Timing-Constrained and Voltage-Island-Aware Voltage Assignment

Huaizhi Wu
Cadence Design Systems, Inc.
huaizhi@cadence.com

Martin D.F. Wong
U. of Illinois at Urbana-Champaign
mdfwong@uiuc.edu

I-Min Liu
Atoptech, Inc.
imliu@atoptech.com

ABSTRACT

Multi-Vdd is an effective method to reduce both leakage and dynamic power. A key challenge in a multi-Vdd design is to limit the design cost and the demand for level shifters. This can be tackled by grouping cells of different supply voltages into a small number of voltage islands. Recently, an elegant algorithm [7] is proposed for generating voltage islands that balance the power versus design cost tradeoff under performance requirement, according to the placement proximity of the critical cells. One prerequisite of [7] is an initial voltage assignment at the standard cell level that meets timing. In this paper, we present a novel method to produce quality voltage assignment to [7], which not only meets timing but also forms good proximity of the critical cells to provide [7] with a smooth input. The algorithm is based on effective delay budgeting and efficient computation of physical proximity by Voronoi diagram. Our experiments on real industrial designs show that our algorithm leads to 25 - 75% improvement in the voltage island generation, with the computation time only linear to the design size.

Categories and Subject Descriptors: B.7.2 Design Aids

General Terms: Algorithms, Design

Keywords: Low power, Voltage assignment, Voronoi diagram

1. INTRODUCTION

Power managing is increasingly one of the most pressing problems in large chip design. Power consumption at the 90nm process node, if not tamed, is of orders of magnitude higher than that at the 180nm node [**?**]. Design for low power is of the primary concern in designing 90-nm chips and below.

Multi-Vdd is an effective method to reduce both dynamic and leakage power. In a multi-Vdd design, high-Vdd is assigned to timing critical cells while low-Vdd is assigned to noncritical cells, so that power can be saved without degrading the overall circuit performance. However, the resulting complex power supply system increases the design cost, as more routing resource and heavy human intervention are required. Moreover, level shifters need to be inserted at the boundaries between low-Vdd and high-Vdd, causing extra area, delay and power penalty. Therefore, it is desired that cells of different supply voltages are grouped into a small number of voltage islands (each having a single Vdd), so that the increase of design cost and the demand for level shifters are limited.

The voltage island generation problem need to balance the power versus design cost tradeoff under performance requirement (where the design cost can be measured by the number of the voltage islands generated). [7] formulated this as an optimization problem and proposed an elegant algorithm to solve it. In [7], voltage islands are generated after a placement is given and the voltage requirement of each cell is known. Based on the physical proximity of the high voltage (critical) cells in the placement, the algorithm efficiently explores the solution space of all possible island boundaries and finds the voltage islands (also called the *voltage island partitioning*) for the best tradeoff between the total power and the number of islands. The timing constraints are respected by honoring the given voltage requirement of all cells.

One prerequisite of [7] is the initial voltage assignment at the standard cell level that meets timing. How to generate such an initial voltage assignment, has remained an unstudied problem. Besides, as [7] generates the voltage islands based on the physical proximity of high voltage cells in the placement, such proximity has a great influence on its output. Specifically, an initial voltage assignment with high voltage cells physically close to each other and low voltage cells forming large continuous areas, promises a better voltage island partitioning. However, in current industry low power design, people only focus on the maximum power reduction on the cells. This may lead to an inferior voltage assignment for the voltage island generation. E.g., in Figure 1, even though the voltage assignment in (a) has more power reduction than the one in (b), however, as the high voltage cells in (b) has better physical proximity than those in (a), the final voltage island partitioning for (b) (shown in (d)) is better than that for (a) (shown in (c)).

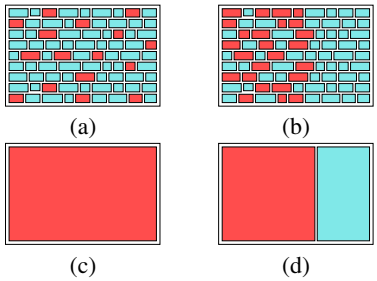


Figure 1: (a) Voltage assignment with more power reduction. (b) Voltage assignment with less power reduction. (c) Voltage island partitioning with less power reduction. (d) Voltage island partitioning with more power reduction.

