Early Detection of Current Hot Spots in Power Gated Designs

Dipanjan Sengupta[1,2], Erhan Ergin[2], Andreas Veneris[1]
[1]University of Toronto, ECE Department, Toronto, ON M5S 3G4
[2]Advanced Micro Devices,33 Commerce Valley Dr. East Markham, ON L3T 7N6
(dipanjan.sengupta@utoronto.ca, Erhan.Ergin@amd.com, veneris@eecg.utoronto.ca)

***Abstract*—With the growing popularity of hand-held battery-powered devices, leakage power is a major concern in the nanometer CMOS era. Power gating technique is an effective and widely adopted solution to this problem. The challenge of implementing power gating is the sizing and placement of the sleep transistors that are used to gate the power supply. In a placed design, due to non-uniform current demand of logic cells, some regions of the chip can have sleep transistors with very high current demand, causing power grid noise violations. Identifying these regions early in the design cycle is critical to the success of power gating implementation. This paper presents a novel methodology to calculate the current demand of each sleep transistor and locate regions in the chip where multiple sleep transistors experience very high current demand. In this paper, we model the spatial locality of the current drawn by each logic cells in the form of a bounding box. We explore techniques to identify the appropriate size of the bounding boxes. Furthermore, we extend the current distribution technique to handle placement blockages that do not share the sleep transistor network of the chip. Experimental results on industrial circuits show that the proposed algorithm can identify over 90% of such regions with a 20x run-time reduction compared to state-of-the-art commercial CAD tool.**

***Index Terms*—Distributed Sleep Transistor Network , Discharge Current, kd-Tree, Multiple Power Gated Domains**

I. Introduction

High power consumption is one of the major impediments to the advancement of VLSI designs in the nanometer CMOS era. Device scaling, and the associated reduction of threshold voltage, channel length, and gate oxide thickness [1] have introduced different forms of leakage current and caused static power to be a dominant part of the total power consumption of the chip. Sub-threshold leakage, one of the major contributors to static power, can be reduced by disconnecting a logic block from either the ground or supply voltage during idle mode of operation using sleep transistors. This technique is commonly referred to as power gating.

Power gating is implemented by placing sleep transistors between chip-level power grid and the virtual power grid. In general, the sleep transistor network should be able to deliver the maximum current demand of the design without incurring a performance loss. However, in a placed design due to locality of current distribution, not all sleep transistors have equal current demand. This uneven distribution leads to local clusters of logic cells with higher current requirement than the maximum discharge current of the local sleep transistors. We refer such regions as *Discharge Current Hot spots (DCHs)*. This can lead to power grid noise violations (IR drop) and commensurate performance loss. Identifying such regions during the design sign-off stage leads to unwanted changes in the design that can reduce the potential power savings. This paper presents a new algorithm to identify DCHs in a placed design early in the design cycle such that design engineers have greater flexibility in taking corrective measures without impacting overall power savings. Experiments show a large performance gain of the proposed methodology as compared to the current state-of-the-art industrial tools.

Power gate insertion methodologies optimize simultaneously the size and position of the sleep transistors. While a smaller sleep transistor leads to unacceptable performance loss, a large sleep transistor leads to significant area and power overhead, thereby negating the purpose of the power gating technique. Based on the granularity of the blocks, different sleep transistor networks have been proposed. Module-based sleep transistors are inserted at the root of the power distribution network of large modules [2]. A fine-grained sleep transistor insertion approach is proposed in [3]–[7] where sleep transistors are wired together forming a *Distributed Sleep Transistor Network (DSTN)*. In reality, the placement of the sleep transistor is highly constrained by custom layout design rules [8]. Thus, insertion of sleep transistors is done prior to automatic place and route of other logic cells [9]. Such an approach is widely used in industrial designs and is the adopted power gating technique.

The goal of this work is to quickly locate DCHs in a *placed* design. In detail, we make the following contributions:

1) We present a novel algorithm for calculating the maximum discharge current demand of each sleep transistor in a standard cell-based placed design and identify potential discharge current hot spots.
2) The different parameters that affect the current distribution among the sleep transistors are explored.
3) We extend our algorithm to handle discontinuity in DSTN due to presence of hard IP blocks such as memory and I/O blocks that have their own power grid.

Results show that, on average, our approach can accurately identify 90% of DCHs and outperform existing techniques by 20x.

