Network Flow-based Power Optimization under Timing Constraints in MSV-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*— Power consumption has become a crucial problem in modern circuit design. Multiple Supply Voltage (MSV) design is introduced to provide higher flexibility in controlling the power and performance trade-off. One important requirement of MSV design is that timing constraints of the circuit must be satisfied after voltage assignment of the cells. In this paper, we will show that the voltage assignment task on a given netlist can be formulated as a convex cost dual network flow problem and can be solved optimally in polynomial time using a cost-scaling algorithm when the delay choices of each module are continuous in the real or integer domain. We can make use of this approach to obtain a feasible voltage assignment solution in the general cases with power consumption approximating the minimum one. Furthermore, we will propose a framework to optimize power consumption and physical layout of a circuit simultaneously during the floorplanning stage, by embedding this cost-scaling solver into a simulated annealing based floorplanner. This is effective in practice due to the short running time of the solver. We compared our approach with the latest work [9] on the same problem, and the experimental results show that, using our framework, significant improvement on power saving (18% less power cost on average) can be achieved in much less running time (7X faster on average) for all the test cases, which confirms the effectiveness of our approach.**

I. Introduction

Power optimization has become one of the most important issues to be addressed in modern circuit design because of the increasing power density and the wide use of portable systems. There are two kinds of power consumption: dynamic and static. Static power comes from leakage current, while dynamic power is caused by the charging and discharging of the load capacitance during switching. *Multiple Supply Voltage* (MSV) design [8] was introduced as an effective mean to reduce both dynamic and static power. In MSV designs, the timing slacks of the cells are traded for power saving by assigning appropriate supply voltage levels to the cells while still satisfying the timing constraints. Basically, high voltage level will be assigned to critical cells and low voltage level is assigned to non-critical cells, so that power can be saved without violating the timing constraints. Therefore, an effective and efficient voltage assignment method is desirable to minimize power consumption under timing requirements. Moreover, it is beneficial to consider the problem of voltage assignment, voltage island generation and floorplanning simultaneously under the timing, power, area and other physical constraints, since these steps will significantly affect each other.

A number of previous works addressed the voltage assignment and island generation problem in floorplanning and placement. In these previous works, MSV is considered at various design stages, including the floorplanning and placement stages [6], [7], [9], [11], [12], and the post-floorplanning and post-placement stages [3], [10], [13], [17], [18]. For the latter category, Wu *et al.* [17] and Ching *et al* [3] targeted at minimizing the number of voltage islands generated, and Mak *et al.* [13] formulated the voltage assignment and island generation problem as an *integer linear program* (ILP), but none of them had taken timing constraints into consideration explicitly. In [18], Wu *et al.* proposed an approximation algorithm based on a *zero slack algorithm* to minimize the power cost and to simplify the power network resource under timing constraints; Lee *et al.* [10] tackled the voltage assignment problem using an ILP based approach with a set of linear inequalities representing the timing requirements. An inevitable deficiency of all these post-floorplanning and post-placement works is that physical layout and power optimization are done separately, hence the solution quality is restricted to a certain extent. Concurrent physical layout and power optimization will be much more favorable. Hu *et al.* [6] and Ma *et al.* [12] considered this simultaneous voltage assignment, voltage island generation, and floorplanning problem, however, neither of them explicitly considered timing in their problem formulation.

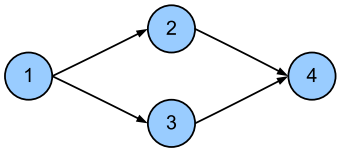

Fig. 1. A Reconvergent Structure in a Netlist.

In one recent work [9], Lee *et al.* handled the voltage assignment problem under timing constraints by dynamic programming, and developed a power network aware floorplanner to generate floorplans with voltage islands after the voltage assignment step. Given a netlist represented by a directed acyclic graph (DAG), voltage assignment is first performed according to the timing requirements before the floorplanning step. Level shifters are then inserted into the nets according to the voltage assignment result. 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 half-perimeters of the voltage islands, will be minimized. As a result, blocks in the same voltage island will be placed close to each other. This paper proposed a reasonable design flow, however, their solution quality suffers in several aspects. First of all, the optimality of its voltage assignment step can only be guaranteed when there are no reconvergent structures in the netlist, but there are surely many of such structures in real circuits, which can significantly worsen the quality of their voltage assignment result. Fig. 1 shows an example of a reconvergent structure, in which the signals emanating from vertex one re-converges at vertex four, after traversing two disjoint paths. Secondly, physical information is not taken into account while performing the voltage assignment step. Lastly, the length of an interconnect is restricted to stay within a certain range during floorplanning in order to satisfy the timing constraints, which may degrade the solution quality.

In this paper, we will first show that the voltage assignment problem under timing constraints can be solved optimally under some conditions, i.e., given a netlist and a number of working voltage levels for each module, a specific voltage level can be assigned to each module on the netlist in such a way that maximum power saving can be achieved without violating the timing constraints, if a feasible

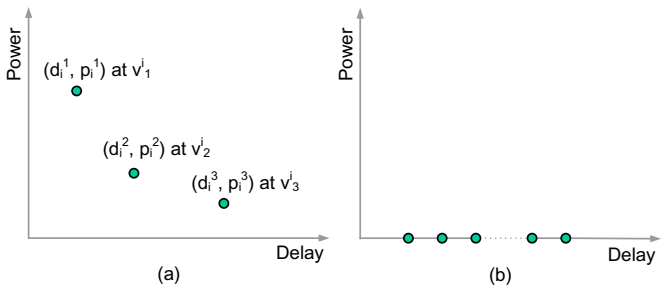

Fig. 2. DP Curve of (a) a Vertex and (b) an Edge in G.

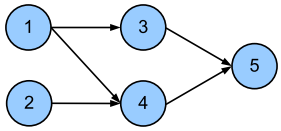

Fig. 3. The DAG Representation of a Netlist.

