Efficient Space Management Techniques for Large-Scale Interconnect Capacitance Extraction With Floating Random Walks

Chao Zhang and Wenjian Yu, *Senior Member, IEEE*

***Abstract*—In the capacitance extraction with the floating random walk (FRW) algorithm, the space management approach is required to facilitate finding the nearest conductor. The Octree and grid-based spatial structures have been used to decompose the whole domain into cells and to store information of local conductors. In this letter, the techniques with the distance limit of cell and only searching in cell's neighbor region are proposed to accelerate the construction of the spatial structures. A fast inquiry technique is proposed to fasten the nearest conductor query. We also propose a grid–Octree hybrid structure, which has advantages over existing structures. Experiments on large very large scale integration structures with up to 484 441 conductors have validated the efficiency of the proposed techniques. The improved FRW algorithm is faster than RWCap for thousands times while extracting a single net, and several to tens times while extracting 100 nets.**

***Index Terms*—Capacitance extraction, floating random walk (FRW), space management, spatial data structure, very large scale integration (VLSI) circuit.**

I. Introduction

The floating random walk (FRW) algorithm [1]–[6] is a major field-solver method for capacitance extraction. Compared with the deterministic algorithms (e.g., boundary element method [7]), the FRW algorithm has the advantages of lower memory usage, better scalability, and tunable accuracy.

Today, the FRW algorithm has become the kernel of commercial capacitance solvers (such as QuickCap). With parallel computing techniques, they have been applied to the block- or chip-level extraction task in the sign-off verification of very large scale integration (VLSI) circuits. Recently, a general FRW algorithm [3] and a hierarchical FRW algorithm [4] were proposed to deal with arbitrary dielectric configuration and for a fabric-aware extraction problem, respectively. In 2013, an FRW algorithm [5] was proposed for the interconnect structure with multilayered dielectrics, where the cross-interface transition probability and weight value are precharacterized for a given process technology. The algorithm was further accelerated with a comprehensive variance reduction scheme, and has been developed to a program called RWCap [5]. Although employing an Octree-based space management approach, RWCap is not efficient for handling the large-scale structures with thousands of conductors. Another approach based on an array data structure has been used, and achieved remarkable speedup over the K-D tree based technique [6]. However, it is not well compared with other techniques, and its details are not published. In [8], several data structures were discussed to speed up the distance queries in the FRW algorithm. But, they either have large space complexity, or lack actual implementation.

Manuscript received January 19, 2013; revised April 3, 2013; accepted July 15, 2013. Date of current version September 16, 2013. This work was supported in part by NSFC under Grant 61076034, the Beijing Natural Science Foundation under Grant 4132047, the Opening Foundation of ASIC and System State Key Laboratory (Fudan University), and the Tsinghua University Initiative Scientific Research Program. This paper was recommended by Associate Editor A. Elfadel.

The authors are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: eric.3zc@gmail.com; yu-wj@tsinghua.edu.cn). *(Corresponding author: W. Yu)*.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2013.2273985

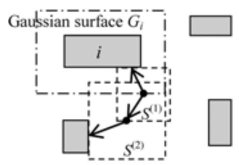

Fig. 1. Two random walks in the FRW algorithm.

Below, preliminaries of FRW based capacitance extraction and the space management are reviewed in Section II. Three accelerating techniques for the construction and inquiry of space management structure are proposed in Section III. After that, the improved Octree-based approach and a grid-Octree hybrid structure are proposed. Numerical results are given in Section V to validate the efficiency of proposed techniques.

