1-score is selected. Fig. 6.13 shows the most common fully connected NN

architecture that is used to develop extraction selectors. In the NN architecture, the number of neurons for each hidden layer depends on the input vector size. For example, layout patterns with an input vector size of 6075 segments use a NN architecture that has 3 hidden layers with 760, 868, and 868 neurons, respectively.



Layer	**1 (input)**	**2**	**3**	**2**	**4 (output)**
Neurons =	n	$n/8$	$n/7$	$n/7$	3
Activation =	RELU	RELU	RELU	RELU	softmax

Fig. 6.13. A neural-networks architecture of the proposed multi-class classifier.

6.2.4. Classifiers Training and Tuning

The training dataset is divided into training and test sets that are randomly selected, where 70% of the dataset (175K) are used for training, whereas 30% of the dataset (75K) are used for testing. For 28nm process node, the average data distribution across the rule-based, DNN-based, and field-solver classes, at 5% required accuracy level, are 63.02%, 36.77%, and 0.21%, respectively. The training used Adam optimizer, 5K as a maximum number of epochs, 10% as a validation set, categorical cross-entropy as a loss function, 1e-3 as a learning rate, 1000 as a batch size, and he_normal initializations. Moreover, the rule-based decision threshold is set to 0.65 and the weights of the DNN-based and field-solver classes are set to 2 and 9, respectively. This helped in minimizing the probability of passing any non-rule-based pattern to the rule-based class. All parameters are obtained using a grid search algorithm. An early stopping is used to stop the training epochs once the training converges. The convergence is achieved when

the accuracy of the validation set starts to degrade, or after 50 epochs of not observing any further accuracy improvements in both the validation and training sets. The average prediction accuracy of a classifier is 85%. The average training runtime of a classifier is 37 minutes using Intel Xeon(R) E5-2680, 2.70GHz with 4 CPUs and 16GB of RAM. Table 6.4 shows the test set confusion matrix of 28nm process node for 92 layers combinations, at 5% accuracy level, highlighting the number of patterns passed to each class. The results show that the number of non-rule-based patterns that are passed to the rule-based class is 2400 patterns (0.053% of the overall test set). Also, the precision of the rule-based class exceeds 99.8%, and the recall is 72%, where the precision is the ratio between the patterns predicted accurately as class *A* and all patterns predicted as class *A*, and the recall is the ratio between the patterns predicted accurately as class *A* and all actual class *A* patterns.

6.2.5. Comparison to Other Layout Representations

The classification accuracy using density-map (i.e., density-based) and CCAS layout feature representations are tested. The testing used 28nm process node and covered 92 layers combinations. The density-map is a layout rasterization method that provides a detailed information of layout patterns using an appropriate segment size as described in the previous section. On the other hand, the CCAS method provides an encoded geometrical information of layout patterns. However, CCAS representation misses the details of layout patterns that are required for parasitic extraction problems [92], [94]–[96]. Table 6.4 shows the confusion matrix, precision, and recall of 28nm process node test sets, using density-map and CCAS representations. The results show that the prediction accuracy of CCAS is 61.2%, and the number of non-rule-based patterns that are passed to the rule-based class is 405,540 patterns (9% of the overall test set). Also, the precision of the rule-based class is 76.78%, and the recall is 50.88%. Table 6.4 shows that the density-map representation managed to provide outstanding accuracy results as compared to CCAS representation.

Table 6.4. The confusion matrices of extraction classifiers using density-map and CCAS layout representations for 28nm test sets, at 5% required accuracy level.

			Actual/True		
			Rule-based	**DNN-based**	**Field-solver**
Density-map	**Predicted**	**Rule-based**	1,846,980	2,220	180
		DNN-based	715,020	1,905,540	300
		Field-solver	4,680	12,420	12,660
	Precision		0.998702	0.727067	0.425403
	Recall		0.719599	0.992376	0.96347
CCAS	**Predicted**	**Rule-based**	1,341,060	405,480	60
		DNN-based	1,255,380	1,412,820	180
		Field-solver	39,120	45,240	660
	Precision		0.767812	0.529467	0.007763
	Recall		0.508833	0.758138	0.733333

6.3. Experimental Results

The accuracy and runtime of the proposed DNN-based and accuracy-based hybrid extraction methods are measured by comparing their results against Calibre xACT3D, i.e., field-solver, across several real designs that are not part of the training sets. The real designs include SRAM (28nm), digital to analog converter (DAC) (28nm), cache memory (CM) (28nm), and phase locked loop (PLL) (7nm). The testing mechanism starts with analyzing a real layout design and its corresponding metal stack. Then, a square sliding window is created with $W \times W$ dimensions, where W represents the sliding window's width and length. The sliding window covers three metal layers including the aggressor (i.e., target) metal layer. It moves in x and y directions with a step size of ($0.5 \times W$), and it moves up in z direction with 1 metal layer step. The moving sliding window is responsible for capturing different layout patterns and passing them to a certain extraction method that extracts all different capacitance components. The overlapping windows improve the extraction accuracy by capturing the coupling capacitances between windows, extracting the capacitances of layout polygons multiple times each in a different context (i.e., window) to consider the surrounding metals, and averaging the overlapping capacitances [99]. The size of a sliding window, W, is the maximum interaction distance of the target metal layer. It depends on the metal stack definition and

the target metal layer. There is a sliding window for each target metal layer covering the different metal layers combinations [99]. Table 6.5 shows approximated sizes of windows, up to metal6, along with the corresponding vector sizes for 28nm and 7nm process nodes.

Table 6.5. Approximate sizes of sliding windows for 28nm and 7nm process nodes.

Target layer	Window size (28nm)	Window size (7nm)
Metal1 and Metal2	1.0μm × 1.0μm Vector size: 45×45×3	0.8μm × 0.8μm Vector size: 75×75×3
Metal3 and Metal4	1.2μm × 1.2μm Vector size: 53×53×3	1.2μm × 1.2μm Vector size: 63×63×3
Metal5	1.4μm × 1.4μm Vector size: 62×62×3	1.35μm × 1.35μm Vector size: 71×71×3
Metal6	1.5μm × 1.5μm Vector size: 66×66×3	1.47μm ×1.47μm Vector size: 77×77×3

6.3.1. DNN-Based Extraction Results

The proposed DNN-based extraction is tested and showed outstanding accuracy results relative to Calibre xACT3D across different real designs. Fig. 6.14 (a), Fig. 6.15 (a), Fig. 6.16 (a), and Fig. 6.17 (a) show accuracy comparisons (i.e., histograms of relative errors) of the DNN-based extraction using SRAM (28nm), DAC (28nm), cache memory (CM) (28nm), and PLL (7nm) designs, respectively, whereas Fig. 6.14 (b), Fig. 6.15 (b), Fig. 6.16 (b), and Fig. 6.17 (b) show the rule-based extraction accuracy comparisons using the same designs, at 5% required accuracy level. The total numbers of extracted windows for SRAM (28nm), DAC (28nm), cache memory (CM) (28nm), and PLL (7nm) designs are 123K, 322K, 304K, and 14.1M, respectively, whereas the corresponding numbers of capacitance components are 345K, 675K, 690K, and 43.5M, respectively. The total numbers of extracted windows that do not exist in the training sets for SRAM (28nm), DAC (28nm), cache memory (CM) (28nm), and PLL (7nm) designs are 109K (88.7%), 248K (77%), 252K (82.8%), 10.7M (75.9%), respectively. The results show that the maximum number of outliers that exceed 5% error in the DNN-based extraction represents around 8% of the overall capacitances, whereas in the rule-based extraction, it represents around 50% of the overall capacitances.

6.3.2. Accuracy-Based Hybrid Extraction Results

The proposed hybrid extraction flow is tested 5% accuracy level. The testing is done across four real designs including SRAM (28nm), DAC (28nm), cache memory (CM) (28nm), and PLL (7nm) designs.

Table 6.6 shows the patterns distribution ratios among field-solver, DNN-based, and rule-based extraction methods using the proposed accuracy-based hybrid extraction flow, at 5% accuracy level. Also, the table shows the relative runtime comparisons among Calibre xACT3D (i.e., field-solver), Calibre PEX (i.e., rule-based extraction), and the proposed accuracy-based hybrid extraction flow assuming that the relative runtime of the proposed accuracy-based hybrid extraction flow is 1.0. The results show that the proposed accuracy-based hybrid extraction flow is way faster than the used field-solver with a speed up of 70 to 100X. Table 6.7 shows the total actual extraction runtime using rule-based, DNN-based, field-solvers, hybrid extraction (without using the DNN-based extraction), and the proposed hybrid extraction methods, at 5% required accuracy level. The results show that the removal of the DNN-based extraction from the hybrid flow caused significant performance degradation, as compared to the proposed hybrid flow, of up to 43X, which emphasizes the need for having an intermediate extraction method. The runtimes are measured using Intel(R) Xeon(R) E5-2680, 2.70GHz with 4 CPUs and 16G of RAM.

As for the accuracy results, Fig. 6.14 (c), Fig. 6.15 (c), Fig. 6.16 (c), and Fig. 6.17 (c) show the proposed hybrid extraction accuracy comparisons using SRAM (28nm), DAC (28nm), cache memory (CM) (28nm), and PLL (7nm) designs, respectively. The results show that the proposed hybrid flow managed to eliminate more than 99% of the outliers that exceed 5% error. As for the prediction accuracy of the extraction classifiers, the prediction accuracy (ACC) is given by:

$$\text{ACC} = \text{correct predictions} / \text{all predictions}, \tag{6.5}$$

whereas the prediction accuracy for parasitic extraction (ACCPEX) is given by:

ACCPEX = patterns passed to appropriate extraction methods

/ all predictions (i.e., patterns), (6.6)

given that the field-solver method is appropriate for all patterns as it can meet the required accuracy level, also the three extraction methods are appropriate for rule-based patterns as the three extraction methods can extract the parasitic capacitances of rule-based patterns with the required accuracy level. Table 6.8 shows the confusion matrices, classification accuracy, and classification accuracy for parasitic extraction of SRAM (28nm), DAC (28nm), cache memory (28nm), and PLL (7nm) designs, respectively, using the extraction selector of the hybrid flow at 5% accuracy level.

Table 6.6. Patterns distribution ratios using the proposed hybrid flow, and the relative runtime comparisons for field-solver and rule-based tools at 5% required accuracy level.

Design	Patterns distribution in a design among different extraction methods			Relative runtime as compared to the proposed hybrid flow.	
	Field-solver	DNN-based	Rule-based	All Rule-based	All Field-solver
SRAM(28nm)	0.78%	57.1%	42.12%	0.348	74.54
DAC(28nm)	0.85%	53.9%	45.25%	0.409	72.36
CM(28nm)	0.81%	58.3%	40.89%	0.455	72.32
PLL(7nm)	0.93%	60.3%	38.77%	0.413	67.1

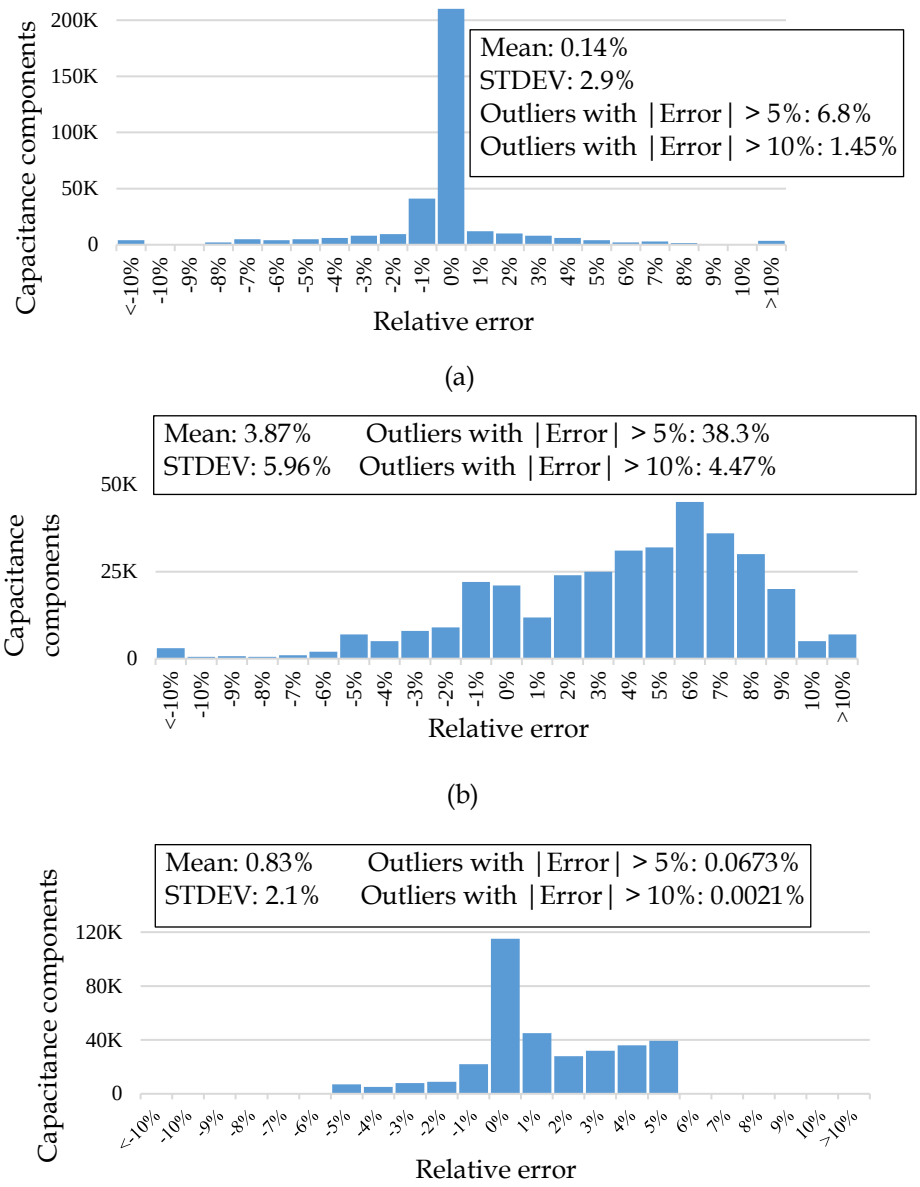
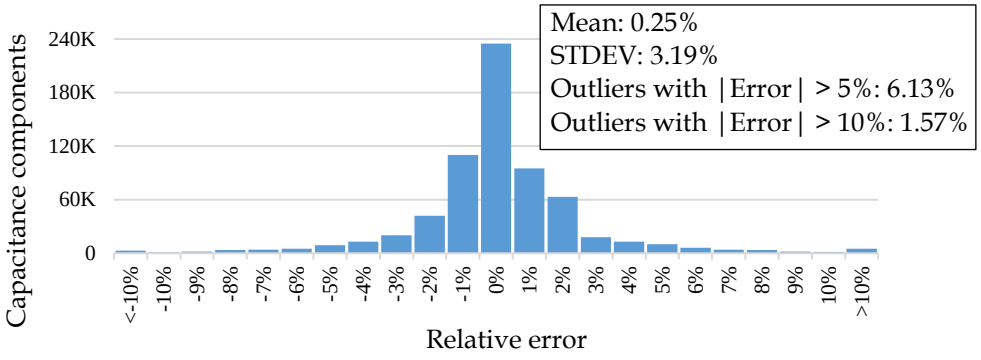
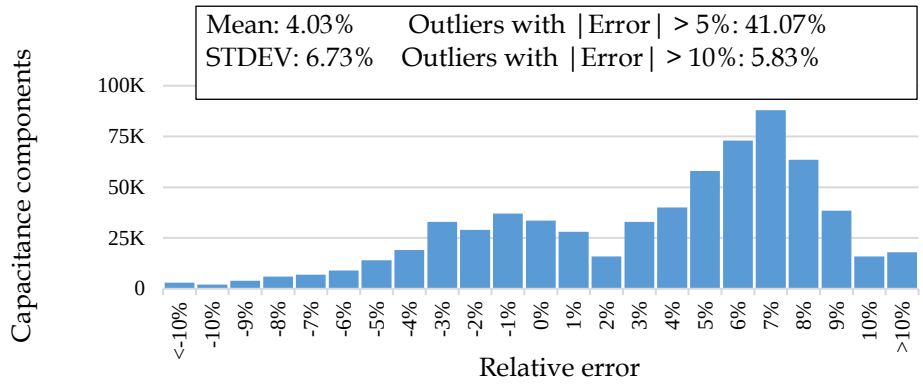


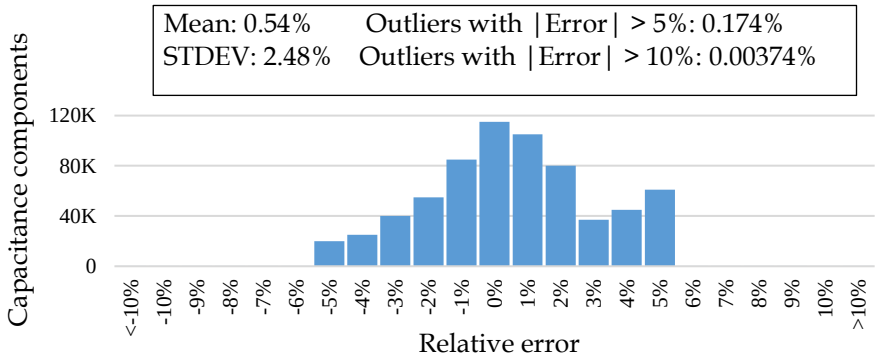
Fig. 6.14. Accuracy comparison histograms, relative to Calibre xACT3D, of the SRAM (28nm) design using: (a) the proposed DNN-based extraction, (b) rule-based extraction, and (c) the proposed hybrid extraction at 5% accuracy level.