The rest of the paper is organized as follows. Section II discusses sleep transistor architectures and the locality property of current distribution. Section III presents a kd-Tree-based data structure for representing the DSTN. The discharge current calculation for each sleep transistor is presented in Section IV. Section V shows our proposed technique to include the effects discontinuity in DSTN due to IP blocks. Section VI reports experimental results, followed by conclusion in Section VII.

II. Sleep Transistor Placement

To facilitate the power gate implementation, sleep transistors are inserted before placing the design. Different insertion architectures have been proposed for sleep transistor insertion.

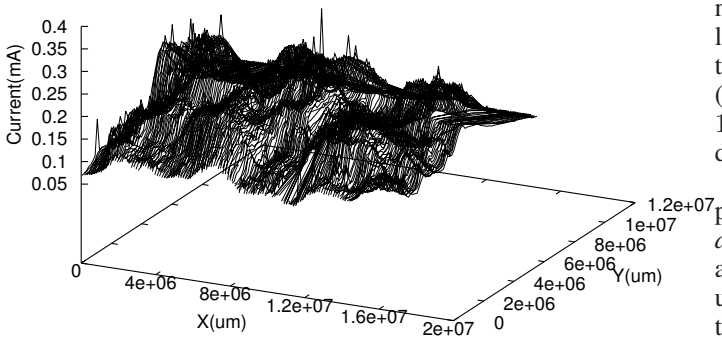


Fig. 1. Maximum Discharge Current

In [10], sleep transistors are inserted as a ring around the logic block to be power gated. Recently, row-based approach has been proposed in [11] in which dedicated rows are inserted for placing sleep transistors. However, the most common sleep transistor architecture is a *grid* [12]. Such an implementation reduces the effects of process variation and introduces less IR drop variation [8].

Assuming equal current demand across the layout, the size of sleep transistors is calculated based on worst-case power [13]. In reality, the sleep transistors encounter unequal current demand. Fig. 1 illustrate this situation. The discharge current demand of sleep transistors in a placed design with $\sim$1 million logic cells is shown in Fig. 1. We observe that the current distribution is non-uniform with regions in the floorplan having sleep transistors with very high current demand (the peaks in Fig. 1). The current demand is based on a generic activity that tries to identify hot spots. Typical applications show similar uneven current distribution.

Definition 1 *Discharge current hot spot is a region of the floorplan with multiple sleep transistors residing contiguously to one another and having current demand greater than the maximum discharge current.*

As logic cells located in the DCH starve for charge, these regions can lead to IR-drop violations. Thus current demand of the sleep transistors residing within DCH must be reduced. Modification of the logic cell placement is one way of addressing this problem. However, such modification can adversely affect the timing and thus is highly discouraged. Solving this problem by up-sizing sleep transistors is proposed in [14]. Changing wire width and adjustment of fake vias are some of the other techniques used to address this problem [15]. However, these methods either are not scalable for industrial designs or require detailed information of the design that can be obtained only towards the end of the design cycle.

The focus of this paper is detecting DCHs $early$ in the design cycle so that necessary steps such as decap insertion, power grid wire width adjustment etc. can be taken well before the design is close to tape-out. Our first step is to model the DSTN as a kd-Tree and then use this data structure for calculating the discharge current for each sleep transistor.

III. DSTN MODELLING USING KD-TREES

In a standard cell-based power gated design, each logic cell acts as a sink, drawing current from virtual supply network (VV_{DD}). Sleep transistors deliver this current from the supply network (V_{DD}). For a given design, let N be the number of logic cells and M be the number of power gating cells. Due to the locality property, current is drawn by each logic cell $(n_i, i = 1, \ldots, N)$ from a set of sleep transistors $(S_i, i = 1, \ldots, N)$ in it's vicinity. The goal is to compute S_i and then distribute the current among the sleep transistors in S_i.

