I/O Buffer Placement Methodology for ASICs

Joseph N. Kozhaya
University of Illinois, Urbana

Sani R. Nassif
IBM Austin Research Labs

Farid N. Najm
University of Toronto

Abstract

In modern designs, voltage drop on the power grid is becoming a critical concern. One important technique to avoid severe voltage drops is to spread the highly *power hungry* buffers, such as I/O buffers, around the grid. In this paper, we study the problem of I/O buffer placement so as to avoid *hot spots* which are locations on the chip characterized by high voltage drops. The problem is defined mathematically and formulated as an ILP problem. Then an efficient greedy heuristic is proposed as an alternative to the expensive ILP solution.

1 Introduction

ITRS'99 [1] indicates that modern and future VLSI designs are characterized by reduced feature size, lower supply voltage, and higher currents. Lower supply voltage makes the voltage variation across the power grids very critical since it may lead to chip failures. Voltage drops on the power grid reduce the supply voltage at logic gates and transistor cells to less than the ideal reference. This leads to reduced noise margins, higher logic gate delays, and overall slower circuits. Reduced noise margins may lead to false switching at certain logic gates and latches. Higher logic gate delays, on the other hand, may slow down the circuit enough so that timing requirements can not be met. Consequently, once voltage drops exceed certain designer-specified thresholds, there is no guarantee that the circuit will operate properly [2, 3].

In order to avoid hot spots, which are locations on the chip characterized by severe voltage drops, several approaches are possible including but not limited to [4]:

1. Making the grid more robust by using a denser grid structure or wider metal lines.
2. Placing on-chip decoupling capacitors that significantly suppress high frequency components of the voltage drop.
3. Moving *power hungry* buffers and modules so that they are spread all over the chip as opposed to keeping them densely populated in certain areas thus creating *hot spots*.

The technique of spreading power hungry buffers around is particularly interesting since it is typically less expensive than either of the other proposed techniques. Making the grid more robust involves expensive redesign effort and adding decaps can be significantly expensive in terms of chip area.

Specifically, I/O buffers are identified as highly power hungry and consequently, the analysis of I/O buffer placement effects on voltage drop is critical in modern designs. The electrical problems due to I/O buffer placement are particularly severe in area-array (also called flip-chip) designs as discussed in [5]. This is because in area-array designs, the I/O buffers may be placed anywhere on the die and are not limited to the periphery. The authors in [5] clearly identify the severity of electrical problems in *area-array* ASIC designs due to the placement of I/O buffers and the reasons for such problems.

Thus, in this paper, we study the problem of I/O buffer placement in *area-array* ASIC designs and propose a placement methodology that avoids severe voltage drops. Note that the proposed placement technique is not meant to replace existing I/O placement techniques but rather complement those techniques by accounting for potential electrical problems. Furthermore, although our technique targets only I/O buffers, it also applies for the electrically sound placement of all power hungry buffers in the design.

2 Problem Definition

The analysis of the effect of buffer placement on the performance of power grids requires modeling the grids as well as the power sources and drains. For purposes of *efficient* analysis, power grids are modeled as *linear* RC networks, power *sources* are modeled as simple constant voltage sources, and power *drains* are modeled as independent time-varying current sources [3, 6]. On-chip inductance is ignored since in today's technology, on-chip power grid inductance is too small to affect the analysis results.

The behavior of such a system can be expressed following the modified nodal analysis (MNA) [7] formulation as the following ordinary differential equation:

$$Gx + C\dot{x} = u(t) \tag{1}$$

where x is a vector of node voltages, and source currents; G is the conductance matrix; C includes the capacitance terms, and $u(t)$ includes the contributions from the sources and the drains.

Applying Backward Euler (BE) numerical integration to (1) results in a set of linear equations:

$$(G + C/h)x(t+h) = u(t+h) + x(t)C/h \tag{2}$$

which can be written as $Ax(t+h) = u(t+h) + x(t)C/h$, where $A = G + C/h$.