(a)

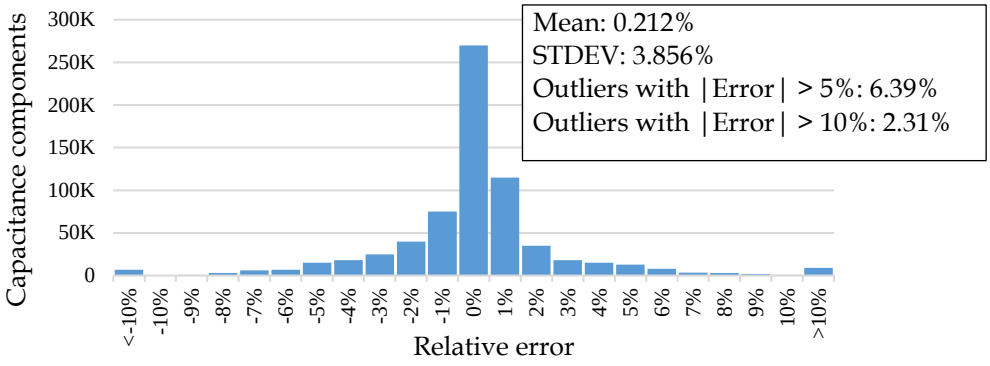


(b)

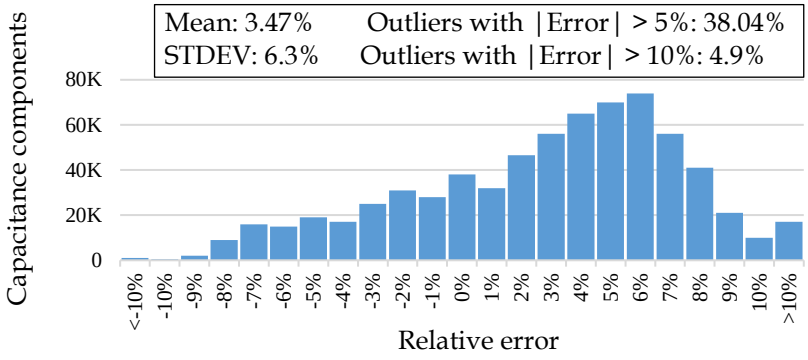


(c)

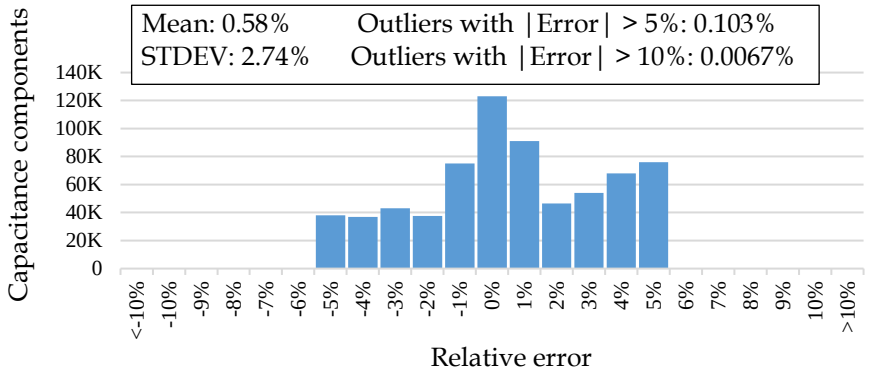
Fig. 6.15. Accuracy comparison histograms, relative to Calibre xACT3D, of the DAC (28nm) design using: (a) the proposed DNN-based extraction, (b) rule-based extraction, and (c) the proposed hybrid extraction at 5% accuracy level.



(a)

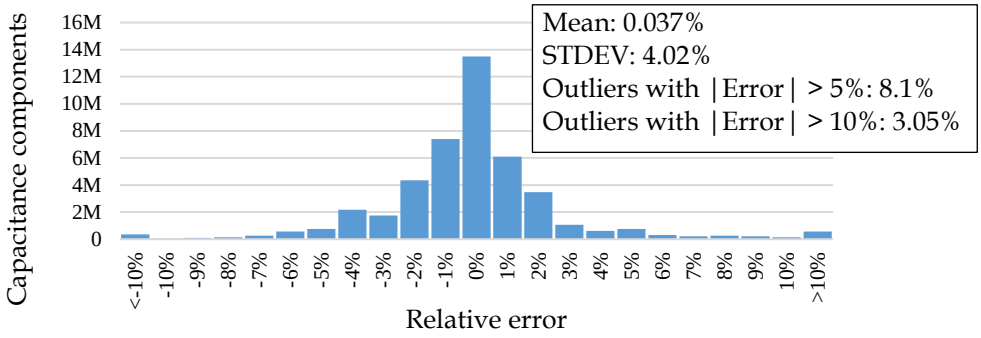


(b)

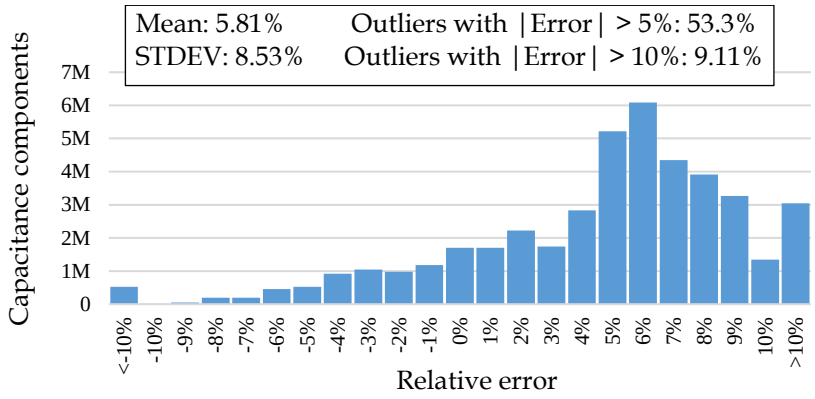


(c)

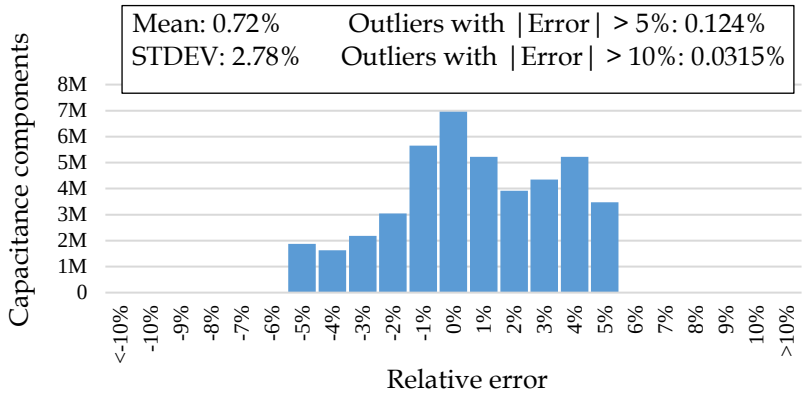
Fig. 6.16. Accuracy comparison histograms, relative to Calibre xACT3D, of the cache memory (28nm) using: (a) the proposed DNN-based extraction, (b) rule-based extraction, and (c) the proposed hybrid flow at 5% accuracy level.



(a)



(b)



(c)

Fig. 6.17. Accuracy comparison histograms, relative to Calibre xACT3D, of the PLL (7nm) using: (a) the proposed DNN-based extraction, (b) the rule-based extraction, and (c) the proposed hybrid extraction at 5% accuracy level.

Table 6.7. Total runtimes of capacitance extraction using rule-based, DNN-based, field-solver, hybrid flow (without using DNN-based), and accuracy-based hybrid extraction methods, at 5% accuracy.

Design	Runtime (Hours)				
	Rule-based	DNN-based	Field-solver	Hybrid (No DNN)	Accuracy-based Hybrid
SRAM(28nm)	0.23	0.42	49.2	28.49	0.66
DAC(28nm)	0.45	0.78	79.6	43.8	1.1
CM(28nm)	0.51	0.81	81	47.9	1.12
PLL(7nm)	2.17	3.425	352.8	216.33	5.26

Table 6.8. The confusion matrices of SRAM, DAC, CM, and PLL designs using the proposed multi-class extraction selector at 5% required accuracy level.

Design			Actual/True			ACC	ACC-PEX
			Rule-based	DNN-based	Field-solver		
SRAM (28nm)	**Predicted**	**Rule-based**	50,644	470	327	79.4%	99.07%
		DNN-based	23,473	45,934	339		
		Field-solver	186	371	394		
DAC (28nm)		**Rule-based**	143,925	964	706	80.26%	99.23%
		DNN-based	59,722	112,873	820		
		Field-solver	506	784	1,438		
CM (28nm)		**Rule-based**	122,537	905	613	78.53%	99.33%
		DNN-based	61,837	114,586	510		
		Field-solver	451	842	1,152		
PLL (7nm)		**Rule-based**	5,964,848	51,109	9,286	84.7%	99.05%
		DNN-based	1,957,103	5,944,609	7,3871		
		Field-solver	19,929	41,603	19,242		

6.4. Conclusion

A novel accuracy-based hybrid parasitic capacitance extraction flow is introduced. The proposed hybrid flow divides the chip into windows and extracts the parasitic capacitances of each window using one of three extraction methods: field-solver, rule-based, and novel deep neural-networks based extraction methods by directing each layout pattern to the fastest extraction method that meets the user pre-determined accuracy level. The proposed flow uses neural-networks classifiers to determine the capacitance extraction method for each window.

On the other hand, a novel deep neural-networks-based extraction method is developed as an intermediate parasitic extraction method between rule-based method

and field-solver method. The proposed hybrid flow is tested on different real designs. The results showed outstanding accuracy and runtime as compared to existing commercial field-solver and rule-based tools. Also, the proposed hybrid flow introduced a novel deep neural-networks modeling methodology that extracts parasitic capacitances of complicated structures with high accuracy levels (less than 3% average error) and 100X faster than field-solvers. Furthermore, the experimental results showed that the proposed hybrid flow managed to meet the required accuracy levels (of less than 5% error) with more than 99% accuracy, and with a speed up of 70X as compared to field-solvers.

Chapter 7

Parasitic-Aware Routing Optimization

The continuous scaling down of process technology nodes enabled the integration of more functionalities and systems together on a single chip. Such an integration significantly increased the complexity and density of layouts introducing more parasitic elements. The impact of interconnect parasitic elements on the overall circuit performance keeps increasing from one technology generation to the next. Moreover, the number of interconnect parasitic elements significantly increased in recent advanced processes. Therefore, the effects of interconnect parasitic elements are no longer second order effects. They are now dominating the overall circuit performance [3], [4], [100]. As a result, it is very important to consider the parasitic effects during placement and routing processes to reduce the overall turn-around-time of a circuit design and improve the yield.

This chapter aims to provide a new parasitic-aware routing optimization methodology. The proposed methodology can be applied either after or within the detailed routing. The proposed methodology enables circuit designers to debug and analyze the impact of parasitic elements on a circuit performance. Also, it provides a mechanism to identify the problematic parasitic elements and correlate them with specific layout geometries. Moreover, it uses nonlinear programming to re-route the problematic paths (i.e., routes) in order to achieve the required specifications with a full consideration of the surrounding environment. The proposed methodology uses a novel incremental parasitic extraction method in order to extract the parasitic elements of a modified layout during the optimization process. The proposed incremental extraction method provides very accurate parasitic extraction results with a maximum error < 1% as compared to a full layout extraction.

7.1. Incremental Parasitic Extraction

The layout parasitic extraction is an essential step in conventional integrated circuit (IC) design flows. It is used to extract parasitic elements of a given layout in order to perform a post-layout simulation. If the post-layout simulation results did not meet the required circuit's specifications, layout designers would modify the corresponding layout until its post-layout simulation results meet the required specifications. Usually, this process requires several iterations of layout modifications, parasitic extractions, and post-layout simulations until convergence.

There are two approaches to reduce the turn-around-time of the layout parasitic extraction step in design loops. First, some approaches use simplified parasitic models to speed up the extraction process and reduce the parasitic network such as in [20], [43], [44]. This approach is not efficient in advanced process technology nodes as it handles the parasitic effects as second order effects ignoring that the interconnect parasitic effects became one of the dominant factors on a circuit's performance in such advanced nodes [11]. Second, other approaches may use an incremental parasitic extraction to limit the parasitic extraction process to the modified polygons in a given layout. As a result, the execution time (i.e., runtime) of the layout parasitic extraction step in design loops decreases significantly with minimal impact on the extraction accuracy as compared to the use of a full layout parasitic extraction.

The incremental parasitic extraction aims to identify the modified layout geometries, extract the corresponding parasitic elements, and update the corresponding circuit network (i.e., netlist) with the newly extracted parasitic elements. In our work, the incremental parasitic extraction is used to extract parasitic resistances and capacitances of modified areas in a given layout.

7.1.1. Incremental Parasitic Resistance Extraction

As for parasitic resistances, they only depend on the geometrical shapes of modified layouts, and they do not depend on the surrounding environment. Therefore, the incremental parasitic resistance extraction identifies the modified layout polygons and

re-extracts their parasitic resistances smoothly without any consideration of the surrounding environment. After that, the corresponding circuit network (i.e., netlist) are updated with the newly extracted parasitic resistive elements.

7.1.2. Incremental Parasitic Capacitance Extraction

The incremental extraction of parasitic capacitances is more complicated than the incremental extraction of parasitic resistances because parasitic capacitances are highly correlated with the surrounding environment. In other words, if a layout polygon is modified, the modifications will not only impact the associated parasitic capacitive elements, but also, they will impact the parasitic capacitive elements among nearby metal polygons. Therefore, the incremental parasitic capacitance extraction needs to select and re-extract the parasitic capacitive elements that are impacted by layout modifications.

Existing incremental parasitic extraction methods can re-extract parasitic resistances efficiently; however, they cannot efficiently re-extract parasitic capacitance. This is because existing incremental methods only re-extract parasitic capacitances that are directly coupled with modified shapes (i.e., first order parasitic capacitances), and they ignore all coupling capacitances that are not directly coupled to modified layout shapes, such as second order coupling capacitances as shown in Fig. 7.1, even if those capacitances are significantly impacted by layout modifications [18], [101]. As a result, they provide a low extraction accuracy as compared to a full layout parasitic capacitance extraction. Fig. 7.2 shows an example of modifying the position of a nearby polygon on the second order coupling capacitance between two other fixed polygons.

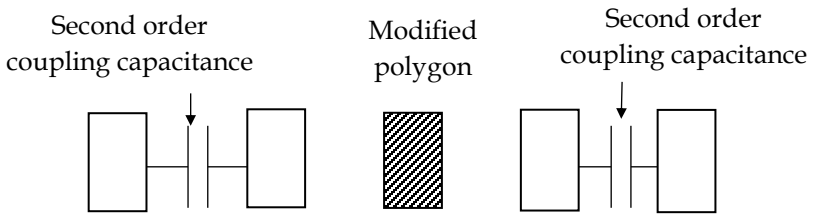


Fig. 7.1. An example of second order coupling capacitances due to modifying a certain metal polygon.

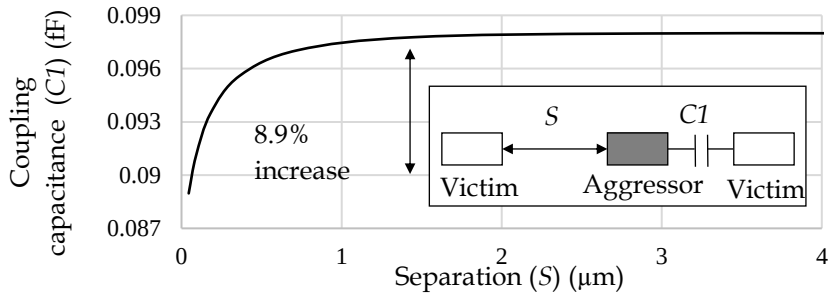


Fig. 7.2. The impact of increasing the separation between the aggressor and left victim polygons on the coupling between the aggressor and right victim polygons. The experiment used metal5 layer of 28nm process technology node.