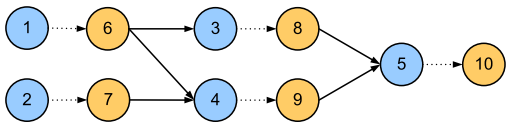

Fig. 4. The Transformed DAG $\bar{G}$ of the Graph G in Fig. 3.

solution exists. This problem can be solved optimally when the delay choices (each delay choice corresponds to a choice of the working voltage level) are continuous in the real or integer domain. We will present a general formulation of this voltage assignment problem as a set of linear inequalities to represent the timing requirements, with an objective to minimize the sum of the power cost in each cell. This problem can be transformed to a convex cost network flow problem [1], after applying the Lagrangian relaxation technique and can be solved optimally by a cost-scaling algorithm [5] in polynomial time. We can then make use of this method to obtain a feasible voltage assignment solution in which each module has just a discrete number of delay choices. Because of the short running time of this cost-scaling algorithm and, particularly, its capability of handling efficiently incremental changes on edge costs, we can embed this voltage assignment step in a simulated annealing based floorplanner, so that power consumption and physical information can be taken into consideration simultaneously. We compared our approach with [9], the most updated previous work on this problem, on the data sets provided by the authors of [9]. Experimental results show that our approach performs much better than [9] in terms of both power saving and running time. An average of 24.61% (v.s. 6.68% by [9]) power saving can be achieved in an average running time of 273s (v.s. 1911s by [9]) with our approach on those data sets.

The remainder of this paper is organized as follows. We 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

We are given a set of n modules $m_1, m_2, ..., m_n$ with areas and aspect ratio bounds. Each module m_i is given k_i choices of supply voltages v_q^i, for $1 \le q \le k_i$, and 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) is the corresponding delay and power consumption when m_i is operated at v_q^i. Note that each d_i^q is in Z^+ (positive integer set), since the delay of a module is measured in terms of the number of clock cycles. Fig. 2(a) is an example of a DP Curve, which contains three choices of supply voltages for this module. Moreover, we assume that for each module, power is a convex function of delay when each point (d_i^q, p_i^q) is connected to its neighboring point(s) by a linear segment in the DP Curve. We are also given a timing requirement T_{cycle}, called clock cycle time, of this circuit and require that the critical path delay is at most T_{cycle}.

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 $e(i, j) \in E$ denotes an interconnect through which a signal is sent from m_i to m_j. Fig. 3 is an example of the DAG representation of a netlist. 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.

To facilitate our problem formulation, we will first transform $G = (V, E)$ into $\bar{G} = (\bar{V}, \bar{E})$ in such a way that each vertex $i \in V$ is split into an input vertex and an output vertex, and the input vertex is connected to the output vertex by a directed edge. We denote this set of newly created edges and the set of original edges by E_1 and E_2, respectively. Hence $\bar{E} = E_1 \cup E_2$. Fig. 4 illustrates the resultant DAG of the graph in Fig. 3 after this transformation, where dashed edges are in E_1 and solid edges are in E_2. After this transformation, the DP Curve of a vertex in G will be associated with its corresponding edge in E_1. For example, after this transformation, vertex one and vertex six in $\bar{G}$ of Fig. 4 are, respectively, the input and output vertex of vertex one in G of Fig. 3, thus the DP Curve of vertex one in G is now associated with the edge $e(1,6)$ in $\bar{G}$, and we will use $k_{1,6}$ to denote the number of voltage level choices of this DP Curve. We can assume that each edge $e(i, j) \in E_2$ (corresponding to a wire) is also associated with a DP Curve in which the delay is an integer ranging from the wire's minimum delay to the maximum allowed delay T_{cycle}, and the power consumption is zero at all points. Fig. 2(b) is an example DP Curve corresponding to an edge $e(i, j) \in E_2$. Therefore, each edge $e(i, j) \in \bar{E}$ of $\bar{G}$ is now associated with one and only DP Curve, while the vertices in $\bar{G}$ simply represent the starting or ending point of a time interval and are not associated with any DP Curves. For each edge $e(i, j) \in \bar{E}$, we use d_{ij} to denote the delay on it, and use $\bar{P}_{ij}$ to represent the function that maps delay to power consumption on edge $e(i, j)$, i.e., $\bar{P}_{ij}(d_{ij}^q) = p_{ij}^q$ for $1 \le q \le k_{ij}$ if $e(i, j) \in E_1$, and $\bar{P}_{ij}(d_{ij}) = 0$ if $e(i, j) \in E_2$. Moreover, we use l_{ij} and u_{ij} to denote the lower and upper bound of the delay d_{ij} on edge $e(i, j)$, i.e., $l_{ij} \le d_{ij} \le u_{ij}$, $\forall e(i, j) \in \bar{E}$. For each edge $e(i, j) \in E_1$, $l_{ij} = d_{ij}^1$ and $u_{ij} = d_{ij}^{k_{ij}}$. For each edge $e(i, j) \in E_2$, u_{ij} is T_{cycle}, while l_{ij} is the minimum wire delay of the corresponding connection, and is estimated by scaling wire length to delay according to equation (1), when physical information is available.

$$l_{ij} = \delta \cdot wl_{ij} \tag{1}$$

where wl_{ij} is the estimated length of the wire connecting module i and module j, and δ is a constant scaling factor. Let $t_i \in Z^+$ denote the arrival time at vertex i in $\bar{V}$, then it follows that

$$t_j - t_i \ge d_{ij}, \quad \forall e(i, j) \in \bar{E} \tag{2}$$

and

$$0 \le t_i \le T_{cycle}, \quad \forall i \in \bar{V} \tag{3}$$

The static timing constraint and the voltage assignment problem can be defined as follows:

Definition 1: **Static Timing Constraint** - Given a clock cycle time T_{cycle} and a DAG, $\bar{G} = (\bar{V}, \bar{E})$ corresponding to a netlist, the static timing constraint of the netlist is that the arrival time t_i satisfies equation (2) and (3), $\forall i \in \bar{V}$.