II. Background

The FRW algorithm originates from

$$\phi(\boldsymbol{r}) = \oint_S P(\boldsymbol{r}, \boldsymbol{r}^{(1)})\phi(\boldsymbol{r}^{(1)})d\boldsymbol{r}^{(1)} \tag{1}$$

where $\phi(\boldsymbol{r})$ is the electric potential at point $\boldsymbol{r}$, S is the boundary of a domain enclosing point $\boldsymbol{r}$, and $P(\boldsymbol{r}, \boldsymbol{r}^{(1)})$ is called surface Green's function. The domain is called the transition domain, because (1) expresses the transition of a calculated quantity from the point inside the domain to its boundary. $P(\boldsymbol{r}, \boldsymbol{r}^{(1)})$ is of nonnegative value for any point $\boldsymbol{r}^{(1)}$ on S, and can be regarded as the probability density function for random sampling. To compute the capacitances related to conductor i, a Gaussian surface G_i is constructed to enclose it (Fig. 1). With the Gauss theorem, charge Q_i of conductor i becomes [5]

$$Q_i = \oint_{G_i} F(\boldsymbol{r})g \oint_{S^{(1)}} \omega(\boldsymbol{r}, \boldsymbol{r}^{(1)})P^{(1)}(\boldsymbol{r}, \boldsymbol{r}^{(1)})\phi(\boldsymbol{r}^{(1)})d\boldsymbol{r}^{(1)}d\boldsymbol{r} \tag{2}$$

where $F(\boldsymbol{r})$ is the dielectric permittivity at point $\boldsymbol{r}$, g is a constant, and function $\omega(\boldsymbol{r}, \boldsymbol{r}^{(1)})$ is called the weight value [1], [5]. Here, $P^{(1)}$ denotes the surface Green function for $S^{(1)}$ enclosing $\boldsymbol{r}$. With the Monte Carlo (MC) method, Q_i can be estimated as the statistical mean of sampled values on G_i, which is further the mean of sampled values on $S^{(1)}$ multiplying the weight value. With (1) substituted into (2) recursively, the above sampling procedure repeats until the potential of the sample point is known. This ends a walk, which may include several hops as shown in Fig. 1. With sufficient walks starting from the Gaussian surface, Q_i can be calculated accurately. It is revealed that the statistical mean of the weight values for the walks terminating at conductor j approximates the coupling capacitance C_{ij} ($j \neq i$) between conductors i and j.

Although the surface Green function for a spherical transition domain has a simple analytical expression, we only consider the cubic transition domain that is well suited to the Manhattan-shaped interconnects in a VLSI circuit. The surface Green function only depends on the relative position of $\boldsymbol{r}^{(1)}$. So, we can precalculate and tabulate the sampling probability and weigh value on a unit-size cube.

It should be pointed out that the major error of the FRW algorithm is the statistical error. The capacitance result obtained with n walks approximately obeys the normal probability distribution. Its standard deviation (Std) can be estimated with the variance of the sample values according to the central limit theorem. This Std is also called 1-σ error, which reflects the

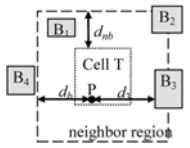

Fig. 2. Point in cell, cell's neighbor region, and calculating the nearest distance.

accuracy of the capacitance result. In the FRW algorithm, the 1-σ error is checked after every certain number of walks. Once it is below the specified accuracy goal, the algorithm terminates.

A major step in each FRW hop is constructing a conductor-free transition cube (Fig. 1). To be efficient, the cube should be as large as possible. This asks for finding the nearest conductor (Fig. 2). Since millions of hops are performed, the distance to the nearest conductor should be calculated as fast as possible. An Octree-based space management approach was presented in [5] for this aim. It divides the whole 3-D domain into organized small subdomains. Here, we call the subdomain *spatial cell*. Each cell is attached with a *candidate list* of conductors such that for any point in the cell its nearest conductor is in the list. With this approach, the inquiry of nearest conductor can be executed very quickly.

A primary operation for constructing the space management structure is generating the candidate list for a cell. We shall check the conductors one by one to see if they should be added to the list. To be rigorous, we give the following definitions, where $x_{\min}(\cdot)$ and $x_{\max}(\cdot)$ denote the minimum and maximum x-coordinates of a cuboid, respectively.

Definition 4: The x-distance between a cuboid A and a point P (x, y, z) is the larger one of $x_{\min}(A)-x$ and $x-x_{\max}(A)$. The y-distance and z-distance between A and P are defined similarly.

Definition 5: The distance between a cuboid A and a point P is the maximum of their x-, y-, and z-distances.

