Power-State-Aware Buffered Tree Construction

Iris Hui-Ru Jiang
Dept of Electronics Engineering
National Chiao Tung University
Hsinchu 30010, Taiwan
hrjiang@faculty.nctu.edu.tw

Ming-Hua Wu
Realtek Semiconductor Corp.
Hsinchu Science Park
Hsinchu 300, Taiwan
aqdest.ee94g@gmail.com

***Abstract*— Interconnect delay and low power are two of the main issues in nano technology. Buffer insertion during routing effectively reduces interconnect delay; power state management and multiple supply voltage significantly lower power consumption. However, buffering without considering power states in multiple supply voltage designs may cause the signal integrity problem. This paper first considers power states into buffered tree construction. Based on a hierarchical approach combined with dynamic programming, we can simultaneously minimize power, satisfy timing constraints and maintain signal integrity.**

I. Introduction

In nano technology, interconnect delay and power consumption are two of the main concerns. Interconnect delay can be reduced by buffer insertion, wire sizing, gate sizing and etc. [1], [2], [3], [4], [5]; among them, buffer insertion is an effective way. Due to limited routing resources at the post-layout stage, buffer insertion during routing has been widely adopted in the past decade.

On the other hand, low power has been extensively studied in literature. Multiple supply voltage (MSV) [6], [7] and power state management [8], [9] have large, and even huge, benefits on power [10]. In a multiple supply voltage design, each cell can be driven by one from several voltage levels. Although this method can reduce dynamic power quadratically, a signal going through differently leveled cells induces leakage currents and raises the integrity issue. To overcome them, an extra level converter is required if a lower voltage cell drives a higher one. Moreover, arbitrarily assigning voltage levels may make power/ground planning difficult. Therefore, we can cluster cells with the same supply source into a group, thus physically partitioning a design into voltage islands [11], [12], [13], [14].

