Dual-V_{dd} Buffer Insertion for Power Reduction

King Ho Tam, Yu Hu, Lei He, Tom Tong Jing, and Xinyi Zhang

***Abstract*—This paper presents the first in-depth study on dual-V_{dd} buffer insertion for power minimization under delay constraint. Compared with delay-optimal single V_{dd} buffer insertion, the dual-V_{dd} buffer insertion reduces power by 16%. Such power reduction increases when the delay specification is relaxed. Whereas the van Ginneken algorithm can be extended to handle the new problem formulation optimally, its time complexity increases from quadratic time ($O(|B|n^2)$) to pseudopolynomial time ($O(|B|n^3c_{\max}^2\log(nc_{\max}))$, where $|B|$ is the size of buffer library, n is the number of buffer stations, and $c_{\max}$ is proportional to the number of all possible subtrees of the net. To improve the time complexity, we propose an approximation technique by sampling subsolutions (i.e., options) and apply predictive min-delay and prebuffer slack pruning rules from a related work. Experiments show that sampling is most effective to reduce run time, whereas the two pruning rules further improve efficiency and accuracy loss due to sampling. We show that our proposed algorithm has linear time complexity with respect to the tree size. It runs over 1000 times faster at a cost of less than 2% delay and power increase over the extended van Ginneken algorithm.**

***Index Terms*—Buffer insertion, delay, dual-V_{dd}, low power.**

I. Introduction

Aggressive scaling of very large scale integration circuits makes interconnects as the performance bottleneck, and buffer insertion is used extensively to reduce the interconnect delay at the expense of more power dissipation. Van Ginneken [1] presented a dynamic programming-based algorithm for the delay-optimal buffer insertion problem. Given a routing tree, partial solutions called options at each tree node are constructed and propagated in a bottom-up fashion. When the optimal solution is identified at the root node, a top–down back trace is performed to get the optimal buffer assignment. Following this dynamic programming framework, various delay optimization buffer insertion algorithms have been developed. Alpert and Devgan [2] proposed wire segmenting with buffer insertion. Lillis and Cheng [3] studied repeater insertion in a multisource net. Alpert *et al.* [4] considered noise and delay optimization simultaneously. Shi and Li [5] presented an efficient algorithm with $O(n\log^2 n)$ time complexity, where n is the number of possible buffer positions. Buffer insertion with variations on wire length and fabrication was considered in [6] and [7], respectively.

Buffer insertion may increase power dissipation if an excessive number of buffers are used. A power-optimal buffer insertion algorithm was proposed in [8], achieving minimal power for a given target delay based on the aforementioned dynamic programming framework. The time complexity is $O(|B|n^3c_{\max}^2\log(nc_{\max}))$, where $|B|$ is the size of buffer library, n is the number of buffer stations, and $c_{\max}$ is the number of different capacitance values among all options, which, in turn, is proportional to the number of all possible subtrees of the net. To reduce run time for large nets due to the increase of uncontrolled options, Rao *et al.* [9] assumed a large buffer library with near continuous buffer sizes and solved the power-optimal buffer insertion problem with five times speedup over [8] with a small loss of delay and power optimality. However, single V_{dd} was assumed in all existing works for power-optimal buffer insertion. Programmable dual-V_{dd} buffers have been used to reduce field-programmable gate array (FPGA) power [10], [11]. Other approaches on FPGA to budget time among dual-V_{dd} buffers [12] and the application of dual-V_{dd} buffer circuitry, using a single power supply network [13], have also been proposed. As buffers are preplaced in FPGAs, the dual-V_{dd} buffer routing can be realized as dual-V_{dd} assignment for buffers.

However, the power-optimal dual-V_{dd} buffer insertion problem in application specific integrated circuit designs is more complicated because the flexible buffer locations increase the solution space substantially. This problem has not been studied in existing work. Compared with the single V_{dd} buffer insertion problem, the dual-V_{dd} version introduces voltage as an extra dimension to the solution space that destroys the true polynomial time complexity, making a straightforward extension of the classical buffer insertion algorithm that is inapplicable in practice.

The major contributions of this paper are as follows. We present the first in-depth study on dual-V_{dd} buffer insertion for power minimization under delay constraint. Furthermore, to cope with the substantial increase of the time complexity, we present an approximation technique to sample options, which is coupled with our extension to predictive min-delay pruning (PMP) and prebuffer slack pruning (PSP) rules that are originally taken from [5] and [14]. Sampling is most effective to reduce run time, and the two pruning rules are needed to further improve the efficiency and accuracy loss due to sampling. Our speed-up techniques reduce the algorithm time complexity from pseudopolynomial time [8] to linear time with respect to the tree size. Experimental results show that dual-V_{dd} buffer insertion reduces power by 16% compared with the delay-optimal single V_{dd} buffer insertion. Such power reduction increases when the delay specification is relaxed. As a result of the speed-up techniques, we achieve a combined speedup of more than 1000 times over the exact power-optimal buffer insertion algorithm (extended from [8]) at the expense of less than 2% delay and power increase, respectively.

