Multi-Voltage Floorplan Design with Optimal Voltage Assignment

Qian Zaichen
Department of CSE
The Chinese University of Hong Kong
Shatin,N.T., Hong Kong
zcqian@cse.cuhk.edu.hk

Evangeline F.Y. Young
Department of CSE
The Chinese University of Hong Kong
Shatin,N.T., Hong Kong
fyyoung@cse.cuhk.edu.hk

ABSTRACT

[1] In this paper, we study the multiple voltage assignment (MVA) problem under timing constraints in floorplanning, which is generally an NP-hard problem. We will present an effective value-oriented branch-and-bound based algorithm to solve it optimally in a reasonable amount of time. A convex cost integer dual network flow approach is used to obtain a feasible upper bound solution, while a lower bound is obtained by a linear relaxation of a general formulation of the problem. We then adopt a value-oriented breadth-first branch-and-bound method with upper and lower bounds as described above to search for the optimal solution. Favorable results can be obtained in comparison with previous methods using a general linear programming solver. We integrate this algorithm into a multi-stage floorplanner. At the first stage, an initial floorplan is obtained by simulated annealing using the convex cost integer dual network flow approach as an evaluator. We then perform optimal voltage assignment to this initial floorplan. Finally, a post-processing step is done to modify the floorplan slightly to optimize the power network routing resource before invoking once more the optimal voltage assignment step at the end. Experimental results show that we can improve over the most updated work on this problem [7] by further reducing 6% of power consumption while maintaining the performance on other factors

Categories and Subject Descriptors

B.7.2 [**Integrated Circuit**]: Design Aid—Layout

General Terms

Algorithms, Performance

[1]The work described in this paper was partially supported by a grant from the Research Council of the Hong Kong Special Administrative Region, China (Project No. 418407).

ISPD'09, March 29–April 1, 2009, San Diego, California, USA.

Keywords

multi-voltage assignment optimization branch-and-bound

1. INTRODUCTION

Multiple Supply Voltages (MSV) is developed to deal with the power problem in today's high performance circuits. In MSV designs, modules lying on critical paths can be operated at higher voltage levels to meet timing constraints, while modules on noncritical paths will be operated at lower voltages to reduce power consumption without affecting the circuit performance. However, a number of issues must be handled first before this approach becomes practical, for example, voltage selection and assignment, MSV-aware floorplanning and placement, etc. Among them, Multiple Voltage Assignment (MVA) is an important problem that has attracted the attention of many researchers in the past few years. Given a netlist of modules which can be operated at several different supply voltages, the MVA problem is to assign a supply voltage to each module to minimize the total power consumption while satisfying the circuit's timing constraints. In MSV designs, the timing slacks of the modules are traded for power saving under a performance guarantee of the circuit. Therefore, an efficient voltage assignment method is desirable to minimize power consumption under timing constraints. Besides, it is absolutely beneficial to consider this multi-voltage design issue at the floorplanning level so that those important physical information like interconnect delay, module location (for estimating the power network routing resources) and area overhead due to level shifters, etc. can be taken into account in an early stage.

A number of previous works were on these MSV related problems. MSV was considered at various design stages, including the floorplanning and placement stage [3, 4], and the post-floorplanning and post-placement stage [2, 5, 6]. In [9], Lee *et al.* used a dynamic programming based algorithm to solve the voltage assignment problem. In [7], Ma *et al.* transformed this power assignment problem to a convex cost integer dual network flow problem [8] which can be solved in polynomial time. However, none of them can produce optimal solution for the problem in general. Lee *et al.* [2] formulated the problem as a mixed integer linear program (MILP), and used a general linear programming solver to obtain optimal solution but the runtime can be excessively long. Chang and Pedram [11] proved that the MVA problem was an NP-hard problem, even if each module had only two voltage choices (the two choices could be different for different modules though).

In this paper, we will first develop a technique that can

solve the general voltage assignment problem under timing constraint optimally in a reasonable amount of time, assuming that the delay-power curve for each module is convex [1]. Given a floorplanning result containing a netlist of modules, a number of working voltage levels for each module with the corresponding delays (delay choices can be continuous or discrete in the real or integral domain), interconnect delays and a timing constraint T, a voltage level will be assigned to each module such that the maximum power saving is achieved without violating the timing constraint. Secondly, we develop a floorplanning strategy with this optimal voltage assignment technique that can perform better on power saving than the most updated work [7] on the NSV-driven floorplanning problem while maintaining the performance on other factors.