Definition 6: The x-distance between two cuboids A and B is the larger one of $x_{\min}(A) - x_{\max}(B)$ and $x_{\min}(B) - x_{\max}(A)$. The y-distance and z-distance between A and B are defined similarly.

Definition 7: The distance between two cuboids is defined as the maximum of their x-, y-, and z-distances.

Note in our problem, there are only Manhattan geometries. It is assumed that each spatial cell or conductor is a cuboid, and can be described by its two opposite vertices. We emphasize that the distances defined above can be a negative value, which is different from the ∞-norm. Below, $d(,)$ denotes the distance between two cuboids or a cuboid and a point.

Definition 8: T is a spatial cell, and B_1, B_2 are two conductors. If for any point $P \in T$, and $P \notin B_1 \cup B_2$, $d(P, B_1) \leq d(P, B_2)$, we say B_1 *dominates* B_2 regarding T.

The domination relationship is the key to generate the candidate list. If B_1 dominates B_2 regarding cell T, and B_1 is already in T's candidate list, B_2 should not be inserted into the list. This is because, for any point in cell T, its distance to B_1 is not larger than that to B_2. So, B_2 can be ignored while finding the nearest conductor for any point in the cell. In Algorithm 4 in [5], the procedure of candidate checking is described.

III. Three Accelerating Techniques

A. Improving the Candidate Checking With Distance Limit

For a large-scale problem including thousands of conductors, generating a candidate list through checking all conductors consumes large computing time. After giving two definitions, we propose a technique to reduce the time for generating the candidate list.

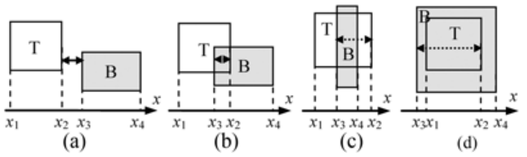

Fig. 3. Four topological relationships between a cell T and a conductor B. For the four cases, we have $d(T, B) = x_3 - x_2$.

Algorithm 1 CandidateCheck (conductor B, cell T)

```
1: $d := d(B, T)$; $l$ is the size of T;
2: If $d \geq L(T)$ then return;
3: For each b in the candidate list of T do
4:     If b dominate B then return;
5:     Elseif B dominate b then
6:         Remove b from the candidate list of T;
7:     Endif
8: Endfor
9: Add B to the candidate list of T;
10: If $(d+l) < L(T)$ then $L(T) := d + l$;
```

Definition 9: The size of a cuboid cell is defined as the maximum of the cell's length, width, and height.

Definition 10: The distance limit of a cell is the minimum distance between the cell and a conductor in its candidate list, plus the cell's size. $L(T)$ denotes the distance limit of cell T.

In Fig. 3, we show four typical relationships between a cell and a conductor. In all these cases, $d(T, B) = x_3 - x_2$, which may be negative. If a cell is intersected, but not fully occupied, by a conductor, the absolute value of their distance will be less than the size of cell [Fig. 3(b) and (c)]. This means that the distance limit of cell must be positive. Otherwise [Fig. 3(d)], the cell is not valid because it is impossible that the random walk would stop in it. So, we shall only consider the candidate list for the cells as shown in Fig. 3(a)–(c). This derives Theorem 1.

Theorem 7: For any valid cell T in the space management structure, its distance limit $L(T) > 0$. And, if the cell is intersected by a conductor, $L(T) \leq l$, where l is the size of T.

Actually, the distance limit $L(T)$ is an upper bound of the distance to nearest conductor from any point in cell T. While generating the candidate list, the value of T's distance limit dynamically changes. During this course, if the distance of a conductor B is not less than the distance limit, i.e., $d(T, B) \geq L(T)$, B must have been dominated by a conductor B_0 in the candidate list, which fulfills $d(T, B_0) + l = L(T)$. This is because, for any point P in the conductor-free space within T, $d(P, B_0) \leq d(T, B_0) + l = L(T)$. Therefore, $d(P, B_0) \leq d(T, B) \leq d(P, B)$, and B_0 dominates B according to Definition 5. This results in Algorithm 1, where the candidate checking can be canceled for some conductors.