Definition 2: **Voltage Assignment Problem** - Given a clock cycle time T_{cycle} and a DAG, $\bar{G} = (\bar{V}, \bar{E})$ where each edge e is associated with a DP Curve, select a delay-power pair for each edge in its DP Curve such that the sum of power consumption is minimized while the static timing constraint is satisfied.

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

$$\begin{array}{lll} \text{Minimize} & \sum_{e(i,j)\in\bar{E}} \bar{P}_{ij}(d_{ij}) & (1a) \\ \text{Subject to} & & \\ & t_j - t_i \geq d_{ij}, \quad \forall e(i,j) \in \bar{E} & (1b) \\ & 0 \leq t_i \leq T_{cycle}, \quad \forall i \in \bar{V} & (1c) \\ & l_{ij} \leq d_{ij} \leq u_{ij}, \quad \forall e(i,j) \in \bar{E} & (1d) \\ & d_{ij} \in \{d_{ij}^1, d_{ij}^2, \ldots, d_{ij}^{k_{ij}}\}, \quad \forall e(i,j) \in E_1 & (1e) \\ & d_{ij} \in Z^+, \quad \forall e(i,j) \in E_2 & (1f) \\ & t_i \in Z^+ \quad \forall i \in \bar{V} & (1g) \end{array}$$

Problem (1) without the discrete choice constraint $(1e)$ is a *convex cost integer dual network flow problem* (or simply a *dual network flow problem*) [1], since its dual can be transformed to a minimum cost flow problem. This problem can be solved optimally in polynomial time, and we will show it in the following sections. Then, we will make use of the optimal solution to this problem to generate a feasible solution to the general cases of discrete choices for the module delay, with a power consumption approximating the optimal one.

In MSV designs, the power network resource should also be minimized. Given a floorplan, the power network resource Φ_q for the modules working at voltage v_q can be estimated by the half-perimeter of the bounding box of all those modules [9], while the total power network resource is the sum of Φ_q, for $q = 1, 2, ..., k$ where k is the number of different voltage levels available. Our ultimate goal in this work is to solve the following MSV-driven floorplanning problem.

Definition 3: **MSV-driven Floorplanning** - Given a netlist, a timing constraint and a set of modules, each of which has multiple choices of supply voltage, generate a floorplan in which each module is assigned to work at a specific voltage level such that the timing constraint is satisfied and a weighted sum of power consumption, power network resource, area and interconnect length is minimized.

III. Methodology

Globally, we use a simulated annealing based FAST-SP [15] floorplanner to explore the solution space. The voltage assignment problem (definition 2) is a sub-problem of the MSV-driven floorplanning problem (definition 3), since the former is needed in order to evaluate a candidate solution for the latter problem. After each random move of the annealing process, the length of some interconnects may be changed which will cause changes in the minimum delay of the corresponding wires, thus l_{ij} for those affected wires $e(i,j) \in E_2$ of $\bar{G}$ should be re-computed with equation (1) after the floorplan is realized. With these changes, the operating voltage level of each module is re-assigned by computing another solution to problem (1). The power network resource is evaluated afterwards. In the following sub-section, we will first focus on solving the voltage assignment problem before discussing how this voltage assignment solver can be incorporated into a MSV-driven floorplanner.

A. Voltage Assignment Problem

The voltage assignment problem is equivalent to the mathematical program (1) in Section II. We will show in the following that problem (1) without the discretization constraint $(1e)$ can be transformed into a minimum cost flow problem and can be solved optimally in polynomial time [1].

To simplify the problem, we will first make some modifications to the DP Curves. First of all, we connect each pair of neighboring points in a DP Curve by a linear segment, resulting in a piecewise linear convex function of power versus delay. It can be shown that there always exists an optimal solution to the dual network flow problem (problem (1) without constraint $(1e)$) with integral variables [1]. Therefore, the constraints $(1f)$ and $(1g)$ can be removed.

In the second step of simplification, the lower and upper bounds on variables d_{ij} and t_i in constraints $(1c)$ and $(1d)$ are eliminated. We first eliminate bounds on d_{ij} by defining a new power delay function for each edge $e(i,j) \in \bar{E}$ as follows:

$$P_{ij}(d_{ij}) = \begin{cases} \bar{P}_{ij}(u_{ij}) + M \times (d_{ij} - u_{ij}), & if \quad d_{ij} > u_{ij} \\ \bar{P}_{ij}(d_{ij}), & if \quad l_{ij} \leq d_{ij} \leq u_{ij} \\ \bar{P}_{ij}(l_{ij}) - M \times (d_{ij} - l_{ij}), & if \quad d_{ij} < l_{ij} \end{cases}$$

where M is a sufficiently large number, and here, we set $M = D + 1$, where $D = \sum_{e(i,j)\in E_1} \bar{P}_{ij}(l_{ij}) - \sum_{e(i,j)\in E_1} \bar{P}_{ij}(u_{ij})$. Actually, M can be viewed as a penalty for violating the lower and upper bounds of the variables, and we set the penalty to be large enough to guarantee that a solution with variables violating the bounds is impossible to be an optimal one, if a feasible solution of problem (1) exists. Therefore, by replacing $\bar{P}_{ij}(d_{ij})$ with $P_{ij}(d_{ij})$, we are able to remove the constraint $(1c)$ and $(1d)$, with variables in the optimal feasible solution guaranteed to stay in the range specified by $(1c)$ and $(1d)$. An example transformed DP Curve of an edge in E_1 and that of an edge in E_2 are illustrated in Fig. 5(a) and Fig. 5(b), respectively.

