A Machine Learning Based Parasitic Extraction Tool

Geraldo Pradipta, Vidya A. Chhabria, and Sachin S. Sapatnekar
University of Minnesota, Minneapolis, MN 55455, USA.

***Abstract*— In this work, we develop a machine learning-based parasitic extractor that takes a routed design in DEF and generates parasitics in SPEF. The software builds regression models that capture the behavior of resistance, capacitance to ground, coupling, crossover, and crossunder capacitance of a net. The characterization of these models is a *one-time* cost to extract per-unit parasitics of the BEOL stack for a given technology. The trained regression models can then be used to rapidly estimate all the parasitic information of a net in the design. Once characterized for a given technology, our tool eliminates the dependency on non-human-readable technology files (e.g., QRC format) provided by the foundry. Our parasitic extraction framework, can be easily used by any standard file-based design flow, since it reads routed design's DEF and generates SPEF. Eventually, this software will be closely integrated with various design stages of the OpenROAD project.**

I. INTRODUCTION

With technology scaling, interconnect parasitics have become dominant in influencing performance of VLSI circuits. For precise timing and power analysis, a fast and accurate parasitic estimation is necessary to aid design closure. Our work [2], builds regression models that accurately captures the behaviour of on-chip resistance (R), capacitance to ground (C), and coupling capacitance (C_c) of a net for a specific technology.

Our parasitic extraction (PEX) engine, can be used as a post-layout parasitic analysis tool in any standard file based RTL-GDS flow [3]. In its current form, it takes as an input, a routed design in Design Exchange Format (DEF), uses the calibrated regression models, and outputs parasitics in a Standard Parasitic Exchange Format (SPEF) [7] file. The generated SPEF file provides R, C, and C_c information of all nets in the design, which can be fed into different sign-off engines to help enable accurate analysis. Eventually, we aim to integrate this software with various design stages in the OpenROAD flow [1], [3], with the ability to handle incremental updates, to provide fast parasitic estimates and help enable a 24 hour turn-around-time flow.

One of the required inputs to parasitic extraction tools are the RC technology files (e.g., in the QRC format) or captables. These files contain parasitic characterization of the technology elements. Newer technology design kits come with encrypted technology files. To enable easy PEX within the inner loop of physical design optimizations, we create regression models to compute the parasitics of on-chip interconnects. The coefficients of these models can then be used to estimate the parasitics of any net, given its length and neighborhood information, very quickly for a given technology and RC corner.

Our work comprises of two flows as shown in Fig. 1.

(a) *Calibration flow* (Fig. 1(a)): a one-time process, for a specific technology, that extracts the coefficients R, C and C_c of various metal layers and via in the BEOL stack in the form of regression models and store the coefficients in a configuration file.

(b) *Inference flow* (Fig. 1(b)): uses the constructed regression models to estimate parasitics on a given routed design, i.e., given a routed DEF, our tool uses the extracted coefficients to write SPEF.

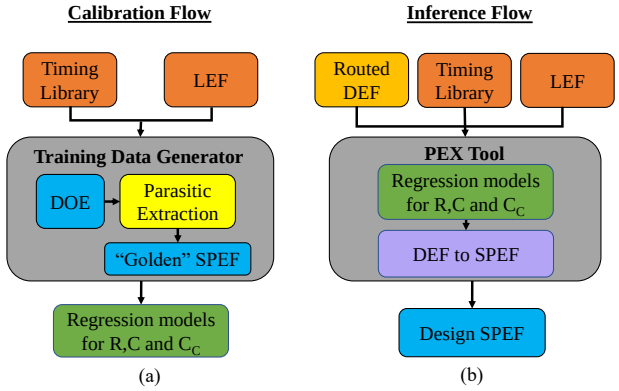

Fig. 1. PEX Framework: (a) calibration flow extracts parasitics from the DOE and trains regression models, and (b) inference which flow uses the trained model to generate SPEF.

II. CALIBRATION FLOW

The calibration flow is a one-time process that is implemented for a specific technology. The goal of the calibration flow is to build regression model for capturing the behavior of these parameters:

- R and C based on per unit length
- C_c based on spacing and overlap between object wire and its same layer neighbor
- Cross-under and cross-over coupling capacitance based on number of neighbors of the object wire, spacing between object wire and its neighbors, and the spacing between the targeted crossing wire with its neighbor.

A. Design of Experiments

In the current implementation, the design of experiments (DOE) assumes that width of the all wires in the design has a fixed value of the minimum width for each metal layer.

