ECO-Aware Obstacle-Avoiding Routing Tree Algorithm

Jui-Hung Hung^1, Yao-Kai Yeh^1, Yu-Cheng Lin^2, Hsin-Hsiung Huang^3 and Tsai-Ming Hsieh^1

^{1} Dept. of Information and Computer Engineering, Chung Yuan Christian University, Chung-Li, Taiwan, R.O.C.
^{2} Dept. of Information and Electronic Commerce, Kainan University, Taoyuan, Taiwan, R.O.C.
^{3} Dept. of Electronic Engineering Lunghwa Univ. of Science and Technology, Taoyuan, Taiwan, R.O.C.
{g9777036, g9777023}@cycu.edu.tw, linyu@mail.knu.edu.tw, pp022@mail.lhu.edu.tw, hsieh@cycu.edu.tw

Abstract—**This study formulates a novel routing problem of engineering change order- (ECO for short) aware Steiner minimal tree with obstacles and solves it by a multiple-stage approach, including partitioning, analysis distribution of spare cells, virtual node insertion and diagonal-based routing tree construction. The objective of this paper is to construct an ECO-aware routing tree in the sense of ECO resources. The number of available spare cells near the routing tree significantly increases while minimizing the additional length compared to the original tree algorithm. To efficiently analyze, an entire chip is divided into a set of fixed-size grids and the number of spare cells in each grid is calculated. To reduce the additional length, we insert the number of user-defined virtual nodes, which represent the grids with more spare cells. Furthermore, a graph-based routing algorithm is used to construct an X-architecture tree. To further reduce total wire length, each segment in the spanning tree is transferred into the corresponding combination of vertical, horizontal and diagonal segments. Experimental results show that the number of available spare cells is increases by 66.5%, while leading to only 2.8% additional total wire length.**

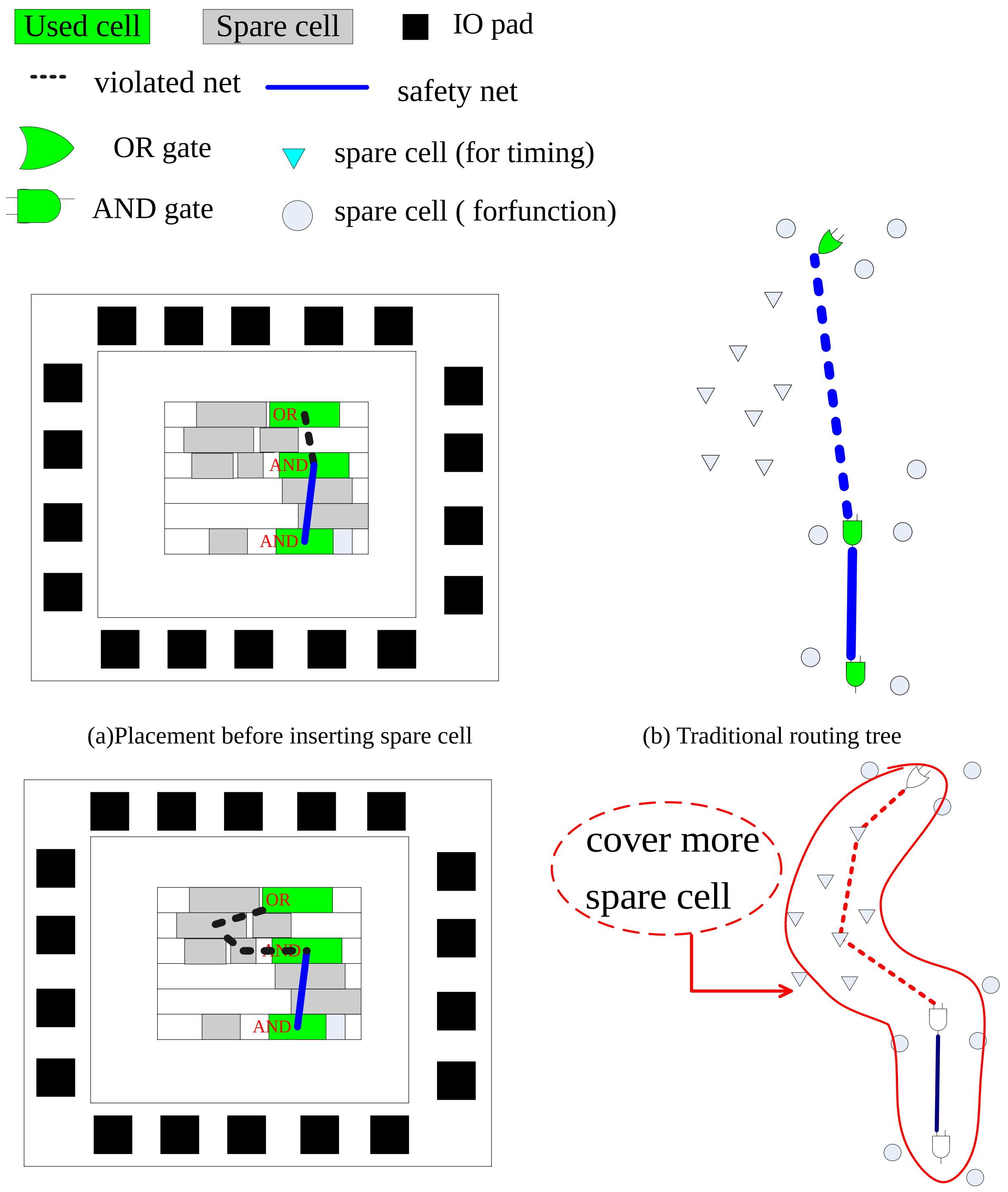
Key-Words: Engineering change order, Obstacle-avoiding, Spare cell, Routing algorithm, X-architecture.

