

Voltage Island-Driven Floorplanning

Qiang Ma and Evangeline F. Y. Young
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Email: {qma,fyyoung}@cse.cuhk.edu.hk

Abstract—Energy efficiency has become one of the most important issues to be addressed in today’s System-on-a-Chip (SoC) designs. One way to lower the power consumption is to reduce the supply voltage. Multi-supply voltage (MSV) is thus introduced to provide higher flexibility in controlling the power and performance trade-off. In region-based MSV, circuits are partitioned into “voltage islands” where each island occupies a contiguous physical space and operates at one supply voltage. These tasks of island partitioning and voltage level assignments should be done simultaneously in the floorplanning process in order to take those important physical information into consideration. In this paper, we consider this core-based voltage island driven floorplanning problem including islands with power down mode, and propose a method to solve it. Given a candidate floorplan solution represented by a normalized Polish expression, we are able to obtain *optimal* voltage assignment and island partitioning (including islands with power down mode) *simultaneously* to minimize the total power consumption. Simulated annealing is used as the basic searching engine. By using this approach, we can achieve significant power savings (up to 50%) for all data sets, without any significant increase in area and wire length. Our floorplanner can also be extended to minimize the number of level shifters between different voltage islands and to simplify the power routing step by placing the islands in proximity to the corresponding power pins.

I. INTRODUCTION

Energy efficiency has become one of the most important issues to be addressed in today’s System-on-a-Chip (SoC) designs because of the increasing power density and the wide use of portable systems. There are two kinds of power consumption: dynamic and leakage. Dynamic power is caused by the charging and discharging of the load capacitance during switching. Leakage power is due to the sub-threshold currents when a device is turned off. There are many techniques to reduce power consumption. One of the most effective methods is by lowering the voltage supply. Multi-voltage design is thus introduced to provide “just enough” power to support different functional operations. Both dynamic and leakage power consumption can be reduced in multi-voltage designs. For dynamic power, since the consumption is proportional to the square of the voltage, a minor adjustment to the voltage level can result in a significant reduction. For leakage power, the consumption can be reduced by powering down parts of a chip when the functions are inactive.

Multi-voltage designs involve the partitioning of a chip into areas called “voltage islands” that can be operated at different voltage levels, or be turned off when idle. With the use of voltage islands, the chip design process is becoming more complicated. We need to solve the problems of island partitioning, voltage assignment and floorplanning simultaneously under area, power, timing and other physical constraints. These problems must be solved at the same time since their results will significantly affect each other. In addition, there are other issues to be considered. For example, the voltage islands should be placed close to the power pins in order to minimize the power routing complexity and the IR drop. Besides, each island requires level shifters to communicate with others and overhead in area and delay will be resulted. These additional issues have created many new challenges in generating floorplans for designs using voltage islands. An example is shown in Figure 1. In this example,

the possible voltage levels of each core and groupings of similar inactive periods (to generate islands with power down mode) are shown on the right hand side. Assuming that the number of islands is three, one possible partitioning is to group cores *A*, *B* and *C* as one island operating at voltage 1.0V, core *D* on its own as one island at voltage 1.5V and cores *I*, *K*, *L* and *M* as one island at 1.2V. Notice that other cores will be operated at the chip-level voltage and the island containing *I*, *K*, *L* and *M* can be powered down during sleep. A candidate floorplan solution for such a partitioning and voltage assignment is shown on the left hand side.

There are several previous papers addressing similar voltage island-driven floorplanning problem. One recent work is by Lee *et al.* [3]. Given a netlist without reconvergent fanouts, voltage assignment (with two voltage levels of VDDL and VDDH) is first performed on the netlist according to the timing requirement before the floorplanning step. Level shifters are then inserted into the nets according to the voltage assignment result when a VDDL block drives a VDDH block. At last, a power-network aware floorplanner is invoked to pack the blocks such that the power-network resource, estimated as the sum of the perimeters of the voltage islands, will be minimized. As a result, blocks in the same voltage island will be placed close to each other. In their approach, the voltage assignment step and the floorplanning step are done separately. Hu *et al.* [2] have also considered this simultaneous island partitioning, voltage assignment and floorplanning problem in SoC designs. Simulated annealing is used as the basic searching engine. Given a candidate solution, perturbations are performed to split an island, change the voltage of an island or change all the islands of one voltage to another voltage. Chip-level floorplanning is then performed to find a floorplan in which compatible islands (islands with the same voltage) are likely to be adjacent. An island merging process is then applied to reduce the number of islands. At the end, island-level floorplanning is done to each newly formed island to shrink its area. The whole process is repeated until a satisfactory solution is obtained. Their approach does not consider islands with power down mode and the search space is large. Mak and Chen [7] have also addressed this problem on SoC designs. Given a floorplanning input, the voltage assignment and island partitioning problem is formulated as a 0-1 integer linear program. In their approach, a few candidate floorplan solutions are generated based on metrics like area and interconnect cost, then voltage assignment and partitioning are performed on these candidate floorplans using the ILP approach to identify the best candidate solution. A fragmentation cost (number of adjacent cores operating at different voltages) is used to model the power network complexity but this cost is not related to the number of islands directly. There are other works addressing issues like reliability [4] and temperature reduction [5] in SoC voltage island partitioning and floorplanning. For island partitioning, Wu *et al.* [6] and Ching *et al.* [8] minimized the number of voltage islands after placement.