The system matrix, A, can be shown to be *symmetric*. In fact, the system of linear equations can be reformulated in such a manner to produce a system matrix, A, which can be shown to be a *nonsingular* M-matrix [9]; thus leading to the following useful properties:

$$\sum_{i=1}^{N} a_{ij} \geq 0 \tag{3}$$

$$a_{ij}^{-1} \geq 0 \quad \forall\, i,\ j \tag{4}$$

where N is the dimension of the A matrix and a_{ij}^{-1} is the entry at the i^{th} row and j^{th} column of the matrix A^{-1}, the inverse of the system matrix A. Note that N also denotes the number of power grid nodes. In all what follows, we assume that the system is formulated in such a manner to result in a system matrix, A, which is a *nonsingular* M-matrix [9].

Before going further, we will restrict the problem to a DC analysis step, for at least two reasons. One reason is that, in order to be conservative, so that the placement is safe irrespective of the timing of the different buffers, it makes a lot of sense to use a DC value of current for each I/O buffer equal to the peak current that it can draw, and to solve the system under this condition. This would seem to be a prudent practical approach. Another reason is that it makes a lot of sense from a methodology standpoint to first do a fast DC analysis, using either peak or average current for every buffer, in order to uncover any gross problems, and then follow that up with more detailed AC analysis to fine tune the result. Thus, the DC solution is a prudent, conservative, and practical approach to the problem. In this case, the system equation becomes $Ax = u$, where $A = G$.

Note that the solution of the resulting linear system provides the voltages at all the grid nodes. However, it is more useful to represent the system in terms of an $N \times 1$ vector of voltage drops, δ. That is, $\delta = V - x$ where V is an $N \times 1$ vector whose entries are all V_{DD}; that is, $V(i) = V_{DD} \quad \forall\, i$. Straightforward algebraic simplification of the original system $Ax = i$ results in the following linear system:

$$A\delta = b \qquad \Leftrightarrow \qquad \delta = A^{-1}b \tag{5}$$

where A is the original system matrix, δ is the vector of voltage drops and b is the vector of current sources. That is, the i^{th} entry of the b vector, $b(i)$, corresponds to the sum of all the currents of the drains being supplied by grid node i. Consequently, it is clear that b is a vector whose every entry is given by:

$$b(i) = \sum_{k=1}^{K} c_{ik} I_k \quad \forall\, i \tag{6}$$

where I_k is the current associated with buffer B_k, and c_{ik}, is a variable which can be either 0 or 1: if B_k is connected to node n_i, then $c_{ik} = 1$; otherwise $c_{ik} = 0$. Thus, the entries of the b vector are all non-negative:

$$b(i) \geq 0 \quad \forall\, i \tag{7}$$

Referring to equation (5), we can define the relationship between the voltage drop at a node n_j and all the entries of the right hand side vector, b:

$$\delta(j) = \sum_{i=1}^{N} a_{ji}^{-1} b(i) \quad \forall\, j \tag{8}$$

Similarly, the current drawn at node n_i, $b(i)$ can be expressed as a function of the voltage drops:

$$b(i) = \sum_{j=1}^{N} a_{ij}\delta(j) \quad \forall\, i \tag{9}$$

Note that $\delta(j) \geq 0 \;\forall\; j$ since $a_{ji}^{-1} \geq 0 \;\forall\; i,\ j$ as given by equation (4) and $b(i) \geq 0 \;\forall\; i$ as given by equation (7).

Hence, equations (5), (8), (9) describe the relationship between the voltage drops on the power grid and the currents drawn by the I/O buffers. In the remainder of this section, we discuss the analysis and propose an algorithm for solving the above problem.