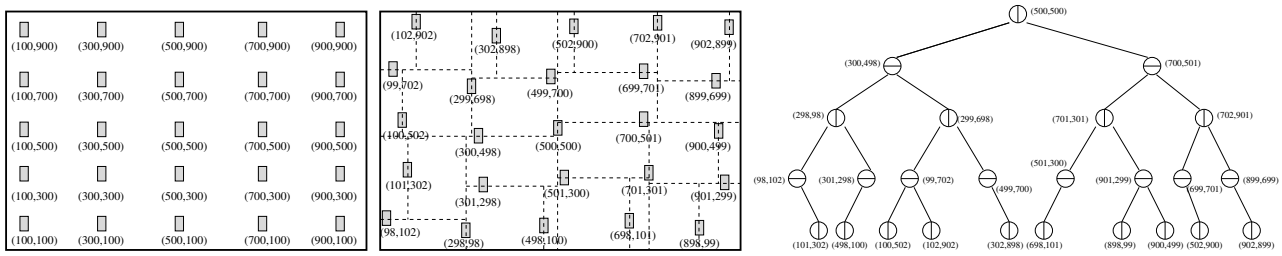
This problem can be modeled as a nearest neighbor search problem in which the search space can be modeled as a *multi-dimensional binary search tree,* or *kd-Tree* [16]. In principle, a kd-Tree is very similar to a binary tree in which the underlying space is partitioned based on just one value of all the d dimensions. Because the floorplan is a two-dimensional Euclidean space, we build a 2-d tree using x and y coordinates of points as keys in a strictly alternating sequence. Given the set of coordinate C of M sleep transistors, the root of the kd-Tree vertically splits the set C into roughly two equal halves. This is done by finding the median x coordinate of the points in C. The coordinate on the splitting line is the root of the tree. All coordinates to the left of the root reside in the left sub-tree and all coordinates to the right of the root resides in the right sub-tree. Next, each sub-tree is split along the y coordinate where the root node of the sub-tree is the median of all the y coordinates in the sub-tree. All points below the point at the root of the sub-tree go to its left sub-tree; all those above, to it's right sub-tree. This process of splitting the coordinates along the x-axis and y-axis is performed iteratively until all nodes have been added to the tree.

A kd-Tree construction requires that there exist only one point on every splitting line. However, due to the regular structure of the DSTN, sleep transistors are aligned to the power grid. This causes multiple sleep transistors to have the same x or y coordinate. To circumvent this problem, we modify the coordinate of each sleep transistor by a small amount (ϵ) such that no two sleep transistors have the same x or y coordinate. This can be expressed as follows:

$$\begin{aligned} &(x^{'}, y^{'}) = (x \pm \epsilon, y \pm \epsilon) \\ &\text{such that } \forall x_1, x_2 \in X_P : x_1 \neq x_2 \\ &\text{and } \forall y_1, y_2 \in Y_P : y_1 \neq y_2 \end{aligned} \tag{1}$$

where X_P and Y_P are x and y coordinate vectors of the sleep transistor locations. Due to high resolution of the floorplan, the minimum distance of the adjacent sleep transistors is much larger than ϵ. Thus, such a change has no effect on the current calculation of the sleep transistors. The following example explains the kd-Tree construction of a DSTN.

Example 1 *Let the sleep transistors be located in positions as shown in Fig. 2(a). Fig. 2(b) shows the necessary modification of the locations as well as the resulting partition of space for kd-Tree construction. The corresponding kd-Tree is shown in Fig. 2(c). The location (500,500) is chosen as the root of the kd-Tree because its x coordinate is median of all x coordinates. The dotted line passing through (500,500) splits the floorplan into two equal halves with sleep transistors residing to the left(right) of the line reside in the left(right) sub-tree of the root node. For the left sub-tree we split the floorplan along the y axis at (300,498). Sleep transistors residing below(above) the splitting line reside in the left(right) sub-tree of (300,498). We continue splitting the floorplan and build the rest of the kd-Tree. The horizontal or vertical line segment at each node in the tree defines the splitting plane at that node.*



(a) Original location (b) Modified location (c) kd-Tree

Fig. 2. Location modification and corresponding kd-Tree

The average time to build the kd-Tree is on the order of $O(M \cdot logM)$ [16] where M is the number of power gating cells. We find the sleep transistors in the vicinity of each logic cell using this data structure, as presented in the next section.

IV. Range Search

This step computes the discharge current demand of each sleep transistor and detects DCHs. The current distribution in present chips tends to be locally uniform and globally non-uniform. This property is referred as spatial locality and is utilized in power grid design [17]–[19]. Because sleep transistors act as switches between the virtual and chip's power/ground network, the current to each logic cell is supplied by sleep transistors in its vicinity. Therefore the principle of locality is applied in finding the discharge current load of individual sleep transistors.