Similarly, we define $B_i(t_i)$ as follows and put it into the objective function so that the lower and upper bounds on t_i can also be eliminated.

$$B_i(t_i) = \begin{cases} M \times (t_i - T_{cycle}), & if \quad t_i > T_{cycle} \\ 0, & if \quad 0 \leq t_i \leq T_{cycle} \\ -M \times (t_i), & if \quad t_i < 0 \end{cases}$$

After all the above simplifications, problem (1) without the discretization constraint $(1e)$ can be transformed into problem (2) as follows.

$$\begin{array}{lll} \text{Minimize} & \sum_{e(i,j)\in\bar{E}} P_{ij}(d_{ij}) + \sum_{i\in\bar{V}} B_i(t_i) & (2a) \\ \text{Subject to} & t_j - t_i \geq d_{ij}, \quad \forall e(i,j) \in \bar{E} & (2b) \end{array}$$

In the following sub-section, problem (2) is further simplified by using the Lagrangian relaxation technique.

1) Lagrangian Relaxation: The Lagrangian relaxation technique is useful to eliminate difficult constraints. Here, we want to eliminate the constraints $t_j - t_i \geq d_{ij}$ in $(2b)$. We apply the Lagrangian relaxation technique to problem (2), by introducing a nonnegative Lagrangian multiplier vector $\vec{x}$, and the corresponding Lagrangian sub-problem is to compute:

$$L(\vec{x}) = \min_{d,t} \sum_{e(i,j)\in\bar{E}} P_{ij}(d_{ij}) + \sum_{i\in\bar{V}} B_i(t_i) - \sum_{e(i,j)\in\bar{E}} (t_j - t_i - d_{ij}) x_{ij}$$

where x_{ij} is the nonnegative Lagrangian multiplier associated with the constraint $t_j - t_i \geq d_{ij}$. Note that:

$$\sum_{e(i,j)\in\bar{E}} (t_i - t_j) x_{ij} = \sum_{i\in\bar{V}} \Big(\sum_{j:e(i,j)\in\bar{E}} x_{ij} - \sum_{j:e(j,i)\in\bar{E}} x_{ji} \Big) t_i$$

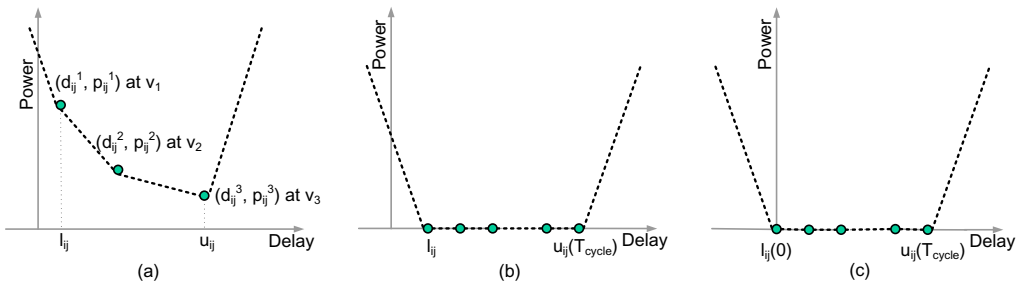

Fig. 5. Delay Power Curve $P_{ij}(d_{ij})$ of (a) an Edge $e(i,j) \in E_1$; (b) an Edge $e(i,j) \in E_2$; (c) an Edge $e(i,j) \in E_3$.

We define

$$x_{0i} = \sum_{j:e(i,j)\in\bar{E}} x_{ij} - \sum_{j:e(j,i)\in\bar{E}} x_{ji}, \quad \forall i \in \bar{V}$$

Then we can rewrite $L(x)$ as follows:

$$L(\vec{x}) = \min_{d,t} \sum_{e(i,j)\in\bar{E}} (P_{ij}(d_{ij}) + x_{ij}d_{ij}) + \sum_{i\in\bar{V}} (B_i(t_i) + x_{0i}t_i)$$

In order to convert this problem to a network flow problem, we add an extra vertex 0 into $\bar{G}$, and add an edge $e(0,i)$ for each vertex $i \in \bar{V}$. Lets denote this set of newly added edges by E_3 and denote the resultant DAG by $G^* = (V^*, E^*)$, so that $V^* = \{0\} \cup \bar{V}$ and $E^* = E_1 \cup E_2 \cup E_3$. Particularly, for each newly added edge $e(0,i)$, we put $d_{0i} = t_i$, $l_{0i} = 0$, $u_{0i} = T_{cycle}$ and $P_{0i} = B_i$. The delay power curve $P_{ij}(d_{ij})$ of an edge $e(i,j) \in E_3$ is shown in Fig. 5(c). Upon these notations, the Lagrangian sub-problem (problem (3)) can be restated as to compute:

$$L(\vec{x}) = \min_{d} \sum_{e(i,j)\in E^*} \{P_{ij}(d_{ij}) + x_{ij}d_{ij}\} \quad (3a)$$

Subject to

$$\sum_{j:e(i,j)\in E^*} x_{ij} - \sum_{j:e(j,i)\in E^*} x_{ji} = 0, \quad \forall i \in V^* \quad (3b)$$

$$x_{ij} \geq 0, \quad \forall e(i,j) \in E_1 \cup E_2 \quad (3c)$$

Since each $P_{ij}(d_{ij})$ is a piecewise linear convex function, $P_{ij}(d_{ij}) + x_{ij}d_{ij}$ is also a convex function of d_{ij} for a given value of x_{ij}. It is important to note that, for a given value of vector $\vec{x}$, each term $\min_{d_{ij}}\{P_{ij}(d_{ij}) + x_{ij}d_{ij}\}$ can be optimized separately, since d_{ij} is now independent of each other among all the edges $e(i,j) \in E^*$. We define function $H_{ij}(x_{ij})$ for each $e(i,j) \in E^*$ as follows:

$$H_{ij}(x_{ij}) = \min_{d_{ij}}\{P_{ij}(d_{ij}) + x_{ij}d_{ij}\}$$

Then problem (3) can be rewritten as to compute:

$$L(\vec{x}) = \sum_{e(i,j)\in E^*} H_{ij}(x_{ij}), \quad \text{subject to (3b) and (3c).}$$

The Lagrange dual problem is to determine x^* such that

$$L(x^*) = \max_x L(x) = \max_x \sum_{e(i,j)\in E^*} H_{ij}(x_{ij}) \quad (4a)$$

Subject to

$$\sum_{j:e(i,j)\in E^*} x_{ij} - \sum_{j:e(j,i)\in E^*} x_{ji} = 0, \quad \forall i \in V^* \quad (4b)$$

$$x_{ij} \geq 0, \quad \forall e(i,j) \in E_1 \cup E_2 \quad (4c)$$

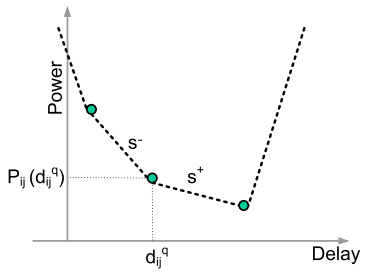

Fig. 6. Left Slope s^- and Right Slope s^+ at $(d_{ij}^q, P_{ij}(d_{ij}^q))$ in a DP Curve.