A novel incremental parasitic capacitance extraction method is developed to extract first and second order capacitances efficiently. The developed method provides outstanding accuracy results as compared to a full layout extraction with a maximum relative error < 1%. Moreover, the impact of extracting second order capacitances on the total extraction runtime is negligible, where the time required to extract second order capacitances represents < 5% of the total incremental extraction runtime. The developed method has three main steps. First, it identifies the modified shapes and the corresponding metal layers. Second, it calculates a maximum coupling capacitance interaction range (MR) for each metal layer. Third, it extracts all coupling capacitances that are enclosed inside the maximum interaction range, and it updates the corresponding circuit's network (i.e., netlist) with the newly extracted parasitic capacitive elements. The three steps of the developed incremental capacitance extraction method are described in more details as follows.

7.1.2.1. Identify Modified Shapes

In this step, all metal polygons that are impacted by layout modifications are marked, where the modified metal polygons are marked, and the metal polygons that were previously interacting with the modified polygons (before modifications) are also marked. This is to ensure that all impacted parasitic capacitances are considered during the incremental extraction process.

7.1.2.2. Calculating the Maximum Capacitance Interaction Range

In this step, a maximum capacitance interaction range (MR) is calculated for each metal layer. The MR of a polygon represents the range (i.e., distance) where coupling capacitances to other polygons are negligible and do not impact the accuracy of a parasitic capacitance extraction. in other words, the MR identifies the valid coupling range for each layout polygon in order to avoid unnecessarily capacitance computations. The calculation of a MR depends on the corresponding metal stack specifications, where each metal layer in a certain process node has a different MR value.

The MR for each metal layer is only calculated once at the beginning of the incremental capacitance extraction process. For a certain metal layer, the MR is calculated by constructing two adjacent metal polygons using the corresponding minimum dimensions. Then, an electrostatic simulator is used to extract the lateral coupling capacitance between the two polygons. Also, the simulator performs a parametric sweep over lateral spacings while it measures the coupling capacitance between the two metal polygons until the MR is achieved, given that the MR represents the distance where the coupling capacitance between the two polygons is less than or equal to 1% of the total capacitance on one of the polygons as shown in Fig. 7.3.

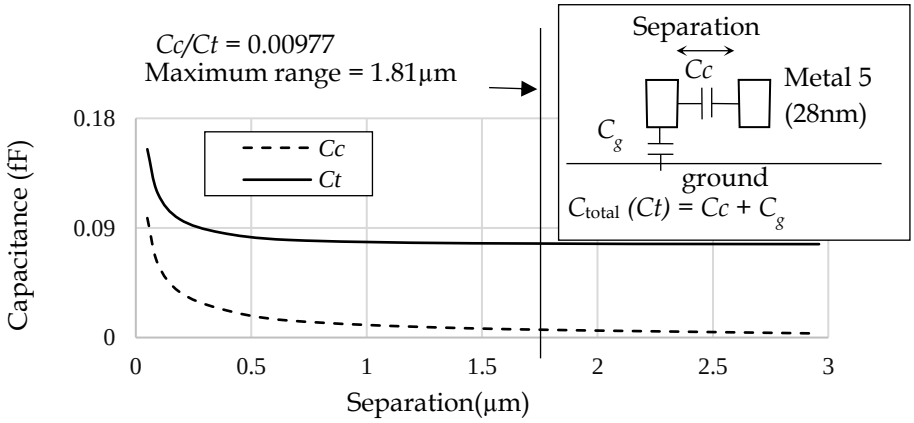


Fig. 7.3. The impact of increasing the separation (i.e., spacing) between two metal polygons on the lateral coupling capacitance between them using metal5 of 28nm process technology node.

7.1.2.3. Capacitance Extraction and Netlist Update

Once the maximum interaction ranges of all modified polygons are identified, all parasitic capacitive elements that are enclosed inside this range are re-extracted including second order parasitic capacitances. This ensures that all impacted capacitive elements are extracted, whereas the capacitive elements that are not enclosed inside the maximum interaction ranges are not extracted as shown in Fig. 7.4. Eventually, the corresponding circuit's network (i.e., netlist) is updated with the newly extracted parasitic capacitive elements.

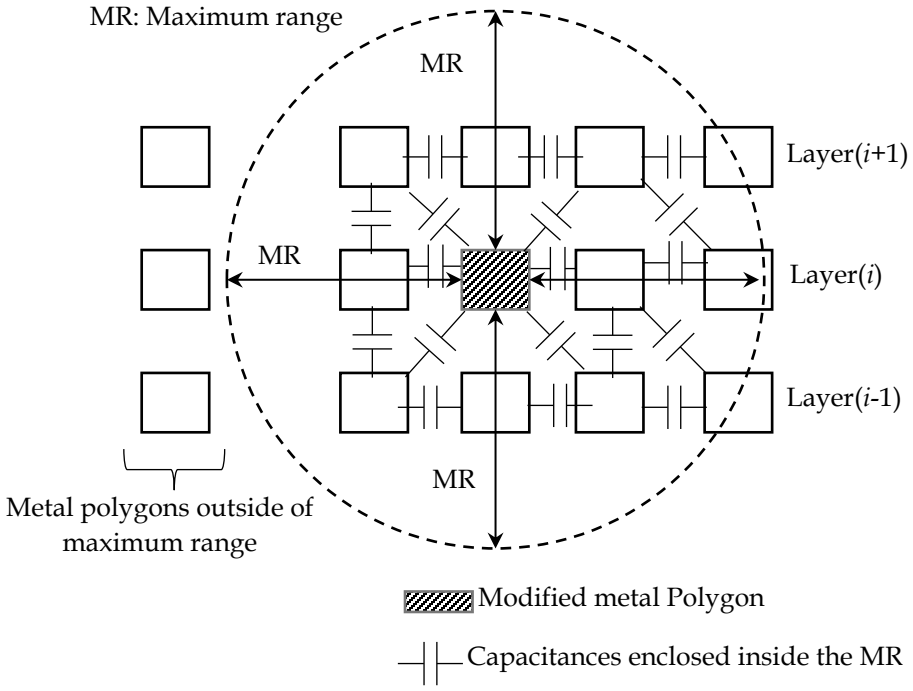


Fig. 7.4. An illustrative example of 2D cross-section metal polygons showing some capacitive elements that are enclosed inside the maximum capacitance interaction range of a modified polygon.

7.2. Parasitic-Aware Layout Routing Optimization Methodology

Parasitic-aware layout routing optimization methodology based on circuit moments is developed. The proposed routing methodology is used as a part of a template-based layout optimization flow. The proposed methodology has three main benefits. First, it helps circuit designers in analyzing the performance of critical routes. This is done by developing a sensitivity circuit model that measures the sensitivity of a route's performance cost function to the corresponding metal geometries. Second, the proposed

methodology efficiently considers the impact of parasitic elements during the optimization of critical routes by using a novel incremental parasitic extraction method. Third, the proposed methodology optimizes critical routes very fast using a cost function and corresponding sensitivity circuit models. The critical routes represent the routes that either hold analog signals or have a considerable impact on a circuit's performance. Such routes are identified by circuit designers after performing a sensitivity analysis across different routes, i.e., the sensitivity of a circuit performance to a route's network including parasitic elements.

The proposed methodology consists of three main steps as shown in Fig. 7.5. First, a performance cost function is developed, for example, a relative cost function that measures the performance difference between two routes. Second, sensitivity circuit models are derived to measure the sensitivities of a cost function to route's geometries. Third, a nonlinear programming is used to minimize a cost function subject to route's geometries considering the obtained sensitivity circuit models. The cost function minimization process considers different geometry constraints such as connectivity, blockages, and net symmetry constraints. Moreover, the optimization process can handle Manhattan and non-Manhattan geometries. It is worth mentioning that the proposed routing optimization method can be applied after the detailed routing step to provide further routing optimization improvements.

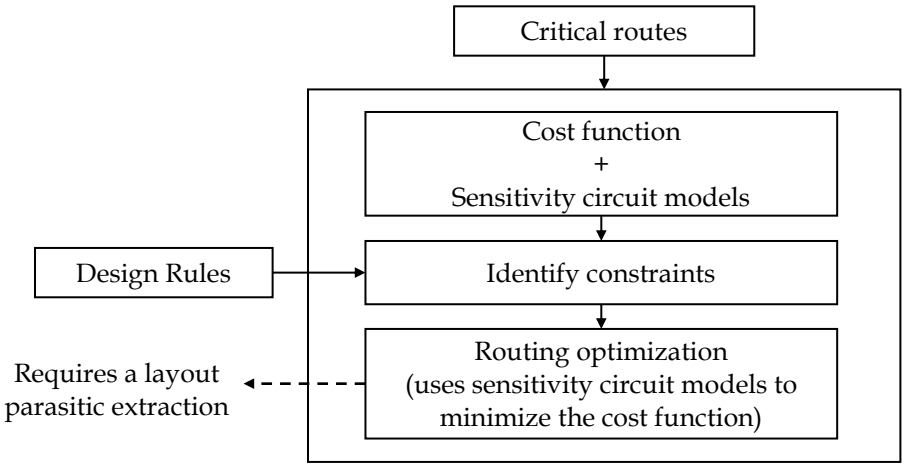


Fig. 7.5. The proposed layout optimization flow for critical routes.

The nonlinear programming requires a layout parasitic extraction process with every optimization iteration to evaluate the developed cost function. Therefore, a novel incremental parasitic extraction method is developed, as described in section 7.1. The developed incremental extraction method employs a full layout extraction tool, Calibre PEX [19], in an incremental manner in order to reduce the parasitic extraction runtime. Moreover, it provides high accuracy numbers as compared to a full layout extraction (<1% error).

7.2.1. Cost Function Development

Two cost functions are developed. The first one represents a net matching (i.e., symmetry), whereas the second one represents a route's delay.

7.2.1.1. Relative Cost Function

A cost function that measures the performance difference between two systems (i.e., routes) is developed as follows. Assuming two systems with output responses S_1 and S_2. The systems can belong to the same net, as shown in Fig. 7.6 (a), or different nets, as shown in Fig. 7.6 (b). The corresponding responses at their terminals are expressed by Taylor series expansions as below:

$$S_1(s) = m_0 + m_1 s + m_2 s^2 + m_3 s^3 + \cdots, \tag{7.1}$$

and

$$S_2(s) = m'_0 + m'_1 s + m'_2 s^2 + m'_3 s^3 + \cdots, \tag{7.2}$$

where m_i and m'_i are circuit moments at i^{th} order.

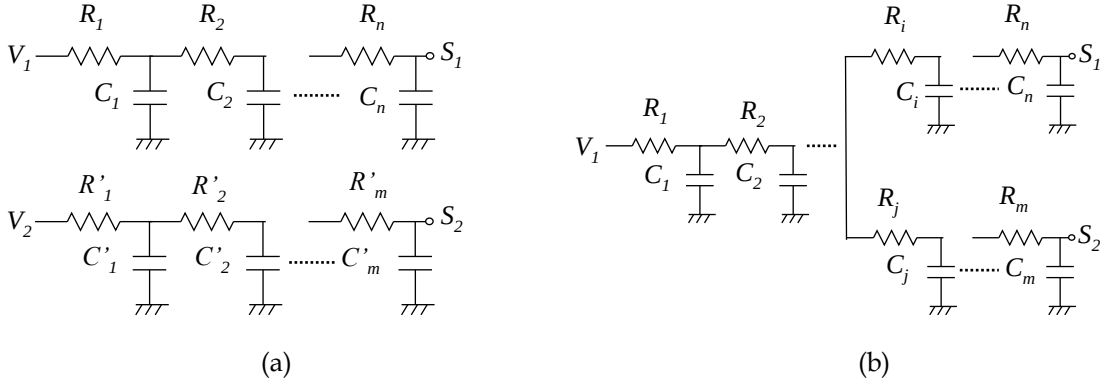


Fig. 7.6. Two different RC systems that belong to (a) two different routes, or (b) the same route.

To ensure that the two systems have the same output response, a relative cost function (RCF) is developed as below:

$$\text{relative cost function (RCF)} = \sum_{i=0}^{q} \frac{(m_i - {m'}_i)^2}{{m'}_i^2}, \tag{7.3}$$

where q represents the required order of circuit moments. The purpose of using a relative formula is to normalize the weights for all required moments to ensure that all required moments are equally considered (regardless of their order of magnitude) during the optimization process.

The RCF has two main uses. First, it is used to meet net symmetry constraints as it measures the performance (or response) error between two routes. Second, it is used to optimize critical layout routes by measuring the performance error between a certain critical route and the corresponding shortest path route assuming no blockages.

7.2.1.2. Delay cost function

Another cost function is developed based on circuit moments in order to minimize a route's delay. According to [102], for a certain network, the crossing time ($t_{rt,q}$) represents the time required by a signal to reach a certain voltage as shown in Fig. 7.7. The crossing time ($t_{rt,q}$) of a signal at a certain threshold ratio of a voltage (r_t) for q moments is given by:

$$t_{rt,q} = a_1 \cdot m_1 + a_2 \cdot \frac{m_2}{m_1} + a_3 \cdot \frac{m_3}{{m_1}^2} + \cdots + a_q \cdot \frac{m_q}{{m_1}^{(q-1)}}, \tag{7.4}$$

where the valid range of r_t is from 0 to 1, $t_{rt,q}$ is the time taken by the signal to achieve (or cross) the threshold voltage, q is the required order of moment, whereas a_1 to a_q are constant coefficients that might have different values based on the required threshold value (r_t). These constants were obtained using curve fitting operations as shown in [102].

In this work, a delay cost function is developed based on (7.4). The threshold voltage ratio of the crossing point is set to 0.5, and the maximum number of moments

(q) is set to 5 moments, as recommended by [102] to achieve a good accuracy. Therefore, the delay cost function (DCF) is given by:

$$\text{Delay cost function (DCF)} = t_{0.5,5}, \tag{7.5}$$

where the values of a_1 to a_5 coefficients are -3.05, 5.59, -4.36, 1.75, and -0.291, respectively as shown in [102].

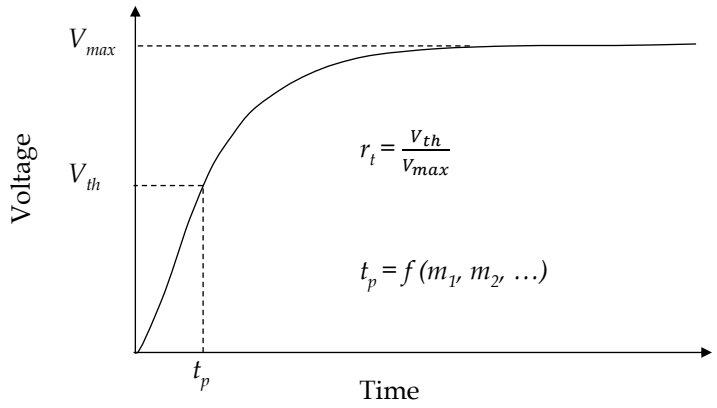


Fig. 7.7. An illustrative example of the threshold ratio (r_t) that represents the threshold-crossing point (t_p, V_{th}), where a time t_p is required by the signal to reach V_{th} voltage.

7.2.2. Sensitivity Circuit Models

In order to measure the impact of modifying layout geometries (i.e., route's geometries) on a cost function (CF), a circuit model that measures the sensitivity of CF to layout geometries is proposed and derived as below:

$$\overline{\frac{\partial \mathrm{CF}}{\partial Ge}} = \left[\frac{\partial \mathrm{CF}}{\partial P}\right]_{1\times n} . \left[\frac{\partial P}{\partial Ge}\right]_{n\times m}, \tag{7.6}$$

where P represents the associated parasitic elements, Ge represents route's geometries, n is the number of parasitic elements, whereas m is the number of corresponding layout geometries. In order to correlate the cost function with layout geometries (Ge), the geometries are represented by using their coordinates (or vertices). Therefore, the sensitivity of a cost function (CF) to layout geometries is given by:

$$\overline{\frac{\partial CF}{\partial Ge}} = \begin{bmatrix} \frac{\partial CF}{\partial R_1} \\ \frac{\partial CF}{\partial R_2} \\ \vdots \\ \frac{\partial CF}{\partial R_i} \\ \frac{\partial CF}{\partial Cc_{i+1}} \\ \frac{\partial CF}{\partial Cc_{i+2}} \\ \vdots \\ \frac{\partial CF}{\partial Cc_n} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial R_1}{\partial x_1} & \frac{\partial R_1}{\partial x_2} & \cdots & \frac{\partial R_1}{\partial y_{m-1}} & \frac{\partial R_1}{\partial y_m} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial Cc_n}{\partial x_1} & \frac{\partial Cc_n}{\partial x_2} & \cdots & \frac{\partial Cc_n}{\partial y_{m-1}} & \frac{\partial Cc_n}{\partial y_m} \end{bmatrix}, \tag{7.7}$$

where x and y represent the coordinates of route polygons as shown in Fig. 7.8, R is a parasitic resistive element, whereas Cc is a parasitic capacitive element. In order to provide a degree of freedom, routes are fractured into quadrilateral polygons (e.g., rectangles). As a result, the sensitivity and cost function calculations consider either Manhattan or non-Manhattan geometries.