In this paper, we present an algorithm to produce an initial voltage assignment that not only meets timing, but also forms good proximity of high voltage cells to provide [7] with a smoother input. Our contributions can be summarized as follows: To our best knowledge, we are the first the consider physical proximity together with timing constraints in standard cell voltage assignment. We

DAC 2006, July 24–28, 2006, San Francisco, California, USA.

formulate this voltage assignment problem as a delay budgeting problem, with the special objective of physical proximity. Based on the framework of a classic delay budgeting algorithm, the zero slack algorithm, we design an effective algorithm to allocate delay budget and assign voltage to cells towards our objective of physical proximity. We efficiently solve a key problem in our algorithm, the dynamic closest point problem, by using incremental construction of Voronoi diagram and fast point location. Our extensive experiments on a wide selection of industry designs demonstrate the effectiveness and efficiency of our algorithm. Specifically, on average, our algorithm leads to 25% - 75% improvement in the voltage island generation, in terms of the number of voltage islands generated. Our running time on the computation of the physical proximity displays a linear behavior, significantly faster than a simple straightforward solution.

2. VOLTAGE ASSIGNMENT

In this section, we first establish the relation between the voltage assignment problem and delay budgeting problem. Then we review the zero slack algorithm for the delay budgeting problem. Last we describe our algorithm for the voltage assignment problem, as well as the efficient computation of physical proximity.

2.1 Voltage Assignment and Delay Budgeting

Cell delays increase with reduced supply voltages. The former can be formulated as a linear function of the latter, when $\Delta V_{dd} = V_{ddH} - V_{ddL}$ is small [2]:

$$d_L - d_H = k * d_H * (V_{ddH} - V_{ddL}) \quad (1)$$

where k is a constant for each cell, and d_H is the cell delay under the nominal voltage.

According to this equation, assigning a specific voltage to a cell is equivalent to allocating a corresponding amount of delay budget to this cell. To reduce cell voltages without violating timing constraints, for every path, the sum of delay increase resulted from reduced voltages can not exceed the total slack of this path. Therefore the voltage assignment problem is essentially a delay budgeting problem, except that, it has a different kind of objective – the physical proximity of high voltage cells – than those of the traditional delay budgeting problem, such as maximizing total (weighted) delay budget or minimizing the maximum budget [3].

2.2 The Zero Slack Algorithm

Among the various existing algorithms for the delay budgeting problem, the zero slack algorithm (ZSA) [6] is both simple, efficient, and can be flexibly adapted to handle different objectives such as physical proximity. Therefore we will adopt its framework and make necessary changes towards our objective.

ZSA iteratively assigns delay budget to non-critical cells in the circuit. At each iteration, it first computes the slacks of all cells and find the minimum positive slack s_{min}; then it finds a path with this minimum positive slack, and distributes s_{min} evenly among the cells on the path. Then it updates the delay of these cells and go to the next iteration. The algorithm stops when the slacks of all cells become zero.

Figure 2 shows an example. The cell delay is shown inside each cell and the net delay is shown on each net. A triplet $t_A/slack/t_R$ is shown at each PI/PO and the output of each cell, where t_A and t_R are arrival time and required time, respectively. (a) shows the first iteration, where (C_1, C_2, C_4) is the path with the minimum positive slack 3, indicated by the circled triplet (we exclude the PI/PO because delay budget will not be assigned to them). Then the path slack is evenly distributed among C_1, C_2 and C_4. Their delays become 3, 5 and 7, respectively. This process repeats until it reaches (b), where the output pin of each cell has zero slack.

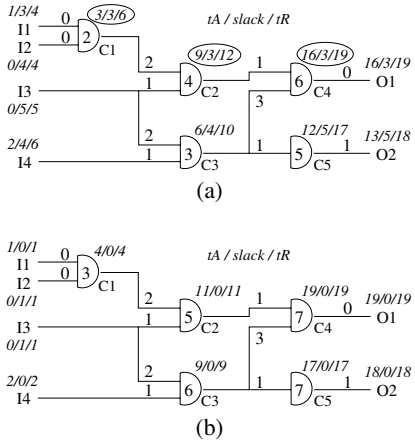


Figure 2: The zero-slack algorithm.

2.3 Physical Proximity Driven Voltage Assignment

For our voltage assignment problem, we adopt the iterative framework of ZSA. The major difference between our algorithm and the original ZSA lies in how we allocate the path slack among the cells on each selected path, as our allocation is driven by a different objective: the physical proximity of the high voltage cells.