In section 2, we present a general formulation of this voltage assignment problem. In section 3, we show how to solve this problem optimally by using a value-oriented branch-and-bound based algorithm. In section 4, we integrate this optimal voltage assignment with floorplanning. In section 5, we show the experimental results.

2. PROBLEM FORMULATION

We are given a set of n modules m_1, m_2,..., m_n with areas and aspect ratios. Each module m_i is given k_i choices of supply voltage v_i^q, for $1 \leq q \leq k_i$, and the corresponding delays d_i^q. We are also given the delay and power for one level shifter that must be inserted in a signal line connecting a low voltage cell to a high voltage cell and a timing requirement T_{cycle}. (This is called the clock cycle time and is an upper bound for all critical path delays.) In the following, we define the multi-voltage floorplanning problem.

DEFINITION 2.1. ***Multi-voltage Floorplanning*** - *Given a netlist of modules, each of which has multiple choices of supply voltages and corresponding delays, and a clock cycle, generate a floorplan with a voltage assignment to each module such that the timing constraint is satisfied and a weighted sum of the total power consumption (due to cells and level shifters), power network routing resources, area and wire length is minimized.*

The power-delay trade-off in m_i is represented by a delay-power curve (DP Curve), $\{(d_i^1, p_i^1), (d_i^2, p_i^2), ...(d_i^{k_i}, p_i^{k_i})\}$, where each pair (d_i^q, p_i^q) for $q = 1, 2, ..., k_i$ is the corresponding delay and power consumption when m_i is operated at v_i^q. Note that each d_i^q and p_i^q can be any real or integer number as long as the DP-Curve (after piecewise linearization) is convex. In our formulation, we use binary variable $u_i(q)$ to represent whether m_i works at voltage p_i^q: $u_i(q) = 1$ if m_i works at voltage p_i^q and $u_i(q) = 0$ otherwise. Therefore, $\Sigma_{q=1}^{k_i} u_i(q) = 1$.

We denote a netlist by a *directed acyclic graph* (DAG), $G = (V, E)$. Each vertex $i \in V$ denotes a module m_i, while each directed edge $(i, j) \in E$ denotes an interconnect from m_i to m_j. In this DAG, each vertex is associated with a DP curve representing the power-delay tradeoff of the corresponding module, while each edge is attributed with a wire delay. For each edge $e(i, j)$, we use a binary variable $LS(i, j)$ to represent whethor a level shifter is needed. Let φ and ρ denote the power consumption and delay respectively of one level shifter.

$$LS(i,j) = \begin{cases} 0 & \text{if a level shifter} \\ & \text{is not needed on } e(i,j) \\ 1 & \text{if a level shifter} \\ & \text{is needed on } e(i,j) \end{cases}$$

The wire delay of $e(i, j)$ is computed as

$$w(i, j) = \delta \cdot l(i, j)$$

where $w(i, j)$ and $l(i, j)$ are the wire delay and wire length of $e(i, j)$ and δ is a constant scaling factor.

Now we can define the voltage assignment problem as follows:

DEFINITION 2.2. ***Voltage Assignment Problem*** - *Given a clock cycle time T_{cycle} and a DAG, $G = (V, E)$, where each vertex $v_i \in V$ is associated with a set of binary variables $\{u_i(1), u_i(2)..., u_i(k_i)\}$ that indicate the working voltage of v_i and each edge $e(i, j)$ is associated with a wire delay $w(i, j)$ and a binary variable $LS(i, j)$ to indicate the existence of a level shifter on $e(i, j)$, select a proper value from $\{0, 1\}$ to each $u_i(q)$ where $q = 1, 2, ..., k_i$ for every v_i such that $\Sigma_{q=1}^{k_i} u_i(q) = 1$ for each v_i, the total power consumption due to the cells and level shifters are minimized and the timing constriant is satisfied.*

For formulation simplicity, we will transform $G = (V, E)$ into a new graph $\bar{G} = (\bar{V}, \bar{E})$ by adding a virtual module m_0 as well as a set of virtual edges E_0. Hence $\bar{V} = V \cup \{m_0\}$ and $\bar{E} = E \cup E_0$. The set E_0 contains (i) edges $e(0, i)$ for all m_i which have no incoming edges in G and, (ii) edges $e(i, 0)$ for all m_i which have no outgoing edges in G. An example is shown in Figure 1 in which sub-figure (a) is the DAG G and sub-figure (b) is the corresponding $\bar{G}$. We will put $u_0(q) = 0$ $\forall q$ and put $w(i, j) = 0$ and $LS(i, j) = 0$ $\forall e(i, j) \in E_0$.