1 Introduction

Engineering change order is an important approach to fix timing and functional violations [1]. To reduce the very expensive mask cost [2][3], the spare cells re-synthesis techniques [4] are utilized to change the functions of gates or timing of wires. To directly select the available spare cells or synthesize the undirected re-synthesis spare cells are usually time-consumption and can not obtain the spare cell assignment with the optimal additional wire length. Due to the benefit of ECO technique, the chip designer significantly saved the re-taped-out time for integrated circuits.

ECO related researches were usually divided into two types, including the timing-driven buffer insertion of the interconnections [5][6] and functional change of the gate re-synthesis [7][8][9][10]. In the first type, Chen *et al.* [5] repaired the timing violated nets by using the gate sizing and buffer insertion. The cell selection approach repaired the many violations with the accepted time. Lu *et al.* [6] studied the violations of input slew and output loading and gave an effective approach to repair violated nets by reconnecting the spare cells. In the second type, Chang *et al.* [7] proposed the effective approach to fix a number of errors by using a post-processing to shorten the manufactured time. Chang *et al.* [8] further explored the analysis-based method for spare cells selection and equally utilized the spare cells in the chip. Kuo *et al.* [9] synthesized the desired function by rewiring some interconnections for spare cells. The constant insertion technique really increases the flexibility under the resource constraints of spare cells. Some ECO-aware researches fixed the timing violations at the later stages such as the placement and IC taped-out. However, the penalties, including the additional total length and time of post-processing, to repair the timing or functional violations are huge.

Many algorithms were proposed to solve the routing problem with (or without) obstacles because system-on-chip design flow widely used predefined intellectual properties (IPs for short) and macros in the chip. Pan *et al.* [12] constructed an initial length-driven routing tree and applied the branch-moving technique to improve the delay of the critical path in the routing tree. Hu *et al.* [13] incorporated an idea of the soft edge into a timing-constrained router and adjusted the Steiner points with consideration of the timing and congestion. Wu *et al.* [14] proposed a graph-based



(a)Placement before inserting spare cell

(b) Traditional routing tree

(c)Placement after inserting spare cell

(b) ECO-aware routing tree

Figure 1. ECO for violated wires (Timing change).

approach which efficiently calculated total wire length with obstacles to construct the routing tree. Huang *et al.* [15] provided two routing algorithms with different objectives. Some approaches presented the X-architecture, which supports the horizontal, vertical and diagonal segments, to reduce the wire length [16][17][18]. Ho *et al.* [16] developed the multilevel framework to construct an X-architecture routing tree which supported diagonal segments. However, most methods did not consider spare cells during the routing tree construction. Some of them could not support the X-architecture to further reduce the wire length of the routing tree.

The major contributions are as follows. First, the ECO-aware routing issues are considered during the routing tree construction. Before the IC taped-out, the routing paths are guided to pass through the regions with a lot of spare cells. Hence, the timing violations could be easily repaired with the short additional lengths. Second, the concept of virtual nodes is provided to insert the desired spare cells during the routing tree construction. An entire chip is partitioned into a set of fixed-size grids. After we analyze the distribution of spare cells, the user-defined extra spare cells (the center of gravity in each partitioned grid) will be inserted. Finally, an efficient spanning graph-based algorithm is utilized to build a better X-architecture routing tree in the sense of ECO resources.

The reminder of this work is as follows. Section 2 illustrates the motivation, terminologies and problem definition. Section 3 explains the proposed ECO-aware routing algorithm in detailed. Experimental results and conclusions are listed in Sections 4 and 5, respectively.

2 Preliminary

In the section, we described the motivation, terminology and the problem definition.

2.1 Motivation

Traditional method constructed a routing tree consideration of the ECO resources, i.e. many spare cells and gates. Therefore, the designer usually repaired the timing violations by using buffers [19] with the large additional length. However, if we analyzed the distribution of spare cells and redundant gates and integrated resources into the routing tree construction, routing paths in the ECO-aware routing tree can be guided to regions with a lot of spare cells. Actually, there are three ECO types, such as the buffer insertion for timing violated wires, the gate replacement for functional change and rewiring for modifying subcircuits. In this work, only the ECO type 1, buffer insertion for violated wires, is explored. For an example in Figure 1(a), there are 2 two-terminal nets in the chip core at the placement stages. One net is timing violation and we need insert a buffer to repair the violation. Figure 1(b) illustrates the graph representation of routing tree. However, there is no buffer near the violated net and there chip designer should modify the long wire length to connect a buffer. Figure 1(c) shows the novel routing tree construction which guide the routing paths to pass through the regions with more spare cells (buffers). Figure 1(d) is the ECO-aware routing tree with less additional length to repair violation.

This observation motivates us to develop an ECO-aware routing algorithm to increase available number of spare cells with the less additional wire length. Hence, we only need less wire length to repair timing violations because the routing tree passing through regions with more spare cells.

2.2 Generation of virtual node

In this work, only user-defined virtual nodes were involved into the ECO-aware routing tree construction because we could balance the improvement of available spare cells and the additional wire length.

To analyze the distribution of the spare cells, the entire chip area is divided into a set of fixed-size grids. Each grid is sorting with the increasing order of the number of spare cells. According to the user-defined number, the top grids with more spare cells are selected. In each grid, there are many spare cells and it is important to take one candidate spare cell as one virtual node.