B. Incomplete Candidate List

If we do not construct the largest transition cube, more conductors can be ignored for generating the candidate list. We may just check the conductors in a cell's neighbor region if during the random walk we need not generate the largest transition cube. This produces an incomplete candidate list for the cell. In this letter, we define the neighbor region of a cell by inflating it with the extension size d_{nb} (Fig. 2). During the random walk, the minimum distance between the current point and the candidate conductors should be compared with the distance between the point and the boundary of neighbor region. The smaller (such as d_b rather than d_3 in Fig. 2) should be used to guarantee that the transition cube is conductor-free.

Because the distance limit is an upper bound of the distance to nearest conductor, we can derive in what situations the incomplete candidate list is actually a complete one.

Theorem 8: After generating the candidate list with cell T's neighbor region with extension size d_{nb}, we get T's distance limit $L(\mathrm{T})$. If $L(\mathrm{T}) \leq d_{nb}$, the candidate list is a complete one, with which we can generate the largest transition cube.

Corollary 1: For a cell T intersected with the conductor, if $d_{nb} \geq l$, where l is T's size, the candidate list generated with T's neighbor region is a complete one.

We still use Algorithm 1 to generate the incomplete candidate list by only checking the conductors in the neighbor region. Alternately, we may check all conductors but set the initial distance limit to be d_{nb} instead of the infinity. Since the distance obtained by inquiring the incomplete candidate list is not the largest, it would increase the number of hops in a walk. However, as shown in Section V, this drawback is marginal if setting a suitable value of d_{nb}.

C. Reducing the Time for Inquiring the Candidate List

To find the nearest conductor from the current location of random walk, we need to find the cell containing the location and then traverse its candidate list to calculate the minimum distance. The computing time of traversal is proportional to the number of visited items. We propose a strategy to terminate the traversal earlier by sorting the candidate conductors in the ascending order of their distance to the cell. Suppose we traverse cell T's candidate list $\{\mathrm{B}_i\}$ for calculating the minimum distance from point P in T. If after visiting the first j items, $d(B_{j+1}, T) \geq d_{min,j} = min_{i\leq j}\{d(P, B_i)$, we can derive $d(P, B_{j+1}) \geq d(B_{j+1}, T) \geq d_{min,j}$. This means B_{j+1} and its succeeding items do not affect the value of the minimum distance since $d(B_{j+k+1}, T) \geq d(B_{j+k}, T), k = 1, 2, \ldots$. So, the traversal can be terminated immediately.

Because the candidate list is usually not long, and we calculate and sort the distances between a cell and its candidates in the construction stage, the overhead of this fast inquiry technique is negligible. With this technique, the inquiring time of the space management structure can be remarkably reduced.

IV. Spatial Structures and Approaches

A. Improved Octree Based Approach

We first consider the construction of the Octree structure. While inserting each conductor into the Octree, we must traverse it from the root node to a leaf node. In the following presentation, we regard the cell's attributes as the node's attributes. With the distance limit defined for each, unnecessary judgments of domination relationship can be avoided. Algorithm 2 describes the procedure of inserting a conductor to an Octree node. The whole structure is constructed by inserting all conductors one by one to the root node.

Algorithm 2 InsertToOctree (conductor B, node T)

```
1: If $d$(B, T) ≥ $L$(T), then return;
2: If T is a leaf node then
3:    CandidateCheck(B, T); //Algorithm 1
4:    If length of T's candidate list > $thres1$ and size of
         T > $thres2$ then
5:       Divide T equally into 8 child nodes: $T_1$, ... , $T_8$;
6:       For each b in the candidate list of T do
7:          For i = 1 to 8 do
8:             InsertToOctree(b, $T_i$);
9:          Endfor
10:      Endfor
11:   Endif
12: Else
13:   For each child node c of T do
14:      InsertToOctree(B, c);
15:   Endfor
16: Endif
```

Because the conductors are randomly inserted into the root node, redundant candidate checking still exists. This can be further reduced by the approach of incomplete candidate list.