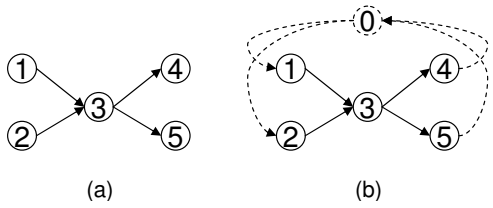

Figure 1: DAG Transformation

According to the definition and notation above, the voltage assignment problem can be easily formulated into the following mathematical program.

$$Minimize: \quad \Sigma_{v_i \in V} \Sigma_{q=1}^{k_i} p_i^q u_i(q) + \Sigma_{e(i,j) \in E} LS(i, j) \cdot \varphi \tag{1a}$$

Subject to:

$$\Sigma_{q=1}^{k_i} u_i(q) = 1 \quad \forall v_i \in V \tag{1b}$$

$$\Sigma_{e(i,j) \in c} (\Sigma_{q=1}^{k_i} d_i^q u_i(q) + \rho LS(i, j) + w(i, j)) \leq T_{cycle} \quad \forall c \in \Phi_k \tag{1c}$$

$$LS(i,j) = \begin{cases} 0 & \sum_{q=1}^{k_i} v_i^q \times u_i(q) \geq \sum_{q=1}^{k_j} v_j^q \times u_j(q) \\ 1 & \text{otherwise} \end{cases} \tag{1d}$$

$$u_i(q) \in \{0, 1\}, \quad q = 1, 2, ..., k_i, \forall v_i \in V \tag{1e}$$

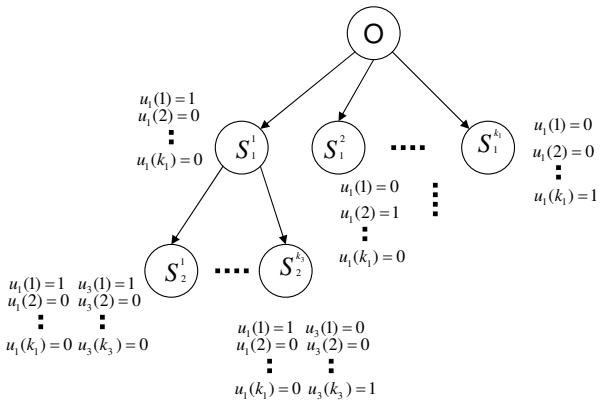

Figure 2: Sample of Branch-and-Bound Method

where Φ_k is the set of all cycles in $\bar{G}$. The objective function (1a) is the sum of all modules' and level shifters' power. Constraint (1b) is used to ensure that each module m_i is working at only one voltage level. Constraint (1c) is the timing constraint that requires the delay of every path to be smaller than the clock cycle.

3. A VALUE-ORIENTED BRANCH-AND-BOUND ALGORITHM

In this section, we will show how to solve the MVA problem optimally. As we mentioned before, the MVA problem is an NP-hard problem, so we developed a branch-and-bound based technique to solve it. We will first describe the basic value-oriented branch-and-bound algorithm and then explain how to compute the upper and lower bounds. The branch-and-bound approach solves problems by searching successive partitions of the solution space. As usual, we will describe the searching process by a search tree in which an internal node represent a set of solutions while a leaf node represent a single solution.

3.1 Branching Rules

In our searching approach, we visit different parts of the solution space by confining the value of some selected variables. An example is shown in Figure 2. Node zero represents the set of all solutions (an optimal one exists for the original problem). In the first iteration, we may select a module m_i (we will describe the selection criteria in the following) and branch to k_i children nodes, which represent the solution space of k_i different subproblems where k_i is the number of supply voltage choices of module m_i. W.l.o.g., we assume that this m_i is m_1 in the following discussion. In figure 2, S_1^1 is the first such subproblem which is obtained by adding the following constraint (2) to constraints (1a)-(1e):

$$u_1(1) = 1, u_1(2) = 0, ..., u_1(k_1) = 0 \quad (2)$$

Similarly, subproblem S_2^1 is obtained by adding to constraints (1a)-(1e) , (2) and a new constraint (3):