For each grid, there is more penalty of wire length if we randomly select the spare cell as one virtual node. In this work, the center of gravity in each grid is selected. The coordinate of one virtual node in each grid is computed by the following formulas,

$$gx_t = \frac{\sum_{i=1}^{num_t} scx_i}{num_t} \tag{1}$$

and

$$gy_t = \frac{\sum_{i=1}^{num_t} scy_i}{num_t} \tag{2}$$

where scx_i and scy_i are the *x*- and *y*- coordinates of each spare cell in the grid g_t. (gx_t, gy_t) is the center of gravity of grid g_t. num_t is the number of spare cells of the grid g_t. The center of gravity, that may be a spare cell or a dummy node, is selected as the virtual node for grid g_t.

2.3 Searching distance

To minimize the degeneration of delay, the proposed algorithm will not increase the huge extra additional wire length. Besides, the routing paths are guided to pass through the region with large amount of buffers. The searching space is a $2\delta \times 2\delta$ square region; wehre δ denotes the searching distance between the node and the boundary of the searching region. Figure 2 illustrates the region with a proper distance δ. We assume a chip with $10000 \times 10000 um^2$ and the searching distance is set to be 5% chip width (about 500 um). In fact, our algorithm is independent to the parameter of searching distance. If the larger searching distance (>500um), more available spare cells will be near the routing tree. However, there is larger additional wire length, which connects the far spare cell, to repair the timing violation.

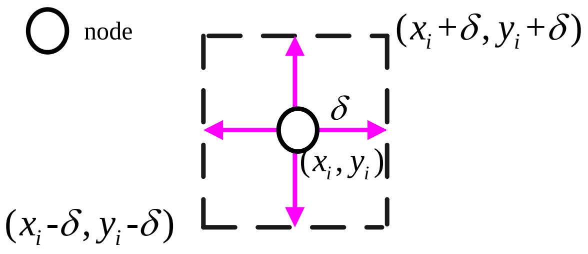


Figure 2. Searching space of a node.

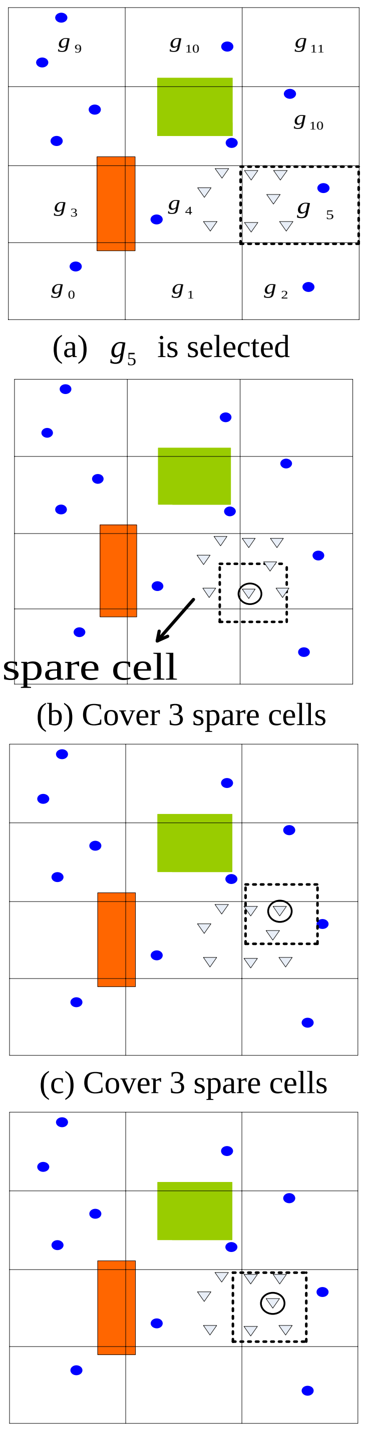
2.4 Available Number of Spare Cells

To increase the number of spare cells to fix timing violations, a routing tree with more number of available spare cells (abbreviated as *ac*) is preferred. In a routing tree, the number of available spare cells *ac* is estimate by the following formula,

$$ac = \sum_{i=1}^{n} sc_i + \sum_{i=1}^{v} sc_i \qquad (3)$$

where n and v are the number of terminals and spare cells, respectively. The first and second terms denote the number of available spare cells of n given terminals and additional virtual nodes v.

For example, there are 3 and 5 spare cells in the grids g_4 and g_5, see Figure 3(a). Because the user-defined virtual number (v = 1) is one, only the grid with the most number of spare cells is selected and the grid g_5 with 5 spare cells is selected. If the spare cell in Figure 3(b) is selected, the covering number of available spare cells is only three. Similarly, the covering number of available spare cells is also three spare cells in Figure 3(c). To increase the available number of spare cells, the center of gravity for the spare cells in each grid is selected as the virtual node, see Figure 3(d). Summary, we integrate the virtual node into the original set of terminals and we obtain the ECO-aware routing tree passing through the regions with more spare cells.



(a) g_5 is selected
(b) Cover 3 spare cells
(c) Cover 3 spare cells
(d) Cover 5 spare cells
Figure 3. Generation of a virtual node ($v = 1$).

2.5 Problem Definition

According to the discussion of generation of virtual node, the number of available spare cells is increased. By the experience of the chip designer, some circuit designs only consumed about 70~80% utilization at the placement phase. Therefore, the percentage of the dummy cells, including the spare cells and gate array, is 20%. The ratio between the used cells and unused spare cells are about 4:1. How to manage the more spare cells to repair the timing violations properly while reducing the additional wire length is a challenge and a very important issue. The following is the problem formulation of this work.