In this paper, we propose a floorplanning method for SoC designs that is tightly integrated with the island partitioning and voltage

assignment steps. Simulated annealing is used with normalized Polish expression [1] as the floorplan representation.¹ Normalized Polish expression is used because the slicing tree is a suitable data structure on which island partitioning and voltage assignment can be done *optimally and efficiently* given one slicing floorplan. Simulated annealing is adopted to perform the random search. In each step of the annealing process, a candidate floorplan solution is generated on which optimal island partitioning and voltage assignment will be performed simultaneously to compute the smallest possible power consumption for that candidate floorplan solution. This is done by dynamic programming with an efficient cost table update technique. In this way, we can integrate the three steps closely, and reduce the searching space (instead of doing voltage assignment by the “move” operations of the annealing process as in [2]). In this floorplanning framework, we can also generate islands with power down mode to optimize the total power consumption further. Our floorplanner can be extended to consider the number of level shifters and the ease of power network routing (proximity to power pins and shapes of voltage islands). By using this approach, we can achieve significant power savings (up to 50%) for all data sets, without any significant increase in area and wire length.

We will define the problem in section II, then the methodology used will be discussed in section III. Experimental results will be reported in section IV before the conclusion and discussion in the last section.

II. PROBLEM FORMULATION

In this problem, we are given a set of n cores with areas $A_1, A_2 \dots A_n$ and aspect ratio bounds $[l_i, u_i]$ for $i = 1 \dots n$. Each core i is associated with a power table T_i that specifies the legal voltage levels for the core and the corresponding average power consumption values. The power table of a core can be characterized by a core designer. For example, they can run simulations that try applying different supply voltages to the core, and a voltage level will be regarded as a legal one as long as the timing assertion² can be satisfied [2]. The power consumption corresponding to each legal voltage can then be estimated. In this work, we compute the power cost of a core i operated at voltage v as $v^2 A_i$. We are also given a set of m nets $\{N_1, N_2 \dots N_m\}$ and a set of groupings $\{G_1, G_2 \dots G_p\}$ between the cores such that the cores in each group G_i have similar inactive periods and will have a $s_i\%$ saving in power consumption if they are grouped together as an island with power down mode.

Given a constant K and a chip-level voltage V_c , our goal is to generate a floorplan F with K rectangular voltage islands so that the total power consumption is minimized. Each island will be supplied with the lowest possible voltage level common to all the cores in that island while the remaining cores not assigned to any island will be operated at the chip-level voltage. Islands containing blocks all belonging to the same group G_i can have a further reduction in power consumption by $s_i\%$ by shutting it down during sleep.

III. METHODOLOGY

Our floorplanner is based on simulated annealing using normalized Polish expression (NPE) as the representation. For each candidate floorplan solution represented by an NPE, we will perform an optimal island partitioning and voltage assignment to maximize the total power saving. The cost function of the annealing process is to

¹Since many input cores have flexibilities in shape at this stage, the restricted use of slicing floorplan can also give satisfactory results.

²We assume that the given voltage levels of each core can meet timing, so we do not consider timing explicitly in our formulation.