$$u_3(1) = 1, u_3(2) = 0, ..., u_3(k_3) = 0 \quad (3)$$

In this way, we will branch out to many subproblems and search the whole solution space of the original problem. Next, we will discuss selection criteria for braching cut. In [7], the author presented an algorithm that can give an optimal solution for the MVA problem when the voltage choices are continuous. But this may not be true in many real applications. For example, in a problem with two supply voltage choices for each module, e.g., module m_i has two choices (p_i^1, d_i^1) and (p_i^2, d_i^2), the approach in [7] may return a delay value $d_i^{optimal}$ for m_i where $d_i^1 \leq d_i^{optimal} \leq d_i^2$ (assuming $d_i^1 < d_i^2$). Notice that if $d_i^{optimal}$ is equal to d_i^1 or d_i^2 exactly, we say that the solution is **feasible**; otherwise, it is **infeasible**. Therefore, to branch out from a node, we will select the module with the lowest index whose solution returned by [7] is infeasible. For instance, given a MVA problem, we first solve it using the approach in [7] and obtain a solution $\{(p_1^*, d_1^*), (p_2^*, d_2^*), ..., (p_k^*, d_k^*)\}$ for module $m_1, m_2, ...m_k$, we will check starting from $i = 1$ to k whether module m_i can work at (p_i^*, d_i^*), and select the first module whose solution is infeasible to branch out to the children nodes. After selecting one module m_i, we will construct k_i subproblems with additional constraints to confine the voltage level of m_i where k_i is the number of voltage choices for m_i.

3.2 Upper Bounds

For a branch-and-bound based algorithm, tight upper and lower bounds are important for efficiency. As we mentioned above, the approach in [7] may produce infeasible solution, because the convex cost integer dual network flow algorithm may return voltage values lying between two feasible voltage levels.

Table 1: NUMBER OF CELLS WITH INFEASIBLE VOLTAGE LEVELS

	n10	n30	n50	n100	n200	n300
No. of cells with infeasible voltage	3	6	9	5	21	13

We have collected some statistics (Table 1) that shows the number of infeasible assignments voltages returned by the approach in [7]. Since the number of infeasible voltages is relatively small, one good and straight forward way to obtain an upper bound for the searching process is to choose a feasible solution as close as possible to the solution returned by [7]. Given a solution $\{(p_1^*, d_1^*), (p_2^*, d_2^*), ...(d_n^*, d_n^*)\}$ for modules $m_1, m_2, ...m_n$, n is the number of modules, if some (p_i^*, d_i^*) for $1 \leq i \leq n$ is not feasible, we can use the largest possible d_i in $(p_i^{'}, d_i^{'})$ such that

$$(p_i^{'}, d_i^{'}) \in \{(p_i^1, d_i^1), (p_i^2, d_i^2), ...(p_i^{k_i}, d_i^{k_i})\}$$

and satisfying $d_i^{'} \leq d_i^*$ as the voltage level for m_i.

After assigning supply voltages to all modules, we will check for each net (i, j), whether a level shifter should be inserted. We will then use the sum of the power consumptions due to the modules and the level shifters as an upper bound. Note that we will do this computation whenever a new node in the search tree is visited and will update the upper bound (which is global) if a better one is found.

3.3 Lower Bounds

The lower bound is obtained by a linear relaxation of the problem. Before we explain it, let us rebuild the formulation of Problem (1). Let k be the total number of supply voltages among *all* the modules. For example, if we have three modules m_1, m_2 and m_3 where m_1 has only one supply voltage choice $p_1^1 = 1.0v$, m_2 has two, $p_2^1 = 1.0v$ and $p_2^2 = 1.2v$, and m_3 has two, $p_3^1 = 1.1v$ and $p_3^1 = 1.4v$, then we will have $k = 4$ different voltage levels $1.0v, 1.1v, 1.2v, 1.4v$ for all the modules. Now, we will substitute each k_i in Problem (1) by k and replace $u_i(q)$ for $q = 1, 2, ..., k_i$ by $u_i(q)$ for $q = 1, 2, ..., k$. For the above example, before this step, module m_3 will have $k_3 = 2$ voltage choices, and after this step, $k = 4$ but $u_3(1)$ and $u_3(3)$ are always zero because they represent $1.0v$ and $1.2v$ respectively and m_3 cannot work at these two voltages.