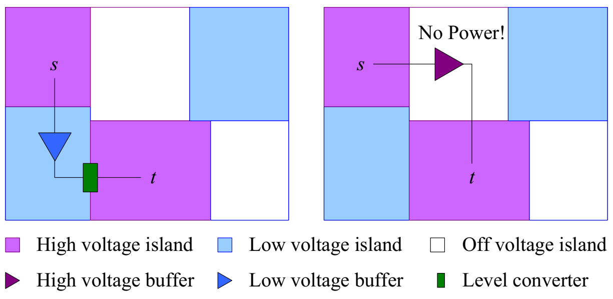


Fig. 1. Under a given power state, the right-hand sided path is infeasible

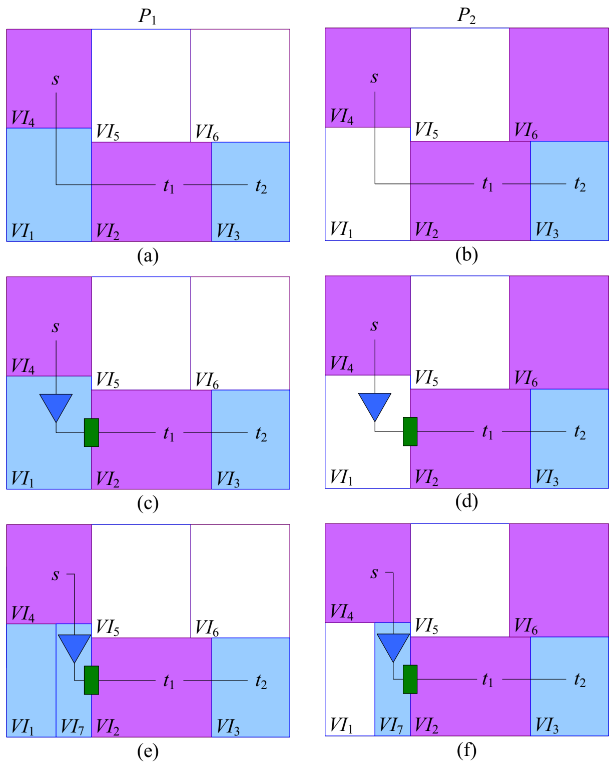


Fig. 2. A buffered tree should be feasible for all possible power states

In addition, circuits are not always running at the high performance mode, e.g., several voltage islands may idle sometimes, but still consume power. With power state management, we can adjust the supply voltages, and further turn off idle islands. However, in this case, we shall carefully do routing and buffer insertion. Fig. 1 shows an MSV design with two voltage islands turned off under a given power state. The right-hand sided path costs only one high voltage buffer, but it conducts an incorrect signal because of no power supply. Moreover, a design may dynamically switch among several power states. As demonstrated in Fig. 2, given a source s and two sinks t_1 and t_2, the net should work at two power states P_1 and P_2. Voltage islands VI_5 and VI_6 are turned off at P_1, while VI_1 and VI_5 are turned off at P_2. To maintain signal integrity for both states, the net is routed as Fig. 2(a) and 2(b). However, this long wire may violate timing constraints. Then, a buffer is inserted in VI_1; this

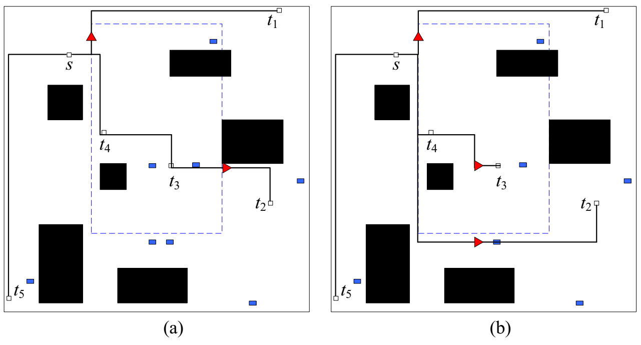


Fig. 3. The dotted block is the low voltage island and the small rectangles are available buffer locations (a) Sink t_2 might violate its timing constraint and incur a long transition (b) Sinks t_2 and t_3 can meet timing constraints for always

buffer solves timing violation at P_1 as Fig. 2(c) but fails at P_2 as Fig. 2(d) because of no power supply. If we can create a new voltage island VI_7 as Fig. 2(e) and 2(f), then the net will maintain signal integrity and satisfy timing constraints for both states.

Based on these observations, it is desirable to construct a buffered routing tree with power state consideration, such that the net is feasible for all possible power states. However, no existing works considered power states during routing and buffer insertion so far.

Several works handled simultaneous routing and buffer insertion for single supply voltage designs [2], [3], [5]; most of them used dynamic programming in a bottom-up fashion derived from [4]. Both [2] and [5] considered only two-pin nets, while [3] grew a multi-pin buffered tree under fix buffer locations. [3] constructed subtrees from sinks first, and then recursively merged subtrees with disjoint sinks and pruned inferior partial solutions until completing a tree. Furthermore, [13] and [14] extended buffered tree construction to dual supply voltage designs. [13] recursively visited children nodes from the source first, and then enumerated all possible solutions in a bottom-up fashion. However, [13] assumed the source was driven by high voltage, and forbade low voltage buffers to drive high ones. Thus, level converters were not used inside the routing tree, and only few level converters were needed right before high voltage sinks. [14] removed this impractical restriction and proposed an algorithm for a simplified dual voltage design with only one low voltage island. Because the low voltage island may be turned off sometimes, if the source and any sink are outside of it, [14] prohibited buffers from being placed inside. As shown in Fig. 3(a), the source s and three sinks t_1, t_2 and t_5 are outside the low voltage island; therefore, no buffer is inserted inside it. If sink t_3 is timing critical, this restriction may make it violating its timing constraint and incurring a long transition. Fig. 3(b) shows an alternative solution, where the timing of sink t_3 can be improved by an inserted buffer, and the timing of t_2 still can be well-maintained. Hence, this restriction is unnecessary, and considering only one low voltage island is somewhat over-simplified.

This paper first considers power states into buffered tree construction for dual supply voltage designs. The goal is to minimize total power consumption under timing constraints and to maintain signal integrity. We adopt a hierarchical approach combined with dynamic programming. First of all, we construct a local buffered tree within each voltage island. Secondly, we connect these local trees into a global one with power state consideration. If no feasible solution meets timing constraints, we will create new voltage islands around the existing ones. Power state diagrams are available in system level [8], [9], and voltage islands are planned at floorplanning. Therefore, we can combine them into a power state table that indicates each state with active islands. The newly created islands are expected to be small and few; their main purposes are to place buffers for timing issues. With careful creation, the overhead on power/verification can be small. Moreover, with the modified power states and voltage islands, we can negotiate with the system level or floorplanning for further improvement. In addition, our method can easily be extended to multiple supply voltage designs and sophisticated delay models. The experimental results show that our approach can not only minimize power, satisfy timing constraints, but also maintain signal integrity effectively and efficiently. On average, our algorithm can achieve 33.39% power reduction and speedup 16.31% runtimes.