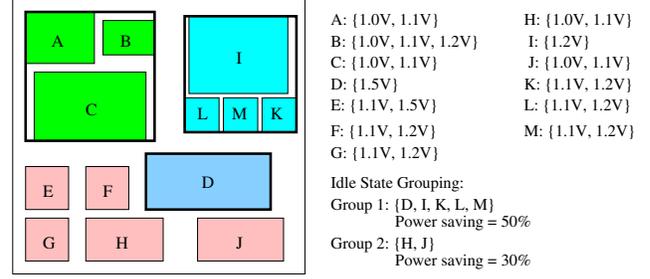


Fig. 1. An example of the voltage island driven floorplanning Problem.

minimize a weighted sum of the area, wire length and power. We can also extend our floorplanner to consider level shifters and proximity to power pins. Details will be given in the following sections.

A. Optimal Island Partitioning and Voltage Assignment

Given a candidate floorplan solution represented by a normalized Polish expression, we can construct the corresponding slicing tree and perform optimal island partitioning and voltage assignment on it. This can be done efficiently by dynamic programming. The pseudo code is shown below:

Pseudocode $TreePart(u, k)$

// Partition the subtree under node u into k subtrees to
// minimize the total power consumption such that the cores
// (leaf nodes) in each of these k subtree will form one island
// operated at one common voltage possibly with power down
// mode while the remaining cores not belonging to any of
// these k subtrees will be operated at the chip level voltage V_c

1. $min_cost = \infty$
2. If k is 0, return($power(u)$).
3. If $cost_table[u][k]$ is updated, return($cost_table[u][k]$).
4. If k is 1,
5. $C_1 = TreePart(lchild(u), 1) + power(rchild(u))$
6. $C_2 = TreePart(rchild(u), 1) + power(lchild(u))$
7. $C_3 = nonSubtree(u, 1)$
8. $C_4 = cost(\{u\})$
9. $min_cost = \min\{C_1, C_2, C_3, C_4\}$
10. Store min_cost into $cost_table[u][1]$.
11. Return (min_cost).
12. Else
13. $min_cost = nonSubtree(u, k)$
14. For $i = 0$ to k
15. $C = TreePart(lchild(u), i) +$
 $TreePart(rchild(u), k - i)$
16. If $min_cost > C$, $min_cost = C$
17. Store min_cost into $cost_table[u][k]$.
18. Return(min_cost).

At the beginning, $TreePart(root, K)$ is called to obtain an optimal island partitioning and voltage assignment of the whole floorplan, where $root$ is the root of the slicing tree corresponding to the normalized Polish expression under consideration and K is the number of voltage islands we want to produce. When k is zero (line 2), no voltage island is formed in the subtree of u , so the power consumption $power(u)$ will be computed as $(V_c)^2 A_{\{u\}}$, where V_c is the chip-level voltage and $A_{\{u\}}$ is the total area of the cores in the subtree of u . For non-zero k , we will first check whether this

optimal cost has been computed before and is available in the table for immediate return (line 3). If this value is not available, we will consider the situations when k is one and when k is larger than one separately. When k is one (line 4), there are four cases: case 1 (line 5), we continue to search for a voltage island in the left subtree of u and let the right subtree operate at the chip-level voltage V_C ; case 2 (line 6), similarly, we look for a voltage island in the right subtree of u and let all the cores in the left subtree work at V_C ; case 3 (line 7), use the function $nonSubtree()$ to group the cores across a number of subtrees along the left tree branches of u into a voltage island (details will be given in the next sub-section); case 4 (line 8), the entire subtree u is regarded as one voltage island and the power consumption $cost(\{u\})$ will be computed as $(v_{\{u\}})^2 A_{\{u\}}$, where $v_{\{u\}}$ is the smallest common voltage among all the cores in the subtree rooted at u and $A_{\{u\}}$ is the total area of the cores in the subtree rooted at u . We will compute the costs of the four cases respectively, and the smallest one will be returned and recorded in the table for future use. When k is more than one (line 12), we will recursively call the procedure $TreePart()$ to exhaust all the possible partitionings of the subtree of u , including both inter-subtree partitionings (line 13) and intra-subtree ones (line 14-16). The minimum one will be returned and recorded in the table.