The remainder of this paper is organized as follows. Section II presents our dual-V_{dd} modeling and the problem formulation. Section III proposes the baseline algorithm and the related speed-up techniques. We conclude in Section IV. An extended abstract of this paper without using V_{dd}-level converters and speed-up techniques, such as adaptive 3-D sampling and predictive pruning, was presented in [15].

II. Preliminaries and Problem Formulation

A. Delay, Slew Rate, and Power Model

We use a distributed Elmore delay model as in [1]–[5] and recent works [16]–[19]. We use a simple power model similar to that in [15].

B. Dual-V_{dd} Circuits

Dual-V_{dd} buffering uses both high and low V_{dd} buffers in interconnect synthesis. Designs using low V_{dd} buffers consume less buffer

Manuscript received July 4, 2007; revised November 5, 2007 and February 17, 2008. This work was supported in part by the National Science Foundation under CAREER award CCR-0401682, by the Semiconductor Research Corporation under Grant 1100, by Analog Devices, Fujitsu Laboratories of America, Intel, and LSI Logic through a University of California (UC) MICRO grant, and by IBM. The work at the University of California, Los Angeles was supported in part by the National Science Foundation under CAREER Award CCR-0401682 and in part by Fujitsu Laboratories of America and Intel through a University of California MICRO grant. This paper was recommended by Associate Editor M. Poncino.

The authors are with the Electrical Engineering Department, University of California, Los Angeles, CA 90095 USA (e-mail: ktam@ee.ucla.edu; hu@ee.ucla.edu; lhe@ee.ucla.edu; tomjing@ee.ucla.edu; zxy@ee.ucla.edu).

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.2008.925784

power E_b and interconnect power. By applying this technique to noncritical paths, we reduce power without worsening the delay of the overall interconnect tree.

C. Problem Formulation

We assume that the loading capacitance and the required arrival time (RAT) q_n^s are given at all sink nodes (terminals and pins) n_s. We assume that the driver resistance at the source node $n_{\rm src}$ is given, and all types of buffers can be placed only at the buffer candidate nodes n_b^k. We use the RAT at the source $n_{\rm src}$ to measure delay performance. Our goal is to minimize the power of the interconnect subject to the RAT constraint at the source $n_{\rm src}$.

Definition 1: The RAT q_n at node n is defined as

$$q_n = \min_{n_s \forall s} (q_n^s - d(n_s, n))$$

where $d(n_s, n)$ is the delay from the sink node n_s to node n.

Dual-V_{dd} buffer insertion (dBIS): Given an interconnect fanout tree, which consists of a source node $n_{\rm src}$, sink nodes n_s, Steiner nodes n_p, candidate buffer nodes n_b, and the connection topology among them, the dBIS problem is to find a buffer placement, size assignment, and V_{dd} level assignment solution such that the RAT $q_n^{\rm src}$ at the source $n_{\rm src}$ is met and the power consumed by the interconnect tree is minimized, whereas the slew rate at every input of the buffers and the sinks n_s are upper bounded by $\hat{s}$.

III. Dual-V_{dd} Buffer Insertion Algorithm

A. Baseline Algorithm

Power-optimal solutions, namely, *options* in this paper, are constructed from partial solutions to the subtrees, which are defined hereafter.

Definition 2: An option Φ_n at the node n refers to the buffer placement, size, and V_{dd} assignment for the subtree T_n rooted at node n. To perform delay and power reduction, the option is represented as a four-tuple $(c_n, p_n, q_n, \theta_n)$, where c_n is the downstream capacitance at node n, p_n is the total power of T_n, q_n is the RAT at node n, and θ_n signifies whether there exists any high V_{dd} buffer at the downstream. The option with the smallest power $p_n^{\rm src}$ at the source node $n_{\rm src}$ is the power-optimal solution.

Our algorithm is based on [8] with a few improvements. We add support for dual-V_{dd} buffer insertion, which inserts level converters where necessary. To facilitate explanation, we define the concept of option dominance here.

Definition 3: An option $\Phi_1 = (c_1, p_1, q_1, \theta_1)$ dominates another option $\Phi_2 = (c_2, p_2, q_2, \theta_2)$ if $c_1 \le c_2$, $p_1 \le p_2$, and $q_1 \ge q_2$.