In the linear relaxation, we first replace (1e) by

$$0 \leq u_i(q) \leq 1, q = 1, 2, ..., k, \forall v_i \in V$$

Then we can substitute (1d) by

$$LS(ij) \geq u_i(q_1) + u_j(q_2) - 1 \quad \forall e(i,j) \in E, \forall q_1 \forall q_2 \text{ s.t. } (0 \leq q_1, q_2 \leq k) \wedge (q_2 > q_1) \tag{4}$$

So Problem (1) becomes the following after this linear relaxation step:

$$Minimize: \quad \Sigma_{v_i \in V} \Sigma_{q=1}^{k} p_i^q u_i(q) + \Sigma_{e(i,j) \in E} LS(i,j) \cdot \varphi \tag{5a}$$

Subject to :

$$\Sigma_{q=1}^{k} u_i(q) = 1 \quad \forall v_i \in V \tag{5b}$$

$$\Sigma_{e(i,j) \in c} (\Sigma_{q=1}^{k} d_i^q u_i(q) + \rho LS(i,j) + w(i,j)) \leq T_{cycle} \quad \forall c \in \Phi_k \tag{5c}$$

$$LS(i,j) \geq u_i(q_1) + u_j(q_2) - 1 \quad \forall e(i,j) \in E, \forall q_1 \forall q_2 \text{ s.t. } (0 \leq q_1, q_2 \leq k) \wedge (q_2 > q_1) \tag{5d}$$

$$0 \leq LS(i,j) \leq 1, \forall e(i,j) \in E \tag{5e}$$

$$0 \leq u_i(q) \leq 1, q = 1, 2, ..., k, \forall v_i \in V \tag{5f}$$

Note that the optimal solution to Problem (1) is included in the solution space of Problem (5), so the power consumption of the optimal solution of (5) is a lower bound for our Problem (1). Problem (5) is a simple linear program which can be solved easily with the simplex algorithm in polynomial time to give a lower bound to our original problem.

3.4 Pruning Rules and Value-Oriented Searching Rules

With the upper and lower bounds obtained above, we can construct our pruning rules. Some nodes with their subtrees together will never contribute an optimal solution, and good pruning rules help to find such nodes to reduce runtime. If a node is infeasible, that is, we cannot find a feasible solution satisfying the timing constraint even assuming a continuous domain for the module voltages (this can be verified by invoking the approach in [7]), it will be pruned. Besides, if a node's lower bound is greater than or equal to the global upper bound, it will also be pruned. This is correct since we can never find a better solution by traversing into its subtree.

With the above bounds and pruning rules, we can search the solution space faster. Our value-oriented branch-and-bound searching algorithm is divided into rounds. In each round, we set a *target* which is used as a threshold to decide whether a node should be visited in this round. For example, in the first round, the *target* will be computed as:

$$target = \alpha(O_{upbound} + O_{lowbound})$$

where $0 \leq \alpha \leq 1$ is a constant (set to 0.6 in our experiments), and $O_{upbound}$ and $O_{lowbound}$ are the upper and lower bounds of the original problem (node 0 in Figure 2). When we reach a node S, we will check whether $S_{lowbound} < target$ where $S_{lowbound}$ is the lower bound at S. If so, we will continue to visit the subtree of S; otherwise, this node S will be marked and will not be visited in this round (may be visited in the following rounds). After one round, the value of *target* will be updated as $target = target + C$ where C is a constant (set to 5% of $O_{upbound}$ in our experiments). Note that a subtree which is explored in a previous round will not be repeatedly visited in any subsequent rounds. In general, this value-oriented searching strategy will search nodes with smaller lower bounds in earlier rounds and we can upper bound the deviation of a solution obtained from such value-oriented search from the true optimal solution. This idea is derived from the target-oriented brand-and-bound algorithm in [10],

In our value-oriented branch-and-bound algorithm, There are two conditions under which we can stop searching: (i) a feasible solution is found at a node of which the objective function value is equal to the smallest lower bound (obtained by the linear relaxation) ever seen during the search; (ii) all possible nodes are visited and searched.