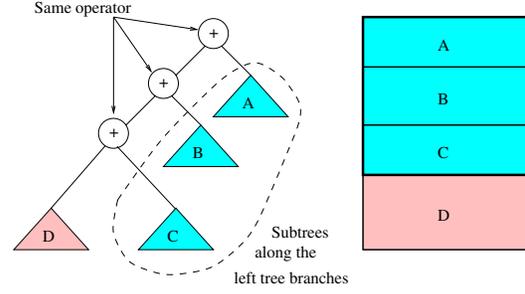
1) *Voltage Islands in Non-subtrees*: Notice that a voltage island (a rectangular region) may be formed by a set of contiguous right subtrees linked by internal nodes of the same operational type. An example is shown in Figure 2. Therefore, we need the procedure $nonSubtree()$ to enumerate these cases. The pseudo code of $nonSubtree()$ is shown in the following. In this procedure, we exhaust all the cases of forming one island with two or more contiguous right subtrees and the one with the smallest power consumption will be returned. On line 7, we compute the cost of grouping the right subtrees in S as one island and having the remaining $k - 1$ islands in the last left subtree (subtree D in the example of Figure 2).

Pseudocode $nonSubtree(u, k)$

// Exhaust the cases of forming one island by a number
// of contiguous right subtrees, while the remaining $k - 1$
// islands are formed in the remaining left subtree.

1. $min_cost = \infty$
2. $S = rchild(u)$
3. $op = operator(u)$
4. While $operator(lchild(u))$ is op ,
5. $u = lchild(u)$
6. $S = S \cup rchild(u)$
7. $C = TreePart(lchild(u), k - 1) + cost(S)$
8. If $min_cost > C$, $min_cost = C$
9. Return(min_cost).

2) *Proof of Optimality*: The procedure $TreePart()$ will give the optimal partitioning to minimize the total power consumption. Given a candidate floorplan solution represented by a normalized slicing tree rooted at u and the number of voltage islands required k , $TreePart()$ will exhaust all the possible cases recursively and return the best solution. When k is zero, there is only one case that all the cores in the tree rooted at u (called T_u) are operated at the chip voltage. When k is one, there is only one voltage island among all the cores in T_u . Three of the cases are obvious: (1) the island is in the left subtree of u , (2) the island is in the right subtree and (3) all the cores in T_u form one island. There is still a case that the island is formed between the left and right subtrees of u . This may happen only when



A, B and C, not in one subtree, can form one voltage island.

Fig. 2. An example of forming island across subtrees.

two consecutive internal nodes are of the same type (both “+” or both “*”). In a normalized slicing tree, an internal node will not be of the same type as its right child, so this will happen only along the left branch. $nonSubtree()$ will exhaust this last case of forming one island by a set of contiguous right subtrees along the left branch rooted at u . When k is larger than one, the cases are similar to those when k is one and $TreePart()$ will exhaust all different ways of distributing the k islands between the left and right subtrees of u and the case of having an island lying between the two subtrees. Since $TreePart()$ has exhausted all different cases of forming k islands in a given candidate floorplan, the solution returned by $TreePart()$ is optimal.

3) *Handling Island with Power Down Mode*: Voltage islands with power down mode can be easily handled in our framework. When computing the power consumption of an island formed with the cores in the set of subtrees rooted at a node in set X by calling $cost(X)$ in the procedures $TreePart()$ and $nonSubtree()$, we only need to check if all the cores within this island belong to one group G_i for some $i = 1 \dots p$. If this is true, the island formed can be shut down during sleep and have an additional power saving of $s_i\%$. In this way, our floorplanner can give optimal island partitioning and voltage assignment taking into account islands with power down mode given any candidate floorplan solution.

4) *Speedup in Implementation and Complexity*: A table $cost_table[v][j]$ for $j = 1 \dots K$ is kept at each internal node v of the slicing tree to record the optimal power consumption of partitioning the cores in the subtree rooted at v into j islands. This data structure can help to minimize the number of recursive calls and to avoid repetitive computations. It can be seen from the the procedure $TreePart()$ that whenever we want to find the optimal power saving at a node u with k voltage islands, we will first check whether this is computed before and the required information is available from $cost_table[u][k]$. If it is available, the optimal value is returned immediately (line 3). Otherwise, it is computed recursively and the computed value will be saved in the $cost_table$ to be used in some later steps (line 10 and 17). After a move in the annealing process, only a small part of the whole slicing tree will be changed and we only need to update the tables of the affected nodes once. The affected nodes will be those lying on the paths from the modified parts of the tree to the root. For those affected nodes, the corresponding $cost_tables$ will be flagged as “not updated” and will be updated during the recursive calls.