We enhance the dynamic programming framework in [8] to accommodate the introduction of dual-V_{dd} buffers, which is summarized as enhanced dynamic programming (EDP) algorithm in Table I. We use the same notation as in Definition 2 to denote options Φ and their components. Moreover, we use c_b^k, E_b^k, V_b^k, and $d_b^k(c_{\rm load})$ to denote the input capacitance, the power, the V_{dd} level, and the delay with output load $c_{\rm load}$ of the buffer b_k, respectively. $d_{n,v}$ and $E_{n,v}(V)$ are the delay and the power of the interconnect between nodes n and v operating at voltage V, respectively. The set of available buffers $Set(B)$ contains both low and high V_{dd} buffers. We first call the algorithm EDP (line 2 in Table I) at the source node $n_{\rm src}$, which recursively visits the children nodes and enumerates all possible options in a bottom-up manner until the entire interconnect tree $T_n^{\rm src}$ is traversed.

TABLE I
EDP

Algorithm: $EDP(T_n, Set(B))$
0. $Set(\Phi_n) = (c_n^s, 0, q_n^s, false)$ if n is a low V_{dd} sink
$(c_n^s, 0, q_n^s, true), (c_n^c, 0, q_n^c, false)$ if n is a high V_{dd} sink
else $(0, 0, \infty, false)$
1. **for each** child v of n
2. $Set(\Phi_v) = EDP(T_v, Set(B))$
3. $Set(\Phi_{temp}) = Set(\Phi_n)$
4. $Set(\Phi_n) = \emptyset$
5. **for each** $\Phi_i \in Set(\Phi_v)$
6. **for each** $\Phi_t \in Set(\Phi_{temp})$
7. **for each** buffer $b_k \in Set(B)$
/* also contains the no buffer option ϕ */
8. **if** $b_k = \phi$
9. V_n = V_H if θ_i or θ_t is true, else V_L
10. $\Phi_{new} = (c_i + c_t, p_i + p_t + E_{n,v}(V_n),$
$\min(q_t, q_i - d_{n,v}), \theta_i$ or $\theta_t)$
11. **else if** i. V_b^k is high; or
ii. V_b^k is low and θ_i is false
12. $\Phi_{new} = (c_b, p_i + p_t + E_{n,v}(V_b^k) + E_b^k,$
$\min(q_t, q_i - d_{n,v} - r_b^k(c_i + c_{n,v}),$
θ_t or $(if\ V_b^k = V_H))$
13. **else**
/* consider the option for level converter insertion */
14. $\Phi_{new} = (c_{LC}, p_i + p_t + E_{n,v}(V_b^k) + E_{LC},$
$\min(q_t, q_i - d_{n,v} - r_{LC}(c_i + c_{n,v}), false)$
15. **if** i. slew rate violation at down-stream buffers; or
ii. Φ_{new} *dominated* by any $\Phi_z \in Set(\Phi_n)$
16. drop Φ_{new}
17. **else**
18. remove all $\Phi_z \in Set(\Phi_n)$ dominated by Φ_{new}
19. $Set(\Phi_n) = Set(\Phi_n) \cup \{\Phi_{new}\}$
20. return $Set(\Phi_n)$

TABLE II
Test-Case Characteristics

net	Node#	Sink#	High V_{dd} sink#
s1	86	19	10
s2	102	29	15
s3	142	49	27
s4	226	99	47
s5	375	199	105
s6	515	299	147
s7	784	499	238
s8	1054	699	340
s9	1188	799	389

There are several new features in our EDP algorithm to support the insertion of dual-V_{dd} buffers with potential level converters. Our implementation considers potential level converters, as shown in line 0 in Table I. Lines 10 and 12 in Table I produce the new options $\Phi_{\rm new}$ for the cases of no buffer insertion and buffer b_k insertion, respectively, between nodes n and v. In the case of no buffer insertion, we set V to either V_H for high V_{dd} or V_L for low V_{dd} in line 9 in Table I, according to the downstream high V_{dd} buffer indicators θ_i and θ_j, and use V to update the power consumed by the interconnect (line 10 in Table I). Note that, when $\theta =$ false (i.e., there are no high V_{dd} buffers in the downstream), only the low V_{dd} option has to be created because the high V_{dd} counterpart is always inferior. In the case of buffer insertion, we simply add $E_{n,v}(V_b^k)$, according to the operational voltage of buffer b_k to $p_{\rm new}$, and update θ accordingly. To consider level converters, we propose the following strategy. If high V_{dd} buffers drive low V_{dd} buffers (as shown in line 11 in Table I), level converters are not needed. Otherwise (as shown in line 13 in Table I), a level converter is inserted to drive downstream buffers with high V_{dd}. This algorithm theoretically allows level converters inserted at nonsink nodes in the buffered tree, but this never happens in any buffering solution in our experimental experience.