Given a set of n terminals $\{t_1, t_2, \cdots t_n\}$*, a set of m obstacles* $\{r_1, r_2, \cdots r_m\}$*, a set of k spare cells* $\{sc_1, sc_2, \cdots sc_k\}$*, a searching distance* δ*, and the user-defined number of virtual node* v*, the objective of this work aims to construct a better ECO-aware routing tree which maximizes the available spare cells while minimizing total additional wire length.*

3 An ECO-Aware Routing Tree

In this section, we describe a proposed graph-based method in detailed. The proposed approach contains several steps. Each step is explained in the following sub-sections.

First, the entire chip is divided into a set of $p \times q$ fixed-size grids. The number of spare cells in each grid is computed and we sort the grids by the ascending order of number of spare cells for each grid.

Second, some spare cells are chosen to be candidate virtual nodes according the number of spare cells in $p \times q$ grids and the number of user-defined values. The top grids with the more spare cells are generated the corresponding virtual nodes. To balance the improvement on available spare cells and the additional wire length, the v virtual nodes, which correspond to top v grids, are produced.

Third, for the terminals and virtual nodes with obstacles, the graph-based approach is applied to construct the X-architecture routing tree. The approach, which is similar to the algorithms in [14], contains three steps, including the spanning graph construction, the spanning tree generation and an X-architecture routing tree. For the virtual nodes, terminals and obstacles, a spanning graph is generated by a set of potential interconnections. For the graph, a minimal spanning tree is obtained to connect all terminals and virtual nodes. Finally, each segment of the minimal spanning tree is transferred into the combination of vertical, horizontal and diagonal segments.

For an example in Figure 4(a), the circuit containing 2 obstacles, 8 spare cells and 12 terminals, is utilized to explain the proposed algorithm. To minimize the additional wire length, the number of virtual node is set to be one. First, the entire chip is partitioned into a set of 4×3 fixed-size grids, see Figure 4(b). Second, one virtual nodes in Figure 4(c) is produced when we evaluate the distribution of spare cells 12 grids and the user-defined value v (=1). Furthermore, the spanning graph is built according to terminals, virtual nodes, and obstacles, see Figure 4(d). Moreover, the minimal spanning tree in Figure 4(e) is constructed from the spanning graph. Finally, the X-architecture spanning tree is obtained by transferring each segment of the spanning tree in Figure 4(f). The algorithm is shown in Figure 5.

```
Algorithm: ECO-Aware Routing Tree Construction
Input: (1) A set of n terminals
       (2) A set of m rectangular obstacles
       (3) A set of k spare cells
       (4) A user-defined searching space $\delta$
       (5) The number of virtual nodes $v$
Output: An ECO-aware X-architecture routing tree
begin
   1.Partition an entire chip into a set of fixed-size
     grids and sort the grids by the ascending
     order of number of spare cells for each grid;
   2. Produce the virtual nodes by the user-defined
     number;
   3. Perform the graph-based approach to obtain
     an X-architecture routing tree;
   3.1 Construct the spanning graph;
   3.2 Build the minimal spanning tree;
   3.3 Obtain the X-architecture tree;
end.
```

Figure 5. Outline of the proposed algorithm.

4 Experimental Results

We implemented the proposed algorithm by using C++ language on an Intel Core2 Duo 1.87G and 1.86G machine with 3GB main memory. The objective of this paper is to maximize the numbers of available spare cells. Due to the limitation of paper space, only the results of the searching distance δ (=500*um*) is listed in experimental results. We observed that the more numbers of virtual nodes lead to the more available numbers of spare cells while the additional total wire length is a lot. To balance the additional wire length and the available number of spare cells, the number of virtual nodes is the user-defined value. If the designer wants to construct a routing tree with more available number of spare cells, they should increase the number of virtual nodes, vice versa. To the space limitation, we list the results of 0, 2 and 4 available numbers of spare cells. There is no standard benchmarks

