A Parasitic Extraction Method of VLSI Interconnects for Pre-Route Timing Analysis*

Weibing Gong*†, Wenjian Yu†, Yongqiang Lü†, Qiming Tang†, Qiang Zhou†, Yici Cai†
*School of Mathematics and Statistics, Lanzhou University, Lanzhou, China
E-mail: gongwb@yahoo.cn
†Department of Computer Science and Technology, Tsinghua University, Beijing, China
E-mail: {yu-wj, luyq}@tsinghua.edu.cn, venustom85@gmail.com

***Abstract*— For faster timing closure, a parasitic extraction method is developed for the pre-route VLSI design. This method generates virtual route and estimates congestion using the placement information of standard cells, and then extract the interconnect parasitics with the pattern-library method. The techniques of generating parasitic RC tree according to the improved FLUTE algorithm, and capacitance extraction of route segment considering congestion are presented. Experiments are carried out on industrial design cases, whose results show that the proposed method has high computational speed and comparable accuracy as commercial tool.**

I. INTRODUCTION

Currently with the technology scaling, the parasitic effects of the interconnects have become dominant influencing the performance of VLSI circuits. For the timing verification with high precision, fast and accurate parasitic extraction of interconnects is required.

For high-performance circuit design, it is important to make timing verification at the early stage of physical design. This can ensure a faster design closure and reduce the time to market. So, after the placement of cells, we can perform a static timing analysis (STA) to find out the signal path violating the timing constraints. This step is very crucial and provides guidelines for the sequent physical synthesis. However, at the pre-route design stage, the routing topology does not exist. Some kind of routing estimation should be applied to enable the parasitic extraction. Recently, a kind of rectilinear Steiner minimal tree (RSMT) algorithm called FLUTE [4] was proposed to estimate net wire length at the pre-route stage. An improved FLUTE algorithm was then proposed, which has faster speed and more flexibility [2]. This method can be employed for routing estimation at the pre-route stage.

Parasitic extraction has been widely investigated during the past decades. Most existing works are about the field solver algorithm [8] or layout parasitic extraction (LPE) method [9]. The former employs 2D or 3D numerical algorithm to simulate electrostatic field, and thus has low computational speed and capacity. The latter is aimed at the post-route circuit layout, and performs full-chip capacitance extraction with the pattern-library method. Although the pre-route timing analysis requests good parasitic model, the capacitance extraction method for this stage is rarely seen in literature, and not exposed in public domain.

In this paper, for effective timing analysis at the pre-route stage, a parasitic extraction method of VLSI interconnects has been developed. This method takes in the locations of cell pins, and constructs the virtual routes for signal nets with the improved FLUTE algorithm. A congestion map is generated according to the cell placement, which estimates the demanded route tracks for each routing grid, and provides the coupling information among nets. Based on the virtual route and the congestion information, capacitance and resistance are extracted for the segments in the virtual routes. Finally, they form a parasitic network for delay calculation. The presented method is compared with a commercial tool with the same capability. The comparison reveals that results of our method have good correlation with those from the commercial tool. And our method has high computational speed, which extracts several thousands nets per second.

The rest of this paper is organized as follow. Section II presents the background of parasitic extraction and pre-route timing analysis. In Section III, the techniques in the parasitic extraction method for pre-route stage are discussed. The experimental results are given in Section IV. And finally, Section V concludes this paper.

II. BACKGROUND

A. Parasitic Extraction and Timing Analysis

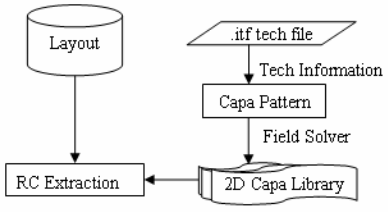

Fig. 1. RC Extraction with the pattern-library method.

Fig. 1 shows a general flowchart of parasitic extraction with the pattern-library method. The .itf technology file

*The work is partially supported by China National Major Science and Technology Special Project 2008ZX01035-001-4 during the 11th five-year plan period, and the Tsinghua National Laboratory for Information Science and Technology (TNList) Cross-discipline Foundation. W. Yu is the corresponding author.

includes the description of metal layers and dielectric layers. The height of each layer and the minimal width and spacing of each metal layer are defined. This file also defines the electrical properties like dielectric permittivity, metal square resistance, etc. For each process technology, field solver algorithms are employed to build up the capacitance library for some predefined interconnect patterns. This procedure runs only once for a technology. Then for each designed circuit, the capacitance library can be utilized for layout parasitic extraction.