TABLE III
EXPERIMENTAL RESULTS FOR SINGLE AND DUAL-V_{dd} BUFFER INSERTIONS

	runtime (s)		RAT^* (ps)	power(fJ)@RAT^*		power(fJ)@105% RAT^*	
net	BIS	*dBIS*	BIS/*dBIS*	BIS	*dBIS* (power reduction %)	BIS	*dBIS* (power reduction %)
s1	176	176	-1444.99	15967.6	13075.5 (18%)	13794.2	10740.4 (22%)
s2	188	276	-1600.45	17010.5	14521.1 (14%)	15833	12179.8 (23%)
s3	1602	2356	-2221	24654.2	20411.2 (16%)	21038.2	17244.4 (18%)
s4	6547	>10000	-1808.54	34537.3	29641.7 (15%)	30562.1	16684.3 (45%)
average				23042.4	19412.4 (16%)	20306.9	14212.2 (30%)

B. Experimental Results for Baseline Algorithm

We generate nine test cases s1–s9 by randomly placing source and sinks in a 1×1 cm box. We use the GeoSteiner package [20] to generate the topologies of the test cases. We also break interconnect between nodes longer than 500 μm by inserting two-degree nodes. The V_{dd} types of level converters under each sink are set randomly. The characteristics of our test cases are shown in Table II.

In the first experiment, we assume that every nonsink node is a candidate buffer node. We set the RAT at all sinks to zero and that at the source to RAT*, which is the optimal delay found by the baseline buffer insertion algorithm [1]. The slew rate upper bound $\hat{s}$ is set to 100 ps. We have made buffers by using an inverter cascaded with another inverter that is four times larger. There are six types of buffers (high and low V_{dd} buffers of 16×, 32×, and 64×) in our buffer library.

We conduct experiments to show the effectiveness of power reduction due to dual-V_{dd} buffer insertion. Without influencing solution quality, we compare single and dual-V_{dd} buffer insertions in Table III without employing any advanced pruning rules shown in Table I. The experiments are on cases s1 to s4 only, whereas the rest cannot finish in a reasonable run time.[1] In Table III, RAT* is the maximum achievable RAT at the source. The percentages in the brackets show the relative power change from BIS (single V_{dd} buffer insertion) to dBIS. On average, dBIS reduces power by 16% compared with BIS. When we relax the RAT at the source to 105% of RAT*, dBIS saves 30% of power over BIS.

To handle large nets such as s5–s9 efficiently, we propose effective speed-up techniques for the dBIS problem as follows.

C. Speedup

1) Three-Dimensional Sampling: The 2-D sampling *under the same capacitance* has shown speedup [15]. However, the run time in our experiments for large test cases (those with over 200 buffer stations) is still very long. The number of distinct capacitance values among Φ grows rapidly with the tree size; therefore, the number of solutions is still growing at a tremendous rate. This observation indicates a need to extend sampling to 3-D, where capacitance is included as the third dimension. As a result, we sample a 3-D space formed by the triplet dimension of power, delay, and capacitance.

The idea is to pick only a certain number of options among all uniformly over the power-delay-capacitance space for upstream propagation. Fig. 1(a) shows a presample option set and Fig.1(b) shows an after-sample option set. Each black dot corresponds to an option. We divide each side of the bounding box of all options into equal segments such that the entire 3-D domain is superposed by cubic grids. For each grid cube shown in Fig. 1(a), we retain only one option if there is any and obtain the sampled nondominated option set shown in Fig. 1(b).