Main Algorithm To start with, we compute for each cell c the delay budget it needs for voltage reduction, according to Equation 1. We denote it as r_c, and we denote the slack of the cell c as s_c. Those cells satisfying $s_c < r_c$ will be assigned high voltage because they can not have any voltage reduction. We call these decided high voltage cells as *sites*. They are going to be used as the initial seeds to guide subsequent voltage assignment.

Then at each iteration, for each cell c on the selected path, we calculate its distance to the closest site, denoted as d_c. The farther away a cell to any site, the more likely it's sitting in a potentially non-critical low-voltage region, the more cost there is if this cell becomes a high voltage cell, and the higher priority we should give it for voltage reduction. Therefore we sort all cells on the selected path with decreasing value of d_c, and allocate the path slack in the order of this sorted list, giving each cell the delay budget it needs for voltage reduction. If at certain cell in the list, the remaining slack is less than its needed delay budget, then we simply skip it and proceed to the next one (different cells have different amount of needed delay budget for voltage reduction). The allocation stops when it reaches the end of the list, or when there is no more slack left. All cells whose voltages remain high will be added as new sites. Such allocation guarantees that high voltage are always assigned to cells as close to the existing sites as possible, thereby maintains the physical proximity of the high voltage cells.

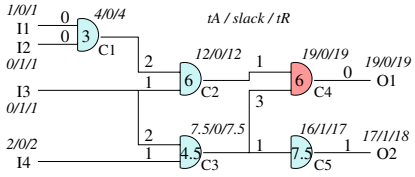


Figure 3: The physical proximity driven voltage assignment.

There may be a surplus slack left at the end of the list. As they may be shared by other unselected paths, we just leave it there and proceed to the next iteration. However, we will mark all cells on the current path as "dead", so that they will be excluded from the search for the minimum positive slack and the corresponding path in subsequent iterations.

The just described algorithm (PDVA) is summarized as follows.

***procedure* Proximity-Driven-Voltage-Assignment**

```
1    begin
2      compute all slacks;
3      find all cells satisfying s_c < r_c, set them as sites;
4      find a path p with minimum positive slack s_min;
5      repeat
6        for each cell c on p :  calculate d_c;  endfor
7        sort all cells on p according to d_c;
8        s = s_min;
9        for each cell c in the sorted list
10         if    s >= r_c
11         then  reduce the voltage of c;
12               s = s - r_c;
13         else  add c as a high voltage site;
14         endif
15         mark c as "dead";
16       endfor
17       update all slacks;
18       find next path p with minimum positive slack s_min;
19     until no path p is found;
20   end
```

We use Figure 3 as a simple illustration of the algorithm. Starting from the same circuit as shown in Figure 2(a), suppose that $k = 0.5$ for each cell, and $d_{C_1} > d_{C_2} > d_{C_4}$ on path (C_1, C_2, C_4). The algorithm will produce the result shown in Figure 3, where the light-colored cells have reduced voltage, while the dark-colored cell has high voltage. Note that, compared to the result in Figure 2(b), cell C_2 got reduced voltage due to the "non-uniform" allocation of slack; cell C_5 got reduced voltage by using the remaining slack from the previous path (C_3).

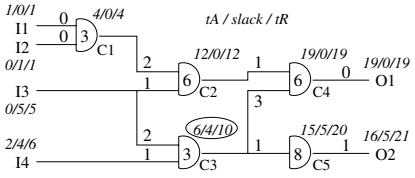


Figure 4: A situation where skipping a cell on an earlier selected path enables a cell on a later selected path to reduce voltage.

Further improvement In the above algorithm we showed how to use existing physical proximity information to allocation slack (and assign voltage) to cells on the same selected path. We further observe that the proximity information can also be used to improve slack allocation between cells on *different* selected paths. Specifically, an earlier selected path may contains some cells closely surrounded by already decided sites. Assigning slack to reduce their voltage will not lead to any potential power reduction, as the whole region will be raised to high voltage in the subsequent voltage island partitioning. In such case, we can skip these cells and save the slack for cells on the intersecting paths that will be selected later. For example, in Figure 4, the needed delay budget for voltage reduction by C_3 and C_5 are 1.5 and 4, respectively. Cell C_3 is detected as being closely surrounded by decided sites, while C_5 is far away from any site. If we skip C_3 and save the 1.5 amount of slack, C_5 will have enough slack to reduce voltage, thus avoiding generating a new site in a low voltage region.