The following well-known theorem [2] establishes a connection between problem (2) and problem (4):

Theorem 1: Let x^* be a solution to the Lagrange dual problem, then $L(x^*)$ equals the optimal objective value of the original convex dual network flow problem.

2) Transformation into the Primal Minimum Cost Flow Problem: We first derive an explicit expression for $H_{ij}(x_{ij})$. As shown in Fig. 6, $P_{ij}(d_{ij})$ is a piecewise linear convex function. Let $(d_{ij}^q, P_{ij}(d_{ij}^q))$ be a break point (where the slope changes) on this function, and let s^- and s^+ denote the left slope and the right slope at this point, respectively. We can prove the following two lemmas.

Lemma 1: $H_{ij}(x_{ij}) = P_{ij}(d_{ij}^q) + x_{ij}d_{ij}^q$, if $-s^+ \leq x_{ij} \leq -s^-$.

Proof: When $d_{ij} \leq d_{ij}^q$, let $f(d_{ij}) = s^- d_{ij} + c$ (c is a constant) be the equation of the straight line on the left hand side of d_{ij}^q (with slope s^-). Due to the convexity of P_{ij}, $P_{ij}(d_{ij}) \geq f(d_{ij})$, for $d_{ij} \leq d_{ij}^q$. Thus $P_{ij}(d_{ij}) + x_{ij}d_{ij} \geq f(d_{ij}) + x_{ij}d_{ij} = (s^- + x_{ij})d_{ij} + c$. Note that $s^- + x_{ij} \leq 0$, so $(s^- + x_{ij})d_{ij} + c$ is monotonically decreasing. Therefore, for $d_{ij} \leq d_{ij}^q$, $P_{ij}(d_{ij}) + x_{ij}d_{ij} \geq (s^- + x_{ij})d_{ij} + c \geq (s^- + x_{ij})d_{ij}^q + c = P_{ij}(d_{ij}^q) + x_{ij}d_{ij}^q$. Similarly, when $d_{ij} \geq d_{ij}^q$, let $f(d_{ij}) = s^+ d_{ij} + c'$ (c' is a constant) be the equation of the straight line on the right hand side of d_{ij}^q (with slope s^+). Then $P_{ij}(d_{ij}) + x_{ij}d_{ij} \geq f(d_{ij}) + x_{ij}d_{ij} = (s^+ + x_{ij})d_{ij} + c'$. Since $s^+ + x_{ij} \geq 0$, $(s^+ + x_{ij})d_{ij} + c'$ is monotonically increasing. Therefore, $P_{ij}(d_{ij}) + x_{ij}d_{ij} \geq (s^+ + x_{ij})d_{ij} + c' \geq (s^+ + x_{ij})d_{ij}^q + c' = P_{ij}(d_{ij}^q) + x_{ij}d_{ij}^q$, for $d_{ij} \geq d_{ij}^q$. Combining the two cases completes the proof. □

Lemma 2: $H_{ij}(x_{ij}) = -\infty$ if $x_{ij} < -M$ or $x_{ij} > M$.

Proof: Consider the DP Curve $P_{ij}(d_{ij})$ of an edge $e(i,j)$ in E^*. If $x_{ij} < -M$, $P_{ij}(d_{ij}) + x_{ij}d_{ij}$ is minimum (with a value of $-\infty$) when d_{ij} tends to ∞ since $x_{ij}d_{ij}$ dominates $P_{ij}(d_{ij})$ when d_{ij} tends to ∞ (the slope of the rightmost segment of P_{ij} is M). Similarly, if $x_{ij} > M$, $P_{ij}(d_{ij}) + x_{ij}d_{ij}$ is minimum (with a value of $-\infty$) when d_{ij} approaches $-\infty$ since $x_{ij}d_{ij}$ dominates $P_{ij}(d_{ij})$ as d_{ij} tends to $-\infty$ (the slope of the leftmost segment of P_{ij} is $-M$). Therefore, $H_{ij}(x_{ij}) = \min_{d_{ij}}\{P_{ij}(d_{ij}) + x_{ij}d_{ij}\} = -\infty$, if $x_{ij} < -M$ or $x_{ij} > M$. □

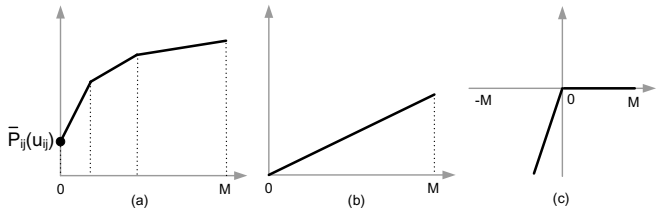

Fig. 7. Function $H_{ij}(x_{ij})$ of (a) an Edge in E_1; (b) an Edge in E_2; and (c) an Edge in E_3.

Recall that the objective of the Lagrange dual problem is to maximize $\sum_{e(i,j)\in E^*} H_{ij}(x_{ij})$ over $\vec{x}$. Therefore, we can safely add the constraint $-M \le x_{ij} \le M \ \forall e(i,j) \in E^*$, according to Lemma 2.