The parasitic extraction is the basis of timing analysis. The fundamental task of timing analysis is to calculate the delay of signal net and circuit cell. With parasitic extraction the interconnects are transformed into RC circuits, from which the circuit delay can be derived. In physical design stage, the extraction and timing analysis usually includes the following steps:

1) According to .itf technology file, generate capacitance pattern files. Employ field solver to extract capacitances for the patterns, and with them establish the capacitance library.
2) Cut the actual design layout into pieces. Then for each piece, extract its capacitance and resistance with the capacitance library. The interpolation or formula fitting is used to lookup the library.
3) Construct the parasitic circuit with extraction results for each net. And then use some algorithm, like Elmore algorithm, to calculate time delay from net driver to receiver.
4) With the parasitic circuit, the driver capacitance seen at the cell output can be obtained. Then, with the driver capacitance and input signal transition time, the cell delay can be calculated by looking up the delay table stored in .lib file.

Parasitic extraction for pre-route stage has difficulty due to the lack of route information after the placement design step. Since the parasitics, especially capacitance are highly dependent on interconnect geometry, we can only estimate the RC values at this stage. Based on this analysis, we use the improved FLUTE [2] algorithm to construct the virtual route, and choose techniques of extraction and delay calculation with fair accuracy and high efficiency.

B. Interconnect Delay Calculation

Nowadays there are many algorithms used to calculate time delay of RC interconnect model, for example, Elmore algorithm [3], AWE algorithm [7], and etc. Without route on VLSI circuits at the stage of placement, it is not necessary to spend too much computing time for evaluating the time delay. Elmore RC model (see Fig. 2) is adopted here. Each wire segment is represented by a π-type equivalent circuit.

The Elmore algorithm is adopted to calculate the net delay, which has the following formula [3]:

$$m_{st} = \sum_{(u,v)\in P(s,t)} r_{uv} c(T_v), \tag{1}$$

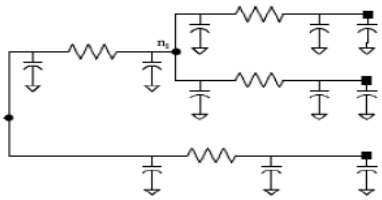

Fig. 2. π-type RC tree modeling the interconnect parasitics.

where s,t,u,v denotes nodes of RC tree, and m_{st} is the time delay from the s node to t node. (u,v) denotes a segment of path $P(s,t)$ from s to t. r_{uv} denotes parasitic resistance of segment (u,v). $c(T_v)$ denotes total capacitance of subtree whose root is v. If v is leaf node of RC tree, $c(T_v)$ is just the capacitance of node v, c_v. Otherwise, it is the sum of the capacitance of node v and the total subtree capacitances of v's child nodes $DD(v)$:

$$c(T_v) = \sum_{k\in DD(v)} c(T_k) + c_v. \tag{2}$$

To calculate the interconnect delay, we firstly traverse path from t to calculate the total capacitance of subtree with (2). Then, the delay between node s and node t is obtained with (1). The task of parasitic extraction is to calculate the R and C for each route segment and convert them to the circuit elements in the RC tree model.

III. Parasitic Extraction Based on Virtual Route and Congestion Estimation

A. Parasitic RC Tree Generation with Virtual Route

Most signal nets in VLSI circuits have a low degree (the number of pins in the net). So in VLSI applications, rather than having a low runtime complexity, it is more important for RSMT algorithms to be simple so that it can be efficient for small nets. Improved FLUTE [2] is a very fast and accurate RSMT algorithm, which extends the technique of wire length estimation in FLUTE [4] and construct the RSMT. With this algorithm, low-degree nets (up to degree 9) are handled optimally and efficiently by a lookup table approach, and high-degree nets are recursively broken down until lookup table can be used. The runtime complexity of the improved FLUTE is $O(nlogn)$ for a degree n net.