II. Preliminaries and Problem Formulation

In this Section, we introduce delay and power models, and then give the problem formulation.

A. Delay and Power Models

Fig. 4 shows the circuit models of a wire, a buffer, and a level converter. Two types of buffers–low voltage one and high voltage one, and one type of level converter are used throughout this paper. TABLE I lists the parameter settings in our experiments. (In order to show the effectiveness of our method, we adequately adjust the parameters used in [13] so that buffers tend to be inserted.) The Elmore delay [15] of a wire D_w and that of a buffer D_b are:

$$D_w(L) = r_w L(\frac{1}{2}c_w L + C_l), \qquad D_b = D_i + R_b C_l,$$

where c_w is unit length capacitance, r_w is unit length resistance, L is wire length, C_l is the downstream capacitance, D_i is the intrinsic delay of a buffer, C_b is the input capacitance of a buffer, R_b is the output resistance of a buffer. The delay model can easily be extended to consider the inductive effect or to more sophisticated ones. A level converter is similarly modeled. The wire power consumption P_w is measured by energy per switch,

$$P_w(L) \quad = \quad \frac{1}{2}c_w L {V_{dd}}^2,$$

where V_{dd} is the supply voltage.

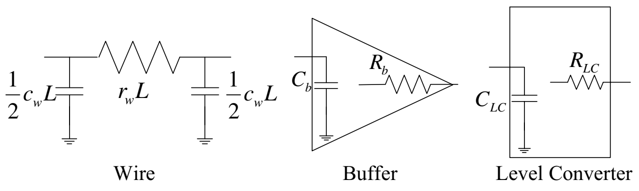


Fig. 4. The circuit models of a wire, a buffer and a level converter

TABLE I
PARAMETERS USED IN THIS PAPER

Model		Value
Wire		$c_w = 0.15\ fF/\mu m$, $r_w = 1.0\ \Omega/\mu m$
Buffer	High	$C_b = 3.4\ fF$, $R_b = 1.0\ k\Omega$, $D_i = 36.4\ ps$
	Low	$C_b = 3.4\ fF$, $R_b = 1.2\ k\Omega$, $D_i = 40.0\ ps$
Level converter		$C_{LC} = 3.4\ fF/\mu m$, $R_{LC} = 1.2\ k\Omega$, $D_{LC} = 100.0\ ps$

B. Problem Formulation

- **The Power-State-Aware Buffered Tree Problem**: Given a multi-pin net, blockages, voltage island planning, power states, and buffer and level converter libraries, construct a buffered routing tree with minimum power (and modify voltage island planning and power states if new voltage islands are created), such that timing constraints are satisfied.

In addition, low voltage buffers can only be inserted in low voltage islands; high voltage buffers and level converters can only be inserted in high voltage islands. Moreover, if no feasible solutions exist, new voltage islands are introduced and thus power states and voltage island planning should be modified accordingly. Please note that we consider on/off power states, free buffer locations, and dual supply voltages for a design.

III. THE PSA ALGORITHM

To solve the power-state-aware buffered tree problem, we propose a hierarchical approach combined with dynamic programming. Fig. 5 gives the overview of the PSA algorithm. First of all, sinks within each voltage island are connected to a local tree; we handle one island at a time. Secondly, local trees are connected to a global one with power state consideration. If no feasible solutions exist, new voltage islands are created at global tree construction.