When finding the nearest conductor in the FRW procedure, the traversal of Octree ends at a leaf node containing the current point. The left job is to traverse the candidate list of the leaf node for calculating the minimum distance whose time is proportional to the length of the candidate list. The fast inquiry technique in Section III-C is helpful to reduce the time.

B. Two Grid Based Approaches

The first step of inquiring the space management structure is to locate the cell containing the current point. To reduce its cost, a 3-D array representing the uniform partition of the whole domain is useful [6]. We call this the grid structure. Note that the number of conductors in each cell is not uniform, and so is the length of candidate list. Defining smaller cell size reduces the maximum length of candidate list, but causes a large number of cells. If a complete candidate list is generated for every grid cell, the construction time of the grid will be very huge.

To reduce the construction time, a strategy different from the idea of using candidate list can be adopted [6]. Only the intersected conductors are recorded for each cell without candidate checking. During the random walk, the conductors in the current cell and its adjacent cells are inquired to calculate the distance to conductor, similar to the approach with an incomplete candidate list. Because redundant conductors are handled, the random walk will perform slower with this approach.

We can also generate the candidate list for each cell if we only search in a neighbor region, which includes the cell's adjacent cells. For the problem with densely routed VLSI interconnects, this grid structure actually has the complete candidate list for many cells (according to Corollary 1). This largely reduces the construction cost, but does not degrade the efficiency of performing random walks. For the cell with too many candidates, it can be divided as a second-level grid. However, more levels of division should be avoided because it removes the advantage of grid structure for locating a point.

C. Hybrid Approach Using Grid and Octree

The distance limit is defined with the maximum dimension of cell. The pruning effect of distance limit can be weakened if the cell has a large aspect ratio, because with a same size of cell, the cube-shape cell has the largest volume. This occurs when the Octree-based approach handles the large-scale VLSI layout with a much larger lateral dimension. To overcome this drawback, we propose a hybrid approach using both the grid and Octree structures. First, the grid-based approach with a candidate list is used to represent the whole domain, where each grid cell is a cube. Then, the Octree structure is constructed for each grid cell.

The size of grid cell should be set as a large value (e.g., the height of the whole domain) so that the incomplete candidate list is almost a complete one and the expense of

Algorithm 3 GenerateGridOctree

```
1:  Suppose G is a 3-D array storing $n_x \times n_y \times n_z$ cells;
2:  $x_0$, $y_0$, $z_0$denote the minimum coordinates of the whole domain;
3:  s: = the size of cell; Set the distance limit of each cell in G to s;
4:  **For** each conductor B **do**
5:      Get B's extreme coordinates: ($x_{min}$,$y_{min}$,$z_{min}$), ($x_{max}$, $y_{max}$,$z_{max}$);
6:      ix1 := max(⌊($x_{min} - x_0$)/s⌋, 1); ix2 := min(⌈($x_{max} - x_0$)s⌉ + 1, $n_x$);
7:      iy1 := max(⌊($y_{min} - y_0$)/s⌋, 1); iy2 := min(⌈($y_{max} - y_0$)/s⌉ + 1$n_y$);
8:      iz1 := $max$(⌊($z_{min} - z_0$)/s⌋, 1); iz2 := min(⌈($z_{max} - z_0$)/s⌉ + 1, $n_z$);
9:      **For** i from ix1 to ix2, j from iy1 to iy2, k from iz1 to iz2 **do**
10:         CandidateCheck(B, G[i][j][k]); //Algorithm 1
11:     **Endfor**
12: **Endfor**
13: **For** each grid cell $E_i$ **do**
14:     Define an Octree root node $T_i$ for the domain of $E_i$;
15:     **For** each conductor $B_{i,j}$ in $E_i$'s candidate list;
16:         InsertToOctree($B_{i,j}$, $T_i$); //Algorithm 2
17:     **Endfor**
18: **Endfor**
```

TABLE I

VARIABLE PARAMETERS IN THE SPACE MANAGEMENT APPROACHES

Octree and grid-Octree hybrid structure	The lower bound of the dimension of leaf node (l_t); the length threshold of the candidate list (n_t); the extension size of neighbor region (d_{nb}); the cell size of the top-level grid (l_{tt}), only for the hybrid structure.
Two Grid structures	The size of the first-level grid cell (l_t); the length threshold of the candidate list for creating the second-level grid (n_t).

constructing the first-level grid is small. Algorithm 3 describes the construction procedure of the grid–Octree hybrid structure.