In VLSI circuits, usually each net has only one signal input pin, and may have multiple signal output pins. Fig. 3 shows a circuit layout at cell level. Small rectangles on the both sides of the design denote pins connecting outside of this circuit. Small ellipses inside the design denote pins connecting the inside cells. We use *Driver* to denote the input pin of a net and *Receiver* to denote the output pin. Below we illustrate how to generate the parasitic RC tree with virtual route, and take Fig. 3 as an example.

For each net, which has been logically defined, its driver and receiver has been specified location after placement. With this information we can firstly perform the improved FLUTE algorithm to get the virtual route of the net. There

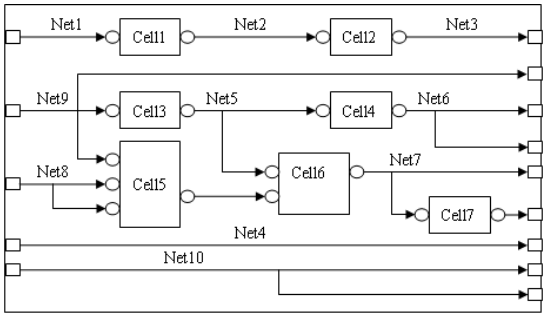

Fig. 3. Cell Level Layout of a Circuit Design

are two kinds of net, two-terminal nets and multi-terminal (more than two) nets. We handle them separately.

1. For two-terminal net, the virtual route is simply the half of bounding box, including one x-direction segment and one y-direction segment. For each segment, the capacitance and resistance are extracted according to the congestion information. Its details will be given in the following subsections. To form the parasitic network like that in Fig. 2, there are two different situations further:

1) The Receiver is the pin of an inside cell, for example, Net1 and Net2 in Fig. 3. Except for the interconnect R and C, we also need the pin capacitance of the inside cell. For the Net1 in Fig. 3, its the input capacitance of Cell1.

2) The Receiver is the pin connecting the outside circuit, for example, Net3 and Net4 in Fig. 3. Except for the interconnect R and C, we also need the outside load capacitance of circuit. This is usually specified in the .sdc file.

2. For multi-terminal net, the virtual route is generated with the improved FLUTE algorithm. The output of this algorithm is a tree structure where each node stands for a layout position. Fig. 4 shows such a Steiner tree structure. With the tree data structure, from each node one can traverse and reach the root of the tree (In Fig. 4 we traverse along the blue arrowheads). The algorithm of generating interconnect parasitics for the multi-terminal net is given in Table I.

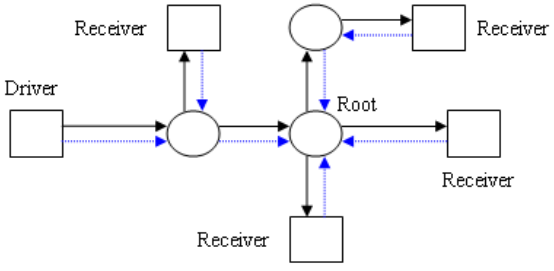

Fig. 4. Steiner RC Tree

As discussed for two-terminal net, we insert different capacitance at the Receiver side of multi-terminal net as well, according to the type of Receiver. It is also noted that for each Receiver, an Elmore delay will be calculated.

The input capacitance of a inside cell can be obtained from the .lib file, while the outside load capacitance is usually

TABLE I

THE ALGORITHM OF GENERATING INTERCONNECT PARASITICS FOR THE MULTI-TERMINAL NET

Parasitic Extraction from Steiner Tree
1. Create Steiner Tree through Improve FLUTE with coordinates of Pins;
2. currentNode=driverNode;
3. While(currentNode != rootOfSteiner)
4. nextNode =getNext(currentNode);
5. Calculate parasitic parameters between current and next;
6. currentNode=nextNode;
7. Endwhile
8. For(all Reveivers)
9. currentNode=reveiverNode;
10. While(currentNode != rootOfSteiner)
11. If(NodeOfCurrent != NULL) break;
12. nextNode =getNext(currentNode);
13. Calculate parasitic parameters between current and next;
14. currentNode=nextNode;
15. Endwhile
16. Endfor

specified in the .sdc file. To calculate the interconnect capacitance and resistance, the pattern-library method is employed, which is introduced in the following two subsections.