A. Local Tree Construction

The procedure of local tree construction is listed in Fig. 6. First of all, the pseudo sink is found in line 1. The pseudo sink is an artificial sink representing all sinks within a voltage island, and also the root of a local tree. Secondly, grid lines are constructed in line 2. Thirdly, grid nodes are initialized in line 3. Finally, in lines 5–15, we iteratively propagate solutions from each sink and prune redundant solutions if necessary until finding a feasible solution. We use a priority queue Q_s and a working queue Q_w in these while loops. Q_w maintains the ordering of sinks according to the distances

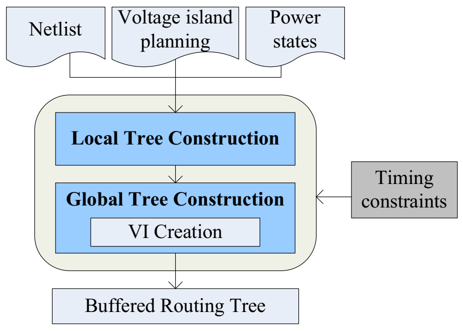


Fig. 5. The overview of the PSA algorithm

```
Algorithm: Local Tree Construction
Input:  Voltage island $VI_i$
        Timing constraints
Output: A local buffered routing tree
1.   Find the pseudo sink $T_i$ of $VI_i$
2.   Construct grid lines within $VI_i$
3.   Initialize grid nodes
4.   Push sinks within $VI_i$ into priority queue $Q_s$
5.   while $Q_s$ is not empty do
6.       Select sink $t_j$ with max. distance to $T_i$ from $Q_s$
7.       Push sink $t_j$ into queue $Q_w$
8.       while $Q_w$ is not empty and
                 no feasible solution is found do
9.           Select node $w$ from $Q_w$
10.          Propagate solution from $w$ to each neighbor $u$
11.          if a feasible solution is found then
12.              Update solution
13.          else
14.              Prune redundant solutions on $u$
15.          Push neighbor $u$ into $Q_w$
```

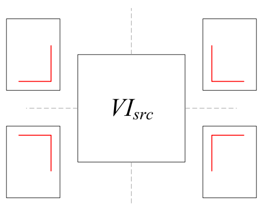
Fig. 6. The procedure of local tree construction

between sinks and the pseudo sink. Q_w records grid nodes which should propagate solutions to neighbors. We detail the procedure as follows.

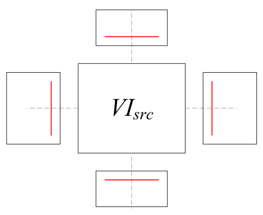
1) Finding the Pseudo Sink: Sinks within each voltage island are locally connected. Except the island of the source VI_{src}, the pseudo sink is used to guide the direction toward the source. We expect that the upstream of local trees can approach the source to reduce delays. Therefore, the nodes in proximity to the source within the current island could be the pseudo sink. If the current island is overlapped with the horizontal or vertical centerlines of VI_{src}, grid nodes on the side close to VI_{src} are selected (indicated by straight lines in Fig. 7(b)). Otherwise, candidates are the L-shaped lines shown in Fig. 7(a). Considering obstacle penalties, the pseudo sink is the candidate with the shortest distance to the source. The obstacle penalty is estimated by the formula derived by [16].

2) Grid Line Construction: Horizontal and vertical grid lines are constructed not only at sinks and the pseudo sink but also around blockages. As shown in Fig. 8, T is the pseudo sink, t_1, t_2 and t_3 are sinks, and vertical/horizontal lines are grid lines of a given voltage island. These grid lines are bounded by the voltage island for local tree construction to reduce the time complexity.

3) Grid Node Initialization: A seven-tuple $(R, C, rat, POW, HW, BV, PP)$ is used to represent a solution.



(a)



(b)

Fig. 7. L-shaped lines in (a) and straight lines in (b) indicate the set of pseudo sink candidates

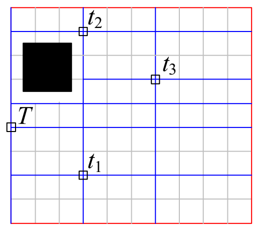


Fig. 8. The grid graph

Parameters are listed in TABLE II. Each grid node is initialized according to the type of the node.

- For being a sink, node i is initialized with a loading capacitance:
  $Solution(0, C_l, rat_i, POW_{C_l}, 0, BV_i, \{i\})$.
- Otherwise, node i is initialized with/without a buffer:
  $Solution(R_b, C_b, \infty, POW_b, HW_b, BV_i, \{i\})$,
  $Solution(0, 0, \infty, 0, 0, BV_i, \{i\})$.