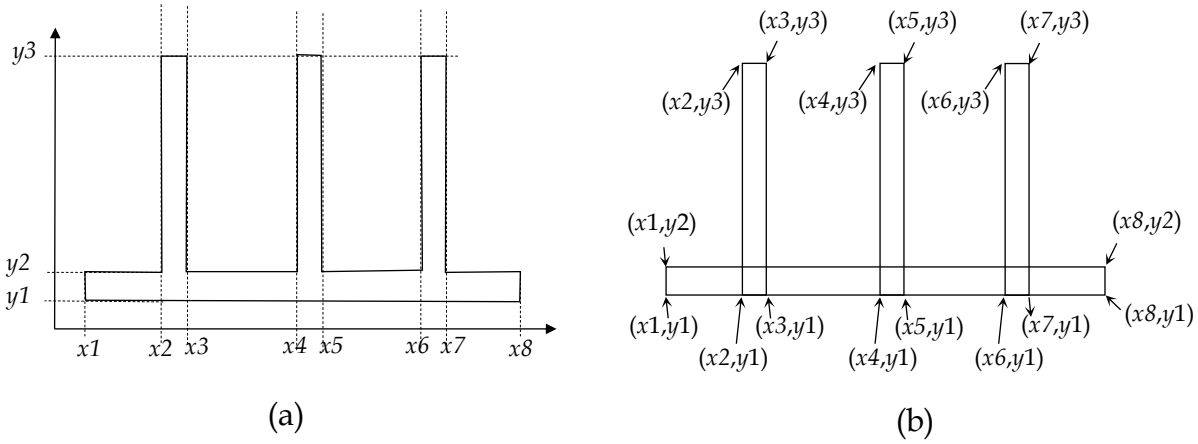


Fig. 7.8. An illustrative example of a geometry representation in the proposed sensitivity models showing (a) an unfractured polygon and (b) a fractured polygon.

The proposed model in (7.6) has two main components. First, the CF sensitivity to parasitic elements ($\partial \mathrm{CF}/\partial P$), which is different from one cost function to another. Second, the sensitivity of parasitic elements to system (i.e., route) geometries ($\partial P/\partial Ge$).

As for a cost function sensitivity to parasitic elements ($\partial \mathrm{CF}/\partial P$), two sensitivity models are developed. First, the relative cost sensitivity to a parasitic element, which is derived from the relative cost function in (7.3). Second, the delay cost sensitivity to a parasitic element, which is derived from the delay cost function in (7.5). Both of them are derived for each parasitic element (P_i) in order to fill the corresponding matrix. As for the sensitivity of parasitic elements to system geometries, it does not rely on the used cost function, and it can be used in (7.7) regardless of the used cost function. The three sensitivity models are derived as follows.

7.2.2.1. The Relative Cost Function Sensitivity to a Parasitic Element

As for the relative cost function sensitivity (RCF) to a parasitic element, it is obtained by differentiating (7.3) with a parasitic element (P_i) as below, given that the detailed derivations are found in the Appendix:

$$\frac{\partial \mathrm{RCF}}{\partial P_i} = \frac{\partial}{\partial P_i}\left(\frac{(m_0 - {m'}_0)^2}{{m'}_0{}^2} + \frac{(m_1 - {m'}_1)^2}{{m'}_1{}^2} + \cdots\right), \tag{7.8}$$

Let

$$\mathrm{RCF_{mk}} = \frac{(m_k - {m'}_k)^2}{{m'}_k{}^2}. \tag{7.9}$$

Therefore, by using m_k as an intermediate variable,

$$\frac{\partial \mathrm{RCF}}{\partial P_i} = \sum_{k=0}^{q} \frac{\partial \mathrm{RCF_{mk}}}{\partial m_k} \frac{\partial m_k}{\partial P_i}, \tag{7.10}$$

where m_k is a certain degree moment at a given node, q is the maximum required degree of moments, and $\mathrm{RCF_{mk}}$ is the relative cost function for a certain moment (i.e., relative moment cost function). This model has two components that include the sensitivity of a relative moment cost function to a circuit moment ($\partial \mathrm{RCF_{mk}}/\partial m_k$) and the sensitivity of a moment to a parasitic element ($\partial m_k/\partial P_i$).

As for the relative moment cost function sensitivity to a circuit moment, it is obtained by differentiating (7.9) with a moment (m_k) as below:

$$\frac{\partial \mathrm{RCF_{mk}}}{\partial m_k} = 2\ \frac{(m_k\ - {m'}_k)}{{m'}_k{}^2}. \tag{7.11}$$

As for the sensitivity of each moment to a parasitic element ($\partial m_k/\partial P_i$), it is obtained by differentiating (2.13) with a parasitic element (P_i) as below, given that the detailed derivations are found in the Appendix:

$$\frac{\partial m_0}{\partial P_i} = -G^{-1}\ \frac{\partial G}{\partial P_i} m_0, \tag{7.12}$$

and

$$\frac{\partial m_k}{\partial P_i} = -G^{-1}\left(\frac{\partial G}{\partial P_i}\ m_k\ + \frac{\partial C}{\partial P_i} m_{k-1} + C\,\frac{\partial m_{k-1}}{\partial P_i}\right), \qquad k \geq 1 \tag{7.13}$$

where C is the capacitors matrix, G is the admittance matrix, and m_0 to m_k are circuit moments at a given node.

Eventually, the sensitivity of an RCF to a parasitic element (P_i) is obtained by substituting (7.11), (7.12), and (7.13) in (7.10) as below, given that the detailed derivations are found in the Appendix:

$$\begin{aligned}\frac{\partial \mathrm{RCF}}{\partial P_i} = \ & 2\ \frac{(m_0\ - {m'}_0)}{{m'}_0{}^2} \\ & \cdot\left(-G^{-1}\ \frac{\partial G}{\partial P_i} m_0\right) + \sum_{k=1}^{q}\left(2\ \frac{(m_k\ - {m'}_k)}{{m'}_k{}^2}\right. \\ & \left.\cdot\left(-G^{-1}\left(\frac{\partial G}{\partial P_i} m_k\ + \frac{\partial C}{\partial P_i} m_{k-1} + C\,\frac{\partial m_{k-1}}{\partial P_i}\right)\right)\right).\end{aligned} \tag{7.14}$$

7.2.2.2. The Delay Cost Function Sensitivity to a Parasitic Element

As for the delay cost function (DCF) sensitivity to a parasitic element (P_i), it is obtained by differentiating (7.5) with a parasitic element (P_i) as below, given that the detailed derivations are found in the Appendix:

$$\frac{\partial DCF}{\partial P_i} = \ a_1 \cdot \frac{\partial m_1}{\partial P_i} + \sum_{k=2}^{q}\left[a_k\left(\frac{\partial m_k}{\partial P_i} \cdot \frac{1}{{m_1}^{k-1}} + \ m_k \cdot \frac{(1-k)}{{m_1}^{k}} \cdot \frac{\partial m_1}{\partial P_i}\ \right)\right], \tag{7.15}$$

where $\partial m_k/\partial P_i$ is obtained in (7.13).

7.2.2.3. A Parasitic Sensitivity to Layout Geometries

As for parasitic sensitivities to layout geometries ($\partial P/\partial Ge$), they are measured by using the proposed incremental parasitic extraction flow, which provides very fast and localized sensitivity numbers. For a certain parasitic element (P_i) and geometry parameter (x_j), the sensitivity is calculated using the below formula:

$$\frac{\partial P_i}{\partial x_j} = \frac{P_i(x_{j+1}) - P_i(x_j)}{x_{j+1} - x_j}, \tag{7.16}$$

where $P_i(x_{j+1})$ is the value of a parasitic element (P_i) when a geometry x equals x_{j+1}, $P_i(x_j)$ is the value of a parasitic element (P_i) when a geometry x equals x_j.

7.2.3. Performance Analysis to Identify Critical Geometries

It is very important to understand and analyze the impact of layout geometries on a route's performance. This would help identifying the most sensitive geometries to a route's performance cost function, speeding up the optimization process, and achieving better optimization results.

The performance analysis is performed by using the cost sensitivity to layout geometries model in (7.7). However, the sensitivity analysis mainly relies on the required performance cost function. In case of performing net matching analysis, the sensitivity models of the relative cost function in (7.7), (7.14), and (7.16) are used. In case of performing a delay analysis, the sensitivity models of the delay cost function in (7.7), (7.15), and (7.16) are used. The higher the sensitivity value, the higher the impact on a route's performance.

As for a general performance analysis, the sensitivity models of the relative cost function may be used in three steps. First, identify the critical routes. Second, create a shortest path route assuming no blockages as a reference route. Third, use (7.7), (7.14), and (7.16) in order to calculate the sensitivity of the RCF to route's geometries using the moments of a shortest path route as reference moments.

7.2.4. Geometrical Constraints

Once the most sensitive geometries are selected, they are used as optimization parameters for the routing optimization process; however, this requires maintaining constraints such as the corresponding process design kit (PDK), net blockage constraints, connectivity, and net symmetry constraints. The constraints are obtained using a symbolic template approach.

7.2.5. Layout Routing Optimization Process

The purpose of this step is to minimize a cost function with respect to the most sensitive route's geometries (i.e., coordinates) using a nonlinear programming. The sequential linear-quadratic programming (SLSQP) algorithm is used as a nonlinear programming algorithm because it is an iterative approach for nonlinear optimization problems that accepts multiple constraints. In order to provide degrees of freedom for the routing optimization process, the target routes are fractured into quadrilateral shapes. The number of fractured polygons relies on the required number of degrees of freedom. The fracturing is done in two steps. First, the polygons are scanned in the x direction and fractured vertically. Second, the polygons are scanned in the y direction and fractured horizontally as shown in Fig. 7.8 (b). Each fractured polygon holds four vertices conforming a quadrilateral polygon. The fractured polygons are used to create and evaluate the sensitivity circuit models in (7.7).

The optimization algorithm is shown in Fig. 7.9. The inputs of the algorithm are: 1) the target routes and 2) the constraints including the new design requirements, whereas the outputs are new routes that are represented by their coordinates. It is worth mentioning that the minimization of a cost function uses the derived sensitivity model, in (7.7), to create the Jacobean matrix that are used by the nonlinear programming algorithm. It is worth mentioning that the routing optimization algorithm is implemented using Python [84].

```
1  Inputs:
2    Routes[1..n]: List of routes that require optimization, and their count is n.
3    Constraints[1..m]: List of constraints, and their count is m.
4  Output:
5    New_Routes[1..n]: final list of optimized routes
6  Begin
7    Routes = initial current routes.
8    CF = initial values of a cost function across all routes.
9    for i ∈ [1..n]       //foreach route
10     R = Routes[i]   //in case of a delay optimization, it contains one route
                       //and in case of a net matching optimization, it contains the two routes.
11     while (optimization is needed) //i.e., gradient is needed
12       Parasitics ← extract_parasitics(R)  //extract parasitics of routes in (R)
13       dPdGe ← calculate_dPdGe(R, Parasitics)  // ∂P/∂Ge  using (7.16)
14       Moments ← calculate_moments(Parasitics) //  using (2.13)
15       dCFdP ← calculate_dCFdP(Moments, Parasitics)  //  calculate ∂CF/∂P using (7.14) or (7.15)
16       dCFdGe ←calculate_dCFdGe (dCFdP, dPdGe)
           // calculate ∂CF/∂Ge using (7.7) to identify the most sensitive geometries for optimization.
17       R ← optimize_route(R, dCFdGe, Constraints, SLSQP)
                          // At this point, R holds an updated route.
18       New_Parasitics← extract_parasitics(R)
19       New_Moments←calculate_moments(New_Parasitics) //using (2.13)
20       CF[i]←calculate_cost_value(New_Moments)  // using (7.3) or (7.5) to calculate new cost value
21     end while
22     New_Routes[i] = R
23   end for
24 End
```

Fig. 7.9. The proposed routing optimization algorithm pseudo code.

7.3. Experimental Results

The testing covered the proposed incremental parasitic capacitance extraction method, the derived sensitivity models, and the proposed parasitic-aware routing optimization method. The testing used Intel Xeon(R) E5-2680, 2 CPUs, 2.50GHz, and 16GB of RAM.

7.3.1. Testing the Proposed Incremental Capacitance Extraction

The accuracy and runtime of the proposed incremental parasitic capacitance extraction were tested and compared against a full layout parasitic capacitance extraction across three designs that include Ring Oscillator (RO) (7nm), Digital to Analog converter (DAC) (28nm), and voltage-controlled oscillator (VCO) (40nm) designs. Calibre PEX is used as an extraction tool for both incremental and full layout parasitic extractions. The testing methodology involves modifying metal shapes for some critical nets. The modifications include deleting, moving, stretching, and adding new metal polygons. Each modified layout is tested by running a full layout parasitic extraction, the proposed incremental extraction, and the incremental extraction without considering the second order capacitances.

As for the RO (7nm), some input and output nets of RO stages were modified in three different ways: 1) modifying two metal layers with 1075 parasitic capacitive elements (i.e., small), 2) modifying three metal layers with 2037 parasitic capacitive elements (i.e., medium), and 3) modifying four metal layers with 3524 parasitic capacitive elements (i.e., large). As shown in Table 7.1, The maximum relative errors in the three scenarios after applying the proposed incremental parasitic extraction flow as compared to the full parasitic extraction are 0.14%, 0.25%, and 0.5%, respectively. Moreover, the relative speedup of the proposed incremental flow as compared to the full layout extraction in the three scenarios is 40.4, 27.8, and 21.15, respectively. Furthermore, the results show that the consideration of the second order parasitic capacitances has a very small impact on the runtime as compared to the incremental extraction that does not consider the second order parasitic capacitances.

Table 7.1 Testing results of the proposed incremental capacitance extraction method using a RO with 31 stages (7nm).

Component	Modification Type		
	Small	**Medium**	**Large**
Capacitive elements	1075	2037	3524
Metal layers	2	3	4
Max error of the proposed method	0.14%	0.25%	0.5%
Incremental extraction runtime in seconds (secs)	11 secs	16 secs	21 secs
Full extraction runtime (minutes)	7.4 minutes		
Relative speedup as compared to a full extraction run	40.4	27.8	21.15
Incremental extraction runtime without second order capacitances	10.4 secs	15.1 secs	19.5 secs

As for the DAC (28nm), several nets were modified in three different ways: 1) modifying two metal layers with 9015 parasitic capacitive elements (i.e., small), 2) modifying three metal layers with 11422 parasitic capacitive elements (i.e., medium), and 3) modifying four metal layers with 14372 parasitic capacitive elements (i.e., large). As shown in Table 7.2, the maximum errors in the three scenarios after applying the proposed incremental parasitic extraction flow as compared to the full parasitic extraction are 0.21%, 0.47%, and 0.67%, respectively. Moreover, the relative speedup of the proposed incremental flow as compared to the full layout extraction in the three scenarios is 43.16, 32.12, and 21.14, respectively.

Table 7.2 Testing results of the proposed incremental capacitance extraction method using a DAC (28nm).

Component	Modification Type		
	Small	**Medium**	**large**
Capacitive elements	9051	11422	14372
Metal layers	2	3	4
Max error of the proposed method	0.21%	0.47%	0.67%
Incremental extraction runtime in minutes (mins)	4.31 mins	5.79 mins	8.8 mins
Full extraction runtime	3.1 hours		
Relative speedup as compared to full run	43.16	32.12	21.14
Incremental extraction runtime without second order capacitances	4.12 mins	5.47 mins	8.25 mins

As for the VCO (40nm), several nets were modified in three different ways: 1) modifying two metal layers with 11768 parasitic capacitive elements (i.e., small), 2) modifying three metal layers with 12794 parasitic capacitive elements (i.e., medium), and 3) modifying four metal layers with 17724 parasitic capacitive elements (i.e., large).

As shown in Table 7.3, the maximum errors in the three scenarios after applying the proposed incremental parasitic extraction flow as compared to the full parasitic extraction are 0.19%, 0.38%, and 0.63%, respectively. Moreover, the relative speedup of the proposed incremental flow as compared to the full layout extraction in the three scenarios is 54.2, 43.07, and 35.1, respectively.

Table 7.3 Testing results of the proposed incremental capacitance extraction method a VCO (40nm).