B. Capacitance Pre-Characterization with 2-D Patterns

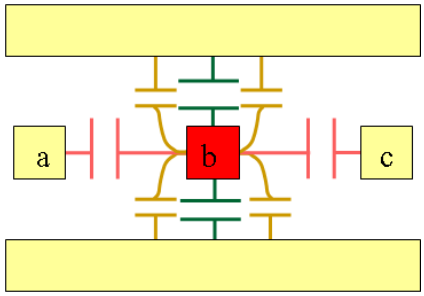

Fig. 5. 2D interconnect pattern for capacitance extraction

At the pre-route stage, 2D interconnect pattern is enough to estimate capacitance with pattern-library method. In Fig. 5, there is the cross section of an interconnect structure. The interconnect under consideration is *b*, which has two neighbor wires *a* and *c*. The interconnects at upper layer and lower layer are assumed to form two conductor plane. Therefore, the whole structure has 2D character, and the capacitance per unit length can be extracted with 2D field solver. The wire width and spacing at the current layer are varying parameters. And the positions of current layer, upper layer and lower layer are also changeable. So, Fig. 5 stands for many patterns with different parameter combinations. The pre-characterization is to solve these patterns and store the results in a capacitance table.

In this paper, field solver Raphael RC2 [10] is applied to extract capacitance of these 2D patterns. RC2 is a general-purpose, two-dimensional program for solving Poisson's equation. It is based on the finite-difference method with an automatically adjustable rectangular mesh. For the process

technology with 6 metal layers, there are 84 combinations of the three layers. For each combination, we set 16 values of wire spacing and 5 values of wire width.

C. Parasitic Extraction of Route Segment with Congestion Estimation

1. The method of capacitance calculation

In the 2D capacitance table, a combination of wire width spacing value correspond to a capacitance per unit length. While the minimal wire width of current metal layer is assumed to be default value, wire spacing should be calculated considering the congestion of placement. A congestion map is usually employed to help optimize the placement design. It is a two-dimensional grid imposed on the whole chip. For each grid cell, the supply resource of routing track and demanded routing track are estimated with algorithms like that in [5]. With the demand count of routing cell, we can calculate the wire spacing to capture the congestion of routing tracks. For current routing segment, the position of its center point can be used to obtain the grid index of congestion map.

In following formulas, *Demand* denotes horizontal or vertical demand counts for routes. *gHeight* and *gWidth* respectively denote height and width of grid cell of chip. *TotalHLayers* denotes the number of metal layers that wires goes in horizontal direction, and *TotalVLayers* denotes the number of metal layers that wires goes in vertical direction. *minWidth* denotes minimal wire width of the metal layer. *spaceH* and *spaceV* respectively denotes wire spacing in the horizontal and vertical direction. *space* and *width* respectively denote wire width and wire spacing of metal layer need to be calculated.

If $Demand = 0$,

$$spaceH = gHeight/2 \tag{3}$$

$$spaceV = gWidth/2 \tag{4}$$

else,

$$spaceH = \frac{TotalHLayers \times gHeight}{Demand} - minWidth \tag{5}$$

$$spaceV = \frac{TotalVLayers \times gWidth}{Demand} - minWidth \tag{6}$$

Therefore, in horizontal and vertical metal layer, $width = minWidth$, $space$ is $spaceH$ and $spaceV$, respectively. Let cap be the bilinear interpolation function with respect to $width$ and $space$, and denote: $cap = cap(w,s)$. In 2D pattern capacitance database, through obtaining wire width $w1$, $w2$ most close to $width$ and wire space $s1$, $s2$ most close to $space$, it is easy to get unit length capacitance $cap11 = cap(w1,s1)$, $cap12 = cap(w1,s2)$, $cap21 = cap(w2,s1)$ and $cap22 = cap(w2,s2)$.