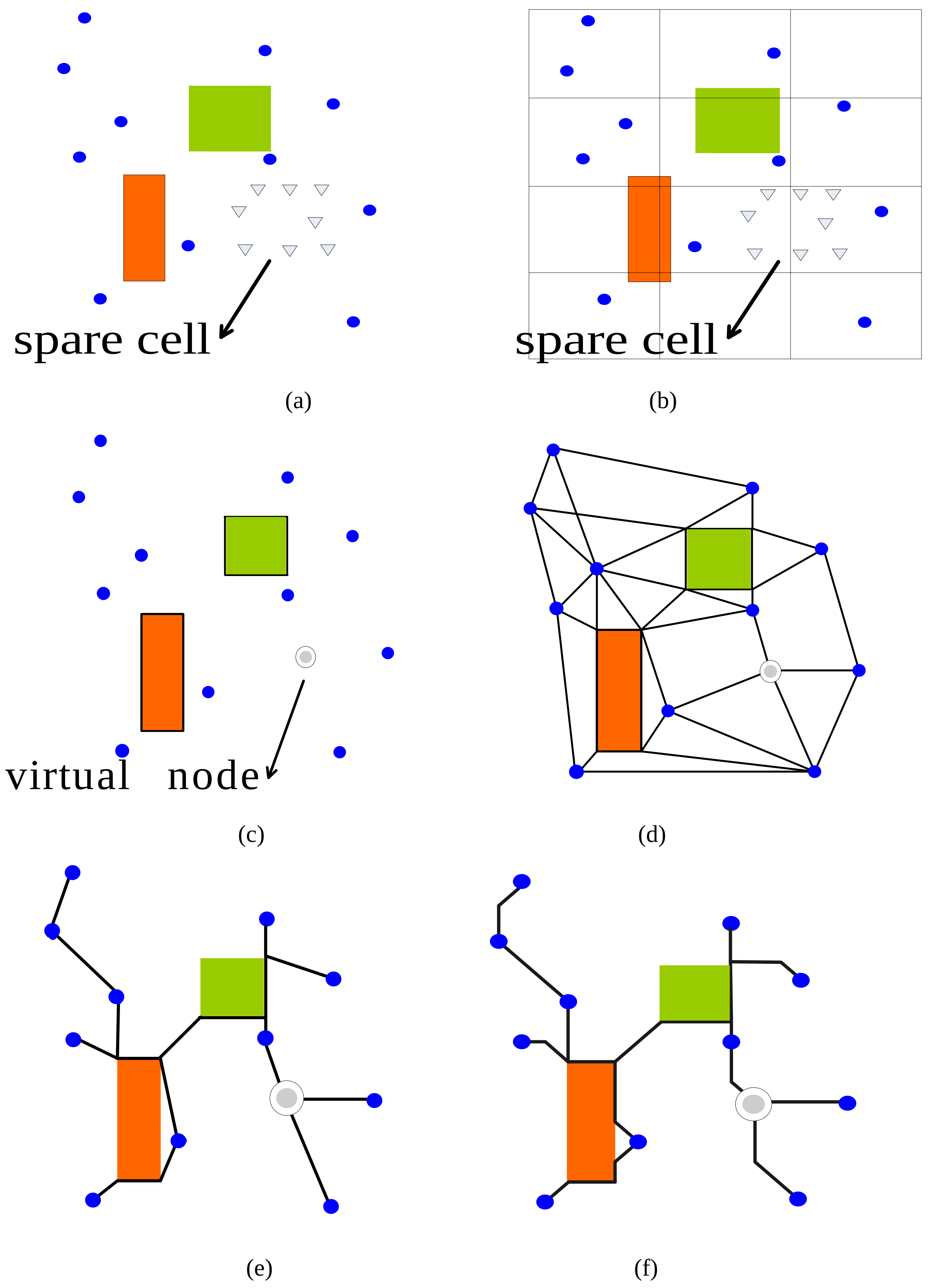


Figure 4. An example of our proposed method.

announced, circuits in [14] are modified to evaluate the performance of the proposed algorithm. The number of terminals, spare cells and obstacles are shown in Table 1. Table 2 illustrates the circuit names, the available numbers of spare cells and the improvement on available spare cells, respectively. Similarly, Table 3 lists the circuit names, total wire length and the percentage of additional wire length, respectively.

First, we explored the improvement on the available number of spare cells at the routing stage. In Table 2, symbols I_sc and II_sc are the results of the traditional and ECO-aware (two virtual nodes) routing algorithms, respectively. From Table 2, we observe that improvement on the available number of spare cells is up to 41.4% (100×(287-203)/203). Similarly, the symbol III_sc is the result of ECO-aware routing tree with four virtual nodes ($v=4$), and the improvement is about 66.5% (=100×(338-203)/203). Summary, the more virtual nodes we insert during the routing tree construction, the more numbers of spare cells we can obtain.

Second, we observed the additional total wire length at the ECO-aware routing tree construction. In table 3, symbols I_wl and II_wl are the total wire lengths with zero ($v=0$) and two ($v=2$) virtual nodes, respectively. Compared to the results of traditional method, we observe the additional wire length of ECO-aware routing tree ($v=2$) is 1.5% (=100 × (1087234.306-1071041.71)/ 1071041.71). Similarly, the symbol III_wl denotes total wire length of four virtual nodes ($v=4$) and the additional wire length is up to 2.8% (=100 × (1101397.302-1071041.71)/1071041.71) compared to result of traditional approach ($v=0$).

Third, we design a graph-user-interference (GUI) to illustrate the ECO-aware routing tree under the different number of virtual nodes. The blue and red nodes in Figure 6(a) are the terminals and spare cells, respectively. The green blocks represent obstacles which the wires are not allowed to pass through obstacles. Figure 6(b) is the original routing tree with two local views which are shown in Figure 6(c) and (d). Figure 7 illustrates the routing tree with the different number of virtual nodes. Figure 7(a) shows one virtual node insertion and (b) is the routing results with only one virtual. Similarly, Figures 7(c), (d), (e) and (f) denote the routing results with two, three, four and five virtual nodes. Compared to the original routing result in Figure 6(a), the routing results in Figures 7(b), (c), (d), (e) and (f) indeed increase the available number of spare cells.

Summary, the available number of spare cells is significantly increase up to 66.5% by applying the virtual node insertion while the additional wire length is only 2.8%

5 Conclusion

In the paper, we formulate the ECO-aware X-architecture minimal Steiner tree problem and solve it by integrating the virtual nodes insertion with the graph-based routing algorithm. The better ECO-aware routing tree in the sense of ECO resources is constructed to improve the available number of spare cells. The analyzed approach, which divided the chip into a set of fixed-size grids followed by calculating the number of spare cells for each grid, is presented to choose the top grids with more spare cells. The concept of the virtual nodes which represent the regions with more resources of spare cells is incorporating during the routing tree construction. Our experimental results show that we improve the number of spare cells by 66.5% while total additional wire length is 2.8%.

References

[1] Reports of the International Technology Roadmap for Semiconductors, 2007, http://www.itrs.net/.