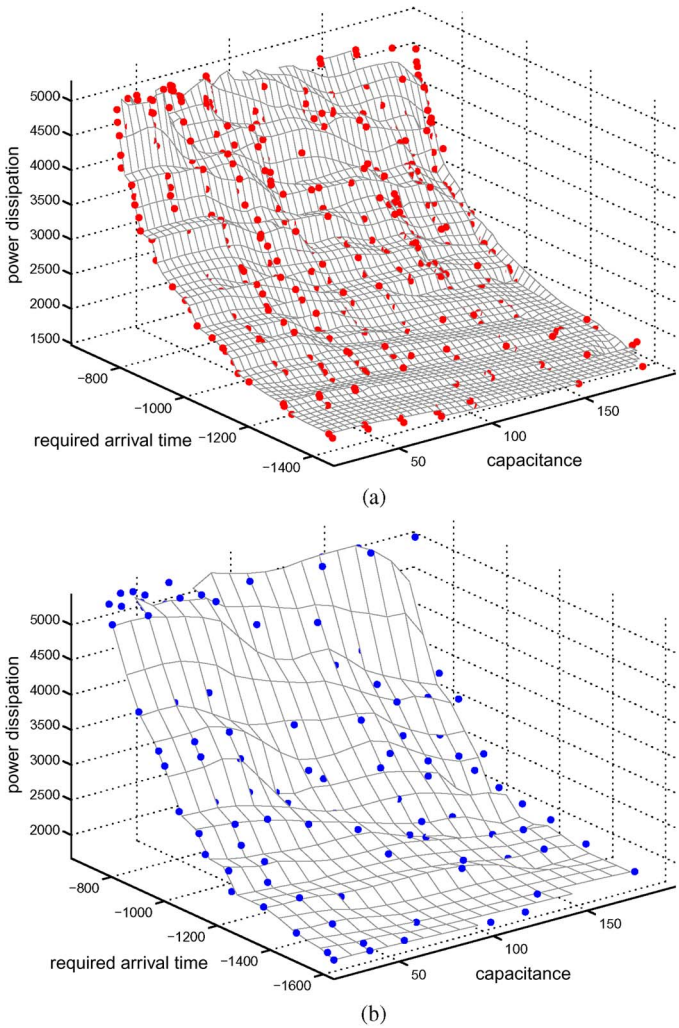

Fig. 1. Three-dimensional sampling for nonredundant options. (a) Before sampling. (b) After sampling.

Both the density of the sampling grid and the frequency of occurrence of the sampling candidate may affect the accuracy of the solutions. We study the effects of 3-D sampling, using uniform and adaptive grid sizes. Let N be the number of grids along the span of each dimension. For *uniform* grid size 3-D sampling, the whole space is divided evenly into N-by-N-by-N cubes. In the *adaptive* sampling, we discretize the cubic grid only if there exists any pair of options that are more than $1/N \cdot \delta$ apart in any dimension within the grid, where δ is the overall span of values in one dimension. This

[1]A straightforward extension of [8] destroys the nice property of option ordering. In [8], options are generated in increasing order of capacitance, making it possible to efficiently merge option sets during propagation. Here, we have voltage as an extra dimension. Therefore, the complexity of merging options and option growth quickly go out of control even for medium-sized test cases like s4.

TABLE IV
PERFORMANCE COMPARISON BETWEEN UNIFORM GRID SIZE AND ADAPTIVE GRID SIZE 3-D SAMPLING

net	RAT^* (ps)	RAT (ps)				power(fJ)@RAT			
		NS	US (*%)	ADS1 (*%)	ADS2 (*%)	NS	US (**%)	ADS1 (**%)	ADS2 (**%)
s1	-1700	-1699.63	-1697.29 (-0.18%)	-1694.61 (-0.30%)	-1698.65 (-0.06%)	8061.85	8309.69 (3.07%)	8197.63 (1.68%)	8085.42 (0.29%)
s2	-1900	-1894.25	-1894.35 (0.005%)	-1893.96 (-0.02%)	-1886.07 (-0.43%)	9698.94	10208.4 (5.25%)	9899.32 (2.07%)	9899.83 (2.07%)
s3	-2700	-2698.83	-2693.58 (-0.19%)	-2696.66 (-0.08%)	-2690.61 (-0.30%)	11415.1	11733.2 (2.79%)	11756 (2.99%)	11733.2 (2.79%)
s4	-2200	-2197.06	-2187.58 (-0.43%)	-2192.54 (-0.20%)	-2195.61 (-0.07%)	17496.8	17954 (2.61%)	17785.5 (1.65%)	17815.3 (1.82%)
s5	-4000	-3998.83	-3995.79 (-0.08%)	-3998.45 (-0.01%)	-3982.91 (-0.40%)	23055.3	24855 (7.81%)	24153 (4.76%)	23454.4 (1.73%)
s6	-3700	-3698.82	-3695.08 (-0.10%)	-3696.13 (-0.07%)	-3699.17 (0.01%)	28302.7	30151.5 (6.53%)	29369.2 (3.77%)	28779 (1.68%)
s7	-5700	-5698.5	-5690.96 (-0.13%)	-5689.56 (-0.16%)	-5699.22 (0.01%)	41839.7	44006.2 (5.18%)	42570.5 (1.75%)	42603.7 (1.82%)
s8	-4700	-4697.95	-4699.73 (0.04%)	-4697.59 (-0.01%)	-4695.55 (-0.05%)	46510.9	49063.4 (5.49%)	47546.6 (2.23%)	47259.7 (1.61%)
average			(-0.13%)	(-0.11%)	(-0.16%)		(4.84%)	(2.61%)	(1.73%)
*%: the difference of RAT between US/NS, ADS1/NS, and ADS2/NS, respectively									
**%: the power increment between US/NS, ADS1/NS, and ADS2/NS, respectively									