Linear interpolation is carried out in $width$ direction and obtain:

$$cap(w,s1) = \frac{w2-w}{w2-w1}cap11 + \frac{w-w1}{w2-w1}cap21 \tag{7}$$

$$cap(w,s2) = \frac{w2-w}{w2-w1}cap12 + \frac{w-w1}{w2-w1}cap22 \tag{8}$$

Then Linear interpolation is carried out in $space$ direction and obtain:

$$cap(w,s) = \frac{s2-s}{s2-s1}cap(w,s1) + \frac{s-s1}{s2-s1}cap(w,s2) \tag{9}$$

The formula (9) is unrelated to interpolation's sequence to $width$ and $space$. The metal layer's unit length capacitance $cap = cap(width, space)$ can be obtained. $caph$, $capv$ denote respectively unit length capacitance in horizontal and vertical direction. If location of two nodes in nets is respectively $(x1,y1)$ and $(x2,y2)$, wire segment capacitance between two nodes is :

$$capaEdge = caph \times |x1-x2| + capv \times |y1-y2| \tag{10}$$

2. The method of resistance calculation

Formula $R = \rho L/S$ is used to calculate wire resistance. ρ is the resistivity, L is the wire length, and S is the cross-section area. In the .itf technology file there is square resistance (R_{SQ}) information for each metal layer. If W denotes wire width, H denotes wire thickness,

$$R = \rho\frac{L}{S} = \rho\frac{L}{WH} = R_{SQ}\frac{L}{W} \tag{11}$$

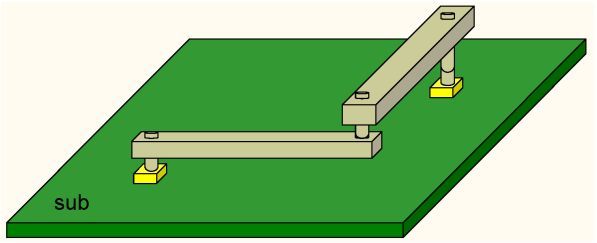

Fig. 6. Multi-layer wire connection with vias

The resistance of via is also considered. The resistance of each via for different layer is specified in the .itf file. As shown in Fig. 6, we consider the multi-layer wire connection with vias, and assign via resistances to the route segments. This contributes to complete RC values for the parasitic circuit network.

IV. Experimental Results

Our extraction method is tested for several design cases. These designs are based on a $0.18\mu m$ process technology. The method is implemented with OpenAccess (OA) [1]. OA is an object-oriented, open-source database. It is exceptionally useful in complex processing accompanied by data swaps. Our program runs on a server with 8*Intel(R) Xeon(TM) CPU 3.00GHz under CentOS release 4.4 (Final).

For each design case with placement information, our program extract all interconnect nets, and generate the RC trees. We first run our program for a set of ISCAS89 designs, the computational time is listed in Table II. Then, four industrial cases are tested, whose results are shown in Table III. In both tables, we show the CPU time of our method varies with the number of nets extracted. The results show that our method has high speed, which extracts several thousands nets per second.

TABLE II

THE COMPUTATIONAL TIME FOR ISCAS89 CASES (IN SECOND)

Design Name	Number of Nets	Time
s208_1_bench	71	0.29
s386_bench	160	0.47
s641_bench	261	0.33
s820_bench	357	0.53
s1494_bench	630	0.69
s9234_1_bench	1351	1.01
s13207_bench	2712	1.70
s15850_bench	4580	2.40
s35932_bench	11623	3.50
s38417_bench	14791	5.57

TABLE III

THE COMPUTATIONAL TIME FOR FOUR INDUSTRIAL CASES (IN SECOND)

Design Name	Number of Nets	Time
SBOX	168	0.46
MMC	9568	4.90
CFHC	12478	6.13
ORCA	23029	12.25

To evaluate the accuracy of our extraction method, we compare it with a commercial tool, Astro of Synopsys Inc. Astro is also capable of parasitic extraction for pre-route stage. It employs a method based on virtual route, and has good result correlation with post-route extraction. For one industrial design case, we show the capacitance results of our method in Table IV and V, and the resistance results of our method in Table VI. Table IV includes the capacitance results for several single nets, with two terminal and multiple terminals. Table V includes the capacitance results obtained while analyzing the delay of one signal path. Table VI includes the resistance results composed of the resistance of wire and via for several single nets. From Table IV, Table V and Table VI, we can see that our results has good correlation with those of Astro. This means that the presented parasitic extraction method has good accuracy.