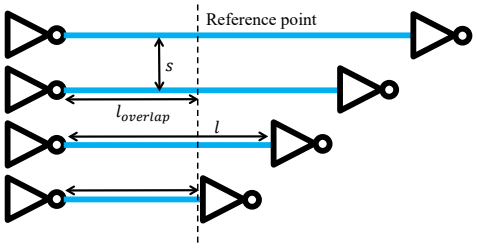

Fig. 2. DOE: wirelength l is varied to extract data points for per unit R and C calibration while $l_overlap$ and s are varied to extract data points for C_c separately for each metal layer.

Resistance and Capacitance: A simple schematic, in Fig. 2 shows a two-pin net that uses inverter cells for both the load and the driver, which forms the basis of our experiments for per-unit R and C extraction. Multiple data points which comprise of length and its corresponding R and C values are extracted by varying the wirelength, l. This experiment is repeated for all layers within the BEOL stack. To remove the contribution of the R and C within the cell, we subtract the parasitics associated with the smallest test structure. Thus, all data extracted from the DOE corresponds to parasitics of wires beyond the smallest test structure, as shown in Fig. 2.
Via resistance and capacitance: The via resistances are extracted using the same training data set as the per unit R and C, which comprises of information across multiple layers. The calibrated per unit R and C is used the in equation below to estimate via R and C from the previously extracted training set:

$$RC_{via,i} = \tfrac{1}{2}RC_{total} - \tfrac{1}{2}\sum_{x\in M} RC_{unit,x}\cdot l_x - \sum_{j=1}^{i-2} RC_{via,j} \quad (1)$$

where $RC_{via,i}$ is the via resistance or capacitance that connects metal layer i and $i+1$, RC_{total} is the total interconnect parasitic, M is the a set of metal layers used in the net, $RC_{unit,x}$ is the per unit R or C value for metal layer x, l_x is the wirelength for metal layer x, and $RC_{via,j}$ is the R or C of the via j.
Coupling capacitance: this is the capacitance between two metal stripes on the same layer. and it depends on two parameters (i) the distance between the two wires, s, and (ii) their overlap length, $l_{overlap}$. We extract several data points for coupling capacitance between metal stripes within the same layer by varying wire spacing and their overlap lengths. Similarly, the experiment is repeated across all metal layers in the BEOL stack.
Crossunder and crossover capacitance: The crossover capacitance is formed between any three adjacent layers of the BEOL stack. Consider a three level set of wire crossings, as shown in Fig. 3, where the second-level metal lines, cross the first-level, and the third-level metal lines, then based on the work in [6], the crossover/crossunder capacitance depends on two parameters: (i) spacing between the object wire and its neighbors in the same layer, s_1 and s_2, and (ii) spacing of the crossing wire to its closest neighbor, s_u and s_t.

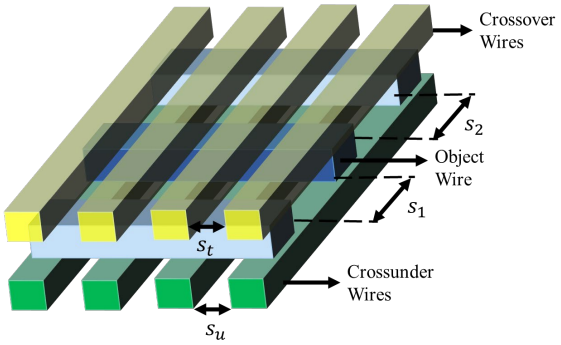

Fig. 3. DOE for crossunder and crossover coupling capacitance: Varying number of neighbors of the object wire, spacing with its neighbors and spacing of targeted crossing wire with its closes neighbors.

III. Regression Models and Configuration File

A. Regression Models for Resistance and Capacitance

We observe that the resistance R and C on each metal layer can be modeled by a linear equation of the form:

$$RC_{wire} = a\cdot l_{wire} + b \quad (2)$$

where l_{wire} represents the length of the wire and RC_{wire} is the R or C of the wire.

The via resistance is independent of length and neighborhood information, but depends on the number of via cuts. The via resistance of a k-cut via is given by the resistance of a 1-cut via divided by k.

The coupling capacitance has a more complex equation since it is also dependent on wire spacing. The coupling capacitance follows the equation of a parallel plate capacitor and varies inversely with the wire spacing. Thus, for two wires run parallelly for a length $l_{overlap}$ and have a spacing of s, the coupling capacitance, C_c, can be modeled as:

$$C_c = c/s + d\cdot l_{overlap} + e\cdot l_{overlap}/s + f \quad (3)$$

where c, d, e, and f are characterization constants and $s \in \{s_1, s_2, s_t, s_u\}$.

Our regression model fits coefficients of s_t, s_u, s_1, and s_2 for each neighboring wires, to estimate the crossunder and crossover coupling capacitance based on the work in [6].

B. Configuration File

We employ a configuration file that stores the coefficients of the regression model, to replace parasitic technology files (e.g., QRC tech file or captable) in the design methodology. The format of the configuration file is depicted in Fig. 4. The variable n in the configuration file represents the total of metal layers for a specific technology, "xxx" represents the coefficient values.