We divide the floorplan into N overlapping regions, where N is the number of logic cells in the design that share the same DSTN. Each region has one logic cell in its center acting as the current sink; all sleep transistors residing within this region act as the current source. These regions are modeled as bounding boxes. Fig. 3 demonstrates the current flow from the sleep transistors surrounding the logic cell in the form of a bounding box. Based on the sleep transistors residing within the bounding box, logic cells $L1$ and $L2$ draw current from sleep transistors $S1, S2, S3, S4$ and $S4, S6$ respectively. Utilizing the spatial locality property, we distribute the current in inverse proportion to the distance between the logic cell and the sleep transistors. Thus, sleep transistors located closest to the logic cell have higher contribution to the current demand. Note that the size of the bounding box is different for the two logic cells. We present our technique of calculating the size in the next sub-section.

To find the sleep transistors within the bounding box, we first compute the four corners of the box. Let (x, y) be the coordinate of a logic cell. Assuming equal height and width of the bounding box, let $maxD$ be the dimension of the bounding box. Starting from the bottom left corner, the four corners of the box are $(x - maxD/2, y - maxD/2), (x + maxD/2, y - maxD/2), (x + maxD/2, y + maxD/2), (x - maxD/2, y + maxD/2)$ respectively in clockwise direction. Starting from the root, for each node we test the point against the range along the splitting dimension. If the x or y coordinate of the node falls within the bounding box, then we have to search both the right and left sub-trees; otherwise, we traverse to the right/left sub-tree depending on the value being greater/less than the range. The following example explains the search technique in the kd-Tree in Fig. 2(c).

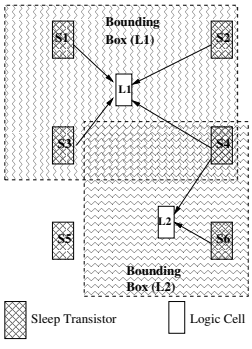


Fig. 3. Bounding Box for Each Logic Cell

Example 2 *Let the coordinates of the logic cell be (310,290) and $maxD$ be 20. Then, the x range and y range are [300,320] and [280,300]. Starting from the root node, we traverse to the left sub-tree because the x coordinate of the root is greater than x range (dotted line in Fig. 4). The y coordinate of the node (300,498) falls within the y range, but the x coordinate is beyond the x-range. Therefore we first traverse the left sub-tree followed by the right sub-tree. Because the node (298,98) is beyond the range, we traverse only to the right sub-tree. The node (301,298) falls within the range and is appended to the location of the sleep transistors in the vicinity of (310,290). In similar fashion we visit the nodes (498,100), (299,698), (99,702), and (100,502). Finally, (301,298) is the only node found within the bounding box (shaded node in Fig. 4).*

The kd-Tree data structure acts as a pruning device of the search space. For example all nodes in the right sub-tree of the root node in Example 2 need not be examined at all. This makes the range search extremely efficient. The run-time for the range search is $O(2 \cdot \sqrt{M} + F)$ where F is the number of nodes found within the bounding box [20].

Using this search technique, we calculate the discharge current demand of each sleep transistor. Next, we identify DCHs as regions in the DSTN having higher current demand than the maximum discharge current.

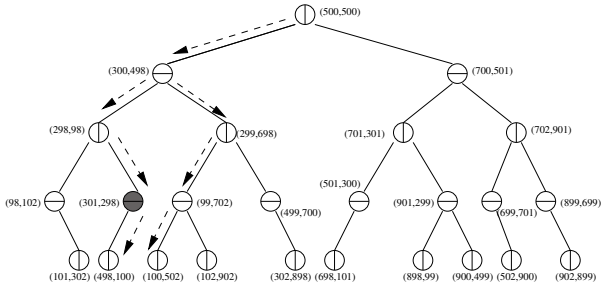


Fig. 4. Range Search in kd-Tree

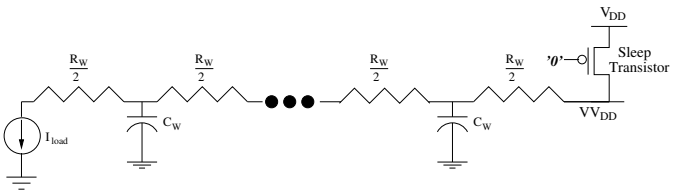


Fig. 5. Power Grid Model

A. Max Distance Calculation

The size of the bounding box is critical to the current distribution among the sleep transistors. In reality, the size of the bounding box is problem dependent. Having a large bounding box leads to unnecessary run time due to an overly conservative choice of current distribution.