To summarize, we can add the following lines between line 9 and 10 in PDVA, and call the improved algorithm PDVA*.

```
    if c is surrounded by high voltage sites
    then continue;
    endif
```

2.4 Efficient Computation of Physical Proximity

In last section, we have described our main algorithm and a further improvement. However, we have not yet described how to solve a key problem for the computation of the physical proximity: Among a set of given sites, finding the one closest to the query cell. (Commonly called the *closest point problem*.) A naive solution is to use a linear search of the sites for each query cell. This will lead to an overall $O(nm)$ running time. (m is the number of sites, n is the number of queries.) When the number of cells is very large, which is the case in modern IC designs, m and n will also be large, causing significant running time. Therefore, a more efficient solution is desired.

Fortunately, this closest point problem, due to its wide applications, has been well studied in computational geometry. A most efficient solution is by constructing the Voronoi diagram of the given sites, and then locating the query point in one of its regions.

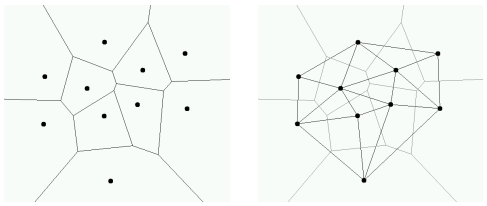


Figure 5: (a) The Voronoi diagram of ten points in the plane. (b) The corresponding Delaunay triangulation.

Let P be a set of n points in the plane, called *sites*. They can be preprocessed into a data structure, so that given a new query point q, we can efficiently locate the nearest neighbor of q among the sites. The n sites in fact subdivide the plane into n regions, one for each site in P, such that the region of a site $p \in P$ contains all points in the plane for which p is the closest site. This subdivision is called the *Voronoi diagram of* P, and the region associated with a site p is called the *Voronoi region of* p. The closest site problem can thereby be solved by constructing the Voronoi diagram and then locating the query point in it. An example Voronoi diagram for a set of ten sites is shown in Figure 5(a).

Among the various algorithms for constructing the Voronoi diagram, the incremental algorithm is the only one that allows the sites to arrive on-line, instead of requiring all of the n sites to be given in advance. Therefore only the incremental algorithm is suitable to solve the dynamic closest point problem for the voltage assignment, because the high voltage sites are decided dynamically. An efficient implementation of the incremental algorithm by using the dual of the Voronoi diagram – the Delaunay triangulation (Figure 5(b)), and an efficient way to locate a query point in the Voronoi diagram, can be found in [5, 4].

To detect whether a given cell is surrounded by high voltage sites during PDVA*, we can simply calculate the radius of the Delaunay triangle containing the cell, and compare it with a threshold value.

Design		# of VI for ΔP_1			# of VI for ΔP_2		
Name	# of Cells	EZSA	PDVA	PDVA*	EZSA	PDVA	PDVA*
bench1	5926	19	12	11	39	24	21
bench2	43677	35	27	25	42	32	29
bench3	76406	38	21	14	OM	28	25
bench4	243188	40	29	27	66	40	33
bench5	306326	44	36	34	OM	44	42
bench6	317752	20	14	13	29	21	19

Design			Running Time (sec)		
Name	# Sites	# Queries	LS	PDVA	PDVA*
bench1	352	4691	0.28	0.04	0.04
bench2	459	43537	3.51	0.3	0.29
bench3	1861	76232	24.41	0.76	0.7
bench4	592	254849	26.85	1.86	1.78
bench5	2326	236198	95.65	2.32	2.08
bench6	6680	304192	364.11	4.19	3.69

Table 1: (a) Comparison of output results(OM: out of memory due to large # of VI) (b) Comparison of running time for closest point computation

3. EXPERIMENTAL RESULTS

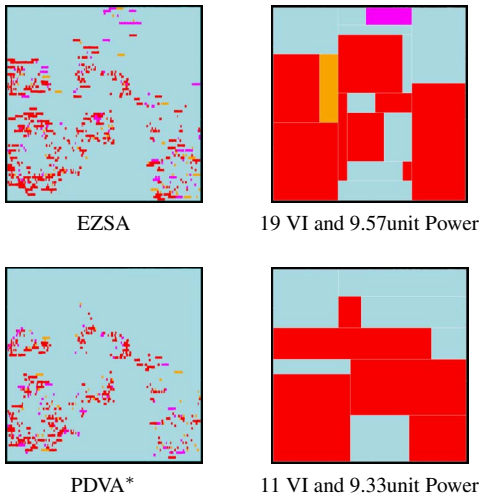
3.1 Experiment setup and snapshots of results

We perform our experiments on a set of industry designs. For each design, the experiment is carried out in the following steps: 1) We use the Cadence's SoC Encounter [1] to do physical prototyping under nominal voltage (the highest allowed voltage). 2) We run the voltage assignment algorithm to generate timing-met initial voltage assignment. 3) We run voltage island partitioning on this initial voltage assignment to generate voltage islands.