For each affected node v , we need to update all the K entries of its table once. Since each entry is just updated once, the time complexity will be the same as that of updating all the affected nodes in a bottom-up fashion from the leaves to the root. If the nodes were

updated from the leaves to the root, the time taken to update a table at a node v was $O(K^2)$, because there were K entries and each entry took $O(K)$ time (the tables of v 's children have already been updated). Therefore the total time to perform all the updates in each iteration is (number of affected nodes) $\times O(K^2)$. This is $O(K^2 n)$ in the worst case, and is $(K^2 \log n)$ on average.

5) *Varying Background Chip-level Voltage*: In the problem formulation, it is assumed that a background chip-level voltage V_c is given. Our approach can also be applied when there is no such voltage and the final chip-level voltage is determined by the minimum feasible voltage level among all the cores which are not grouped into any voltage island. To achieve this, we only need to expand the table at each node L times where L is the number of possible voltage levels among all the cores. In this case, each entry $cost_table[v][j][V_l]$ will be the optimal power consumption of partitioning the cores in the subtree rooted at v into j islands when the background chip-level voltage is V_l where $1 \leq l \leq L$. Notice that all entries in the tables corresponding to different chip-level voltages can be updated simultaneously during the recursive calls and the run time is only linearly scaled up by L . This is very affordable in practice since the number of possible voltage levels L is usually small.

B. Moves

There are three kinds of moves to change the normalized Polish expression of a candidate floorplan solution in the annealing process. This set of moves has been proven to be complete to change any arbitrary solution to any other arbitrary solution.

- 1) **Swap** - Swap two adjacent blocks.
- 2) **Complement** - Complement a chain of operators.
- 3) **SwapOp** - Swap a block with its adjacent operator.

C. Cost Function

We use the cost function $\psi = A + \lambda_w W + \lambda_p P$ to evaluate a floorplan where A is the area of the floorplan, W is the total wire length estimated by the half perimeter bounding box method and P is the total power consumption. The parameters λ_w and λ_p are the weights which will be set at the beginning of the annealing process by random walks to make the three terms similar in weighting. This cost function can be modified to consider the fixed-outline constraint by replacing the area term A (since we are not minimizing the area) by $\lambda_\infty (\max\{0, w - W'\} + \max\{0, h - H'\})$ where λ_∞ is a large positive constant, w and h are the width and height of the candidate floorplan solution, and W' and H' are the fixed width and height required for the final floorplan design.

IV. EXPERIMENTAL RESULTS

We have done experiments on the GSRC floorplanning benchmarks. Since no voltage information is provided in those benchmarks, we have randomly generated the voltage levels for each block from the set $\{1.0V, 1.1V, 1.2V, 1.3V, 1.5V\}$ and 1.5V is assumed to be the chip-level voltage. In each data set, groups of blocks with similar inactive periods are also randomly generated. Table I shows the details of each data set, the fourth column indicates the grouping information of similar inactive periods, e.g., for n30, there are two groups: one contains five cores with an additional 30% power saving if being grouped together, and the other one contains five cores with an additional 20% power saving if being grouped together. Our algorithm is implemented in the C programming language and all the experiments were performed on a Sun Blade 2500 with a 1.6 GHz CPU and 2 GB RAM.

TABLE I
DATA SETS

Data	Block No.	Net No.	Groups
n10	10	118	3(30%)
n30	30	349	5(30%), 5(20%)
n50	50	485	5(50%), 5(30%), 5(20%)
n100	100	885	10(30%), 5(50%), 6(40%)
n200	200	1585	10(50%), 10(40%), 9(30%), 9(20%)
n300	300	1893	15(60%), 15(50%), 15(40%), 15(30%)

The results are shown in Table III. For each data set, we performed voltage island driven floorplanning with the number of voltage islands generated ranges from zero to six. We can see from the results that up to 50% power saving can be achieved without any significant degradation in area and wire length. In addition, the speed is very acceptable and promising. Some resultant floorplans are shown in Figure 3 and Figure 4. Figure 3 is a resultant packing of data set n100, with four voltage islands generated. In Figure 4, we aim at testing a particular situation in which some cores can be operated at very low voltages (compared with the chip-level voltage). We use *playout* of the MCNC benchmark as the testing data set, and assign 0.6V to cores $\{2-9\}$ and 0.8V to cores $\{12-19\}$ respectively as their minimum working voltages, while the other cores' working voltages are all set to 1.5V. The floorplanning procedure is then performed with $K = 2$ and 1.5V being the chip-level voltage. In the result, two islands are generated as expected, one with cores $\{2,3,4,8,9\}$ and the other one with $\{12,13,14,15,17\}$. The cores $\{5,6,7\}$ and $\{16,18,19\}$ are not included in the islands due to other factors like interconnect and area. Their sizes are small and will not cause large power wastage even if being operated at a higher voltage.