[2] Synopsys, Inc. http://www.synopsys.com/.

[3] Cadence Design System. http://www.cadence.com/.

[4] M. Pedram and N. Bhat, "Layout Driven Technology Mapping," *Proc. of Design Automation Conference*, pp. 99–105, 1991.

[5] Y.P. Chen, J.W. Fang, and Y.W. Chang, "ECO Timing Optimization Using Spare Cells," in *Proc of IEEE/ACM International Conference on Computer-Aided Design*, pp.530-535, 2007.

[6] C.P. Lu, M. C.T. Chao, C.H. Lo, and C.W. Chang, "A Metal-only-ECO Solver for Input Slew and Output Loading Violations," in *Proc. of ACM/IEEE ISPD*, pp.191-198, 2009.

[7] K.H Chang, I.L. Markov, V. Bertacco, "Automating Post-Silicon Debugging and Repair," in *Proc. of IEEE/ACM International Conference on Computer-Aided Design*, pp. 91-98, 2007.

[8] K.H. Chang, I.L. Markov and V. Bertacco, "Reap What You Sow: Spare Cells for Post-Silicon Metal Fix," in *Proc. ACM/IEEE International Symposium on Physical Design*, pp. 103-110, 2008.

[9] Y.M. Kuo, Y.T. Chang, S.C. Chang, M.M. Sadowska, "Engineering Change Using Spare Cells with Constant Insertion," in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pp.544-547, 2007.

[10] K.Y. Lee and T.C. Wang, "On Using Spare Cells for Functional Changes with Wirelength Consideration," in *Proc. of The Workshop on Synthesis And System Integration of Mixed Information technologies*, pp.64-69, 2009.

[11] Y.L. Li, J.Y. Li and W.B. Chen, "An Efficient Tile-based ECO Router using Routing Graph Reduction and Enhanced Global routing Glow," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 26, No. 2, pp.345-358, 2007.

[12] M. Pan and C. Chou, "A Novel Performance-Driven Topology Design Algorithm," in *Proc. of ACM/IEEE Asia and South Pacific Design Automation Conference*, pp.23-26, 2007.

[13] J. Hu and S. Sapatnekar, "A Timing-Constrained Algorithm for Simultaneous Global Routing of Multiple Nets," in *Proc. of IEEE/ACM International Conference on Computer-Aided Design*, pp. 99-103, 2000.

[14] P.C. Wu, J.R. Gao and T.C. Wang, “A Fast and Stable Algorithm for Obstacle-Avoiding Rectilinear Steiner Minimal Tree Construction,” in *Proc. of ACM/IEEE Asia and South Pacific Design Automation Conference*, pp. 262-267, 2007.

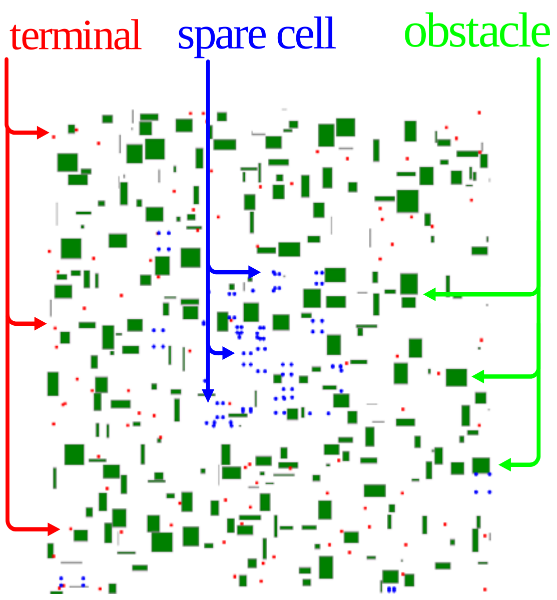
[15] H.H. Huang, H.Y. Huang, D.J. Huang and T.M. Hsieh, “An Obstacle-Avoiding Efficient Rectilinear Steiner Tree Construction,” *WSEAS Transactions on Circuits and Systems*, pp.1775-1782, 2006.

[16] T.Y. Ho, C.F. Chang, Y.W. Chang, and S.J. Chen, “Multilevel Full-Chip Routing for the X-based Architecture,” in *Proc. of ACM/IEEE Design Automation Conference*, pp. 597-602, 2005.

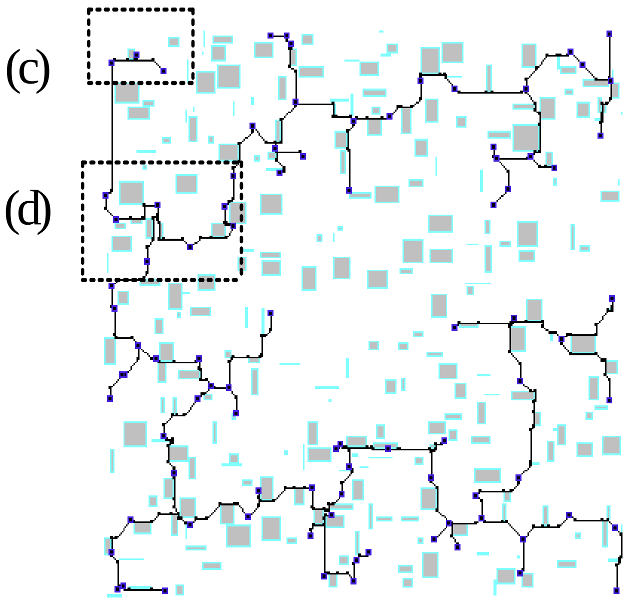
[17] S.P. Chang, H.H. Huang, C.C. Lin and T.M. Hsieh, “Rectangular and Non-Rectangular Obstacles-Avoiding Timing-Driven Steiner Routing Tree for the X-Architecture”, *WSEAS Transactions on Circuits and Systems*, pp. 433- 442, 2009.