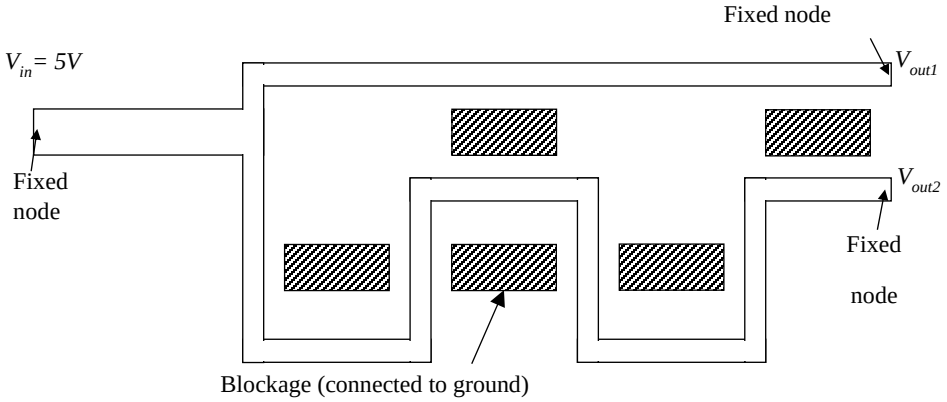
Component	Modification Type		
	Small	**Medium**	**large**
Capacitive elements	11768	12794	17724
Metal layers	2	3	4
Max error of the proposed method	0.19%	0.38%	0.63%
Incremental extraction runtime in minutes (mins)	6.67 mins	8.4 mins	10.3 mins
Full extraction runtime	6.03 hours		
Relative speedup as compared to full run	54.2	43.07	35.1
Incremental extraction runtime without second order capacitances	6.35 mins	7.93 mins	9.65 mins

Table 7.1, Table 7.2, and Table 7.3 summarize the experimental results of the RO (7nm), DAC (28nm), and VCO (40nm) designs, respectively. As shown in the tables, the proposed incremental extraction flow provides an outstanding accuracy as compared to full extraction with maximum errors < 1% and with huge runtime savings of up to 54X. Furthermore, the results show that the consideration of the second order parasitic capacitances has a very small impact on the runtime as compared to the incremental extraction that does not consider the second order parasitic capacitances.

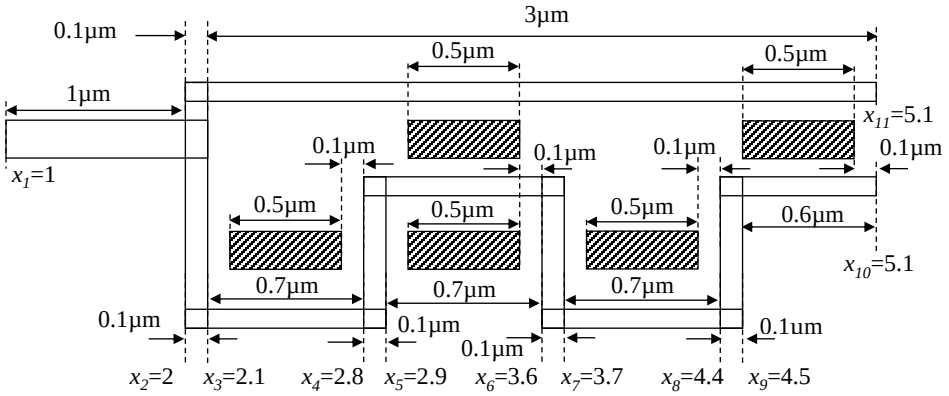
7.3.2. Testing the Proposed Parasitic Sensitivity Models and Routing Optimization Method Using a Simple Interconnect Structure

The proposed sensitivity models were tested using the interconnect structure shown in Fig. 7.10. This experiment has two purposes. First, it aims to measure the sensitivity of the relative cost function (RCF) to each layout geometry (i.e., coordinate) using (7.7), where the relative cost function measures V_{out2} moments relative to V_{out1} moments. Second, it aims to match the signal responses at V_{out1} and V_{out2} by optimizing the geometries of V_{out2} route. This is done by using a nonlinear programming to minimize the relative cost function in (7.3).

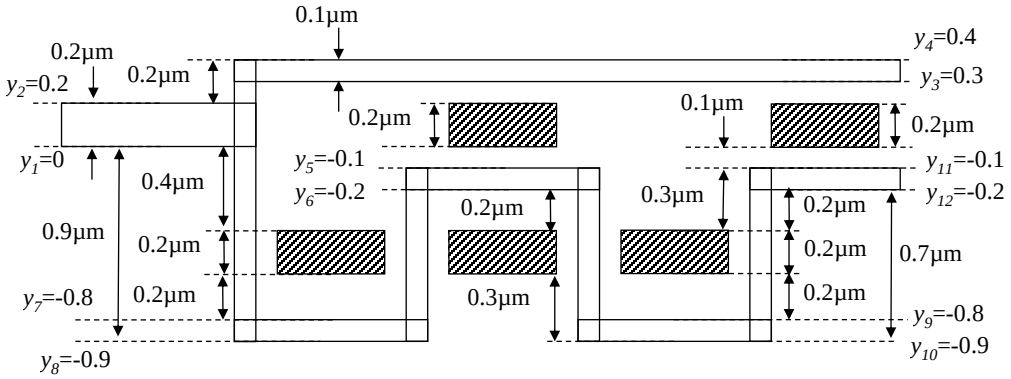
Fig. 7.10 (a) shows the experimental interconnect structure. It contains one input pin, V_{in}, and two output pins that include V_{out1} and V_{out2}. The surrounding dielectric constant is set to 3.9, the elevation of the metal is set to 1 μm, the metal thickness is set to 0.1μm, whereas the sheet resistance is set to 3 Ω/□.



(a)



(b)



(c)

Fig. 7.10. An experimental interconnect structure that is used for verifying the sensitivity circuit models and the optimization algorithm highlighting (a) the nodes, (b) the dimensions in the *x*-direction, and (c) the dimensions in the y-direction, given that all dimensions are in μm.

The experiment aims to match the signal responses of V_{out1} and V_{out2} without moving the fixed nodes that represent the locations of input and output pins. The route of V_{out2} pin has four obstacles (i.e., blockages). Therefore, V_{out2} route should pass through such obstacles with minimal impact on the performance. The dimensions of the interconnect are shown in Fig. 7.10 (b) and Fig. 7.10 (c). The optimization process used Calibre PEX, [19], to extract the parasitic elements of the interconnect structure. Table 7.4 shows the initial values (at the original interconnect dimensions) of the relative cost function sensitivities to the coordinates of V_{out2} route using (7.7). It is worth mentioning that the sensitivities are nonlinear. Therefore, they are calculated with every optimization iteration.

Table 7.4. The values of the sensitivity of the relative cost function to each Vout2 coordinate in the experimental interconnect structure.

Sensitivity parameter	Value	Sensitivity parameter	Value
$\boldsymbol{\partial CF/\partial x2}$	181.226	$\boldsymbol{\partial CF/\partial y5}$	-936.95
$\boldsymbol{\partial CF/\partial x3}$	-1395.2	$\boldsymbol{\partial CF/\partial y6}$	937.001
$\boldsymbol{\partial CF/\partial x4}$	1416.3	$\boldsymbol{\partial CF/\partial y7}$	-903.5
$\boldsymbol{\partial CF/\partial x5}$	-1020.98	$\boldsymbol{\partial CF/\partial y8}$	837.4
$\boldsymbol{\partial CF/\partial x6}$	1307.1	$\boldsymbol{\partial CF/\partial y9}$	926.7
$\boldsymbol{\partial CF/\partial x7}$	-1120.98	$\boldsymbol{\partial CF/\partial y10}$	-843.9
$\boldsymbol{\partial CF/\partial x8}$	1902.3	$\boldsymbol{\partial CF/\partial y11}$	884.2
$\boldsymbol{\partial CF/\partial x9}$	-1813.7	$\boldsymbol{\partial CF/\partial y12}$	-809.7

Moreover, a nonlinear programming is applied using SLSQP method in order to minimize the relative cost function. The nonlinear programming uses V_{out2} interconnect geometries (i.e., coordinates) as optimization parameters. Fig. 7.11 shows the optimized interconnect structure. Fig. 7.12 (a) shows the signal responses at V_{out1} and V_{out2} before the optimization process, whereas Fig. 7.12 (b) shows the signal responses after the optimization process. As for the cost values, the value of the relative cost function before the optimization is 0.391, whereas the value of the relative cost function after the optimization is 0.002047.

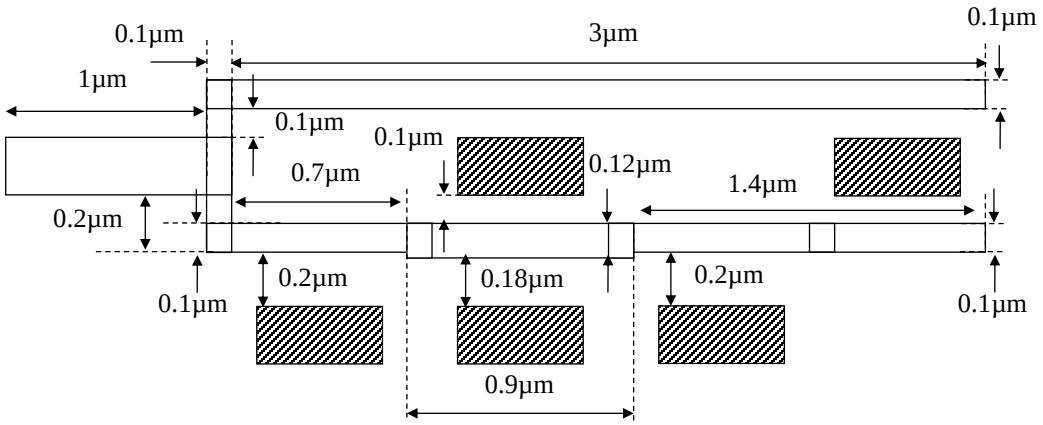
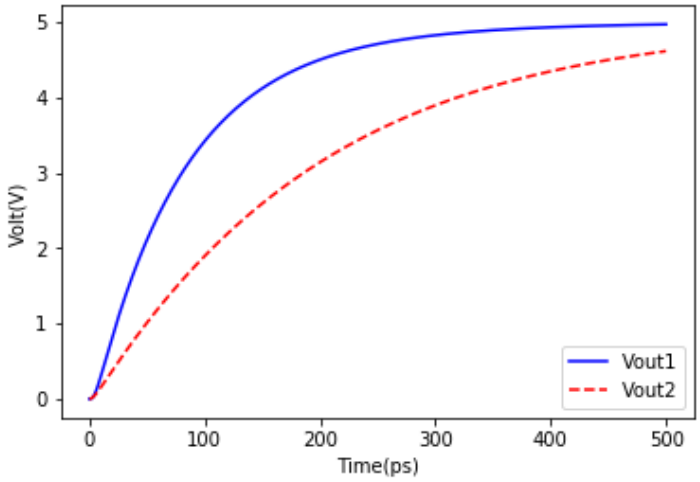
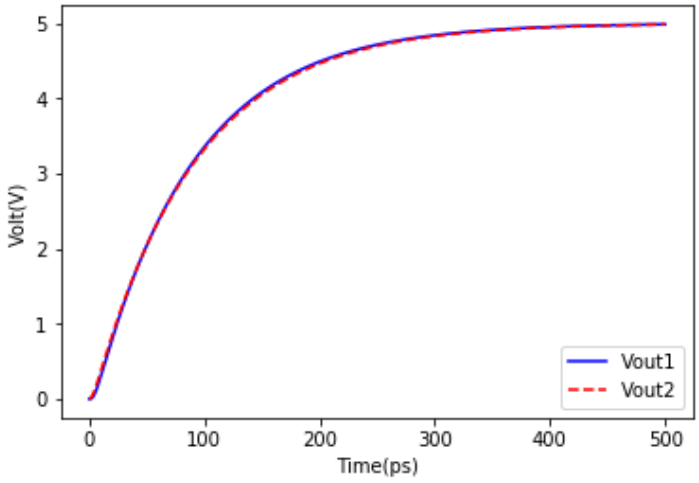


Fig. 7.11. The experimental interconnect structure after the optimization process.



(a)



(b)

Fig. 7.12. The output response of the experimental interconnect structure at Vout1 and Vout2 (a) before the optimization process and (b) after optimization process.

7.3.3. Testing the Layout Routing Optimization Method Using Circuit Designs

The proposed circuit models and layout optimization methodology were tested across different designs that include Ring Oscillators (RO) of 7nm process node and folded cascode operational amplifiers with common mode feedback of 65nm process node. The performance of the proposed optimization method was tested in terms of the accuracy and the optimization runtime. The accuracy was measured and compared to the required circuit specifications. The results were also compared against the traditional template-based layout optimization method that is described in [20], [44], [66], [68].

7.3.3.1. Ring Oscillator (7nm)

As for the RO(7nm), six different RO designs each with 31 stages were tested using 0.75V as an operating voltage. The optimization used the delay cost function in (7.5) and its corresponding sensitivity circuit models. The optimization included the input and output pins (i.e., input and output routes) of each stage. As shown in Table 7.5, the proposed optimization flow managed to reduce the delay of the six RO designs by 9.32%, 10.33%, 10.79%, 9.68%, 10.65%, and 11.1%, respectively, as compared to traditional template-based methods. Moreover, the relative speedup of the proposed method as compared to the traditional template-based method for the six designs is 9.06, 8.91, 9.48, 8.7, 9.27, and 8.54, respectively, as shown in Table 7.5. The reason behind such improvements is that traditional template-based optimization methods use multiple circuit simulations in order to identify the parasitic bounds, and each simulation consumes around 29 minutes. As for the delay improvements, traditional template-based methods use simplified parasitic formulas that are not suitable for advanced process technology nodes, whereas the proposed method uses the proposed incremental extraction method. As for the area, both optimization methods provided almost the same area.

Table 7.5. The testing results of the proposed routing optimization method as compared to a traditional template-based method across six different RO (7nm) designs.

	Traditional template-based routing Method		Proposed Method	
	Delay	**Optimization runtime**	**Delay**	**Optimization runtime**
RO1	7.51ps	4.23 hours	6.81ps	28 minutes
RO2	8.13ps	4.31 hours	7.29ps	29 minutes
RO3	9.27ps	4.11 hours	8.27ps	26 minutes
RO4	8.47ps	4.35 hours	7.65ps	30 minutes
RO5	8.26ps	4.17 hours	7.38ps	27 minutes
RO6	9.10ps	4.41 hours	8.09ps	31 minutes

7.3.3.2. Folded Cascode Differential Amplifier with Common Mode Feedback (65nm)

Folded cascode differential amplifiers with common mode feedback (CMFB) circuits were tested using three different specifications. The Amplifiers were developed using 65nm process node. Fig. 7.13 shows a block diagram of the amplifiers, whereas Fig. 7.14 shows a schematic circuit design of the folded cascode differential amplifier.

The optimization used the relative cost function in (7.3) and its corresponding sensitivity circuit models with 5 moments. The optimization was performed over 7 routes, Route1 to Route7, as shown in Fig. 7.14. The optimization aimed to match the responses (i.e., net matching) at the output terminal of each two similar routes, where Route1 was matched with Route2, Route3 was matched with Route4, and Route5 was matched with Route6. Moreover, the responses at the output terminals (i.e., t_1 and t_2) of Route7 were also matched.

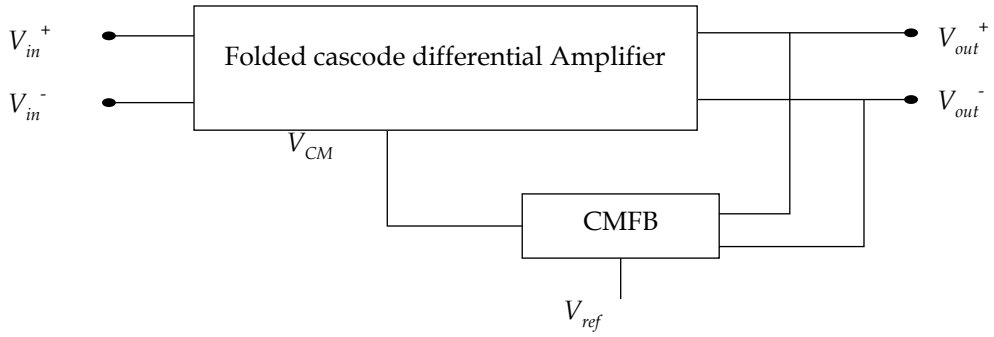


Fig. 7.13. Block diagram of a fully differential folded cascode amplifier with common mode feedback circuit.

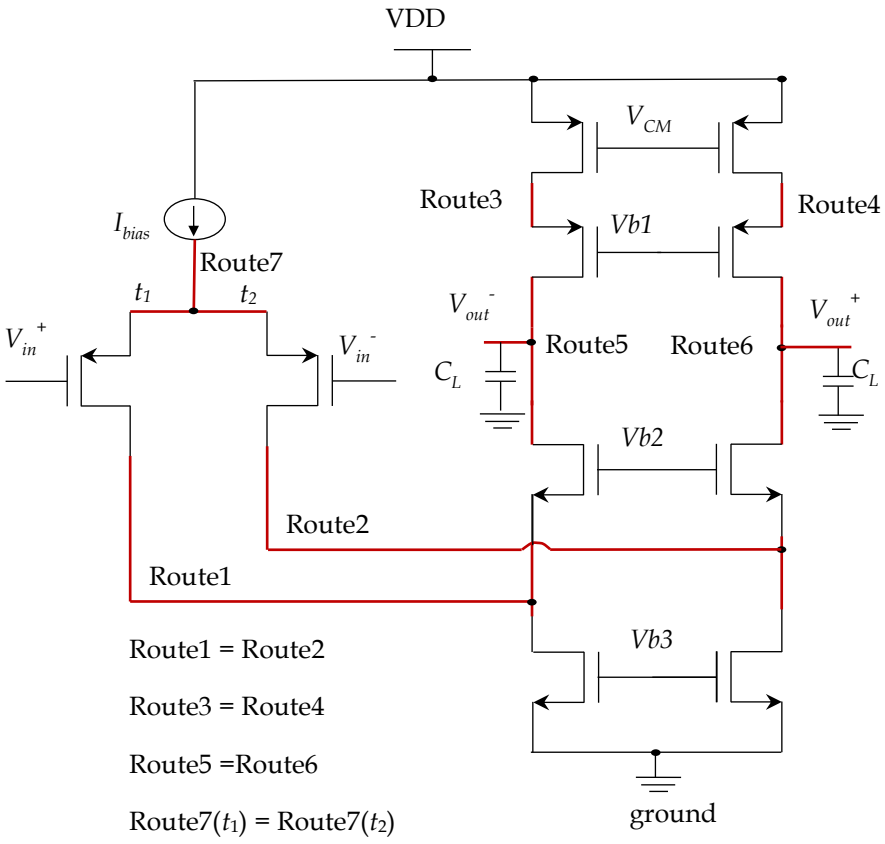


Fig. 7.14. A circuit design of an experimental folded cascode operational amplifier (65nm) showing the optimized routes.