Each grid node is allowed to insert a buffer. (If restricted buffer locations are considered, a grid node on an infeasible buffer location is initialized only without buffer.) In addition, if a pseudo sink is located at a high voltage island, it is also initialized with a level converter, because it would be driven by low voltage cells.

4) Solution Propagation: For local tree construction, solutions are propagated from sinks to the pseudo sink. Solutions of a grid node w are propagated to its neighbors until reaching the pseudo sink or the partially routed tree. For a neighbor u, the current solutions of u and those of w are combined to a new one for u. If a solution is propagated to a routed grid node and the required arrival time is met, the result should be updated along to the root, i.e., the pseudo sink. Otherwise, solutions keep being propagated to other grid nodes to form a feasible solution. We detail how to generate the rat of a new solution as follows. Other parameters can be obtained in a similar way.

$$\begin{aligned} rat_1 &= rat_u - R_u(c_w L + C_v), \\ rat_2 &= rat_v - D_i - R_u(C_x + c_w L + C_v) \\ &\quad - r_w L(\frac{1}{2}c_w L + C_v), \\ rat_{new} &= \min\{rat_1, rat_2\}, \end{aligned}$$

where C_x is loading from other branches, C_v is loading of v. If node u is unrouted, $rat_u = \infty$ and $C_x = 0$; if node u is with a buffer, $R_u \neq 0$; if node u is without a buffer, $R_u = 0$.

TABLE II
PARAMETERS USED IN A SOLUTION

Parameter	Description	Parameter	Description
R	Driving resistance	HW	Hardware cost
C	Loading capacitance	BV	Voltage level
rat	Required arrival time	PP	Propagated path
POW	Power consumption		

5) Redundancy Pruning: During solution propagation, we only store some prior solutions to save memory space. Therefore, if solution A has smaller required arrival time, and larger power and capacitance than solution B, then A is pruned. In addition, in a high voltage island, a solution with level converter at the pseudo sink is kept for maintaining signal integrity in the global buffered tree.

As demonstrated in Fig. 9, we summarize local tree construction with the instance given in Fig. 8. First of all, we begin with sink t_3 with the maximum distance to the pseudo sink T. A solution with capacitance loading of t_3 is propagated to its neighbor grid nodes as shown in Fig. 9(a), then solutions of these neighbors are propagated to their neighbors as in Fig. 9(b). After two more propagation steps, we have Fig. 9(c). (The dots indicate the progress of propagation.) The propagation is repeated until a feasible solution from t_3 to T is found. Solutions are then updated as in Fig. 9(d). Secondly, sink t_2 is selected; as depicted in Fig. 9(e), solutions are propagated in the similar manner with sink t_3 until the partially routed tree is reached. If the required arrival time of the pseudo sink is satisfied, solutions are updated to T. Fig. 9(f) is the resulting tree connecting t_2 and t_3. Although not presented here, t_1 is finally connected.

B. Global Tree Construction

As listed in Fig. 10, global tree construction partitions voltage islands into power state groups in line 1, connects the pseudo sinks of local trees according to their power states in lines 2–7 (lines 4–7 are similar to local tree construction), and creates new voltage islands if necessary in lines 8–10. During global tree construction, the power state table is taken into account for signal integrity. In addition, new voltage islands and level converters should properly be added. Isolation cells are not modeled here, but it can be easily extended.

1) Partitioning Power State Groups: Voltage islands are partitioned into groups according to their power states. Because voltage islands may be turned off, two local trees can be connected by pure wiring or through buffers placed within voltage islands that are turned on at the same power states, at compatible power states, or always. As shown in Fig. 11, the power state table has three states. VI_2 and VI_4 are active at all power states, viewed as an always-on group G_0. G_1 contains VI_5 and VI_6, active only at P_1; G_2 contains VI_1, active at P_2 and P_3; G_3 contains VI_3, active only at P_2. In addition, G_2 is G_3's compatible group, but G_3 is not G_2's. Please note that the number of power state groups is bounded by the number of voltage islands, not exponential with that