V. Numerical Results

Four large VLSI layouts are tested. The first one is a 1000×1000 cross-over structure. Each wire has dimensions of 14 nm $\times$ 28 μm $\times$ 14 nm. The distance between two metal layers is 86 nm. The second case is an actual design, called FreeCPU, based on the 180-nm technology with the minimum wire width (w_{min}) of 200 nm [5]. It includes 37 062 conductor blocks in five layers, which forms 3036 nets. The dimensions of whole structure are about 700 μm $\times$ 700 μm $\times$ 9.4 μm. The third case is an artificially created layout based on the 45-nm technology with w_{min} of 70 nm. It includes 101 595 conductor blocks in three layers with random dimensions and distribution. The last case is a larger case, including 484 441 conductor blocks. Its parameters are similar to FreeCPU. The proposed space management techniques are implemented in the C++ program RWCap [5]. In Table I, we list the parameters of the spatial structures.

The experiments are carried out on a Linux server with Intel Xeon E5-2650 8-core CPU of 2.0 GHz. All results are obtained from the execution of serial computing. The accuracy criterion of FRW algorithm is set to 0.5% 1-σ error.

TABLE II

CONSTRUCTION TIME OF OCTREE STRUCTURE (IN UNIT OF SECOND)

Case	#node	l_{min}(μm)	#conductor	Octree(O)	Octree(DL)	Octree(ICL)	Sp.
1	27145	0.41	2000	81.3	0.36	0.29	**280**
2	83441	7.11	37062	1757.9	3.10	0.74	**2375**
3	253761	2.34	101595	16595.6	8.21	2.43	**6829**
4	1061361	6.44	484441	--	83.12	17.12	--

TABLE III

EFFICIENCY OF FAST INQUIRY TECHNIQUE

Case	Average #walk	Average #hop/walk	Time for extracting a net (s)		Sp.
			FRW(O)	FRW(FastInq)	
1	2.82×10^5	9.1	3.04	1.61	**1.89**
2	3.03×10^5	12.5	2.97	1.41	**2.11**
3	1.89×10^5	10.5	1.73	0.81	**2.14**
4	2.77×10^5	12.7	3.63	1.49	**2.44**

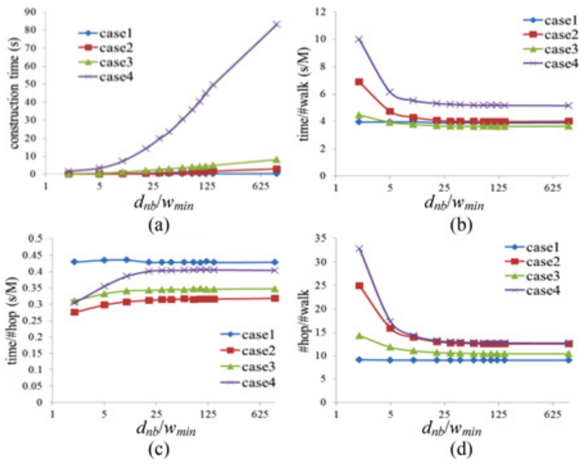

Fig. 4. The curves of the construction time (a), the average time for a million walks (b), the average time for a million hops (c), and the average number of hops per walk (d), with varied d_{nb} in the Octree based approach.

A. Validating the Three Accelerating Techniques

We use the Octree structure to demonstrate the efficiency of the accelerating techniques. In the experiment, $n_t = 21$, $l_t = 40w_{min}$, and $d_{nb} = 25w_{min}$. The total runtime of the FRW algorithm includes two parts: the time for constructing the space management structure and the time for the random walk procedure. The original Octree based approach in [5] is denoted as Octree(O). Octree(DL) and Octree(ICL) denote the versions using the distance limit and using both distance limit and incomplete candidate list, respectively. The construction time of the Octree structure is listed in Table II, where the number of Octree nodes and the minimum size of cell (l_{min}) are also given. From the table, we see that the distance limit brings huge acceleration. The incomplete candidate list further reduces the construction time for 4X. For Case 4, Octree(O) cannot finish the construction in a week, while Octree(ICL) costs only 17 s.