Tables 7.6, 7.7, and 7.8 show the optimization results for three different required specifications. The results show that the proposed optimization method managed to provide closer results to the specifications as compared to the traditional template-based method with minimal impact on the area. Moreover, the optimization runtimes

of the proposed method for the three cases were faster than the traditional template-based method with a speedup of 3.6X, 3.36X, 3.41X, respectively.

Table 7.6. The testing results of the proposed routing optimization method as compared to a traditional template-based method over the first specification requirements of a folded cascode differential amplifier.

	Specifications	**Traditional Method**	**Proposed Method**
Gain (dB)	60.0	61.1	60.3
GBW (Hz)	350M	361M	352M
PM (°)	60.0	63.1	61.5
Output swing (V)	0.8	0.76	0.78
Loading capacitance (pF)	1pF		
Optimization runtime		4.3 minutes	1.2 minutes
Area (μm)²		2958	2962

Table 7.7. The testing results of the proposed routing optimization method as compared to a traditional template-based method over the second specification requirements of a folded cascode differential amplifier.

	Specifications	**Traditional Method**	**Proposed Method**
Gain (dB)	50.0	53.2	51.1
GBW (Hz)	300M	309M	303M
PM (°)	50.0	53.8	50.7
Output swing (V)	0.9	0.88	0.89
Loading capacitance (pF)	1pF		
Optimization runtime		4.5 minutes	1.34 minutes
Area (μm)²		3162	3150

Table 7.8. The testing results of the proposed routing optimization method as compared to a traditional template-based method over the third specification requirements of a folded cascode differential amplifier.

	Specifications	**Traditional Method**	**Proposed Method**
Gain (dB)	60.0	61.3	60.8
GBW (Hz)	600M	612M	604M
PM (°)	55.0	57.1	55.6
Output swing (V)	0.8	0.78	0.79
Loading capacitance (pF)	1pF		
Optimization runtime		4.7 minutes	1.38 minutes
Area (μm)²		3364	3352

7.4. Conclusion

A parasitic-aware layout routing optimization methodology is developed. Existing layout routing optimization methods suffer from three main problems. First, they rely on many circuit simulations to calculate the parasitic bounds. Second, they rely on either simple parasitic models, which provide poor accuracy, or a full layout extraction, which consumes a lot of time, in order to extract the parasitic elements of a given layout during the optimization process. Third, they do not provide a mechanism to analyze the impact of parasitic elements and corresponding geometries on a system's performance. The proposed methodology overcomes such limitations by providing novel sensitivity circuit models that help circuit designers in analyzing the impact of parasitic elements and corresponding layout geometries on a system's performance. Moreover, it provides a novel incremental parasitic capacitance extraction methodology that helps in providing a significant speeding up in the optimization runtime with minimal impact on the accuracy as compared to those methods that use a full layout extraction. The proposed optimization method uses a nonlinear programming technique to modify and optimize the problematic routes based on the proposed sensitivity circuit models. The proposed methodology is tested over different ring oscillator designs of 7nm process node and folded cascode differential amplifiers of 65nm process node. The experimental results show that the proposed methodology managed to achieve better accuracy and runtime results as compared to traditional template-based layout routing optimization methods. The proposed methodology managed to identify and optimize the problematic geometries in critical routes with up to 10% improvements in the performance and a speed up of 3 to 9X as compared to traditional template-based methods.

Chapter 8

Conclusion

This work provided new solutions to: 1) improve the accuracy of the rule-based 2.5D interconnect parasitic capacitance extraction models; 2) provide new models to predict parasitic capacitances of MEOL; 3) introduce new parasitic capacitance extraction method based on NN that provides high accuracy values in a reasonable runtime; 4) introduce new hybrid parasitic capacitance extraction method; and 5) introduce new parasitic-aware routing methodology based on an incremental parasitic extraction and a fast optimization algorithm.

As for the **rule-based 2.5D parasitic capacitance models**, a novel modeling methodology is developed using machine learning methods. The proposed methodology managed to overcome several problems in rule-based extraction tools such as handling systematic process variations, high pattern mismatches, and limited pattern coverages. It aims to create new machine learning compact models in order to predict parasitic capacitances between different metal polygons in 2D cross-section layout patterns. The models are created for each process technology node in order to simplify the modeling process and reduce the number of input variables. The input of the compact models is a given cross-section pattern including the required capacitances and the corresponding systematic process variations. Three different input representations are introduced including: 1) ratio-based; 2) dimensions-based; and 3) vertex-based pattern representations. Moreover, two different machine learning methods are used to implement the compact models including neural networks and support vector regression methods. The proposed methodology is tested over multiple real designs of 28nm, 14nm, and 7nm process nodes with more than 6.7M interconnect patterns. The generated compact models are faster than traditional rule-based models by 2.5X. Also, they managed to achieve outstanding results as compared to field-solvers and rule-based

cross-section models, where the average relative error of the generated models is < 0.15% and the standard deviation of relative errors is < 3.31%.

As for the **MEOL parasitic capacitance models**, a new modeling methodology is developed. The methodology aims to create machine learning models to predict parasitic coupling capacitances for MEOL around FINFETs and MOSFETs. This method overcomes the problems of existing methods that either use field-solvers or pre-characterized libraries to extract MEOL patterns, where field-solvers suffer from runtime and capacity problems, whereas pre-characterized libraries suffer from pattern mismatches and insufficient pattern coverage problems. The proposed modeling methodology selects all devices in a certain layout, identifies their MEOL patterns, and represents MEOL patterns using a novel geometry-based representation to be used as inputs to the required machine learning models. The proposed methodology is tested over two process nodes including: 28nm and 7nm. The testing covered devices in several real designs with more than 40M devices. The proposed methodology provided outstanding results as compared to field-solvers with an average error < 0.2%, a standard deviation < 3%, and a speed up of 100X.

As for the **hybrid extraction method**, a novel accuracy-based hybrid parasitic capacitance extraction flow is developed. The proposed hybrid flow divides the chip into windows and extracts the parasitic capacitances of each window using one of three extraction methods: 1) field-solver; 2) rule-based; or 3) novel deep neural-networks based extraction methods. This is done by extracting the parasitic capacitive elements of each pattern (i.e., window) by using the fastest extraction method that meets the user pre-determined accuracy level. The proposed flow uses neural-networks classifiers to determine the capacitance extraction method for each window. On the other hand, a new DNN-based extraction method is developed as an intermediate parasitic extraction method between rule-based method and field-solver method in terms of runtime and accuracy. The proposed hybrid flow is tested over different designs of 7nm and 28nm process nodes. The results showed that the proposed hybrid flow managed to meet the

required accuracy levels (of less than 5% error) with more than 99% accuracy, and with a speed up of 70X as compared to field-solvers

As for the **parasitic-aware routing optimization** methodology, a new routing optimization methodology is developed based on an incremental parasitic extraction and a fast optimization algorithm. The proposed methodology aims to overcome the problems of existing layout routing optimization that include: 1) the use of many circuit simulations to calculate the parasitic bounds; 2) the use of either simple parasitic models, which provide poor accuracy, or a full layout extraction, which consumes a lot of time, in order to extract the parasitic elements of a given; and 3) the lack of a mechanism to analyze the impact of parasitic elements and corresponding geometries on a system's performance. The proposed methodology provides: 1) novel sensitivity circuit models that help circuit designers in analyzing the impact of parasitic elements and corresponding layout geometries on a system's performance; 2) a novel incremental parasitic capacitance extraction methodology that helps in providing a significant speedup in the optimization runtime; and 3) a nonlinear programming algorithm to modify and optimize the problematic routes based on the proposed sensitivity circuit models. The proposed methodology is tested over different ring oscillator designs of 7nm process node and folded cascode differential amplifiers of 65nm process node. The experimental results show the proposed methodology managed to identify and optimize the problematic geometries in critical routes with up to 10% improvements in the performance and a speedup of 3 to 9X as compared to traditional template-based methods.

Chapter 9

Future Work

The future work includes two main topics: 1) extending the parasitic capacitance modeling to cover new technologies, such as 3DICs; and 2) improving the parasitic-aware routing method to incorporate parasitic inductances.

As for the neural network models of the proposed parasitic capacitance modeling methodologies, the impact of changing and increasing the number of neurons and layers on the network performance need to be studied. This would provide more understanding to the impact of changing neural network architecture on the performance of the different parasitic capacitance extraction models.

Moreover, the proposed models cannot predict parasitic capacitances of three-dimensional integrated circuits (3DIC) technologies, such as stacked-die 3DIC and monolithic 3DIC technologies. The 3DIC technologies aims to combine and integrate multiple systems on a single package. In stacked-die 3DIC technologies, multiple silicon wafers (or chips) are stacked vertically and connected together by using a through-silicon-via (TSV). The stacking may have many forms, such as a face to face or a face to back. In such cases, the capacitance coupling interactions among the interconnects across those chips need to be modeled. As for monolithic 3DIC technologies, the device layers and their corresponding devices are fabricated sequentially, and multiple devices with different elevations may exist. In such cases, there are many different metal and device layers that are vertically overlapped, and the parasitic capacitances among them need to be modeled correctly. Eventually, the proposed models need to be extended to support 3DIC technologies.

As for the parasitic-aware routing optimization method, it might be extended to use different machine learning methods, such as re-enforcement learning, in order to improve

the routing optimization process. On the other hand, the proposed methodology only considers the RC parasitic elements. Hence, their models are appropriate for local interconnect at any frequency and global interconnect at a lower frequency. For high frequency global interconnect, inductance and more complex models need to be included. Therefore, the future work aims to extend this work to consider the different inductance effects.

Appendix

The appendix provides the derivations of the sensitivity circuit models in Chapter 7 as follows.

A. Moments Sensitivity to a Parasitic Element

The derivations of moments sensitivity to a parasitic element, in (7.12) and (7.13), are as below:

By differentiating (2.13) with a certain parasitic element (P_i) we get:

for m_0: differentiating ($G\ \underline{m_0} = \underline{b}$) with P_i

$$\frac{\partial}{\partial P_i}\left(G\ \underline{m_0}\right) = \frac{\partial}{\partial P_i}\left(\ \underline{b}\right), \tag{1}$$

Therefore,

$$\frac{\partial G}{\partial P_i}\ \underline{m_0} + G\ \frac{\partial \underline{m_0}}{\partial P_i} = 0. \tag{2}$$

Then,

$$\frac{\partial G}{\partial P_i}\ \underline{m_0} = \ -G\ \frac{\partial \underline{m_0}}{\partial P_i}, \tag{3}$$

Multiplying both sides by G^{-1} , we get:

$$\frac{\partial \underline{m_0}}{\partial P_i} = -G^{-1}\ \frac{\partial G}{\partial P_i}\underline{m_0}, \tag{4}$$

for m_1: differentiating ($G\ \underline{m_1} + C\ \underline{m_0} = 0$) with P_i

$$\frac{\partial}{\partial P_i}\left(G\ \underline{m_1}\right) + \frac{\partial}{\partial P_i}\left(C\ \underline{m_0}\right) = \ 0. \tag{5}$$

Therefore,

$$\frac{\partial G}{\partial P_i}\ \underline{m_1} + G\ \frac{\partial \underline{m_1}}{\partial P_i} + \frac{\partial C}{\partial P_i}\ \underline{m_0} + C\ \frac{\partial \underline{m_0}}{\partial P_i} = \ 0, \tag{6}$$

Eventually,

$$\frac{\partial \underline{m_1}}{\partial P_i} = -G^{-1} \cdot \left(\frac{\partial G}{\partial P_i}\ \underline{m_1} + \frac{dC}{\partial P_i}\ \underline{m_0} + C\ \frac{\partial \underline{m_0}}{\partial P_i}\right). \tag{7}$$

Similarly, for m_2 till m_k, where ($G\ \underline{m_k} + C\ \underline{m_{k-1}} = 0$) :

$$\frac{\partial}{\partial P_i}\left(G\ \underline{m_k}\right) + \frac{\partial}{\partial P_i}\left(C\ \underline{m_{k-1}}\right) = 0.$$

$$\vdots$$

$$\frac{\partial \underline{m_k}}{\partial P_i} = -G^{-1}\cdot\left(\frac{\partial G}{\partial P_i}\ \underline{m_k} + \frac{\partial C}{\partial P_i}\underline{m_{k-1}} + C\frac{\partial \underline{m_{k-1}}}{\partial P_i}\right), \qquad k \geq 1, \tag{8}$$

where $\underline{m_k}$ is an n vector of moments and n is the number of nodes in an RC network. This model represents a general model for moments sensitivity to a certain parasitic element. For a certain target node, the moment sensitivity to a parasitic element (P_i) is given by:

$$\frac{\partial m_0}{\partial P_i} = -G^{-1}\ \frac{\partial G}{\partial P_i}m_0, \quad \text{and} \tag{9}$$

$$\frac{\partial m_k}{\partial P_i} = -G^{-1}\cdot\left(\frac{\partial G}{\partial P_i}\ m_k + \frac{\partial C}{\partial P_i}m_{k-1} + C\frac{\partial m_{k-1}}{\partial P_i}\right), \qquad k \geq 1, \tag{10}$$

where C is the capacitors matrix, G is the admittance matrix, and m_0 to m_k are circuit moments at a given node.

The parasitic element (P_i) in (9) and (10) can be either a resistive or capacitive element. The derivations for both cases are as follows.

1) Moments Sensitivity to a Parasitic Resistive Element

The moment sensitivity to a parasitic resistive element (R_i) is obtained by substituting a parasitic element parameter (P_i) in (9) and (10) with a resistive element (R_i) as below:

$$\frac{\partial m_0}{\partial R_i} = -G^{-1}\ \frac{\partial G}{\partial R_i}m_0, \qquad \text{and} \tag{11}$$

$$\frac{\partial m_k}{\partial R_i} = -G^{-1}\left(\frac{\partial G}{\partial R_i}\ m_k + \frac{\partial C}{\partial R_i}m_{k-1} + C\frac{\partial m_{k-1}}{\partial R_i}\right), \qquad k \geq 1. \tag{12}$$

However, some terms might have special values when they are differentiated with a parasitic resistive element (R_i) as below:

$$\frac{\partial C}{\partial R_i} = 0, \tag{13}$$

because C is the capacitance matrix and differentiating it with a resistive element gives zero. Moreover, $dG/dR_i$is obtained as below:

$$\frac{\partial G}{\partial R_i} = \frac{\partial G}{\partial g_i}\ \frac{\partial g_i}{\partial R_i}, \tag{14}$$

where $g_i = (1/R_i)$. Therefore,

$$\frac{\partial G}{\partial R_i} = \frac{\partial G}{\partial g_i}\ \frac{\partial (1/R_i)}{\partial R_i}, \tag{15}$$

$$\frac{\partial G}{\partial R_i} = -\frac{1}{{R_i}^2}\ \frac{\partial G}{\partial g_i}. \tag{16}$$

As a result, the moments sensitivity to a parasitic resistive element (R_i) is given by:

for m_0:

substitute (16) in (11), we get:

$$\frac{\partial m_0}{\partial R_i} = G^{-1}\ \frac{1}{R^2}\ \frac{\partial G}{\partial g_i} m_0, \tag{17}$$

which represents the moment (m_0) sensitivity to a certain parasitic resistive element at a given node.

for m_k , $k \geq 1$, substitute (13) and (16) in (12), we get:

$$\frac{\partial m_k}{\partial R_i} = -G^{-1}\left(-\ \frac{1}{{R_i}^2}\ \frac{\partial G}{\partial g_i}\ m_k\ + C\frac{\partial m_{k-1}}{\partial R_i}\right), \qquad k \geq 1, \tag{18}$$

which represents the moment (m_k) sensitivity to a certain parasitic resistive element when $k \geq 1$ at a given node.