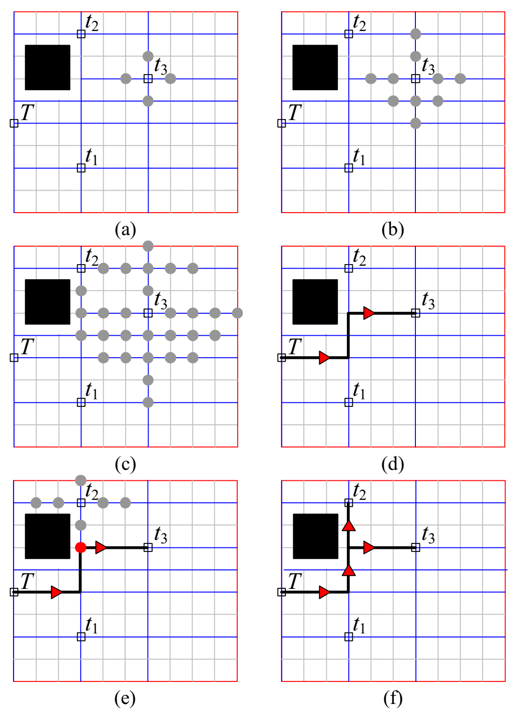


Fig. 9. (a)–(c) Solution propagation from sink t_3 to the pseudo sink T (d) A feasible solution connecting t_3 to T (e) Solution propagation from sink t_2 to the partially routed tree (f) The resulting buffered tree for t_2 and t_3

```
Algorithm: Global Tree Construction
Input:   Power states
         Timing constraints
Output: A global buffered routing tree
1.   Partition power state groups $G$
2.   foreach group $G_j$ in $G$ do
3.       foreach voltage island $VI_k$ in $G_j$ do
4.           Construct grid lines
5.           Initialize grid nodes
6.           Propagate solution from the pseudo sink $T_k$
7.           Prune redundant solutions
8.           if no feasible solutions then
9.               Create new voltage islands
10.              Update solutions and power states
```

Fig. 10. The procedure of global tree construction

of the power states. The extreme case is when each pair of voltage islands have different active behavior, each voltage island forms a power state group.

2) Solution Propagation: During global tree construction, solution propagation is in the same manner with local one. Group by group, solutions are propagated from the pseudo sink of each voltage island toward a partially routed global tree. The node where propagation occurs is located in the voltage island in the always-on group, in the same power state group, or in a compatible power state group. For example, the case in Fig. 12 has the power state table given in Fig. 11. Dashed triangles represent local trees rooted at the pseudo sinks, and dots indicate propagations. As shown in Fig. 12(a), the always-on group, VI_2 and VI_4, first forms a partially routed global tree. Solutions of the pseudo sink of VI_5 are then propagated to the partially routed global tree of

Power state table	Power state group
P_1: VI_2 VI_4 VI_5 VI_6	G_0: VI_2 VI_4
P_2: VI_1 VI_2 VI_3 VI_4	G_1: VI_5 VI_6
P_3: VI_1 VI_2 VI_4	G_2: VI_1
	G_3: VI_3

Fig. 11. The power state table vs. the power state groups

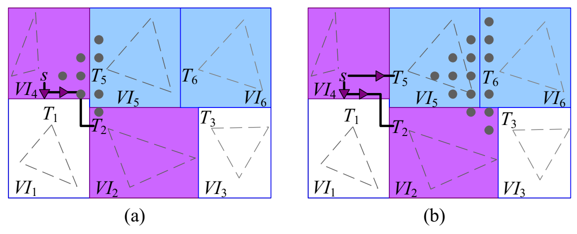


Fig. 12. In global tree construction, solutions are propagated group by group

the always-on group. In Fig. 12(b), solutions of the pseudo sink of VI_6 can be propagated to the partial tree of the always-on group, or to VI_5 in the same group. Similarly, solutions of the pseudo sink of VI_1 are then propagated. Finally, solutions of the pseudo sink of VI_3 can be propagated to the partial tree of the always-on group, or to VI_1 in a compatible group.

To maintain signal integrity, solution propagation from grid node w to its neighbor u should consider their voltage levels. In other words, level converters are only inserted at grid nodes in high voltage islands, receiving signal from low voltage nodes.

3) New Voltage Island Creation: If no feasible solutions can be found after line 7, we shall create a new voltage island and update solutions. In addition, the power state table and groups are modified accordingly. In order to reduce the difficulty of power/ground planning and to minimize the modification of voltage island planning, we prefer to create new voltage islands at the peripheral of original ones. Therefore, for an infeasible solution, some buffers are temporarily allowed to be inserted at the peripheral of islands in incompatible groups to satisfy the timing constraints. New voltage islands are then created to cover these buffers.