Putting all the above steps together, we have the following pseudo-code for our value-oriented brand-and-bound algorithm:

Algorithm 1 VALUE-ORIENTED BRANCH-AND-BOUND

```
1: // Procedure to find optimal solution to the voltage assignment problem
2: P_MVA //solution to the multiple voltage assignment problem assuming continuous domain for voltage choices
3: pool = ∅ // a queue for searching nodes
4: basket = ∅ // a stack for searching nodes
5: node_0 = construct first node from P_MVA
6: put node_0 into pool
7: while (pool is not empty) do
8:    node_x = get a node from pool
9:    push node_x into basket
10:   while (basket is not empty) do
11:      node_x = Pop node out of basket
12:      if (node_x can be cut) then
13:         continue
14:      end if
15:      if (node_x can be stored in the pool for subsequent iterations) then
16:         store node_x in pool
17:         continue
18:      end if
19:      branch at node_x
20:      push all subproblems into basket
21:   end while
22: end while
```

4. FLOORPLANNING

We integrated the above optimal voltage assignment algorithm into the floorplanning step. Our floorplanner is multi-stage. In the first stage, we will invoke the simulated annealing based floorplanner in [4] to give an initial floorplan. After an initial floorplan is generated, we will perform our optimal voltage assignment on it. Finally, a post-processing step will be performed to incrementally change the floorplan to reduce the network routing resources before a final optimal voltage assignment is performed once again. The post-processing step is done by invoking an ordinary simulated annealing based floorplanner which will move blocks around to minimize an objective function which is a weighted sum of area, wire length and power network routing resources (no power consumption in the cost function). After this fast floorplanning step, the timing constraint may be violated, so we need to perform once again the optimal voltage assignment to assign a final voltage to each block. The floorplanner [7] is used to produce the initial floorplan because their approach is very similar in nature with our optimal voltage assignment method and the second stage thus will not make a large number of block voltage changes, which, basically, will invalidate and waste the optimization done in the first stage. To verify this point, we have done a simple experiment to compare the results of the minimum cost network flow approach used in [7] and our optimal voltage assignment method. We randomly generate 10 floorplans for each benchmark and compare our approach (called VOBB) and [7] in terms of average power consumption and the number of modules with different voltage assignments. The results are shown in Table 2.

5. EXPERIMENTAL RESULTS

We implemented our algorithms in the C programming language and experimented on a PC with an Intel 2.6GHz CPU and 1 GB memory. We have done two sets of experiments as described below.

5.1 OPTIMAL VOLTAGE ASSIGNMENT

In this set of experiments, we want to compare our optimal voltage assignment method with the most updated previous work [2] that also tried to solve the same problem optimally using MILP. A general linear programming solver called *lp_solve* is used in [2]. The results are shown in Table 3. The run time limit for each data set is ten hours, and an asterisk indicates that the program does not run to the end after 10 hours (so, the solution is suboptimal). Our technique can obtain optimal solutions in a reasonable amount of time for most testbenches and has out-performed [2] quite a lot. During the experiments, we found that the runtime depends more on the complexity of the circuit rather than the number of modules. The word "complexity" here means the total number of constraints (1c). In the experimental results, the runtime for n200 is even longer than that for n300 [2]. For larger problem instances, we can set a threshold difference between the global upper bound and the lower bound at the end of each round e.t. the $target$, during the seaching process. When the difference is smaller than the threshold, we will accept the upper bound as the final result.

[2]We have also tried to rectify (1c) using edge-based timing constraints [12], but we found that this new formulation did not have any advantages over our formulation.

5.2 FLOORPLANNING RESULTS

In this set of experiments, we compare the results of our multi-stage floorplanner with those in [7]. The results are shown in Table 4. We can see that our floorplanner can out-perform [7] in almost all aspects except having around 0.8% more deadspaces. Our floorplanner can give a floorplan with about 6% more power saving using about 50% less level shifters.

6. CONCLUSION

In this paper, we proposed a method to solve the Multiple Voltage Assignment problem optimally, and integrated it into a floorplanner. We show that the general MVA problem under timing constraints can be solved optimally by our value-oriented branch-and-bound based algorithm in a reasonable amount of time, by constructing proper upper bounds and lower bounds. We embed this technique into a floorplanner to produce multi-voltage aware floorplans. Experimental results show that our floorplanner can save about 6% more power compared with the most updated results on this problem using around 30% less level shifters.

7. REFERENCES

[1] W.R. Davis, A-.M. Sule and H-. Hua, "Multi-Parameter Power Minimization of Synthesized Datapaths", *Proceedings of the ISVLSI*, 2004.