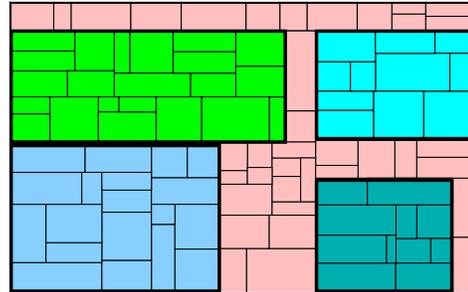


Fig. 3. One resultant floorplan of n100 with four voltage islands.

In order to compare with the previous related work on SoC designs [7], we have done another set of experiments with the benchmarks provided by the authors of [7]. In their data sets, the available voltage levels for each cell are chosen from the set $\{1.1V, 1.3V, 1.5V, 1.8V\}$. The comparisons are displayed in Table II. Note that two different comparisons have been done, one of which enables the idle island option while the other disables it. Result shows that our approach is much more efficient and is able to save more power in most cases, while with less area overhead. In this set of experiments, we set $K = 4$ for all data sets, i.e., four voltage islands are generated for each test case.

A. Extension to Minimize Level Shifters

Level shifters (LS) are needed for connections between two blocks in different power domains. These level shifters will lead to area and delay overhead and should be minimized. We can extend our

TABLE II
COMPARISONS WITH A PREVIOUS WORK [7]

Data Set	Power Saving (%)			Dead Space (%)			Run Time* (s)		
	with idle island	without idle island	[7]	with idle island	without idle island	[7]	with idle island	without idle island	[7]
apte	52.47	47.91	53.78	1.264	1.232	3.422	3.016	3.002	5.482
xerox	50.21	41.76	22.85	1.137	1.085	5.259	3.235	3.185	7.079
hp	47.85	39.19	25.37	1.324	1.318	5.700	3.827	3.367	122.9
ami33	54.26	46.96	44.12	3.865	3.582	5.784	40.69	36.84	89.39
ami49	52.63	45.50	41.13	3.791	3.805	6.440	98.36	91.23	90.46
2xerox	50.86	41.76	33.53	2.062	3.067	3.765	10.64	9.026	74.75
2hp	55.13	39.19	17.47	2.451	2.238	5.650	12.74	11.12	26.86
ami75	49.32	40.05	39.06	4.379	4.871	6.330	276.1	257.2	316.5
ami99	48.68	45.57	41.16	6.388	7.036	7.666	441.4	428.3	684.4
ami200	43.87	39.88	41.90	8.116	8.011	10.88	1657	1598	3851
ami300	42.54	40.54	40.69	10.08	11.42	12.02	3369	3243	7380
Average	49.78	42.57	36.46	4.078	4.333	6.655	537.8	516.7	1150

*[7] is run on a Linux machine with a 2.1 GHz CPU and 4 GB RAM.