To validate the proposed techniques on the inquiry of space management structure, we randomly extract 100 nets for each case. The distance limit does not affect the inquiry time, and the incomplete distance list in this experiment affects it little. The results are listed in Table III, where FRW(O) and FRW(FastInq) denote the FRW algorithms without and with the fast inquiry technique in Section III-C, respectively. The average numbers of walks and hops per walk, and the average time for performing walks for extracting a net are listed.

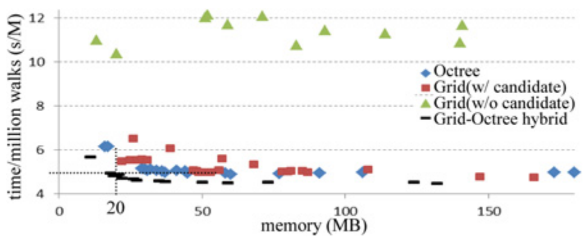

Fig. 5. Comparison of four space management approaches for the tradeoff of memory usage and the time for performing random walks.

TABLE IV
COMPARISON OF RWCAP AND RWCAP2

Case	RWCap				RWCap2					Sp. for 1000 nets
	T_{cons} (s)	T_{walk} (s)			T_{cons} (s)	Mem (MB)	T_{walk} (s)			
		1 net	100nets	1000nets			1 net	100nets	1000nets	
1	81.3	3.00	304	3185	0.13	6	1.17	119	1178	**2.8**
2	1758	1.80	297	2935	0.34	18	0.80	132	1265	**3.7**
3	16596	1.78	173	1729	1.37	87	0.75	71	718	**25.5**

TABLE V
COMPARISON OF RWCAP2 AND A COMMERCIAL SOLVER (CAPACITANCE IN UNIT OF 10^{-18} F, TIME IN UNIT OF SECOND)

Structure	Commercial solver			RWCap2				
	Cap.	Time/walk	Time	Cap.	Error	Time/walk	Time	Sp.
100×100	120.3	11.05×10^{-6}	4.15	120.4	0.1%	4.01×10^{-6}	1.13	**2.8**
500×500	598.9	13.44×10^{-6}	5.04	597.5	-0.2%	3.89×10^{-6}	1.11	**3.5**
1000×1000	1197	15.41×10^{-6}	5.78	1192	-0.4%	4.06×10^{-6}	1.12	**3.8**

It is demonstrated that the proposed technique gets over 2.1X speedup for larger cases.

To evaluate the effect of d_{nb}, we plot the curves of the construction time, the average time of million walks, the average time of million hops, and the average hop numbers of a walk in Fig. 4. It shows that if d_{nb} is larger than $25w_{min}$, there would be no difference in the walking procedure. As d_{nb} decreases, the number of hops per walk increases quickly. It makes the time per walk increase. So, $25w_{min}$ is chosen as an optimal d_{nb} to balance the construction time and the time for random walk.

B. Evaluating Different Space Management Approaches

Among the variable parameters listed in Table I, we set d_{nb} to $25w_{min}$, and l_{tt} to the height of the simulated domain. The left two variables are l_t and n_t. Note that d_{nb} and l_t should be measured in terms of w_{min}. This makes their effect changes little for cases under different process technologies. We have investigated the trends of the construction time, time of random walk procedure, and memory usage for Case 2, with different spatial structures. Numerical results reveal that the construction time always decreases when l_t or n_t increases. So does the memory usage. The time for the random walk procedure usually increases with l_t or n_t. There is a tradeoff between the memory usage or construction time and the random-walk time. We plot the memory usage and time for performing random walk in Fig. 5 under various parameter settings. From it we see that with same memory (∼20 MB), the grid–Octree hybrid approach makes 12% reduction of the random-walk time over the Octree, and 2.1X reduction over the grid without candidate list. With the same performance of random walk (∼5 s), the memory cost of the hybrid structure is less than half of others.