3 Proposed Buffer Placement

Solving the I/O buffer placement problem involves placing the K given I/O buffers while satisfying the user-specified drop thresholds denoted by the $N \times 1$ vector, δ_{max}. Thus, a *feasible* solution requires that:

$$\delta \leq \delta_{max} \qquad \Leftrightarrow \qquad \delta(j) \leq \delta_{max}(j) \quad \forall\, j \tag{10}$$

3.1 ILP formulation

Consequently, the I/O buffer placement problem can be formulated as an ILP (integer linear programming) problem where a feasible assignment of the c_{ik} variables is needed to satisfy the following constraints:

$$\sum_{i=1}^{N} c_{ik} = 1 \quad \forall k \tag{11}$$

$$\sum_{k=1}^{K} c_{ik} I_k - b(i) = 0 \quad \forall\, i \tag{12}$$

$$\sum_{j=1}^{N} a_{ij}\delta(j) - b(i) = 0 \quad \forall\, i \tag{13}$$

$$\delta(j) \leq \delta_{max}(j) \quad \forall\, j \tag{14}$$

where N is the number of nodes in the system and K is the number of I/O buffers as defined earlier. Also, a_{ij} is the entry of the A matrix at row i and column j and c_{ik} is the 0-1 variable defined earlier. Note that the constraints given by equation (13) are nothing but the system equations $A\delta = b \Leftrightarrow A\delta - b = 0$. The constraints given by equation (14), on the other hand, impose the user-specified drop thresholds at all the nodes in the circuit. Furthermore, the constraints given by equation (12) define the i^{th} entry of the right hand side vector to be the sum of the currents of the I/O buffers connected to node n_i. Finally, the constraints given by equation (11) mathematically force the physical observation that every I/O buffer can be placed at exactly one node. Thus, if $c_{jk} = 1$ which indicates that buffer B_k is placed at node n_j, then $c_{ik} = 0 \quad \forall\, i \neq j$.

Note that we are not necessarily searching for an optimal solution, but rather for a feasible solution, simply because a feasible solution suffices to avoid *hot spots* and guarantees no electrical failures. We have used the ILP package, BARON [10], to solve for a feasible assignment of the c_{ik} variables such that the drop threshold at every grid node is 2.0%. It is clear from the results shown in Table 3.1 that finding a feasible solution using ILP suffers from complexity blow-up

Table 1: CPU times using ILP solver, BARON.

Problem	# buffers, K	# nodes, N	CPU time
p_1	12	210	1.53 sec
p_2	25	392	6.59 sec
p_3	53	504	15.31 min
p_4	74	760	48.06 min

as the size of the problem increases. In order to avoid this, we propose a *greedy* heuristic that will be presented in section 3.2 below.

3.2 Greedy algorithm

Notice that the A^{-1} matrix defines the sensitivity of the solution relative to a change in the placement of I/O buffers. This information allows us to propose a *greedy* heuristic, IOPlace, that attempts to produce an *electrically sound* I/O buffer placement. The basic idea is to place the I/O buffers one at a time while guaranteeing that the drop thresholds are satisfied at all the nodes. We define the *drop slack* at a node n_j, denoted $s(j)$, to indicate the amount of drop which node n_j can still sustain before its drop threshold constraint is violated. That is:

$$s(j) = \delta_{max}(j) - \delta(j) \quad \forall\ j \tag{15}$$

It is clear that a drop threshold violation at a node is detected when the slack at that node becomes negative. Define the current at n_i to be *allowable* if the drop at every node n_j due to that current is less than the drop slack at n_j:

$$\delta(j) = a_{ji}^{-1} b(i) \leq s(j) \tag{16}$$

Thus, given the drop slack at all the grid nodes, an *upper bound* on the *allowable* current drawn from node n_i can be computed as follows:

$$b(i) \leq \min_j \frac{s(j)}{a_{ji}^{-1}} = UB_i \tag{17}$$

With these definitions and observations, the greedy algorithm, IOPlace, can be summarized as follows:

1. Sort the I/O buffers in decreasing order of their currents.
2. Initialize the drop slack at all the grid nodes to their specified drop thresholds. That is, $s(j) = \delta_{max}(j)\ \forall\ j$.
3. For every buffer, B_k, let L_k be the list of potential grid nodes to which B_k can be connected (placed). In general, L_k is the list of all grid nodes unless otherwise specified by the user. Then for every node $n_i \in L_k$, compute the upper bound UB_i on the current that can be drawn at n_i as given by (17).
4. Let n_m be the node with the highest upper bound, $UB_m \geq UB_i\ \ \forall\ i$. Assign buffer B_k to node n_m. If $I_k \leq UB_m$, it is guaranteed that placing buffer B_k at node n_m will not cause any drop threshold violations. Otherwise, violation may occur at certain nodes.
5. Update the slack at all the grid nodes:
$$s(j) = s(j) - a_{jm}^{-1} I_k \quad \forall\ j \tag{18}$$

Table 2: CPU times for I/O placement. K is the number of buffers and N is the number of nodes.