TABLE V
RUNTIME COMPARISON BETWEEN UNIFORM GRID SIZE AND ADAPTIVE GRID SIZE 3-D SAMPLING

net	CPU time (s)			
	NS	US (%)	ADS1 (%)	ADS2 (%)
s1	21	2 (9.52%)	4 (19.05%)	5 (23.81%)
s2	34	2 (5.88%)	5 (14.71%)	7 (20.59%)
s3	176	4 (2.27%)	10 (5.68%)	17 (9.66%)
s4	529	7 (1.32%)	14 (2.65%)	22 (4.16%)
s5	2091	19 (0.91%)	37 (1.77%)	64 (3.06%)
s6	5399	29 (0.54%)	58 (1.07%)	101 (1.87%)
s7	6077	48 (0.79%)	91 (1.50%)	156 (2.57%)
s8	6268	69 (1.10%)	141 (2.25%)	224 (3.57%)
average	(100%)	(2.79%)	(6.09%)	(8.66%)

procedure is performed recursively until no cubic grid requires further partitioning. Practically, we set $N = 100$, which indicates that the distances between pruned and sampled options are less than 1% of the overall span. This leads to good abstraction of the distribution of the original (optimal) solution space.

We implement both uniform and adaptive grid size 3-D sampling and test on cases s1–s8. In our second experiment, we compare four methods—no sampling (NS), which is optimal, uniform sampling (US), adaptive sampling with $N = 80$ (ADS1), and adaptive sampling with $N = 100$ (ADS2) in Table IV. Target RATs* at the source node are summarized in the second column. Their run times are summarized in Table V. To achieve the same RAT, adaptive sampling usually has a lower power consumption but longer run time compared with US. The larger the threshold is, the longer the adaptive sampling takes, and the closer it is to the optimal solution. In subsequent discussion, we use adaptive grid size 3-D sampling method with the threshold set to 100.

We present two additional prediction-based pruning rules in the following sections—PMP and PSP, which can further prune redundancy so that 3-D sampling can be performed in a better option pool. This is achieved by regressively removing options that are impossible to result in the optimal solution as the algorithm progresses.

2) Dual-V_{dd} Buffer PMP: Using the formula in [21], we precompute a unit length minimum delay table indexed by buffer, unit length resistance and capacitance, and the path length from the source to each tree node. Given an option, its upstream delay lower bound can be calculated by basing on the unit length minimum delay given by the precomputed table. An option is pruned if the difference between the source's RAT and its RAT is greater than this minimum delay, which renders the option infeasible under the source's RAT requirement.

3) Dual-V_{dd} Buffer PSP: PSP is extended from [5] and [14] for single V_{dd} buffer insertion. The basic idea is to remove options that are going to become inferior with respect to other options upon buffer insertion in the upstream. Due to page limitations, details are not repeated in this context.

D. Experimental Results and Discussions on Speed-up Techniques

1) Study of Individual Speed-up Technique: To evaluate the speed-up capability and the effect on the solution qualities of the three speed-up techniques (PMP, PSP, and adaptive grid size 3-D sampling), we run dBIS by using each of them individually. We then compare all pruning techniques with the exact algorithm (EDP) without any advanced pruning rules presented in the previous section. Owing to the exorbitant computational cost of EDP, we do not give the results of the large test cases. Table VI shows the comparisons on run time and solution qualities. Note that PMP and PSP give the same solution as EDP; thus, we collapse their RAT and power columns under EDP. In Table VI, we find that PMP and PSP can achieve some speedup upon EDP without losing optimality. Adaptive grid size 3-D sampling can achieve ten times speedup at a cost of 3% delay increase for small test cases.

2) Combine Sampling and Other Pruning Rules: To find out the effect of combining sampling and pruning rules, we test dBIS by combining adaptive grid size 3-D sampling with PMP and PSP, respectively. The results are shown in Fig. 2. We also list the results produced by employing adaptive grid size 3-D sampling only in Fig. 2. In Fig. 2, we observe the following.