To overcome this problem, we employ a novel technique for finding the height/width of the bounding box ($maxD$). We model the logic cell as a current source and the virtual power grid connecting the sleep transistor with the logic cell as a distributed RC wire. Fig. 5 depicts the worst-case scenario in which a single sleep transistor delivers the current to the logic cell. I_{load} is the load current of the logic cell, and R_w and C_w are the wire resistance and capacitance of unit length.

Next, we simultaneously modify the switching frequency, average current demand of the logic cell, and the distance between the logic cell and the sleep transistor and explore their effects on IR drop in the virtual power grid. Fig. 6 shows the average IR drop variation ($V_{DD} = 1.0V$) with the change of the above parameters using SPICE simulation. The combined effect of these design parameters cause $maxD$ to vary (as shown inFig. 6). Thus, using a fixed $maxD$ will cause significant error in the current calculation. The problem is acute when the fixed value is less than the variable $maxD$ for a high current consuming logic cell. The current calculation will cause higher current demand for few sleep transistors in the vicinity of the logic cell.

For a given frequency and average current, we select $maxD$ such that a sleep transistor residing beyond $maxD$ will cause an IR drop violation greater than 10% [21] of V_{DD} (0.1V) in Fig. 6). The frequency of switching can be extracted easily from the top-level design by identifying the clock domain for the design block in which the logic cell resides. Also, average current is computed from the power consumed by the logic cell that can be derived by simulating the design. We consider each logic cell switches at the positive clock edge of the clock domain in which it resides.

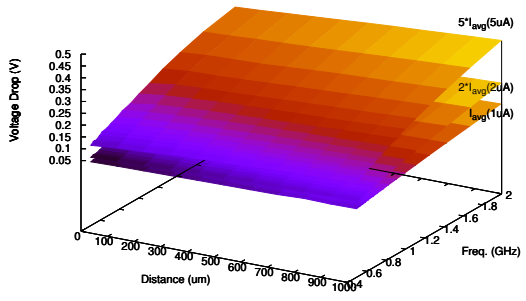


Fig. 6. Effect of Distance, Frequency, and Average Current on IR Drop

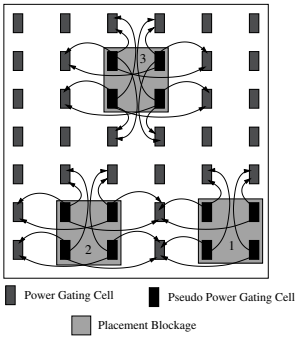


Fig. 7. Pseudo Sleep Transistor insertion

V. Placement Blockage

In general hard IP blocks do not share the power grid network with the rest of the design and act as placement blockage to the DSTN. This causes discontinuity in the DSTN. The power and ground rails, routed over these blocks, connect the power gating cells that surround them. Although these power gating cells are physically located far from one another, the resistance between them is small. Moreover there is no logic cell present between them. Thus, logic cells placed near the placement blockage are likely to draw current from the sleep transistors present at the opposite end of the IP block. In order to include placement blockage, we insert pseudo sleep transistors (PST) in the blockage region and consider them as current sources. The location of each PST is determined by the horizontal and vertical spacing between neighboring sleep transistors in the original DSTN. Moreover each PST coordinate must be unique as they are added to the kd-Tree structure of the DSTN. Fig. 7 shows three different scenarios of placement blockage. The PST placed within the IP block location 1 act as current source for sleep transistors that are directly above it or to it's left (shown as directed edges). In contrast, three sides of the IP block have sleep transistors for IP block at location 2. Thus three sleep transistors are associated with each pseudo sleep transistor within location 2. All four sides of the IP block at location 3 have sleep transistors and each PST is associated with four sleep transistors. In each case, the directed edges between each pseudo sleep transistor and the sleep transistors indicate the association.

The current demand calculation uses both the sleep transistors and pseudo sleep transistors as current sources. However, as PST are not present in the original DSTN, we need to re-distribute the current to the sleep transistors. The association of each PST is used for this purpose. From the set of sleep transistors associated to each pseudo sleep transistor, we remove those sleep transistors that are located within the bounding box of the logic cell. This step ensures that we do not consider the sleep transistors twice, once when the current is distributed among sleep transistors including PST and when we are redistributing the current of the pseudo sleep transistor. Using the same distance-based metric, the current to the pseudo sleep transistor is re-distributed to the remaining sleep transistors.