2) Moments Sensitivity to a Parasitic Capacitive Element

The moment sensitivity to a parasitic capacitive element (Cc_j) is obtained by substituting a parasitic element parameter (P_i) in (9) and (10) with a capacitive element (Cc_j) as below:

$$\frac{\partial m_0}{\partial Cc_j} = -G^{-1}\ \frac{\partial G}{\partial Cc_j} m_0, \qquad \text{and} \tag{19}$$

$$\frac{\partial m_k}{\partial Cc_j} = -G^{-1}\left(\frac{\partial G}{\partial Cc_j}\ m_k\ + \frac{\partial C}{\partial Cc_j} m_{k-1} + C\frac{\partial m_{k-1}}{\partial Cc_j}\right), \qquad k \geq 1. \tag{20}$$

However, some terms might have special values when they are differentiated with a parasitic capacitive element (Cc_j) as below:

$$\frac{\partial G}{\partial Cc_j} = 0, \tag{21}$$

because G is the admittance matrix and differentiating it with a capacitive element gives zero.

As a result, the moments sensitivity to a parasitic capacitive element (Cc_j) is given by:

for m_0, substitute (21) in (19), we get:

$$\frac{\partial m_0}{\partial Cc_j} = 0, \tag{22}$$

which represents the moment (m_0) sensitivity to a certain parasitic capacitive element at a given node.

for m_k , $k \geq 1$, substitute (21) in (20), we get:

$$\frac{\partial m_k}{\partial Cc_j} = -G^{-1}\left(\frac{\partial C}{\partial Cc_j} m_{k-1} + C\frac{\partial m_{k-1}}{\partial Cc_j}\right), k \geq 1 \tag{23}$$

which represents the moment (m_k) sensitivity to a certain parasitic capacitive element when $k \geq 1$ at a given node.

B. Relative Cost Function Sensitivity to a Parasitic Element

The derivations of the relative cost function sensitivity to a parasitic element, in (7.14), are as below:

Assuming two systems, the output response of the first system is given by:

$$S1(s) = m_0 + m_1 s + m_2 s^2 + m_3 s^3 + \cdots, \tag{24}$$

whereas the output response of the second system is given by:

$$S2(s) = m'_0 + m'_1 s + m'_2 s^2 + m'_3 s^3 + \cdots. \tag{25}$$

Therefore, the relative cost function (RCF) between the two systems is given by:

$$\text{RCF} = \sum_{i=0}^{q} \frac{(m_i - m'_i)^2}{{m'_i}^2}, \tag{26}$$

where q represents the required order of circuit moments.

differentiating (26) with a parasitic element (P_i) gives:

$$\frac{\partial \mathrm{RCF}}{\partial P_i} = \frac{\partial}{\partial P_i}\left(\frac{(m_0 - {m'}_0)^2}{{m'}_0{}^2} + \frac{(m_1 - {m'}_1)^2}{{m'}_1{}^2} + \cdots\right). \tag{27}$$

Let

$$\mathrm{RCF_{mk}} = \frac{(m_k - {m'}_k)^2}{{m'}_k{}^2}. \tag{28}$$

Therefore,

$$\frac{\partial \mathrm{RCF}}{\partial P_i} = \frac{\partial}{\partial P_i}\left(\mathrm{RCF_{m0}} + \mathrm{RCF_{m1}} + \cdots\right), \tag{29}$$

Use m_0 to m_k as intermediate variables for differentiation, we get:

$$\frac{\partial \mathrm{RCF}}{\partial P_i} = \frac{\partial \mathrm{RCF_{m0}}}{\partial m_0}\frac{\partial m_0}{\partial P_i} + \frac{\partial \mathrm{RCF_{m1}}}{\partial m_1}\frac{\partial m_1}{\partial P_i} + \cdots. \tag{30}$$

As a result,

$$\frac{\partial \mathrm{RCF}}{\partial P_i} = \sum_{k=0}^{n} \frac{\partial \mathrm{RCF_{mk}}}{\partial m_k}\frac{\partial m_k}{\partial P_i}. \tag{31}$$

This model has two components. The first component is ($\partial RCF_{mk}/\partial m_k$). It is obtained by differentiating (28) with a certain moment (m_k) as below:

$$\frac{\partial \mathrm{RCF_{mk}}}{\partial m_k} = 2\,\frac{(m_k - {m'}_k)}{{m'}_k{}^2}, \tag{32}$$

The second component ($\partial m_k/\partial P_i$) is already obtained in (9) and (10). By substituting (9), (10) and (32) in (31), we get:

$$\begin{aligned}\frac{\partial \mathrm{RCF}}{\partial P_i} = {} & 2\,\frac{(m_0 - {m'}_0)}{{m'}_0{}^2} \\ & \cdot\left(-G^{-1}\,\frac{\partial G}{\partial P_i}m_0\right) + \sum_{k=1}^{n}\left(2\,\frac{(m_k - {m'}_k)}{{m'}_k{}^2}\right. \\ & \left.\cdot\left(-G^{-1}\left(\frac{\partial G}{\partial P_i}m_k + \frac{\partial C}{\partial P_i}m_{k-1} + C\frac{\partial m_{k-1}}{\partial P_i}\right)\right)\right),\end{aligned} \tag{33}$$

which represents the relative cost function (RCF) sensitivity to a certain parasitic element (P_i) at a given node.

C. Delay cost function sensitivity to a parasitic element

The derivations of the delay cost function sensitivity to a parasitic element, in (7.15), are as below:

The delay cost function (DCF) is given by, based on [102]:

$$\mathrm{DCF} = a_1 \cdot m_1 + a_2 \cdot \frac{m_2}{m_1} + a_3 \cdot \frac{m_3}{{m_1}^2} + \cdots + a_q \cdot \frac{m_q}{{m_1}^{(q-1)}}, \tag{34}$$

differentiating (34) with a parasitic element (P_i) gives:

$$\frac{\partial \mathrm{DCF}}{\partial P_i} = \frac{\partial}{\partial P_i}\left(a_1 \cdot m_1 + a_2 \cdot \frac{m_2}{m_1} + a_3 \cdot \frac{m_3}{{m_1}^2} + \cdots + a_q \cdot \frac{m_q}{{m_1}^{(q-1)}}\right). \tag{35}$$

Therefore,

$$\begin{aligned}\frac{\partial \mathrm{DCF}}{\partial P_i} = {} & a_1 \cdot \frac{\partial m_1}{\partial P_i} + \\ & a_2 \cdot \left(\frac{\partial m_2}{\partial P_i}\frac{1}{m_1} + m_2\,(-{m_1}^{-2})\frac{\partial m_1}{\partial P_i}\right) + \cdots + \\ & a_q \cdot \left(\frac{\partial m_q}{\partial P_i}\frac{1}{{m_1}^{k-1}} + m_q\,(-(q-1){m_1}^{-q})\frac{\partial m_1}{\partial P_i}\right).\end{aligned} \tag{36}$$

As a result,

$$\frac{\partial \mathrm{DCF}}{\partial P_i} = a_1 \cdot \frac{\partial m_1}{\partial P_i} + \sum_{k=2}^{q}\left[a_k\left(\frac{\partial m_k}{\partial P_i} \cdot \frac{1}{{m_1}^{k-1}} + m_k \cdot \frac{(1-k)}{{m_1}^{k}} \cdot \frac{\partial m_1}{\partial P_i}\right)\right], \tag{37}$$

which represents the delay cost function sensitivity to a parasitic element (P_i) at a given node.

References

[1] G. Bell, “Growing challenges in nanometer timing analysis,” *EE Times*, Oct. 18, 2004.

[2] G. Bell, “Nanometer scale effects complicate IP characterization -,” *EETimes*, Dec. 19, 2002. Accessed: Jun. 01, 2021. [Online]. Available: https://www.eetimes.com/nanometer-scale-effects-complicate-ip-characterization/

[3] J. H.-C. Chen, T. E. Standaert, E. Alptekin, T. A. Spooner, and V. Paruchuri, “Interconnect performance and scaling strategy at 7 nm node,” in *IEEE International Interconnect Technology Conference*, May 2014, pp. 93–96. doi: 10.1109/IITC.2014.6831843.

[4] M. Bohr, “The new era of scaling in an SoC world,” in *2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, Feb. 2009, pp. 23–28. doi: 10.1109/ISSCC.2009.4977293.

[5] A. Naeemi, C. Pan, A. Ceyhan, R. M. Iraei, V. Kumar, and S. Rakheja, “BEOL scaling limits and next generation technology prospects,” in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, Jun. 2014, pp. 1–6. doi: 10.1145/2593069.2596672.

[6] EDN, “Parasitic extraction must solve advanced node issues,” *EDN*, May 01, 2018. https://www.edn.com/parasitic-extraction-must-solve-advanced-node-issues/ (accessed Jun. 10, 2021).

[7] R. Suaya, R. Escovar, and Q. Salvador, “Modeling and Extraction of Nanometer Scale Interconnects: Challenges and Opportunities,” in *Proceedings of the 23rd Advanced Metallization Conference (AMC)*, Oct. 2006, pp. 1–11.

[8] X. Qi and R. W. Dutton, “Interconnect Parasitic Extraction of Resistance, Capacitance, and Inductance,” in *Interconnect Technology and Design for Gigascale Integration*, J. Davis and J. D. Meindl, Eds. Boston, MA: Springer US, 2003, pp. 67–109. doi: 10.1007/978-1-4615-0461-0_3.

[9] B. Cardoso, R. Martins, N. Lourenço, and N. Horta, “AIDA-PEx: Accurate parasitic extraction for layout-aware analog integrated circuit sizing,” in *2015 11th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, Jun. 2015, pp. 129–132. doi: 10.1109/PRIME.2015.7251351.

[10] C. Venkataiah, K. S. Prasad, and T. J. C. Prasad, “Effect of interconnect parasitic variations on circuit performance parameters,” in *2016 International Conference on Communication and Electronics Systems (ICCES)*, Oct. 2016, pp. 1–4. doi: 10.1109/CESYS.2016.7889958.

[11] W. Yu, M. Song, and M. Yang, “Advancements and Challenges on Parasitic Extraction for Advanced Process Technologies,” in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2021, pp. 841–846.

[12] “FinFETs, 16nm and 14nm nodes, and Parasitic Extraction.” https://www5.cadence.com/2014-July-Newsletter_LP.html (accessed Oct. 12, 2021).

[13] A. Kurokawa, T. Sato, T. Kanamoto, and M. Hashimoto, “Interconnect Modeling: A Physical Design Perspective,” *IEEE Trans. Electron Devices*, vol. 56, no. 9, pp. 1840–1851, Sep. 2009, doi: 10.1109/TED.2009.2026208.

[14] W. Yu and X. Wang, *Advanced Field-Solver Techniques for RC Extraction of Integrated Circuits*. Springer Science & Business, 2014.

[15] A. Zhang *et al.*, “A field-based parasitic capacitance model with 3-D terminal and terminal fringe components,” in *2015 6th Asia Symposium on Quality Electronic Design (ASQED)*, Aug. 2015, pp. 166–170. doi: 10.1109/ACQED.2015.7274028.

[16] A. Zhang *et al.*, “Field-based parasitic capacitance models for 2D and 3D sub-45-nm interconnect,” in *2012 4th Asia Symposium on Quality Electronic Design (ASQED)*, Jul. 2012, pp. 110–116. doi: 10.1109/ACQED.2012.6320485.

[17] W. H. Kao, C.-Y. Lo, M. Basel, and R. Singh, “Parasitic extraction: current state of the art and future trends,” *Proc. IEEE*, vol. 89, no. 5, pp. 729–739, May 2001, doi: 10.1109/5.929651.

[18] W. PINELLO, A. Nieuwoudt, M. DRUT, and B. Qiu, “Determining eco aggressor nets during incremental extraction,” US20160350470A1, Dec. 01, 2016 Accessed: Mar. 07, 2021. [Online]. Available: https://patents.google.com/patent/US20160350470A1/en

[19] “Calibre xRC | Siemens Digital Industries Software.” https://eda.sw.siemens.com/en-US/ic/calibre-design/circuit-verification/xrc (accessed Jul. 19, 2021).

[20] N. Jangkrajarng, L. Zhang, S. Bhattacharya, N. Kohagen, and C.-R. Shi, “Template-Based Parasitic-Aware Optimization and Retargeting of Analog and RF Integrated Circuit Layouts,” in *2006 IEEE/ACM International Conference on Computer Aided Design*, Nov. 2006, pp. 342–348. doi: 10.1109/ICCAD.2006.320056.

[21] A. Domic, “Layout synthesis of MOS digital cells,” in *27th ACM/IEEE Design Automation Conference*, Jun. 1990, pp. 241–245. doi: 10.1109/DAC.1990.114861.

[22] Zheng Liu and Lihong Zhang, “Performance-constrained parasitic-aware retargeting and optimization of analog layouts,” in *2009 Canadian Conference on Electrical and Computer Engineering*, May 2009, pp. 1194–1197. doi: 10.1109/CCECE.2009.5090314.

[23] M. P. Lin, Y. Chang, and C. Hung, “Recent research development and new challenges in analog layout synthesis,” in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2016, pp. 617–622. doi: 10.1109/ASPDAC.2016.7428080.

[24] M. D. Moffitt, “Global routing revisited,” in *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, Nov. 2009, pp. 805–808. doi: 10.1145/1687399.1687549.

[25] N. Gockel, R. Drechsler, and B. Becker, “A multi-layer detailed routing approach based on evolutionary algorithms,” in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC ’97)*, Apr. 1997, pp. 557–562. doi: 10.1109/ICEC.1997.592373.

[26] X. Qi, A. Gyure, Y. Luo, S. C. Lo, M. Shahram, and K. Singhal, “Measurement and characterization of pattern dependent process variations of interconnect resistance, capacitance and inductance in nanometer technologies,” in *Proceedings of the 16th ACM Great Lakes symposium on VLSI - GLSVLSI ’06*, Philadelphia, PA, USA, 2006, p. 14. doi: 10.1145/1127908.1127914.

[27] N. K. Karsilayan, J. Falbo, and D. Petranovic, “Efficient and accurate RIE modeling methodology for BEOL 2.5D parasitic extraction,” in *2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug. 2014, pp. 519–522. doi: 10.1109/MWSCAS.2014.6908466.

[28] N. K.-H. Huang, “Implementation of algorithms to determine the capacitance sensitivity of interconnect parasitics in the Magic VLSI layout tool,” University of British Columbia, 2009. doi: 10.14288/1.0067690.

[29] N. Huang and A. Labun, “Extracting interconnect capacitance sensitivity to linewidth variation,” in *2009 Canadian Conference on Electrical and Computer Engineering*, May 2009, pp. 185–189. doi: 10.1109/CCECE.2009.5090117.

[30] C. Clee, “Parasitic extraction at advanced nodes,” *EDN*, Nov. 15, 2017. https://www.edn.com/electronics-blogs/absolute-eda/4459079/Parasitic-extraction-at-advanced-nodes (accessed May 30, 2021).

[31] Z. Li and W. Shi, “Layout Capacitance Extraction Using Automatic Pre-Characterization and Machine Learning,” in *2020 21st International Symposium on Quality Electronic Design (ISQED)*, Mar. 2020, pp. 457–464. doi: 10.1109/ISQED48828.2020.9136970.

[32] “Calibre xACT 3D | Siemens Digital Industries Software.” https://eda.sw.siemens.com/en-US/ic/calibre-design/circuit-verification/xact-3d/ (accessed Jul. 19, 2021).

[33] "Synopsys StarRC - Golden Signoff Extraction." https://www.synopsys.com/implementation-and-signoff/signoff/starrc.html (accessed Oct. 12, 2021).

[34] L. Lavagno, L. Scheffer, and G. Martin, *EDA for IC Implementation, Circuit Design, and Process Technology*. CRC Press, 2006.

[35] K. J. Kuhn *et al.*, "Process Technology Variation," *IEEE Trans. Electron Devices*, vol. 58, no. 8, pp. 2197–2208, Aug. 2011, doi: 10.1109/TED.2011.2121913.

[36] Y.-S. Song, C.-Y. Chu, J. Jeon, U.-H. Kwon, K.-H. Lee, and S. Kim, "Accurate BEOL statistical modeling methodology with circuit-level multi-layer process variations," in *2017 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, Sep. 2017, pp. 265–268. doi: 10.23919/SISPAD.2017.8085315.

[37] J.-H. Liu, J.-K. Zeng, A.-S. Hong, L. Chen, and C. C. P. Chen, "Process-Variation Statistical Modeling for VLSI Timing Analysis," in *9th International Symposium on Quality Electronic Design (isqed 2008)*, Mar. 2008, pp. 730–733. doi: 10.1109/ISQED.2008.4479828.

[38] V. Mehrotra, "Modeling the effects of systematic process variation on circuit performance," phd, Massachusetts Institute of Technology, USA, 2001.

[39] L. Sun *et al.*, "A Novel Customized RC Tightened Corner Modeling Methodology Using Statistical SPICE Simulation in Advanced FinFET Technology," in *2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, Oct. 2018, pp. 1–4. doi: 10.1109/ICSICT.2018.8564960.

[40] S. Kiamehr *et al.*, "The impact of process variation and stochastic aging in nanoscale VLSI," in *2016 IEEE International Reliability Physics Symposium (IRPS)*, Apr. 2016, p. CR-1-1-CR-1-6. doi: 10.1109/IRPS.2016.7574590.