TABLE III
EXPERIMENTAL RESULTS WITH IDLE ISLANDS

Data	K	Total Power	Power Saving (%)	Total Area	Dead Space (%)	Wire Length ($\times 10$)	Idle Island No.	Run Time (s)	Data	K	Total Power	Power Saving (%)	Total Area	Dead Space (%)	Wire Length ($\times 10$)	Idle Island No.	Run Time (s)
n10	0	498718	0.000	223588	0.854	1832	0	3.236	n100	0	417928	0.000	187899	1.146	17638	0	325.8
	1	406653	18.47	223918	1.003	1865	1	3.259		1	365143	12.63	190173	2.338	17962	0	333.7
	2	357723	28.28	224457	1.238	1986	1	3.343		2	350558	16.12	195880	5.174	18267	0	367.6
	3	303755	39.16	224546	1.277	1929	1	3.351		3	327864	21.55	196858	5.645	18663	1	396.7
	4	277569	44.35	224335	1.184	1897	1	3.374		4	307553	26.41	198599	6.472	18135	1	438.1
	5	268342	46.22	224314	1.175	2106	1	3.388		5	282686	32.36	200187	7.214	19579	2	451.5
6	267145	46.44	224355	1.193	2075	1	3.454	6	250064	40.17	200429	7.326	18851	2	492.4		
n30	0	469330	0.000	211720	1.476	8659	0	29.65	n200	0	419223	0.000	188092	0.942	21501	0	1489
	1	322429	31.30	214040	2.543	8823	0	31.34		1	382449	8.761	195633	4.762	22419	0	1523
	2	251420	46.43	213453	2.278	8425	1	32.47		2	363843	13.21	196582	5.224	22946	1	1568
	3	243300	48.16	214763	2.874	9016	1	32.83		3	350804	16.32	199979	6.831	23017	1	1626
	4	237621	49.37	216481	3.645	8969	2	37.14		4	325484	22.36	203808	8.586	22838	1	1648
	5	233623	50.22	220572	5.432	9113	2	39.88		5	295216	29.58	206267	9.670	23519	1	1723
6	228047	51.41	217022	3.885	9084	2	42.64	6	271572	35.22	209939	11.26	23485	2	1838		
n50	0	446802	0.000	200465	0.941	15324	0	90.26	n300	0	668290	0.000	302007	1.652	27924	0	2843
	1	326612	26.90	200852	1.132	16233	0	94.25		1	624303	6.582	317022	6.310	28318	0	2869
	2	286310	35.92	202101	1.743	17659	0	95.73		2	598988	10.37	325061	8.627	28254	0	2913
	3	255838	42.74	202332	1.855	16983	1	95.46		3	563502	15.68	326493	9.028	29247	1	2966
	4	229835	48.56	202875	2.118	17241	2	100.3		4	515518	22.86	331086	10.29	28761	1	3202
	5	217369	51.35	202495	1.934	16815	3	113.7		5	483508	27.65	335348	11.43	29388	1	3635
6	216788	51.48	206309	3.747	17927	2	134.1	6	453167	32.19	342858	13.37	29289	2	3828		

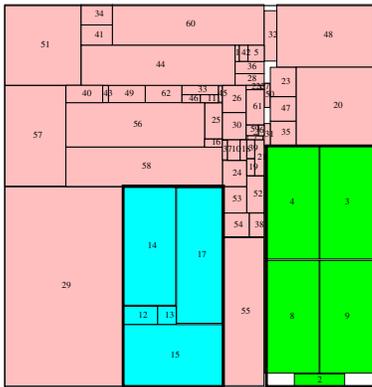


Fig. 4. One resultant floorplan of *layout* with some blocks operated at very low voltages.

floorplanner to minimize the usage of level shifters by having an additional term in the cost function (with a weight determined by random walk before the annealing process) to represent the number of level shifters used. We assume that a level shifter is needed whenever

a wire is connecting two blocks operating at different voltages³. For example, if a net connects a source in voltage island *A* to three sinks, two in island *B* and one in island *C*, two level shifters will be inserted, one between *A* and *B* and one between *A* and *C*. Since the operating voltage of each block is known after the voltage assignment and voltage island partitioning procedure, the number of level shifters needed can be counted trivially. The result is shown in Table IV. In this set of experiments, the numbers of voltage islands *K* are all four. The results have shown that our method can reduce the number of level shifters by 67.1% on average with some penalty in power saving and run time.

B. Extension to Consider Power Network Routing

The voltage islands should be placed in proximity to the power pins to simplify the power routing step and to minimize the IR drop. The power network resources can be modeled by the sum of the half perimeters of the islands [3]. We can also extend our floorplanner to consider these power network issues by having additional terms in the cost function (with weights determined by random walk) to represent

³We can also consider adding a level shifter only when one core with a lower voltage level drives another core with a higher voltage level.

TABLE IV
MINIMIZATION OF LEVEL SHIFTER NUMBER

Data	No. of LS		Power Saving (%)		Run Time (s)	
	w/o LS Opt.	with LS Opt.	w/o LS Opt.	with LS Opt.	w/o LS Opt.	with LS Opt.
n10	43	27	44.35	38.25	3.374	4.635
n30	79	45	49.37	35.89	37.14	42.82
n50	192	137	48.56	40.63	100.3	108.9
n100	267	62	26.41	17.82	438.1	440.6
n200	345	96	22.36	15.34	1648	1702
n300	569	126	22.86	12.86	3202	3228
Avg	249.2	82.1	35.65	26.80	904.8	920.2
Diff(%)	-67.1		-24.82		+1.7	

(1) the total distance of the voltage islands from their respective power pins, and (2) the sum of the half perimeters of the islands. In our experiments, we assume that the positions of the K power pins are given. In each iteration of the annealing process, each island is matched to a power pin such that the total distance between them is the smallest possible. This total distance and the sum of the islands' half perimeters will be minimized during the annealing process.