V. CONCLUSIONS

This paper presents a parasitic extraction method of VLSI interconnects for pre-route stage, which is tested with industrial design cases. Experimental results show this method has high running speed, and reasonable accuracy. In the future work, we will complete the STA of the whole circuit and make more testing of our method. And, the equivalent capacitance method may be investigated for accurate calculation of cell delay.

TABLE IV

THE CAPACITANCE OF SEVERAL SINGLE NETS (IN pF AND μm)

Master/Net	Net Length	Our Method	Astro
U_MMC_INT_IF/cmd_sent_ff1	3.345	0.0039	0.0041
U_MMC_INT_IF/cmd_crcfail_ff1	8.745	0.0048	0.0046
U_MMC_INT_IF/cmd_timeout_ff1	13.165	0.0056	0.0051
U_MMC_FIFO_TX/rd_grad_ff1[0]	24.215	0.0073	0.0064
U_MMC_INT_IF/cmd_respend_ff1	84.565	0.0173	0.0127
U_MMC_CLOCK/resume_ff1	21.465	0.0069	0.0062
U_MMC_CMD/cmd_in_ff2	21.195	0.0069	0.0062
U_MMC_FIFO_TX/wr_grad_ff1[4]	53.385	0.0111	0.0096

TABLE V

THE INTERCONNECT CAPACITANCES IN A SIGNAL PATH (IN pF AND μm)

From_Pin	To_Pin	Net Length	Our Method	Astro
U23/Y	U15/A	10.410	0.0063	0.0059
U15/Y	U13/D	6.630	0.0117	0.0110
U13/Y	U3/A	69.275	0.0242	0.0220
U3/Y	U3/A0	367.450	0.0724	0.0514
U3/Y	U7/A	6.690	0.0072	0.0066
U7/Y	ASTtlrInst210/B	87.660	0.0174	0.0136

TABLE VI

THE TOTAL RESISTANCE OF SEVERAL SINGLE NETS (IN Ω AND μm)

Master/Net	Net Length	Our Method	Astro
n34	174.0700	69.4523	65.0045
n31	20.9700	27.0321	26.4768
inv/n18	82.1400	42.7813	41.2127
inv/n12	77.0000	46.6221	42.9521
inv/n14	15.0700	23.8765	24.1780
inv/n15	73.6400	40.9747	39.1136
inv/n13	417.6100	136.6729	126.4125
inv/n10	81.2100	42.8972	40.9591

REFERENCES

[1] OpenAccess API Tutorial Eighth Edition (OA 2.2 DM4), Published by Silicon Integration Initiative, Inc, 2009.

[2] Chris Chu and Yiu-Chung Wong, Fast and accurate rectilinear steiner minimal tree algorithm for VLSI design, In Proc. IEEE Intl. Symp. on Physical Design, pages 28C35, 2005.

[3] Chung-Kuan Cheng, John Lillis, Shen Lin and Norman Chang wrote, Wenjian Yu and Xu Ning translated, Interconnected Analysis and Synthesis, Tsinghua University Press, 2008 (in Chinese).

[4] Chris Chu. FLUTE: Fast lookup table based wirelength estimation technique. In Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design, pp. 696-701, 2004.

[5] Shenghua Liu, Xianlong Hong, Tong Jing and Jingyu Xu, FREe: A Fast Routability Estimator, In Proc. of IEEE ICCCAS, 2006, Guilin, China, pp. 2454-2458.

[6] C.-C. Chang, J. Cong, Z. Pan, and X. Yuan, Multilevel global placement with congestion control, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 22, NO. 4, pp. 395-409, Apr. 2003.

[7] L. T. Pillage and R. A. Rohrer, Asymptotic waveform evaluation for timing analysis, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 9, No. 4, pp. 352-366, Apr. 1990.

[8] Wenjian Yu, Zeyi Wang, Enhanced QMM-BEM solver for 3-D multiple-dielectric capacitance extraction within finite domain, IEEE Trans. Microwave Theory Tech., 2004, 52(2): 560-566.

[9] William H. Kao, C. Lo, Mark Basel, Raminderpal Singh, Parasitic extraction: Current state of the art and future trends, in Proc. of IEEE, Vol. 89, No. 5, pp. 729-739, 2001.

[10] Synopsys, Raphael Reference Manual, Version 2003.09, Mountain View, California, September 2003.