Design	K	N	# Violations	CPU time
C_1	616	4602	0	3.79 min
C_2	588	3325	0	3.04 min

6. If $s(j) < 0$, report drop threshold violation at node n_j. Flag node n_j so that it is ignored when placing the remaining buffers.
7. Continue at step 3, with the next buffer.

Either because the problem has no feasible solution, or because of the greedy nature of the algorithm, it is possible that the algorithm fails to satisfy the drop thresholds at all the grid nodes. In case a violation at a node is detected, it is reported to the user so that afterwards, he/she can attempt to fix the problem manually. Note that if the algorithm reports no violations, then this means that the resulting proposed placement does satisfy the drop thresholds at *all* the nodes.

It remains to discuss the complexity of the proposed technique. Following the same notation, assume the input to the problem is a power grid of N nodes and K buffers that should be placed. The matrix inversion is done by first performing an LU-factorization and then applying a forward-backward solve in order to obtain every row of the inverse matrix, as needed. The LU is approximately $O(N^{1.5})$ for a sparse matrix, and each forward-backward solve is approximately $O(N^{1.1})$ for a sparse matrix.

As for the algorithm itself, for every I/O buffer B_k (K times), the algorithm goes through the list of nodes in L_k (N nodes), and for each of these nodes, it computes the upper bound on the *allowable* current by going through all the grid nodes (N nodes). Therefore, this algorithm has $O(KN^2)$ complexity. However, note that if $a_{ji}^{-1} \approx 0$, then there is no need to consider node n_j when searching for the upper bound on the *allowable* current at node n_i. In other words, there is no need to consider the effect of node n_j which is *not sensitive* or *slightly sensitive* to node n_i. Thus, we need only consider a node $n_j \in SEN_i$ where $SEN_i = \{n_j \mid a_{ji}^{-1} > \varepsilon\}$ is the sensitivity list associated with node n_i and ε is a small positive constant, $\varepsilon \approx 0$. Furthermore, in *real* power grids, SEN_i typically consists of those nodes which are *geometrically close* to n_i; that is, SEN_i consists of a *small* number of nodes. In order to make use of this observation, we modify the above algorithm so that it computes the upper bound on n_i by going through the nodes $n_j \in SEN_i$ only. Since the number of nodes in SEN_i is small and can be upper bounded by a constant, then the complexity of the modified algorithm is reduced to $O(KN)$ instead of $O(KN^2)$.

4 Experimental Results

The greedy algorithm for I/O buffer placement is implemented and integrated with a power grid linear simulator written in C++. All experimental results reported in this section are obtained by testing the proposed methodology on a

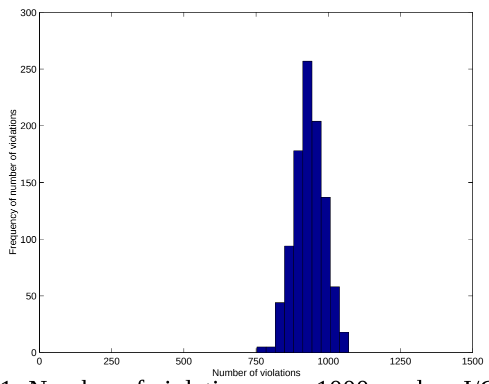

Figure 1: Number of violations over 1000 random I/O placement trials.

400 MHz ULTRA 2 Sun workstation with SunOS 5.7 operating system.

The practicality and efficiency of the proposed methodology are illustrated by testing it for the I/O buffer placement of two *real* industrial ASIC designs. We will refer to these designs as C_1 and C_2. Circuit C_1 is a 0.18μ CMOS design while C_2 is a 0.13μ CMOS design. Both designs have a supply voltage of 1.8 V. Given the power grid, the technique requires as input the currents associated with the different I/O buffers and the drop thresholds at the different nodes of the grid.