[41] W. Yu, H. Zhuang, C. Zhang, G. Hu, and Z. Liu, "RWCap: A Floating Random Walk Solver for 3-D Capacitance Extraction of Very-Large-Scale Integration Interconnects," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 32, no. 3, pp. 353–366, Mar. 2013, doi: 10.1109/TCAD.2012.2224346.

[42] W. Yu, "RWCap2: Advanced floating random walk solver for the capacitance extraction of VLSI interconnects," in *2013 IEEE 10th International Conference on ASIC*, Oct. 2013, pp. 1–4. doi: 10.1109/ASICON.2013.6811859.

[43] S. Bhattacharya, N. Jangkrajarng, and C.-R. Shi, "Template-driven parasitic-aware optimization of analog integrated circuit layouts," in *Proceedings. 42nd Design Automation Conference, 2005.*, Jun. 2005, pp. 644–647. doi: 10.1145/1065579.1065748.

[44] L. Zhang, N. Jangkrajarng, S. Bhattacharya, and C.-R. Shi, "Parasitic-Aware Optimization and Retargeting of Analog Layouts: A Symbolic-Template Approach," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 27, no. 5, pp. 791–802, May 2008, doi: 10.1109/TCAD.2008.917594.

[45] J. K. Ousterhout, "Corner Stitching: A Data-Structuring Technique for VLSI Layout Tools," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 3, no. 1, pp. 87–100, Jan. 1984, doi: 10.1109/TCAD.1984.1270061.

[46] L. Xiao, E. F. Y. Young, X. He, and K. P. Pun, "Practical placement and routing techniques for analog circuit designs," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2010, pp. 675–679. doi: 10.1109/ICCAD.2010.5654239.

[47] Y. I. Ismail, "Improved model-order reduction by using spacial information in moments," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 11, no. 5, pp. 900–908, Oct. 2003, doi: 10.1109/TVLSI.2003.817138.

[48] K.-Y. Tsai, W.-J. Hsieh, Y.-C. Lu, B.-S. Chang, S.-W. Chien, and Y.-C. Lu, "A new method to improve accuracy of parasitics extraction considering sub-wavelength lithography effects," in *2010*

15th Asia and South Pacific Design Automation Conference (ASP-DAC), Jan. 2010, pp. 651–656. doi: 10.1109/ASPDAC.2010.5419805.

[49] Weibing Gong, Wenjian Yu, Yongqiang Lü, Qiming Tang, Qiang Zhou, and Yici Cai, "A parasitic extraction method of VLSI interconnects for pre-route timing analysis," in *2010 International Conference on Communications, Circuits and Systems (ICCCAS)*, Jul. 2010, pp. 871–875. doi: 10.1109/ICCCAS.2010.5581853.

[50] S.-C. Wong, G.-Y. Lee, and D.-J. Ma, "Modeling of interconnect capacitance, delay, and crosstalk in VLSI," *IEEE Trans. Semicond. Manuf.*, vol. 13, no. 1, pp. 108–111, Feb. 2000, doi: 10.1109/66.827350.

[51] R. Kasai, T. Kanamoto, M. Imai, A. Kurokawa, and K. Hachiya, "Neural Network-Based 3D IC Interconnect Capacitance Extraction," in *2019 2nd International Conference on Communication Engineering and Technology (ICCET)*, Apr. 2019, pp. 168–172. doi: 10.1109/ICCET.2019.8726919.

[52] C. Zhang and G. Sun, "Fabrication cost analysis for 2D, 2.5D, and 3D IC designs," in *2011 IEEE International 3D Systems Integration Conference (3DIC), 2011 IEEE International*, Jan. 2012, pp. 1–4. doi: 10.1109/3DIC.2012.6263032.

[53] H. Ren, G. F. Kokai, W. J. Turner, and T.-S. Ku, "ParaGraph: Layout Parasitics and Device Parameter Prediction using Graph Neural Networks," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, Jul. 2020, pp. 1–6. doi: 10.1109/DAC18072.2020.9218515.

[54] B. Shook, P. Bhansali, C. Kashyap, C. Amin, and S. Joshi, "MLParest: Machine Learning based Parasitic Estimation for Custom Circuit Design," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, Jul. 2020, pp. 1–6. doi: 10.1109/DAC18072.2020.9218495.

[55] L. Sun *et al.*, "Extraction and modeling of layout-dependent MOSFET gate-to-source/drain fringing capacitance in 40nm technology," *Solid-State Electron.*, vol. 111, pp. 118–122, Sep. 2015, doi: 10.1016/j.sse.2015.05.033.

[56] R. Bose and J. N. Roy, "Analytical Modeling for Parasitics in a Triple Gate MOSFET Device," in *2020 IEEE VLSI DEVICE CIRCUIT AND SYSTEM (VLSI DCS)*, Jul. 2020, pp. 187–192. doi: 10.1109/VLSIDCS47293.2020.9179934.

[57] L. Sun, Z. Li, W. Wong, and Y. Xia, "MEOL Gate-around Parasitic Capacitance Extraction Verification for Design Enablements in Advanced FinFET Technology," in *2018 IEEE International Conference on Electron Devices and Solid State Circuits (EDSSC)*, Jun. 2018, pp. 1–2. doi: 10.1109/EDSSC.2018.8487070.

[58] L.-J. Sun *et al.*, "Extraction of geometry-related interconnect variation based on parasitic capacitance data," *IEEE Electron Device Lett.*, vol. 35, no. 10, pp. 980–982, Oct. 2014, doi: 10.1109/LED.2014.2344173.

[59] O. Moldovan, D. Lederer, B. Iniguez, and J.-P. Raskin, "Finite Element Simulations of Parasitic Capacitances Related to Multiple-Gate Field-Effect Transistors Architectures," in *2008 IEEE Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems*, Jan. 2008, pp. 183–186. doi: 10.1109/SMIC.2008.52.

[60] S. S. Rodriguez, J. C. Tinoco, A. G. Martinez-Lopez, J. Alvarado, and J.-P. Raskin, "Parasitic Gate Capacitance Model for Triple-Gate FinFETs," *IEEE Trans. Electron Devices*, vol. 60, no. 11, pp. 3710–3717, Nov. 2013, doi: 10.1109/TED.2013.2282629.

[61] R. M. Smey, B. Swartz, and P. H. Madden, "Crosstalk reduction in area routing," in *Automation and Test in Europe Conference and Exhibition 2003 Design*, Mar. 2003, pp. 862–867. doi: 10.1109/DATE.2003.1253714.

[62] N. Lourenço, R. Martins, and N. Horta, “Layout-aware sizing of analog ICs using floorplan amp; routing estimates for parasitic extraction,” in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2015, pp. 1156–1161.

[63] A. Bhaduri and R. Vemuri, “Parasitic-Aware and Moment-driven Constraint Satisfying Non-Linear Routing Methodology,” in *2006 49th IEEE International Midwest Symposium on Circuits and Systems*, Aug. 2006, vol. 2, pp. 84–88. doi: 10.1109/MWSCAS.2006.382214.

[64] A. Bhaduri and R. Vemuri, “Parasitic aware routing methodology based on higher order RLCK moment metrics,” in *19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design (VLSID’06)*, Jan. 2006, p. 6 pp.-. doi: 10.1109/VLSID.2006.129.

[65] A. Bhaduri and R. Vemuri, “Inductive and capacitive coupling aware routing methodology driven by a higher order RLCK moment metric,” in *Design, Automation and Test in Europe*, Mar. 2005, pp. 922-923 Vol. 2. doi: 10.1109/DATE.2005.182.

[66] Zheng Liu and L. Zhang, “A performance-constrained template-based layout retargeting algorithm for analog integrated circuits,” in *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2010, pp. 293–298. doi: 10.1109/ASPDAC.2010.5419880.

[67] R. Martins, N. Lourenço, and N. Horta, “LAYGEN II—Automatic Layout Generation of Analog Integrated Circuits,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 32, no. 11, pp. 1641–1654, Nov. 2013, doi: 10.1109/TCAD.2013.2269050.

[68] F. A. Naguib, S. Ahmed, S. Hamed, and M. Dessouky, “Expert Guided Analog Layout Placement and Routing Automation for Deep Nanotechnologies,” in *2020 37th National Radio Science Conference (NRSC)*, Sep. 2020, pp. 240–247. doi: 10.1109/NRSC49500.2020.9235112.

[69] G. Liu, W. Zhu, S. Xu, Z. Zhuang, Y.-C. Chen, and G. Chen, “Efficient VLSI routing algorithm employing novel discrete PSO and multi-stage transformation,” *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–16, 2020, doi: https://doi.org/10.1007/s12652-020-02659-8.

[70] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, “Coupling aware routing,” in *Proceedings of 13th Annual IEEE International ASIC/SOC Conference (Cat. No.00TH8541)*, Sep. 2000, pp. 392–396. doi: 10.1109/ASIC.2000.880770.

[71] T.-Y. Ho, Y.-W. Chang, S.-J. Chen, and D. T. Lee, “A fast crosstalk- and performance-driven multilevel routing system,” in *ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No.03CH37486)*, Nov. 2003, pp. 382–387. doi: 10.1109/ICCAD.2003.159715.

[72] H. Habal and H. Graeb, “Constraint-Based Layout-Driven Sizing of Analog Circuits,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 8, pp. 1089–1102, Aug. 2011, doi: 10.1109/TCAD.2011.2158732.

[73] S. J. Patel and R. A. Thakker, “Parasitic-Aware Automatic Analog CMOS Circuit Design Environment,” in *2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)*, Jan. 2019, pp. 245–250. doi: 10.1109/VLSID.2019.00061.

[74] S. J. Patel and R. A. Thakker, “Parasitic Aware Automatic Analog CMOS Circuit Design Environment Using ABC Algorithm,” in *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*, Jan. 2018, pp. 445–446. doi: 10.1109/VLSID.2018.105.

[75] “Raphael - Technology Computer Aided Design (TCAD) | Synopsys.” https://www.synopsys.com/silicon/tcad/interconnect-simulation/raphael.html (accessed Oct. 12, 2021).

[76] R. Sharma, Nitin, V. K. Sehgal, and D. S. Chauhan, “Closed-form expressions for extraction of capacitances in multilayer VLSI interconnects,” in *TENCON 2008 - 2008 IEEE Region 10 Conference*, Nov. 2008, pp. 1–4. doi: 10.1109/TENCON.2008.4766573.

[77] S.-C. Wong, T. G.-Y. Lee, D.-J. Ma, and C.-J. Chao, “An empirical three-dimensional crossover capacitance model for multilevel interconnect VLSI circuits,” *IEEE Trans. Semicond. Manuf.*, vol. 13, no. 2, pp. 219–227, May 2000, doi: 10.1109/66.843637.

[78] A. Kurokawa, “Second-Order Polynomial Expressions for On-Chip Interconnect Capacitance,” *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E88-A, no. 12, pp. 3453–3462, Dec. 2005, doi: 10.1093/ietfec/e88-a.12.3453.

[79] T. Sakurai and K. Tamaru, “Simple formulas for two- and three-dimensional capacitances,” *IEEE Trans. Electron Devices*, vol. 30, no. 2, pp. 183–185, Feb. 1983, doi: 10.1109/T-ED.1983.21093.

[80] E. Demircan, “Effects of Interconnect Process Variations on Signal Integrity,” in *2006 IEEE International SOC Conference*, Sep. 2006, pp. 281–284. doi: 10.1109/SOCC.2006.283898.

[81] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, Dec. 2015, pp. 1026–1034. doi: 10.1109/ICCV.2015.123.

[82] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Mar. 2010, pp. 249–256. Accessed: Nov. 30, 2021. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html

[83] “TensorFlow,” *TensorFlow*. https://www.tensorflow.org/ (accessed Mar. 25, 2021).

[84] “Welcome to Python.org,” *Python.org*. https://www.python.org/ (accessed Apr. 30, 2022).

[85] F. Wilcoxon, “Individual Comparisons by Ranking Methods,” *Biom. Bull.*, vol. 1, no. 6, pp. 80–83, 1945, doi: 10.2307/3001968.

[86] P. Liashchynskyi and P. Liashchynskyi, “Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS,” *ArXiv191206059 Cs Stat*, Dec. 2019, Accessed: Jul. 18, 2021. [Online]. Available: http://arxiv.org/abs/1912.06059

[87] “Calibre xACT,” *Siemens Digital Industries Software*. https://eda.sw.siemens.com/en-US/ic/calibre-design/circuit-verification/xact/ (accessed Jul. 19, 2021).

[88] J.- Chern, J. Huang, L. Arledge, P.- Li, and P. Yang, “Multilevel metal capacitance models for CAD design synthesis systems,” *IEEE Electron Device Lett.*, vol. 13, no. 1, pp. 32–34, Jan. 1992, doi: 10.1109/55.144942.

[89] S. Tani *et al.*, “Parasitic capacitance modeling for multilevel interconnects,” in *Asia-Pacific Conference on Circuits and Systems*, Oct. 2002, vol. 1, pp. 59–64 vol.1. doi: 10.1109/APCCAS.2002.1114908.

[90] G. Shomalnasab, H. M. Heys, and L. Zhang, “Interconnect Capacitive Modeling in Submicron and Nano Technologies,” 2011.

[91] H. Li *et al.*, “Design space exploration for early identification of yield limiting patterns,” in *Design-Process-Technology Co-optimization for Manufacturability X*, Mar. 2016, vol. 9781, p. 97810W. doi: 10.1117/12.2218540.

[92] H. Geng, H. Yang, B. Yu, X. Li, and X. Zeng, “Sparse VLSI Layout Feature Extraction: A Dictionary Learning Approach,” in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Jul. 2018, pp. 488–493. doi: 10.1109/ISVLSI.2018.00094.

[93] “Module: tf.keras.initializers | TensorFlow Core v2.4.1,” *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/initializers (accessed Feb. 04, 2021).

[94] A. Gu and A. Zakhor, “Optical Proximity Correction With Linear Regression,” *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 2, pp. 263–271, May 2008, doi: 10.1109/TSM.2008.2000283.

[95] D. Z. Pan, Y. Lin, X. Xu, and J. Ou, “Machine learning for mask/wafer hotspot detection and mask synthesis,” in *Photomask Technology*, Monterey, United States, Oct. 2017, p. 10. doi: 10.1117/12.2282943.

[96] H. Zhang, F. Zhu, H. Li, E. F. Y. Young, and B. Yu, “Bilinear Lithography Hotspot Detection,” in *Proceedings of the 2017 ACM on International Symposium on Physical Design*, New York, NY, USA, Mar. 2017, pp. 7–14. doi: 10.1145/3036669.3036673.

[97] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[98] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[99] W. R. S. Jr, R. A. Brodie, and M. W. Beaven, “Method of computing multi-conductor parasitic capacitances for VLSI circuits,” US5452224A, Sep. 19, 1995 Accessed: May 30, 2021. [Online]. Available: https://patents.google.com/patent/US5452224A/en

[100] “New Parasitic Extraction Requirements In Custom Design For The Next Wave Of SoCs,” *Semiconductor Engineering*, Jan. 30, 2020. https://semiengineering.com/new-parasitic-extraction-requirements-in-custom-design-for-the-next-wave-of-socs/ (accessed Jun. 21, 2021).

[101] K. Kalafala *et al.*, “Incremental parasitic extraction for coupled timing and power optimization,” US9858383B2, Jan. 02, 2018 Accessed: Mar. 04, 2021. [Online]. Available: https://patents.google.com/patent/US9858383B2/en

[102] Y. I. Ismail and C. S. Amin, “Computation of signal threshold crossing times directly from higher order moments,” in *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.*, Nov. 2004, pp. 246–253. doi: 10.1109/ICCAD.2004.1382581.

List of Publications

1. M. S. Abouelyazid, S. Hammouda and Y. Ismail, "Accuracy-Based Hybrid Parasitic Capacitance Extraction Using Rule-Based, Neural-Networks, and Field-Solver Methods," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, doi: 10.1109/TCAD.2022.3161199.

2. M. S. Abouelyazid, S. Hammouda and Y. Ismail, "Fast and Accurate Machine Learning Compact Models for Interconnect Parasitic Capacitances Considering Systematic Process Variations," in *IEEE Access*, vol. 10, pp. 7533-7553, 2022, doi: 10.1109/ACCESS.2022.3142330.

3. M. S. Abouelyazid, S. Hammouda and Y. Ismail, "A Fast and Accurate Middle End of Line Parasitic Capacitance Extraction for MOSFET and FinFET Technologies Using Machine Learning," *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2022, pp. 371-376, doi: 10.1109/ASP-DAC52403.2022.9712514.

4. M. S. Abouelyazid, S. Hammouda and Y. Ismail, "Connectivity-Based Machine Learning Compact Models for Interconnect Parasitic Capacitances," *2021 ACM/IEEE 3rd Workshop on Machine Learning for CAD (MLCAD)*, 2021, pp. 1-6, doi: 10.1109/MLCAD52597.2021.9531300.

Submitted:

1. “Parasitic-Aware Layout Analysis and Routing Optimization Methodology” to IEEE ACCESS