IV. Experimental Results

We implemented the PSA algorithm in C++ language on a 2.4GHz AMD Opteron$^{\text{TM}}$ platform with 4GB memory.

We randomly created six cases listed in TABLE III. The unit grid size is 0.5mm*0.5mm. The statistics of each case is listed in the third column, including the number of sinks, the number of blockages, the number of voltage islands. We compare the delay, power, hardware cost (including high voltage buffers, low voltage buffers, level converters) when power state management is considered or not. The required arrival times at sinks are considered as timing constraints and are set according to presimulation results. Without considering power state management, we cannot turn off any islands to ensure the signal integrity. At this time, the system has only one power state, all voltage islands are always turned on (AlwaysON). On the contrary,

TABLE III

THE COMPARISONS ON DELAY, POWER, HARDWARE COSTS WITHOUT AND WITH POWER STATES CONSIDERATION

Case	Dimenion	#sink/#blk/#VI	AlwaysON					PSA				
			#PS	Delay (ns)	Power (pJ)	HW (#BufH/#BufL/#LC)	Time (s)	#PS	Delay (ns)	Power (pJ)	HW (#BufH/#BufL/#LC)	Time (s)
Case1	40*40	20 / 6 / 6	1	8.55	12.84	85 (38 / 45 / 2)	1.38	3	8.52	7.15	83 (36 / 44 / 3)	1.12
Case2	50*50	26 / 6 / 6	1	9.90	18.21	124 (55 / 67 / 2)	3.17	3	9.74	9.55	123 (52 / 44 / 3)	2.53
Case3	50*50	37 / 8 / 6	1	9.95	23.23	149 (74 / 72 / 3)	5.58	3	9.72	12.80	153 (73 / 77 / 3)	4.66
Case4	100*100	56 / 6 / 6	1	20.59	57.35	356 (197 / 158 / 1)	41.21	2	20.68	48.83	354 (196 / 158 / 0)	35.58
Case5	200*200	101 / 6 / 5	1	58.46	150.84	907 (513 / 392 / 2)	544.43	2	58.46	106.19	906 (513 / 391 / 2)	468.83
Case6	200*250	200 / 5 / 5	1	52.44	236.88	1505 (840 / 663 / 2)	2523.64	2	52.44	191.61	1488 (833 / 654 / 1)	2150.56
Imp.	–	–	–	–	–	–	–	–	0.62%	33.39%	0.38% (– / – / –)	16.31%

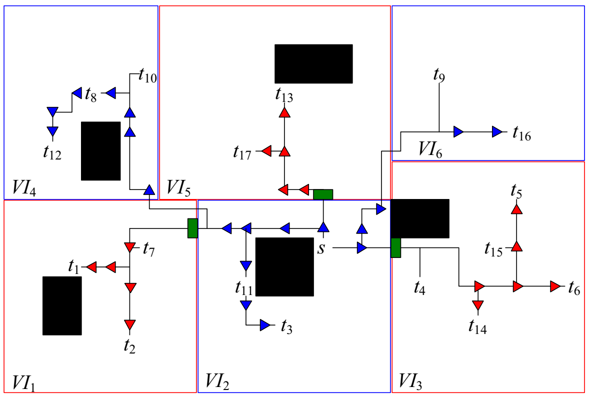


Fig. 13. The buffered tree of a case of 17 sinks

Power state table	Power state group
P_1: VI_1 VI_2 VI_3 VI_5	G_0: VI_2 VI_3
P_2: VI_2 VI_3 VI_4 VI_6	G_1: VI_1 VI_5
	G_2: VI_4 VI_6

Fig. 14. The power states of the case in Fig. 13

with power state management, the PSA algorithm achieves average 33.39%, 16.31% improvement on power, runtimes, respectively without sacrificing delay and hardware costs; the improvement is calculated by (AlwaysON-PSA)/AlwaysON. Without loss of generality, we measure the power consumption for PSA by averaging the power consumed at each power state. The accurate power consumption can be computed by extracting the operating time slots of each power state from simulations. Moreover, the hierarchical approach not only helps on constructing buffered trees that can operate correctly but also improves the runtimes.