Different current measures can be used for the analysis depending on the application of interest. For instance, while peak current is a good representative measure for IR drop, average current is a better measure for electromigration analysis. Any current measure of interest can be obtained by simulating the I/O buffer under *nominal* loads and *realistic* switching factors. In all our experiments, we use the peak current drawn by an I/O buffer as our current measure. As for the drop thresholds at the different grid nodes, they are randomly assigned to vary between 1.0% and 5.0% unless otherwise specified.

In order to quantify the size of the problem being solved, we use the two input parameters, the number of I/O buffers to be placed, K, and the number of power grid nodes, N. As a first experiment, we apply the proposed methodology to place the I/O buffers in each of the given two designs. Table 3.2 shows the CPU run times required by the proposed algorithm to generate a feasible solution (including the time to invert the matrix). The first column indicates the design name. The second and third columns give the number of buffers, K, and the number of nodes, N, respectively. Column 4, reports the number of nodes where the drop thresholds are not satisfied. If a feasible placement is found, then this number is 0. The last column reports the execution time in CPU minutes.

Table 3.2 shows that, indeed, the proposed algorithm *efficiently* finds an I/O placement that satisfies all the user-specified thresholds. In order to verify the results of the algorithm, the I/O buffers are placed according to the proposed solution returned by the algorithm. Then, a circuit simulator is used to simulate the power grid with the given placement of the I/O buffers and the number of drop threshold violations is recorded. In both cases, the algorithm returned a feasible solution with 0 violations.

A question that comes up at this point is whether any random placement may find a feasible solution. To answer this question, our second experiment consists of trying 1000 different random placements of the I/O buffers in an attempt to find a feasible solution. This experiment is applied on design C_2 which consists of 3325 grid nodes and 588 I/O buffers. For a clear comparison between the proposed algorithm and the random placements, a tight drop threshold of 1.0% is specified at all the grid nodes. The proposed algorithm finds a feasible placement in 3.1 CPU minutes while all the 1000 random attempts fail to provide a feasible solution. The histogram in Figure 1 shows the distribution for the number of violations over all the 1000 trials. It is clear from this histogram that the minimum number of violations over all 1000 trials is larger than 750 nodes. That is, at least 750 nodes out of the total 3325 nodes of the grid suffer from drop violations if random placement is attempted. Thus, random placements may fail to find a feasible solution even if one exists.

5 Conclusion

In this paper, we have motivated and discussed the need for an *electrically sound* I/O placement methodology. By proper modeling of the power grids, source, and drains, the problem is defined mathematically and formulated as an ILP problem. To avoid the complexity of an ILP solution, a greedy heuristic is proposed and tested on two *real* ASIC designs with good results.

References

[1] The 1999 International Technology Roadmap for Semiconductors. Semiconductor Industry Association, 1999.

[2] A. Dharchoudhury et. al. Design and analysis of power distribution networks in *PowerPC^{TM}* microprocessors. 35th Design Automation Conference, 1998.

[3] S. R. Nassif and J. N. Kozhaya. Fast power grid simulation. 37th Design Automation Conference, 2000.

[4] K. L. Shepard and V. Narayanan. Noise in deep submicron digital design. IEEE/ACM International Conference on Computer Aided Design, San Jose, CA, 1996.

[5] P. H. Buffet et. al. Methodology for I/O cell placement and checking in ASIC designs using area-array power grid. IEEE Custom Integrated Circuits Conf., 2000.

[6] H. H. Chen and J. S. Neely. Interconnect and circuit modeling techniques for full-chip power noise analysis. IEEE Transactions on Components and Packaging II, 1998

[7] L. T. Pillage, R. A. Rohrer, and C. Visweswariah. *Electronic and System Simulation Methods*. McGraw-Hill, 1995.

[8] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[9] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Academic Press, NY 1996.

[10] N. V. Sahinidis. *BARON: Branch and Reduce Optimization Navigator*. User's manual, University of Illinois, June 2000. http://archimedes.scs.uiuc.edu