1) Adaptive grid size 3-D sampling itself (column "3d") introduces significant loss of optimality for large test cases, although it brings substantial speedup.
2) PSP and PMP with 3-D sampling both result in better run time and solution quality compared with 3-D sampling-only solution. This shows that both rules contribute to removing redundant options, which create a superior candidate pool for sieving through 3-D sampling.
3) By combining PSP, PMP, and 3-D sampling (column "psp + pmp + 3d"), dBIS achieves the best performance on both solution quality and run time among 3d, psp + 3d, and pmp + 3d. The application of PSP and PMP maximizes the efficiency of 3-D sampling. Compared with the exact algorithm in Table VI, "psp + pmp + 3d" achieves over 1000 times speedup for s4 with less than 2% delay and power increase. More speedup is expected for larger test cases.
4) Both "psp + 3d" and "pmp + 3d" run much faster than 3-D sampling for small test cases (s1 and s2), but such speedup degrades for larger test cases. Among the speed-up techniques, only 3-D sampling confines number of options to linear growth with respect to tree size; therefore, the scaling trend of all other configurations follows that of 3-D sampling in larger test cases.

IV. CONCLUSION

This paper presents the first in-depth study on dual-V_{dd} buffer insertion for power minimization under delay constraint. Compared with single V_{dd} buffers, dual-V_{dd} buffers reduce power by 16% at the

TABLE VI
COMPARISON OF INDIVIDUAL SPEED-UP TECHNIQUES

	runtime (s)				RAT (ps)		power (fJ)	
net	EDP	pmp	psp	adaptive 3D	EDP/pmp /psp	adaptive 3D	EDP/pmp /psp	adaptive 3D
s1	176	184	146	15	-1444.99	-1465.03	13075.5	13798.7
s2	329	276	235	19	-1600.45	-1619.07	14521	14379.8
s3	2356	1759	1416	36	-2221.89	-2268.45	20411.1	19478.4
s4	>10000	8390	4391	50	-1804.55	-1841.09	29641.7	28828.4

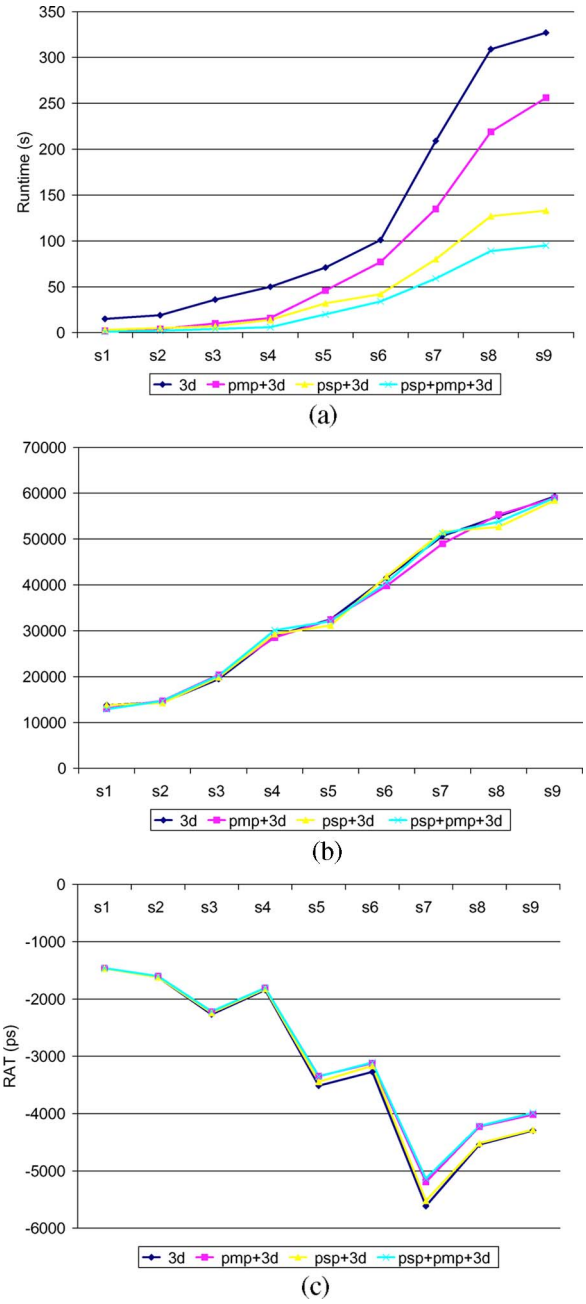

Fig. 2. Combining adaptive grid size 3-D sampling and other pruning rules. (a) Runtime. (b) Power. (c) RAT.

minimum delay specification. To cope with the increased complexity due to simultaneous delay and power consideration and increased buffer choices, we propose a speed-up technique combining sampling and two prediction-based pruning rules. The run time of the resulting algorithm grows linearly with respect to the tree size. We achieve a combined speedup of more than 1000 times over the exact power-optimal buffer insertion algorithm extended from [8] at the expense of 2% delay and power, respectively. In the future, we will study dual-V_{dd} buffered tree construction and consider cross-talk noise for buffering multiple nets.