Fig. 13 shows the buffered tree of a case of 17 sinks, where VI_1, VI_3 and VI_5 are high voltage islands, and VI_2, VI_4 and VI_6 are low ones. The source s is at low voltage island VI_2, and level converters (indicated by small rectangles) are properly inserted at the boundaris of high voltage islands. Its power states are detailed in Fig. 14. It can be seen that signal integrity is maintained well for both power states.

V. CONCLUSION

This paper first considered power states into buffered tree construction and proposed a hierarchical approach combined with dynamic programming. The results showed our method is very promising.

VI. ACKNOWLEDGEMENT

This work was partially supported by National Science Council of Taiwan under grant no. NSC 97-2220-E-009-033.

REFERENCES

[1] J. Lillis, C.-K. Cheng, and T. T. Y. Lin, "Optimal and efficient buffer insertion and wire sizing," in *Proc. IEEE Custom Integrated Circuits Conf. (CICC'95)*, 1995, pp. 259–262.

[2] M. Lai and D. F. Wong, "Maze routing with buffer insertion and wiresizing," *IEEE Trans. Computer-Aided Design*, vol. 21, no. 10, pp. 1205–1209, Oct. 2002.

[3] J. Cong and X. Yuan, "Routing tree construction under fixed buffer locations," in *Proc. ACM/IEEE Design Automation Conf. (DAC'00)*, June 2000, pp. 379–384.

[4] L. P. P. P. van Ginneken, "Buffer placement in distributed rc-tree networks for minimal elmore delay," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS'90)*, vol. 2, May 1990, pp. 865–868.

[5] H. Zhou, M. D. Wong, I.-M. Liu, and A. Aziz, "Simultaneous routing and buffer insertion with restrictions on buffer locations," *IEEE Trans. Computer-Aided Design*, vol. 19, no. 7, pp. 819–824, 2000.

[6] K. Usami and M. Horowitz, "Clustered voltage scaling technique for low-power design," in *Proc. ACM Int. Symp. on Low Power Design (ISLPED'95)*, 1995, pp. 3–8.

[7] H.-Y. Liu, W.-P. Lee, and Y.-W. Chang, "A provably good approximation algorithm for power optimization using multiple supply voltages," in *Proc. ACM/IEEE Design Automation Conf. (DAC'07)*, June 2007, pp. 887–890.

[8] R. A. Bergamaschi and Y. W. Jiang, "State-based power analysis for systems-on-chip," in *Proc. ACM/IEEE Design Automation Conf. (DAC'03)*, June 2003, pp. 638–641.

[9] A. Naveh, E. Rotem, A. Mendelson, S. Gochman, R. Chabukswar, K. Krishnan, and A. Kumar, "Power and thermal management in the intel core duo processor," *Intel Technol. J.*, vol. 10, no. 2, May 2006.

[10] A. Eliopoulos, P. Chen, and Q. Wang. (2007) How to architect, design, implement, and verify low-power digital integrated circuits. EDA DesignLine.

[11] D. E. Lackey, P. S. Zuchowski, T. R. Bednar, D. W. Stout, S. W. Gould, and J. M. Cohn, "Managing power and performance for system-on-chip designs using voltage islands," in *Proc. IEEE/ACM Int. Conf. on Computer-aided Design (ICCAD'02)*, Nov. 2002, pp. 195–202.

[12] W.-P. Lee, H.-Y. Liu, and Y.-W. Chang, "Voltage island aware floorplanning for power and timing optimization," in *Proc. IEEE/ACM Int. Conf. on Computer-aided Design (ICCAD'06)*, Nov. 2006, pp. 389–394.

[13] K. H. Tam and L. He, "Power optimal dual-vdd buffered tree considering buffer stations and blockages," in *Proc. ACM/IEEE Design Automation Conf. (DAC'05)*, June 2005, pp. 497–502.

[14] B. Tseng and H.-M. Chen, "Blockage and voltage island-aware dual-vdd buffered tree construction under fixed buffer locations," in *Proc. ACM Int. Symp. on Physical Design (ISPD'08)*, Apr. 2008, pp. 23–30.

[15] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *J. Applied Phys.*, vol. 19, pp. 55–63, Jan. 1948.

[16] 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. ACM/IEEE Asia and South Pacific Design Automation Conf. (ASP-DAC'07)*, Jan. 2007, pp. 262–267.