The overall design methodology is presented in Algorithm 1. Starting with a placed design PL, the functions ExtractST() and ExtractLC() reads the location of logic cells and sleep transistors for each domain. IP is the set of hard IP blocks in the design. We modify the location of the

Algorithm 1 DCH Detection and Correction

input PL: placed design with sleep transistors
input D: number of power domains
input IP: set of IP blocks present in the design
input I: current demand of each logic cell in PL
input F: clock domain of each logic cell in PL
input P: distance, frequency and current relation
output DCH: location of all DCHs
$n^i_{(x,y)}$: logic cell at location (x, y) in domain i
$DCH = \emptyset$
for each domain $i \in D$ **do**
  $[X^i_P, Y^i_P]$ = ExtractST(PL)
  $[X^i_L, Y^i_L]$ = ExtractLC(PL)
  ModifyST($[X^i_P, Y^i_P]$)
  K_i = BuildkdTree($[X^i_P, Y^i_P]$)
  if $IP \neq \emptyset$ **then**
    InsertPST(X^i_P, Y^i_P, IP)
    K_i = BuildkdTree($[X^i_P, Y^i_P]$)
  end if
  for each $(x, y) \in [X^i_L, Y^i_L]$ **do**
    $maxD$ = FindDist($P, F, n^i_{x,y}$)
    $S^i_{(x,y)}$ = FindST($K_i, maxD$)
    ST_i = DistCur($S_(x, y)^i, LT_i, n^i_{x,y}$)
  end for
  if $IP \neq \emptyset$ **then**
    RedistCur(ST_i, X^i_P, Y^i_P)
  end if
end for
for each domain $i \in D$ **do**
  DCH^i = ComputeDCH(ST_i)
  if $D > 1$ **then**
    Swap($[X^i_P, Y^i_P], ST_i$)
    DCH^i = ComputeDCH(ST_i)
  end if
  $DCH = DCH \bigcup DCH^i$
end for
return (DCH)

sleep transistors using the function ModifyST() and then build each kd-Tree K_i, as presented in Section III. In the presence of IP blocks, InsertPST() inserts the PST. The function FindDist() calculates $maxD$ for each logic cell. Range search is performed by FindST() and the current is distributed using the distance metric in DistCur(). The function RedistCur() redistributes the current in the PST to the neighboring sleep transistors. Based on the current of each sleep transistor, ComputeDCH() identifies each DCH. Next, based on the vicinity of these sleep transistors we identify the DCHs. The output of the proposed algorithm is the set DCH that lists all the DCHs in the design.

VI. Experimental Results

The validity of the proposed approach to identify and mitigate DCHs is presented in this section. The algorithm is implemented using Python and the computations were performed on a Unix workstation with 3 GHz CPU and 18 GB of RAM. In our experiments we use 28-nm technology with $V_{DD} = 1.0V$.

At first, the $maxD$ is calculated using HSPICE simulations with the current source modeled as an inverter. For simplicity, VV_{DD} is modeled as Metal 1 wire as it is used in the DSTN connecting the logic cells to the sleep transistors. The variation of IR drop with current, frequency, and distance is noted. These values are used to select $maxD$ for each logic cell during average current calculation of the sleep transistors. The proposed approach is run on 12 industrial benchmark circuits. The designs are synthesized using Synopsys Physical Compiler. The current load of each logic cell in the gate-level netlist is derived from simulations using Synopsys Power Compiler. OpenAccess Database is used to extract the position of sleep transistors and logic cells. We inserted PST within the placement blockages.

The commercial tool* used for comparison also uses the placed design as it's input. In addition to extracting the design related information, such as location of logic and power gating cells, the tool also gathers detailed routing information. Next, it runs fast SPICE simulation by modeling each logic cell as a simple current source connected to the RC-network generated from the routing information. Similar to the proposed work, the current sources are derived from the power information available from previous simulation runs. It reports the current demand of each power gating cell.

Table I shows the results of of our algorithm. The first five columns provide design-related information such as design name, number of logic cells, number of power domains, number of pre-placed sleep transistors in each domain, and the number of IP blocks. For each design with multiple domains, the number of logic cells and sleep transistors in each domain is mentioned individually. The number of pseudo sleep transistors required for each design is reported in column six. The next two columns indicate the number of DCH found for each domain using the proposed algorithm and the industrial tool. The next column reports the number of matching DCH using both the methods.