REFERENCES

[1] L. P. P. P. van Ginneken, "Buffer placement in distributed RC-tree networks for minimal Elmore delay," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1990, pp. 865–868.
[2] C. J. Alpert and A. Devgan, "Wire segmenting for improved buffer insertion," in *Proc. Des. Autom. Conf.*, 1997, pp. 588–593.
[3] J. Lillis and C.-K. Cheng, "Timing optimization for multisource nets: Characterization and optimal repeater insertion," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 18, no. 3, pp. 322–331, Mar. 1999.
[4] C. J. Alpert, A. Devgan, and S. T. Quay, "Buffer insertion for noise and delay optimization," in *Proc. Des. Autom. Conf.*, 1998, pp. 362–367.
[5] W. P. Shi and Z. Li, "An O(nlogn) time algorithm for optimal buffer insertion," in *Proc. Des. Autom. Conf.*, 2003, pp. 580–585.
[6] V. Khandelwal, A. Davoodi, A. Nanavati, and A. Srivastava, "A probabilistic approach to buffer insertion," in *Proc. Int. Conf. Comput.-Aided Des.*, 2003, pp. 560–567.
[7] J. J. Xiong and L. He, "Fast buffer insertion considering process variations," in *Proc. Int. Symp. Phys. Des.*, 2006, pp. 128–135.
[8] J. Lillis, C. K. Cheng, and T. T. Y. Lin, "Optimal wire sizing and buffer insertion for low power and a generalized delay model," in *Proc. Int. Conf. Comput.-Aided Des.*, 1995, pp. 138–143.
[9] R. Rao, D. Blaauw, D. Sylvester, C. J. Alpert, and S. Nassif, "An efficient surface-based low-power buffer insertion algorithm," in *Proc. Int. Symp. Phys. Des.*, 2005, pp. 86–93.
[10] F. Li, Y. Lin, and L. He, "FPGA power reduction using configurable dual-Vdd," in *Proc. Des. Autom. Conf.*, 2004, pp. 735–740.
[11] F. Li, Y. Lin, and L. He, "Vdd programmability to reduce FPGA interconnect power," in *Proc. Int. Conf. Comput.-Aided Des.*, 2004, pp. 760–765.
[12] Y. Lin and L. He, "Leakage efficient chip-level dual-Vdd assignment with time slack allocation for FPGA power reduction," in *Proc. Des. Autom. Conf.*, 2005, pp. 720–725.
[13] J. H. Anderson and F. N. Najm, "Low-power programmable routing circuitry for FPGAs," in *Proc. Int. Conf. Comput.-Aided Des.*, 2004, pp. 602–609.
[14] Z. Li, C. N. Sze, C. J. Alpert, J. Hu, and W. P. Shi, "Making fast buffer insertion even faster via approximation techniques," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2005, pp. 13–18.
[15] K. H. Tam and L. He, "Power optimal dual-Vdd buffered tree considering buffer stations and blockages," in *Proc. Des. Autom. Conf.*, 2005, pp. 497–502.
[16] J. Cong and X. Yuan, "Routing tree construction under fixed buffer locations," in *Proc. Des. Autom. Conf.*, 2000, pp. 379–384.
[17] C. J. Alpert, G. Gandham, J. Hu, J. L. Neves, S. T. Quay, and S. S. Sapatnekar, "Steiner tree optimization for buffers, blockages, and bays," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2001, pp. 399–402.
[18] W. Chen, M. Pedram, and P. Buch, "Buffered routing tree construction under buffer placement blockages," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2002, pp. 381–386.
[19] J. Hu, C. J. Alpert, S. T. Quay, and G. Gandham, "Buffer insertion with adaptive blockage avoidance," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 4, pp. 492–498, Apr. 2003.
[20] D. Warme, P. Winter, and M. Zachariasen, *GeoSteiner*. [Online]. Available: http://www.diku.dk/geosteiner
[21] K. Banerjee and A. Mehrotra, "A power-optimal repeater insertion methodology for global interconnects in nanometer designs," *IEEE Trans. Electron Devices*, vol. 49, no. 11, pp. 2001–2007, Nov. 2002.