Two resultant packings of the $n300$ data set consisting of 300 blocks are shown in Figure 5, and they are produced with the fixed-outline constraint. The packing in Figure 5(a) is obtained by the original floorplanner, without taking into consideration the power network issues, while the one in Figure 5(b) is obtained by this extended version. There are four power pins located at the center of each boundary in this example. We can see from the figures that the four islands are shifted to the sides of the chip containing the pins in order to be located closer to their respective power pins. Besides the islands are closer to square in shape that will favor IR drop reduction and power network routing.

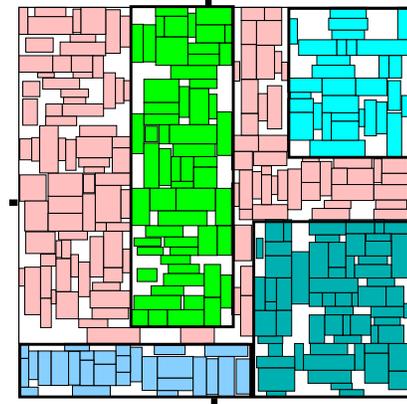
V. CONCLUSION

In this paper, we have proposed a simulated annealing-based approach for the floorplanning problem with simultaneous island partitioning and voltage assignment. The three factors area, wire length and power consumption of the resultant floorplan are concurrently taken into consideration. The experiment results have shown that we are able to achieve a significant power saving of up to 50% for the testing data sets.

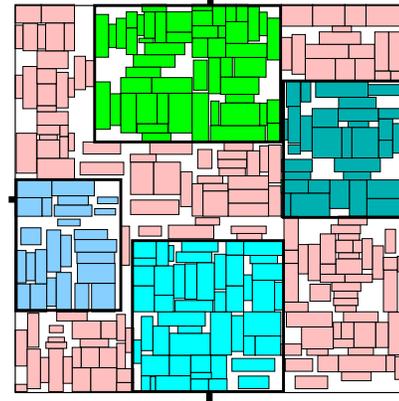
In addition, when extended to minimize the number of level shifters, our method can reduce 67.1% of the LS used on average, with some penalty in power saving. It also functions well to generate voltage islands in proximity to their corresponding power pins, by having additional terms in the cost function to minimize the total distance between voltage islands and power pins, and to restrict the islands to be more square-like in shape.

REFERENCES

[1] D. F. Wong and C. L. Liu. A New Algorithm for Floorplan Design. Proceedings of the 23rd ACM/IEEE Design Automation Conference, pages 101–107, 1986.
 [2] J. Hu, Y. Shin, N. Dhanwada, and R. Marculescu. Architecting Voltage Islands in Core-based System-on-a-chip Designs. *Proceedings*



(a) Without considering proximity to power pins



(b) Considering proximity to power pins

Fig. 5. Resultant floorplans considering proximity constraints to power pins

of the 2004 International Symposium on Low Power Electronics and Design, pages 180–185, 2004.
 [3] W.-P. Lee, H.-Y. Liu, and Y.-W. Chang. Voltage Island Aware Floorplanning for Power and Timing Optimization. *Proceedings of the International Conference on Computer-Aided Design*, 2006.
 [4] S. Yang, W. Wolf, N. Vijaykrishnan and Y. Xie. Reliability-aware SoC Voltage Islands Partition and Floorplan. *Proceedings of the Emerging VLSI Technologies and Architectures*, 2006.
 [5] W.-L. Hung, G.M. Link, Y. Xie, N. Vijaykrishnan, N. Dhanwada and J. Conner. Temperature-aware Voltage Islands Architecting in System-on-chip Design. *Proceedings of the Computer Design*, 2004.
 [6] H. Wu, I.-M. Liu, D.-F. Wong, and Y. Wang. Post-Placement Voltage Island Generation under Performance Requirement. *Proceedings of the International Conference on Computer-Aided Design*, 2005.
 [7] Wai-Kei Mak and Jr-Wei Chen. Voltage Island Generation under Performance Requirement for SoC Designs. *Proceedings of the Asian South Pacific Design Automation Conference*, 2007.
 [8] Royce L.-S. Ching and Evangeline F.-Y. Young. Post-placement Voltage Island Generation. *Proceedings of the International Conference on Computer-Aided Design*, 2006.