IV. Inference Flow

Fig. 1(b) demostrates the inference flow that can be deployed into a standard design methodology (e.g., [3]). The inference flow takes four inputs: (i) the configuration file generated from the technology-specific calibration flow and (ii) the

```
RESISTANCE
metal_layer      a      b
1               xxx    xxx
2               xxx    xxx
.                .      .
.                .      .
n               xxx    xxx

CAPACITANCE TO GROUND
metal_layer      a      b
1               xxx    xxx
2               xxx    xxx
.                .      .
.                .      .
n               xxx    xxx

COUPLING CAPACITANCE
metal_layer      c      d      e      f
1               xxx    xxx    xxx    xxx
2               xxx    xxx    xxx    xxx
.                .      .      .      .
.                .      .      .      .
n               xxx    xxx    xxx    xxx

VIA RESISTANCE
metal_layer    1_CUT
1               xxx
2               xxx
.                .
.                .
n-1             xxx

VIA CAPACITANCE
metal_layer    1_CUT    2_CUT  3_CUT  4_CUT
1               xxx      xxx    xxx    xxx
2               xxx      xxx    xxx    xxx
.                .        .   .        .
.                .        .      .      .
n-1             xxx      xxx    xxx    xxx
```

Fig. 4. The configuration file format.

routed design in DEF format [4] (iii) LEF (iv) timing library. The advantage of using the characterized configuration file for a given technology, as described in Section III-B, is that we can rapidly predict the parasitics for a given net in any technology-specific design. This is enabled by building standard file readers and writers which process the four inputs and generate parasitics for every net in the design in a SPEF [7]. The SPEF can be fed to any timing analyzer (e.g., [5]) to assess design performance. For each net in the design, we run the parasitic analyzer that processes information of a particular net and its neighborhood to obtain the required parameters for the regression model. Then, the tool calculates the resistances and total capacitance of every net in the design.

V. Results

Our experiments are carried out using Python 3.6 and Cadence Innovus RC extractor as the golden extractor [9]. The regression models are built using TensorFlow [8]. We have several test nets represented in a DEF file that we use as the input to the inference flow. The outputs of the inference flow (SPEF) are then compared with the golden outputs generated by Cadence Innovus extractor.

Figure 5 shows the "goodness of fit" plot for the constructed regression models. The figure compares the normalized predicted and actual (commercially extracted) parasitic (R and C) values from the test nets. The normalized root mean square errors of the two plots are 0.016 and 0.021 respectively. This shows that our regression models fit the data well.

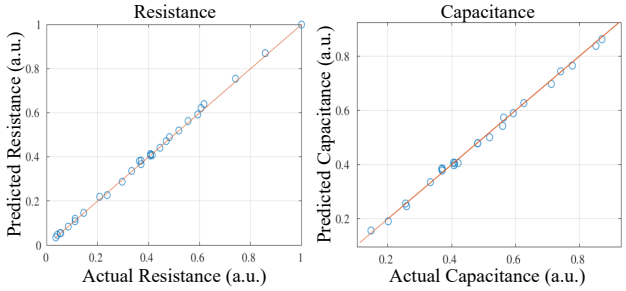

Fig. 5. The predicted versus the actual resistance and capacitance values.

VI. Future Directions

In its current form, while the tool builds accurate regression models for resistance and capacitance, and fits into a file-based RTL-GDS flow, the software relies on commercial tools for the *one-time* calibration flow. The models can be used within fast optimization iterations of a timer and support incremental parasitic updates during each stage design flow. We aim to integrate closely with every stage of an end-to-end hardware design methodology as a part of the OpenROAD [3] design flow. Continuous efforts are being put into addressing the scalability issues of the routed DEF to SPEF converter and its ability to handle a wide variety of testcases.

References

[1] T. Ajayi, *et al.* "Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project." in *Proc. DAC*, pp. 76.1–76.4, 2019.

[2] OpenROAD-PEX, https://github.com/The-OpenROAD-Project/def/tree/master/def

[3] The OpenROAD Project, https://theopenroadproject.org

[4] LEF/DEF reference 5.8, http://www.si2.org/openeda.si2.org/projects/lefdefnew

[5] OpenSTA, https://github.com/abk-openroad/OpenSTA

[6] J. Cong, *et al.* "Analysis and justification of a simple, practical 2 1/2-D capacitance extraction methodology," in *Proc. DAC*, pp. 627–632, 1997.

[7] IEEE Standard for Integrated Circuit (IC) Delay and Power Calculation System, https://standards.ieee.org/project/1481.html

[8] TensorFlow framework, https://www.tensorflow.org/

[9] Cadence Innovus, https://www.cadence.com/content/cadence-www/global/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html