We define accuracy as the number of DCH that are common in both the techniques and report it in the next column. In all the designs, our method identified equal or greater number of DCH than the commercial tool. In the additional DCHs found by our method, the commercial tool reported relatively high average current demand. Addressing these regions is beneficial in terms of reducing over all fluctuation in current demand across the chip. The run time of our method and that of the industrial tool is reported in the following two columns. The last column indicates the relative speed-up achieved using the proposed method. On average, there is a run-time speedup of 20x with good accuracy in identifying the DCH locations.

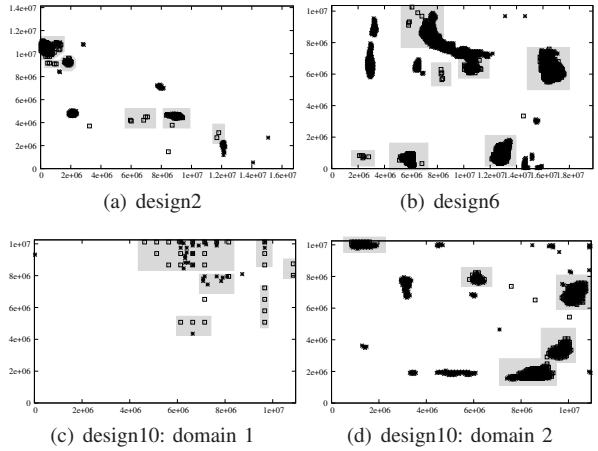
To measure the quality of our solutions, the location of DCH in three circuits in Table I, identified using the proposed method and the industrial tool, is shown in Fig. 8. The location of sleep transistors violating maximum discharge current detected by the industrial tool and the proposed method are indicated by '□' and '$*$' respectively. The shaded regions are the DCH found using the industrial tool.

Fig. 9 compares the runtime of our method with the number of logic cells. The positive linear correlation highlights the benefit of using the kd-Tree model for searching power gating cells. Thus, future technology generations with larger logic cells will benefit from this approach.

*Name of vendor cannot be disclosed due to legal agreement.

TABLE I
DCH RESULTS

Circuit Info						# DCH				Run time		
Instance name	# logic cells	# power domains	# Sleep Transistors	#IP block	#PST	Industrial Tool	Proposed Method	Match	Accuracy	Industrial Tool	Proposed Method	Improvement
design1	493K	1	5138	5	12	2	3	2	100%	4h 23min	27min 19s	9.7x
design2	1.32M	1	15620	20	36	5	7	5	100%	6h 22m	13in 13s	29.4x
design3	1.34M	1	13425	11	13	5	8	5	100%	4h 49min	12min 27s	24.08x
design4	207K	1	2490	14	46	2	4	2	100%	1h 55min	9min 10s	12.77x
design5	1.28M	1	14368	19	29	2	4	2	100%	4h 57min	11min 4s	27x
design6	1.01M	1	10885	18	33	6	9	6	100%	5h 50min	16min 1s	21.87x
design7	1.4M	1	16793	34	60	1	3	1	100%	38h 46min	1h 33min	25.01x
design8	753K	1	7981	11	65	4	8	3	75%	5h 14min	12min	26.16x
design9	1:14K 2:475K	2	1:240 2:5094	7	1:4 2:16	1:2 2:3	1:5 2:5	1:2 2:2	80%	4h 2min	11min 50s	20.1x
design10	1:9K 2:935K	2	1:448 2:9410	7	1:3 2:124	1:6 2:5	1:5 2:8	1:4 2:5	81%	4h 59min	16min 52s	17.58x
design11	1:14K 2:475K	2	1:240 2:5098	5	1:4 2:16	1:2 2:3	1:5 2:5	1:2 2:2	80%	1h 16min	6min 33s	11.69x
design12	1:183K 2:479K	2	1:275 2:5063	7	1:4 2:19	1:2 2:3	1:2 2:6	1:2 2:2	80%	2h 13min	9min 20s	14.3x
AVG									**93%**			**20x**



(a) design2 (b) design6

(c) design10: domain 1 (d) design10: domain 2

Fig. 8. Comparison of DCH

VII. CONCLUSION

A kd-Tree-based range search technique for identifying discharge current hot spots has been proposed in this paper. Locality-based current distribution using a bounding box was explored. Identifying these regions early in the design cycle gives greater leverage to fix the issues with minimal impact to power saving goals. Experimental results on industrial benchmark circuits show that the overall methodology is highly effective in identifying the DCHs and it is also considerably faster compared to state-of-the-art industrial solutions. Future work would concentrate on considering the presence of decaps in the design that act as local source of ccharge there-by reducing the impact on the power gating cells.