[18] H.H. Huang, S.P. Chang, Y.C. Lin, and T.M. Hsieh, “Timing-Driven Non-Rectangular Obstacles-Avoiding Routing Algorithm for the X-Architecture,” in *8th WSEAS International Conference on Instrumentation, Measurement, Circuits and Systems*, pp. 31-34, 2009.

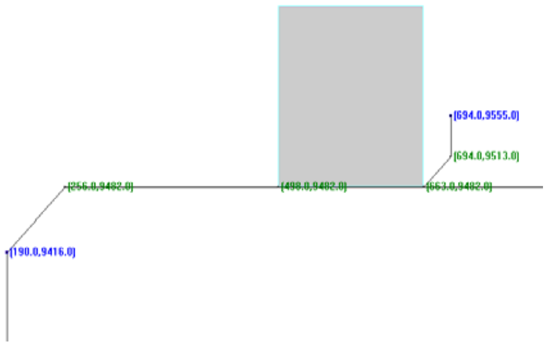
[19] T. Sharma, K.G. Sharma, B.P. Singh, N. Arora, “Efficient Interconnect Design with Novel Repeater Insertion for Low Power Applications,” *WSEAS Transactions on Circuits and Systems*, pp. 153 -162, 2010.



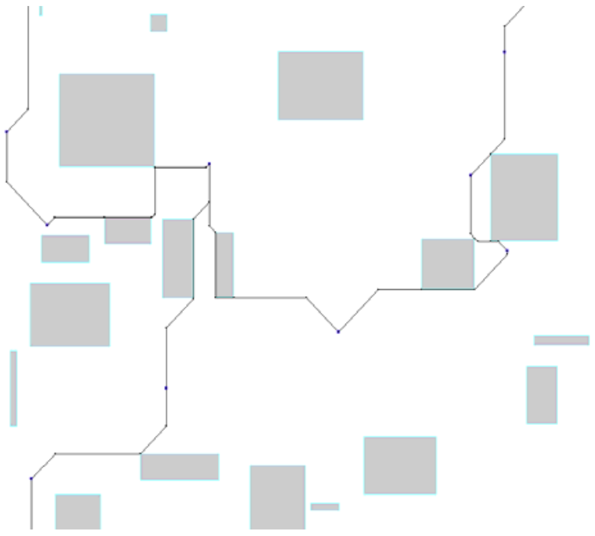
(a) Distribution of resource



(b) Original tree with two local views



(c) Local view with coordinates



(d) Local view of diagonal segment

Figure 6. Illustration of the original routing tree.

Table 1. Benchmark circuits.

	No. of terminal	**No. of obstacle**	**No. of spare cell**
R1	90	250	100
R2	182	250	100
R3	171	400	100
R4	183	500	100
R5	884	5000	500

Table 2. Improvement on the available spare cells.

	No. of available spare cell (v)			**Imp_sc (%)**	
	I_sc ($v=0$)	**II_sc** ($v=2$)	**III_sc** ($v=4$)	**Imp1 (%)**	**Imp2 (%)**
R1	17	35	43	105.9	152.9
R2	44	69	76	56.8	72.7
R3	33	47	60	42.4	81.8
R4	37	58	73	56.8	97.3
R5	72	78	86	8.3	19.4
all	203	287	338	-	-
norm	**1**	**1.414**	**1.665**	**41.4**	**66.5**

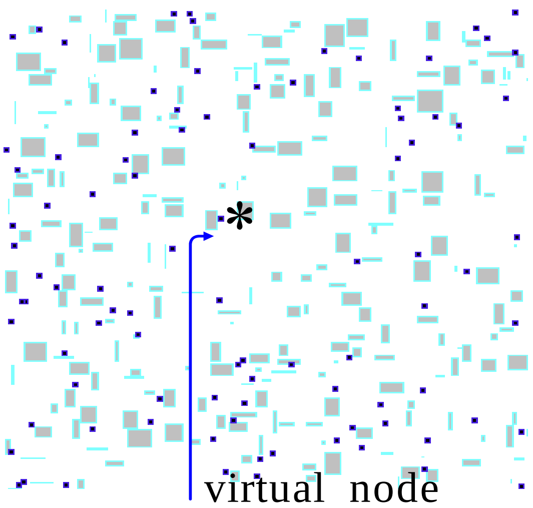
I, II and III denote the routing tree with zero, two and four virtual nodes with the searching distance δ = 500*um*.
Imp1 = 100 (II_sc-I_sc)/I_sc; Imp2 = 100 (III_sc-I_sc)/I_sc;

Table 3. Results of the additional wire length.