[2] W.-P.Lee, H.-Y. Liu and Y.-W. Chang, "An ILP Algorithm for Post-Floorplanning Voltage-Island Generation Considering Power-Network Planning", *Proceedings of International Conference on Computer-Aided Design*, 2007.

[3] J.Hu, Y.Shin, N.Dhanwada, and R. Marculescu, "Architecting Voltage Islands in Core-based System-on-a-chip Designs", *Proceedings of International Symposium on Low Power Electronics and Design*, pp.180-185, 2004

[4] Q. Ma and Evangeline FY. Young, "Voltage Island-Driven Floorplanning", *Proceedings of International Conference on Computer-Aided Design*, pp.644-648, 2007.

[5] W.-K. Mak and J.-W. Chen, "Voltage Island Generation under Performance Requirement for SoC Designs", *Proceedings of Asia and South Pacific Design Automation Conference*, 2007.

[6] H. Wu, D.F. Wong, and I.-M. Liu, "Timing-Constrained and Voltage-Island-Aware Voltage Assignment", *Proceedings of Design Automation Conference*, pp.429-432, 2006.

[7] Q. Ma and Evangeline FY. Young, "Network Flow Based Power Optimization Under Timing Constraints in MSV-Driven Floorplanning" *Proceedings of International Conference on Computer-Aided Design*, 2008.

[8] R.K Ahuja, D.S. Hochbaum, and J.B. Orlin, "Solving The Convex Cost Integer Dual Network Flow Problem", *management Science* , 49(7):950-964, July 2003.

[9] W.-P Lee, H.-Y.Liu, and Y.-W. Chang, "Voltage Island Aware Floorplanning for Power and Timing Optimization", *Proceedings of Asia and South Pacific Design Automation Conference*, 2006.

[10] V. Stix, "Target-Oriented Branch and Bound Method for Global Optimization", *Journal of Global Optimization*, 26:261-277, 2003.

[11] J.-M. Chang, M. Pedram, "Energy Minimization Using Multiple Supply Voltages", *IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS*, VOL.5, NO.4, DECEMBER 1997.

[12] C.Chen, C.C.N.Chu, and D.F. Wong, " Fast and exact simultaneous gate and wire sizing by Lagrangian relaxation", *IEEE Trans. CAD*, 18(7):1014-1025, July 1999.

Table 2: VOLTAGE ASSIGNMENT COMPARISONS BETWEEN VOBB AND [7]

Testbenches	Power		Ratio	Average No. of Blocks with Different Voltages
	[7]	VOBB		
n10	202709	185270	91.4%	1.7
n30	162534	155853	95.9%	2.9
n50	166931	157163	94.1%	7.8
n100	137608	126855	92.2%	9.9

Table 3: COMPARISONS BETWEEN VOBB AND [2]

Testbenches	Power		Run time	
	VOBB	[2]	VOBB	[2]
n10	169058	169058	1.2 s	0.0 s
n30	143460	143460	12.1 s	10 h
n50	138983	138983	35.0 s	11.1 m
n100	113231	*117761	10.0 m	10 h
n200	*119229	*116341	10 h	10 h
n300	142641	*143041	32.4 m	10 h
Average	137767	138107	-	-

Table 4: FLOORPLANNING RESULT COMPARISONS BETWEEN OUR VOBB-BASED FLOORPLANNER AND [7]

Data Set	Max Power (MaxP)	Power Cost with Level Shifters (P)		Power Saving (%)		Power Network Routing Resources		Level Shifter Number		Dead Space (%)		Wire Length	
		VOBB	[7]	VOBB	[7]	VOBB	[7]	VOBB	[7]	VOBB	[7]	VOBB	[7]
n10	216841	169058	189942	22.04	12.40	1373	1530	8	4	2.12	1.77	6920.7	7781.3
n30	205650	143460	151483	30.24	26.34	1354	1577	21	25	7.05	9.12	28814.2	29283.0
n50	195140	138983	153084	28.78	21.76	1662	1641	32	34	10.82	9.72	64532.2	64623.6
n100	180022	113231	120850	37.10	33.09	1446	1528	50	77	9.59	8.64	116552.8	116681.6
n200	177633	121222	130489	31.76	26.54	1626	1584	94	129	14.30	12.49	198205.8	210457.2
n300	273499	142641	161464	47.85	40.96	1690	1806	30	92	12.52	10.37	229116.1	240326.2
Average	-	138099	151219	32.96	26.85	1525	1611	39	60	9.46	8.68	107357.0	111525.5