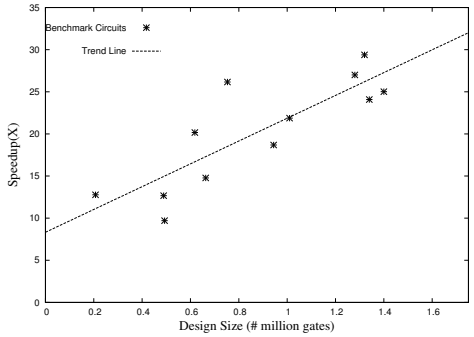


Fig. 9. Comparison of runtime with design size

REFERENCES

[1] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits," *Proc. of the IEEE*, vol. 91, no. 2, pp. 305–327, 2003.

[2] J. Kao, S. Narendra, and A. Chandrakasan, "Mtcmos hierarchical sizing based on mutual exclusive discharge patterns," in *Design Automation Conf.*, 1998, pp. 495–500.

[3] V. Khandelwal and A. Srivastava, "Leakage control through fine-grained placement and sizing of sleep transistors," in *Int'l Conf. on CAD*, 2004, pp. 533–536.

[4] C. Hwang, P. Rong, and M. Pedram, "Sleep transistor distribution in row-based mtcmos designs," in *Great Lakes Symp. VLSI*, 2007, pp. 235–240.

[5] D.-S. Chiou, D.-C. Juan, Y.-T. Chen, and S.-C. Chang, "Fine-grained sleep transistor sizing algorithm for leakage power minimization," in *Design Automation Conf.*, 2007, pp. 81–86.

[6] L. Guo, Y. Cai, Q. Zhou, L. Kang, and X. Hong, "A novel performance driven power gating based on distributed sleep transistor network," in *Great Lakes Symp. VLSI*, 2008, pp. 255–260.

[7] A. Sathanur, A. Pullini, L. Benini, A. Macii, E. Macii, and M. Poncino, "Optimal sleep transistor synthesis under timing and area constraints," in *Great Lakes Symp. VLSI*, 2008, pp. 177–182.

[8] K. Shi, Z. Lin, Y.-M. Jiang, and L. Yuan, "Simultaneous sleep transistor insertion and power network synthesis for industrial power gating designs," *Journal of Computers*, vol. 3, no. 3, 2008.

[9] L. K. Yong and C. K. Ung, "Power density aware power gate placement optimization scheme," in *ASP Design Automation Conf.*, 2010, pp. 38–42.

[10] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low Power Methodology Manual: For System-on-Chip Design*. Springer Publishing Company, Incorporated, 2007.

[11] A. V. Sathanur, L. Benini, A. Macii, E. Macii, and M. Poncino, "Row-based power-gating: A novel sleep transistor insertion methodology for leakage power optimization in nanometer cmos circuits," *IEEE Trans. on VLSI Systems*, vol. 19, no. 3, pp. 469–482, 2011.

[12] K. Shi and D. Howard, "Challenges in sleep transistor design and implementation in low-power designs," in *Design Automation Conf.*, 2006, pp. 113–116.

[13] F. Li and L. He, "Maximum current estimation considering power gating," in *Int'l Symp. on Physical Design*, 2001, pp. 106–111.

[14] Y. Xu and G. K. Yeap, "An mtcmos power network design flow," in *ASP Design Automation Conf.*, 2009, pp. 266–269.

[15] C. Long, J. Xiong, and L. He, "On optimal physical synthesis of sleep transistors," in *ISPD*, 2004, pp. 156–161.

[16] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sept. 1975.

[17] S. Pant and E. Chiprout, "Power grid physics and implications for cad," in *Design Automation Conf.*, 2006, pp. 199–204.

[18] Y. Zhong and M. D. F. Wong, "Fast algorithms for ir drop analysis in large power grid," in *Int'l Conf. on CAD*, 2005, pp. 351–357.

[19] Z. Zeng and P. Li, "Locality-driven parallel power grid optimization," *IEEE Trans. on CAD*, vol. 28, no. 8, pp. 1190–1200, Aug. 2009.

[20] D. T. Lee and C. K. Wong, "Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees," *Acta Informatica*, vol. 9, pp. 23–29, 1977.

[21] A. H. Ajami, K. Banerjee, A. Mehrotra, and M. Pedram, "Analysis of ir-drop scaling with implications for deep submicron p/g network designs," in *Int'l Symp. on Quality Electronic Design*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 35–40.