C. RWCap2 With Hybrid Approach Using Grid and Octree

The RWCap using the grid–Octree hybrid structure is called RWCap2. We compare it with RWCap [5]. The construction time and the random-walk time for extracting 1 net and multiple nets are listed in Table IV. Case 4 is dropped since it causes unbearable construction time of RWCap. The table shows, for extracting 1 net RWCap2 is up to 7829X faster than RWCap, and for extracting 1000 nets the speedup is from 2.8 to 26. Comparing Tables IV and II, III, we can see the advantages of the grid–Octree hybrid structure over the Octree structure. Note the former includes fewer nodes, e.g., 35 653 for Case 2.

RWCap2 is also compared with an advance commercial FRW solver on a Linux Server with AMD 2.4 GHz CPU with several crossover cases. For each one, the middle wire in the M2 layer is extracted. The results are listed in Table V. From the data of time/walk, we see that the space management approach used in RWCap2 brings 3X speedup over the other solver. For the total runtime, RWCap2 has larger speedup because its FRW procedure converges faster [5].

In the above experiments, we have omitted the dielectric configuration. While for the actual cases with multiple dielectrics, the technique in [5] can be used. This affects the space management approaches little, except that the constraint of dielectric interfaces should be considered while constructing the transition cube during random walks. We have used RWCap2 to extract these multidielectric cases. The results show that the proposed techniques bring 2X speedup to the random walk procedure. It is similar to the results of single-dielectric cases (Table III).

VI. CONCLUSION

Efficient techniques are proposed to largely accelerate the construction of space management structures, and to facilitate the fast nearest conductor query in the FRW based capacitance extraction. A new grid–Octree hybrid structure is proposed to achieve better tradeoff between the costs of space management and the efficiency gain on the random walk procedure. Large single- and multidielectric VLSI interconnect structures have been used to validate the efficiency of proposed techniques.

REFERENCES

[1] Y. L. Coz and R. B. Iverson, “A stochastic algorithm for high speed capacitance extraction in integrated circuits,” *Solid State Electron.*, vol. 35, no. 7, pp. 1005–1012, Jul. 1992.

[2] S. H. Batterywala, R. Ananthakrishna, Y. Luo, and A. Gyure, “A statistical method for fast and accurate capacitance extraction in the presence of floating dummy fills,” in *Proc. 19th Int. Conf. VLSI Design*, Jan. 2006, pp. 129–134.

[3] T. A. El-Moselhy, I. M. Elfadel, and L. Daniel, “A capacitance solver for incremental variation-aware extraction,” in *Proc. ICCAD*, Nov. 2008, pp. 662–669.

[4] T. A. El-Moselhy, I. M. Elfadel, and L. Daniel, “A hierarchical floating random walk algorithm for fabric-aware 3-D capacitance extraction,” in *Proc. ICCAD*, Nov. 2009, pp. 752–758.

[5] W. Yu, H. Zhuang, C. Zhang, G. Hu, and Z. Liu, “RWCap: A floating random walk solver for 3-D capacitance extraction of VLSI interconnects,” *IEEE Trans. Computer-Aided Design*, vol. 32, no. 3, pp. 353–366, Mar. 2013.

[6] G. Rollins. (2010, Jul.). “Rapid3D 20X performance improvement,” Online presentation of Synopsys, Inc. [Online]. Available: http://www.synopsys.com/Community/UniversityProgram/Pages/Presentations.aspx

[7] W. Yu, X. Wang, Z. Ye, and Z. Wang, “Efficient extraction of frequency-dependent substrate parasitics using direct boundary element method,” *IEEE Trans. Computer-Aided Design*, vol. 27, no. 8, pp. 1508–1513, Aug. 2008.

[8] N. Bansal, “Randomized algorithms for capacitance estimation,” Indian Instit. Technol. Bombay, Mumbai, India, Tech. Rep., Apr. 1999.