Note that, for the function $H_{ij}(x_{ij})$ corresponding to an edge $e(i,j) \in E_1$, the Lagrangian multiplier x_{ij} must be nonnegative, therefore, it can be further bounded as $0 \le x_{ij} \le M \ \forall e(i,j) \in E_1$. Together with Lemma 1 and the DP Curve in Fig. 5(a), we are able to express $H_{ij}(x_{ij})$ of each edge $e(i,j) \in E_1$ in the following manner.

$$H_{ij}(x_{ij}) = \begin{cases} P_{ij}(d_{ij}^{k_{ij}}) + d_{ij}^{k_{ij}} x_{ij}, & 0 \le x_{ij} \le b_{ij}(k_{ij}) \\ P_{ij}(d_{ij}^{k_{ij}-1}) + d_{ij}^{k_{ij}-1} x_{ij}, & b_{ij}(k_{ij}) \le x_{ij} \le b_{ij}(k_{ij}-1) \\ \vdots & \\ P_{ij}(d_{ij}^{q}) + d_{ij}^{q} x_{ij}, & b_{ij}(q+1) \le x_{ij} \le b_{ij}(q) \\ \vdots & \\ P_{ij}(d_{ij}^{1}) + d_{ij}^{1} x_{ij}, & b_{ij}(2) \le x_{ij} \le M \end{cases}$$

where $b_{ij}(q) = \frac{P_{ij}(d_{ij}^{q-1}) - P_{ij}(d_{ij}^{q})}{d_{ij}^{q} - d_{ij}^{q-1}}$. Note that it directly follows from the convexity of $P_{ij}(d_{ij})$ that $b_{ij}(q+1) \le b_{ij}(q)$.

Similarly, the constraint $0 \le x_{ij} \le M$ can also be added for each function $H_{ij}(x_{ij})$ that corresponds to an edge $e(i,j) \in E_2$, since the variable x_{ij} of an edge $e(i,j) \in E_2$ is also a nonnegative Lagrangian multiplier. According to Lemma 1 and the power delay curve $P_{ij}(d_{ij})$ in Fig. 5(b), the corresponding $H_{ij}(x_{ij})$ of an edge $e(i,j) \in E_2$ can be written as follows:

$$H_{ij}(x_{ij}) = l_{ij} x_{ij}, \quad 0 \le x_{ij} \le M$$

For the function $H_{ij}(x_{ij})$ corresponding to an edge $e(i,j) \in E_3$, the variable x_{ij} is not a Lagrangian multiplier, and it is bounded by $-M \le x_{ij} \le M$, according to Lemma 2. The corresponding function $H_{ij}(x_{ij})$ can be written as follows, according to Lemma 1 and the delay power curve in Fig. 5(c).

$$H_{ij}(x_{ij}) = \begin{cases} T_{cycle} x_{ij}, & -M \le x_{ij} \le 0 \\ 0, & 0 \le x_{ij} \le M \end{cases}$$

Fig. 7 shows the bounds and shapes of all these functions $H_{ij}(x_{ij})$. It is easy to observe that these functions $H_{ij}(x_{ij})$ are piecewise linear concave. Then, we let $C_{ij}(x_{ij}) = -H_{ij}(x_{ij})$, so that $C_{ij}(x_{ij})$ is a piecewise linear convex function. We can subsequently transform problem (4) into problem (5) as shown below, by replacing $H_{ij}(x_{ij})$ with $-C_{ij}(x_{ij})$.

$$\text{Minimize} \quad \sum_{e(i,j)\in E^*} C_{ij}(x_{ij}) \qquad (5a)$$

Subject to

$$\sum_{j:e(i,j)\in E^*} x_{ij} - \sum_{j:e(j,i)\in E^*} x_{ji} = 0, \qquad \forall i \in V^* \qquad (5b)$$

$$-M \le x_{ij} \le M \qquad \forall e(i,j) \in E_3 \qquad (5c)$$

$$0 \le x_{ij} \le M \qquad \forall e(i,j) \in E_1 \cup E_2 \qquad (5d)$$

3) Cost-Scaling Algorithm: Problem (5) is a convex cost flow problem in which the cost associated with the flow on an edge $e(i,j)$ is a piecewise linear convex function containing a number of linear segments. We can transform problem (5) into a minimum cost flow problem in an expanded network $G' = (V', E')$, and solve it using a cost-scaling algorithm [5].

In the transformation from G^* to G', each edge $e(i,j)$ in G^* will be replaced by one or more edges, depending on the number of linear segments in the corresponding function $C_{ij}(x_{ij})$. There are three categories of edges to consider, namely, E_1, E_2, and E_3:

• *Edge in* E_1: An edge in E_1 represents an input and output relationship of a module. For each edge $e(i,j) \in E_1$, $k = k_{i,j}$ edges are introduced in G', with one edge corresponding to one linear segment in the cost function C_{ij}. These edges have costs $-d_{ij}^{k}$, $-d_{ij}^{k-1}$, $-d_{ij}^{k-2}$, ..., $-d_{ij}^{1}$, upper capacities $b_{ij}(k)$, $b_{ij}(k-1) - b_{ij}(k)$, $b_{ij}(k-2) - b_{ij}(k-1)$, ..., $M - b_{ij}(2)$, respectively, and lower capacities of zero for all of them.

• *Edge in* E_2: An edge in E_2 represents an interconnect. Each edge $e(i,j) \in E_2$ is directly copied to G', with its cost, lower and upper capacity set as $-l_{ij}$, 0 and M respectively.

• *Edge in* E_3: Edges belonging to this category emanate from vertex 0 in G^*. For each of them, two edges are introduced in G'. The cost, lower capacity and upper capacity are respectively $-T_{cycle}$, $-M$ and 0 for one edge, and 0, 0 and M for the other edge.