The result from the voltage island partitioning will be the direct measurement of the result quality of our voltage assignment.

To demonstrate the effectiveness of our physical proximity driven voltage assignment algorithms PDVA and PDVA*, we compare them with an algorithm which differs from PDVA only in that it does not sort the cells on each selected path according to their respective distance to the closest site. we call this algorithm EZSA.

To demonstrate the efficiency of the Voronoi diagram based closest point computation, we compare the closest point computation time in PDVA and PDVA* (including diagram construction and point queries) with the straightforward linear search based computation, called LS.



EZSA 19 VI and 9.57unit Power

PDVA* 11 VI and 9.33unit Power

Figure 6: Snapshot of Design bench1.

Snapshots We first give some visual results. Figure 6 shows the comparison of EZSA and PDVA* on an industry design (bench1). The left column shows the voltage assignment produced by EZSA and PDVA*; the right column shows the corresponding voltage island partitioning generated from these initial voltage assignments. We can see that, with roughly the same amount of power reduction, PDVA* leads to a much smaller number of voltage islands.

3.2 Comparison between different algorithms

Comparison on the same design with different power bounds Figure 7 compares EZSA and PDVA* on the same design under different power bound by showing the number of voltage islands

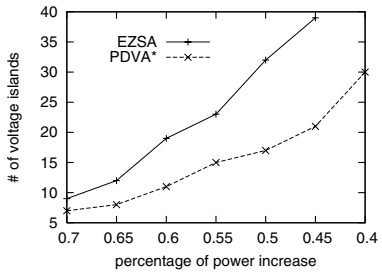


Figure 7: Comparison of EZSA and PDVA* on design bench1.

generated in the subsequent voltage island partitioning. Clearly PDVA* outperforms EZSA significantly and consistently.

Comparison of quality of result on different designs We run our experiments on a wide selection of industry designs of different sizes. For each design, we compare the three algorithms under two power bounds. The results are shown in Table 1(a). Clearly, PDVA and PDVA* outperform EZSA significantly and consistently on all designs in terms of the number of voltage islands generated. PDVA* also achieves consistent improvement over PDVA.

Comparison of running time on different designs The actual running time of the closest point computation based on the incremental construction of Voronoi diagram, depends on the physical distribution of the sites and the query points. With well behaved input distribution, the algorithm can run in $O(n \log m)$ or even $O(n)$ time. (m is the number of sites, n is the number of queries.) From Table 1(b) we can see that the running time of the closest point computation in PDVA and PDVA* actually displays the linear behavior ($O(n)$).

Acknowledgment

The first author would like to thank Dr. Vassilios Gerousis and Yu Zheng for helps on modeling power-delay relation; thank Jean-Pierre Hiol and Leslie Lai for helps on using Cadence's Common Timing Engine.

4. REFERENCES

[1] Cadence software manual: Soc encounter gps. http://www.cadence.com/products/digital_ic/soc_encounter/index.aspx.

[2] Synopsys liberty user guide, version 2003.12.

[3] E. Bozorgzadeh, S. Choudhury, and M. Sarrafzadeh. A unified theory of timing budget management. In *Proc. of the IEEE/ACM Int. Conf. on Computer-aided design*, pages 653–659, 2004.

[4] M. de Berg, M. van Kreveld, O. Overmars, and O. Schwarzkopf. *Computational Geometry - Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[5] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM Trans. Graphics*, 4(2):74–123, 1985.

[6] R. Nair, C. L. Berman, P. S. Hauge, and E. J. Yoffa. Generation of performance constraints for layout. *IEEE Trans. Computer-Aided Design*, 8(8):860–874, 1989.

[7] H. Wu, I. Liu, M. Wong, and Y. Wang. Post-placement voltage island generation under performance requirement. In *Proc. of the IEEE/ACM Int. Conf. on Computer-aided design*, pages 309–316, 2005.