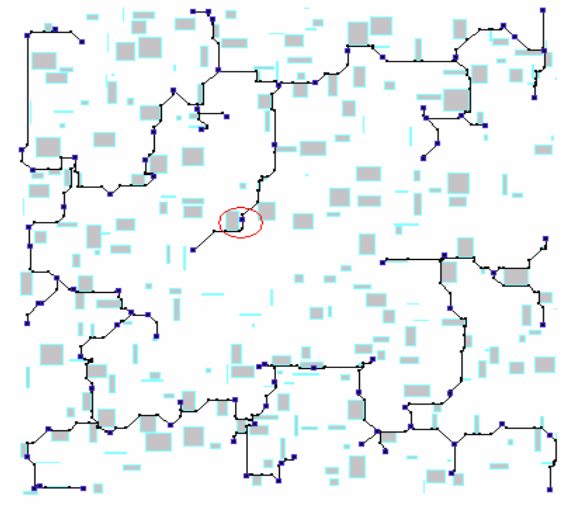
	Totla wirelength			**Add_wl (%)**	
	I_wl ($v=0$)	**II_wl** ($v=2$)	**III_wl** ($v=4$)	**Imp3 (%)**	**Imp4 (%)**
R1	74488.002	77271.165	79506.209	3.7	6.7
R2	100514.948	102606.367	104648.195	2.1	4.1
R3	99913.084	102845.173	104306.758	2.9	4.4
R4	100241.598	102419.678	104193.654	2.2	3.9
R5	695884.078	702091.923	708742.486	0.9	1.8
all	1071041.71	1087234.306	1101397.302	-	-
norm	**1**	**1.015**	**1.028**	**1.5**	**2.8**

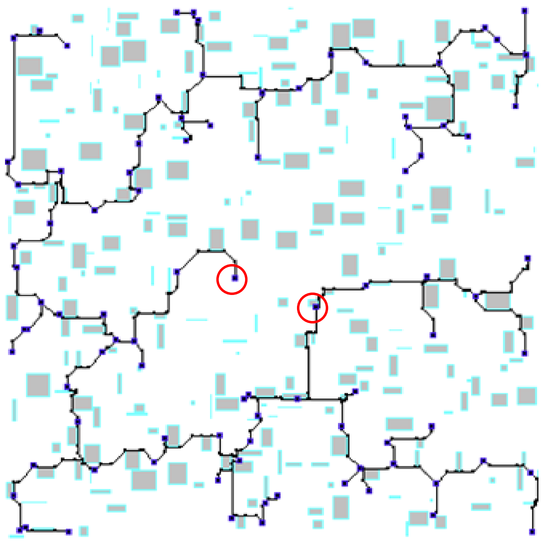
I, II and III denote the routing tree with zero, two and four virtual nodes with the searching distance δ = 500*um*.

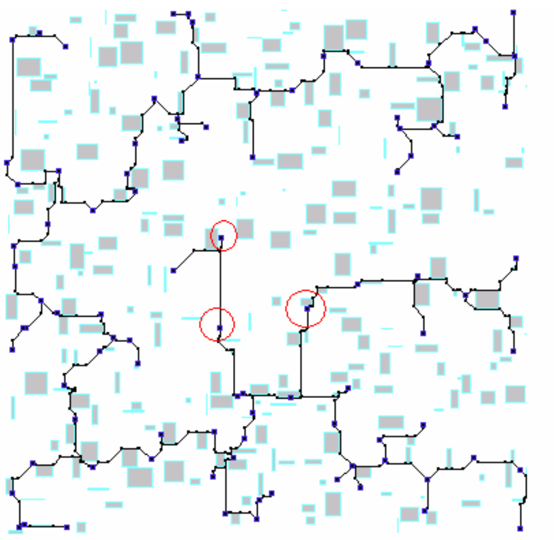
Imp3 = 100 (II_wl-I_wl)/I_wl; Imp4 = 100 (III_wl-I_wl)/I_wl;

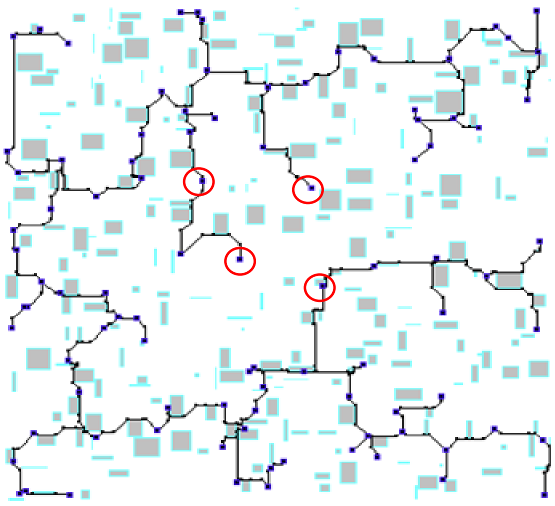


(a) Generate a virtual node


(b) Routing result (v =1)


(c) Routing result (v =2)


(c) Routing result (v =3)


(e) Routing result (v =4)

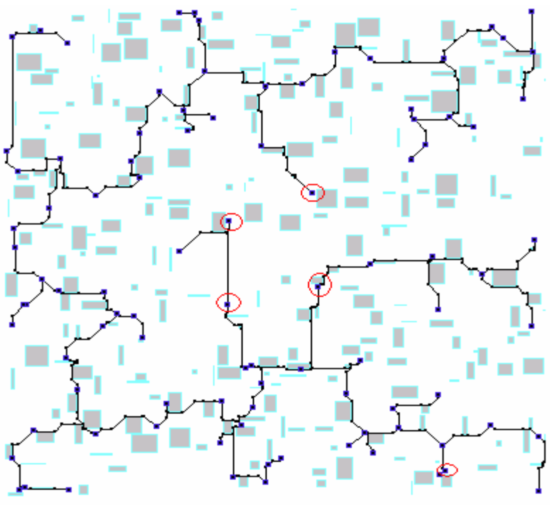

(f) Routing result (v =5)

Figure 7. Illustration of the ECO-aware routing tree.