It is well known that solving the convex cost flow problem in G^* is equivalent to solving the minimum cost flow problem in G'.[1] Particularly, due to the fact that the costs of the edges (corresponding to delay) are all integers in our problem, we can directly apply the cost-scaling algorithm on G' to find a minimum cost flow (as the correctness of the cost-scaling algorithm in [5] relies on the fact that all edge costs must be integer). Since this cost-scaling algorithm is a commonly known method to solve the minimum cost network flow problem, we will just briefly explain its major steps in the following, and more details can be found from [5], [4].

The cost-scaling algorithm maintains a pseudoflow x at each step such that x satisfies the upper and lower capacity constraints for all the edges but might violate the flow conservation constraint. For any pseudoflow x, the excess of a vertex i is defined as $exc(i) = \sum_{j:e(j,i)\in E'} x_{ji} - \sum_{j:e(i,j)\in E'} x_{ij}$ for all $i \in V'$. Vertex i is called an excess vertex if $exc(i) > 0$. A potential $\pi(i)$ is maintained for each vertex i. The cost-scaling algorithm proceeds by constructing and manipulating a residual network $G'(x)$ with respect to a pseudoflow x. The residual network consists only of edges with positive residual capacity. For a given residual network $G'(x)$ and a potential vector π, the reduced cost of an edge $e(i,j)$ is defined as $c_{ij}^{\pi} = c_{ij} - \pi(i) + \pi(j)$, where c_{ij} is the cost of edge $e(i,j)$. A flow or a pseudoflow x is said to be ε-optimal for some $\varepsilon \ge 0$ if for some potential vector π, $c_{ij}^{\pi} \ge -\varepsilon$ for every edge $e(i,j)$ in the residual network $G'(x)$. The cost-scaling algorithm treats ε as a parameter and iteratively obtains ε-optimal flows for successively smaller values of ε. Initially, $\varepsilon = \max\{|c_{ij}| : e(i,j) \in E'\}$ (T_{cycle} in our problem) and any feasible flow is ε-optimal. The algorithm then performs cost-scaling phases by repeatedly applying an improve-approximation procedure that transforms an ε-optimal flow into an $\varepsilon/2$-optimal flow. The improve-approximation procedure terminates when there are no excess vertices. After $O(log(n'T_{cycle}))$ cost-scaling phases, $\varepsilon < 1/n'$ and the algorithm terminates with an optimal flow where n' is the number of vertices in the network G'.

[1] There is a constant difference ($\sum_{e(i,j)\in E_1} \bar{P}_{ij}(u_{ij})$) between the cost of minimum convex cost flow in G^* and the cost of minimum cost flow in G'.

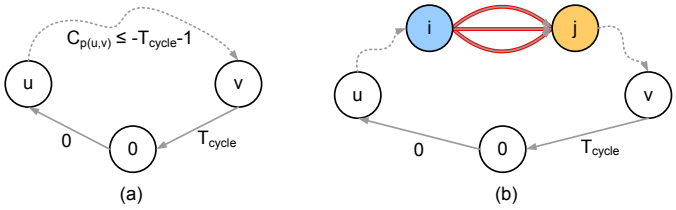

Fig. 8. Paths in the Proof of Lemma 3.

Time complexity analysis: The cost-scaling algorithm solves the minimum cost flow problem in $O(n'm'log(n'^2/m')log(n'U'))$ time, with n', m' and U' being the number of vertices, the number of edges and an upper bound of the edge costs, respectively. As for our problem, it is not difficult to see that there are $O(nk)$ vertices and $O(m+nk)$ edges in the expanded network G' where n and m are the numbers of vertices and edges in the original DAG $G=(V,E)$, and k is the maximum number of voltage level choices for each module. Therefore, the time complexity of solving our power optimization problem under timing constraint by this approach is $O(nk(m+nk)log(k^2n^2/m)log(knT_{cycle}))$, where T_{cycle} is the circuit timing constraint. The overhead is still acceptable since k is usually a very small number.

4) Solution Transformation: The convex cost-scaling algorithm upon termination gives an optimal flow x^* and the corresponding optimal node potentials π^*. Both the solutions x^* and π^* may be non-integer. Since all the edge costs are integers, there always exist integral optimal vertex potentials π. To determine them, we construct $G'(x^*)$ and determine the shortest path distance $sd(i)$ from vertex 0 to every other vertex $i \in V'$. Since all edge costs in $G'(x^*)$ are integers, each $sd(i)$ is also an integer. Then $\pi(i) = -sd(i)$ for each $i \in V'$ gives an integral optimal set of vertex potentials for problem (5). According to [1], an optimal solution to problem (2) can be obtained by first assigning $t_i = \pi_i$, then set $d_{ij} = t_j - t_i$ for each $e(i,j) \in E_1$. Note that we do not need to assign the value of d_{ij} for each $e(i,j) \in E_2$, since the power consumption on a wire is zero, and the timing constraints will be automatically satisfied if there exists a feasible solution to the original voltage assignment problem (problem (1)). If no feasible solutions of the voltage assignment problem exist, the total flow passing through vertex 0 will be larger than D, as stated in the following lemma.

Lemma 3: Given an instance of the voltage assignment problem, there exists a feasible solution to this instance if and only if the flow value through vertex 0 in the min-cost network flow problem in G' as constructed above is such that $f_0 \le D$, where $D = M - 1 = \sum_{e(i,j)\in E_1} \bar{P}_{ij}(l_{ij}) - \sum_{e(i,j)\in E_1} \bar{P}_{ij}(u_{ij})$.

Proof: *(only if part)* Suppose a feasible solution exists for the voltage assignment problem. Let C be the total cost of the flow network. Initially, $C = 0$ when the flow on each edge is zero, and this is automatically the minimum cost flow if there is no negative cycle. Note that every negative cycle must pass vertex 0, since the original netlist is a DAG. Now suppose, without loss of generality, the edge $e(0,u)$, the path from vertex u to vertex v (denoted by $p(u,v)$) and the edge $e(v,0)$ form a negative cycle, as shown if Fig. 8(a). Recall that the edge cost are all integers, thus the cost through $p(u,v)$ (denoted by $C_{p(u,v)}$) must be integral, i.e., $C_{p(u,v)} \le -T_{cycle} - 1$, otherwise the above mentioned negative cycle cannot be formed. This yields that one unit flow through vertex 0 will decrease the total cost C by at least one, and hence $C \le -f_0$, where f_0 denotes the flow value through vertex 0. Note that the optimal objective value of the Lagrange dual problem (problem (4)) $L(x^*) = (-C) + \sum_{e(i,j)\in E_1} \bar{P}_{ij}(u_{ij})$. If $f_0 > D$, then $C < -D$, thus $L(x^*) > D + \sum_{e(i,j)\in E_1} \bar{P}_{ij}(u_{ij}) = \sum_{e(i,j)\in E_1} \bar{P}_{ij}(l_{ij})$. According to theorem 1, $L(x^*)$ equals the optimal objective value of problem (2) (the dual network flow problem), so the objective value of problem (2) is also greater than $\sum_{e(i,j)\in E_1} \bar{P}_{ij}(l_{ij})$, which corresponds to the highest possible total power consumption of all the modules. This is impossible and we can thus conclude that no feasible solutions exist for the original voltage assignment problem exists, contradictory to our original assumption.

(if part) Suppose the flow through vertex 0 is less than or equal to D, i.e., $f_0 \le D$. Again, we prove by contradiction. If there is no feasible solution to the original voltage assignment problem, it means that the timing constraint T_{cycle} will be violated by some critical path $p(u,v)$ even if each module on $p(u,v)$ is assigned to work at its highest legal voltage level. Given this, there must be at least one module with its input vertex as i and output vertex as j in the expanded network G', such that all the edges between vertex i and vertex j are saturated, since a negative cycle will be formed with vertex 0 otherwise (as shown in Fig. 8(b), all the red edges are saturated). Note that the sum of the capacities of these thickened red edges is $M = D+1$ and the flow through these edges must go through vertex 0. Therefore, we can conclude that $f_0 > D$, contradictory to our original assumption. □

If the flow through vertex 0 is not more than D, we can construct a feasible solution for the original problem (problem (1)) with discretization constraint $(1e)$ from this optimal solution of problem (2), by taking the largest possible delay choice d_{ij}^q smaller than or equal to d_{ij} for each edge $e(i,j) \in E_1$. It can be easily shown that this solution is feasible to our original problem, and we can see from the experimental results that this transformation method can approximate the optimal power consumption well and gives large power savings.

B. Simulated Annealing

A candidate floorplan will be generated by a FAST-SP [15] floorplanner in each iteration of the annealing process, after which the wire delays are estimated and the voltage levels of the modules are assigned by solving the dual network flow problem. The quality of the current solution will then be evaluated by an objective function. When the annealing process terminates, the best found solution will be returned. Particularly, if the best found solution violates timing, our program will report that no legal solutions are found.

1) Moves: There are three kinds of moves to perturb the sequence pair representation 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.

- *Change aspect ratio* - The aspect ratio of an arbitrary block m_i is changed.
- *Swap two blocks in the* α *sequence* - Two randomly chosen blocks are swapped in the α sequence.
- *Swap two blocks in both sequences* - Two randomly chosen blocks are swapped in both the α and β sequences.

2) Speeding up heuristic: The cost-scaling algorithm is capable of re-computing a minimum cost flow upon incremental changes of the edge costs with relatively less effort, in comparison with solving the whole problem once again. This property can potentially speedup the whole engine when we embed the cost-scaling solver into the annealing process. In practice, only a small portion of nets will get their lengths changed after a random move. This results in a high chance that the minimum cost flow found previously is also optimal (or close to optimal) after the random move. Therefore, in our

implementation, we will first check whether the previous solution is still optimal in the new iteration. If this is true, the previous solution will be directly returned; otherwise, refinement will be conducted based on the previous solution. This heuristic has proven to be very helpful in improving the efficiency of our approach.

3) Cost Function: We use the following cost function to evaluate a candidate floorplan.

$$\psi = A + \lambda_w W + \lambda_{pn}\Phi + \lambda_p P \tag{4}$$

In this function, A is the area of the floorplan; W is the total wire length estimated by the half perimeter bounding box method; Φ represents the power network resource; P denotes the total power consumption, and particularly, P is set to a large number to give a heavy penalty when the candidate floorplan violates timing (when the amount of flow passing through vertex 0 exceeds D). Note that there exists a possibility of accepting a move that results in a floorplan violating timing in our searching process, and this strategy allows us to explore the whole solution space effectively. The parameters λ_w, λ_{pn}, and λ_p can be used to adjust the relative weighting between the contributing factors, and they are determined in a random walk with 1000 random moves before the annealing process starts, in such a way that the four factors, area, wire length, power network resource and power cost, are contributing similarly to the cost function.

4) Annealing Schedule: In our annealing engine, the temperature is set to 10^5 at the beginning and will drop at a rate of 0.95. At each temperature, $n/2$ random moves are performed, where n is the number of modules in the test case. The annealing process stops when the temperature falls below 10^{-5}.

IV. Experimental Results

The previous work [9] is the most updated one in handling the floorplanning problem under timing constraint in the context of MSV. In order to compare with [9], we performed a set of experiments on the test cases provided by the authors of [9]. There are totally six test cases, with the number of blocks ranging from 10 to 300. These test cases are based on the GSRC benchmarks, with power and delay specifications added for each block. In these test cases, every block has two legal working voltage levels, each of which is associated with both a power consumption value and a delay value.

Note that the approach in [9] performed level shifter insertion into the netlist, after the voltage assignment step. Whenever there is a net through which signal is sent from a module at low voltage level to a module at high voltage level, a level shifter will be inserted into this net. Each level shifter is associated with fixed delay and power values, and will affect the timing and power consumption of the whole circuit. In order to consider level shifters so as to have a fair comparison with [9], we extended our floorplanner with a conservative approach to take level shifters into consideration. When the DAG representation of the input netlist is given, we will first compute the longest path L (in terms of the number of edges on a path) from primary input to primary output in the DAG. It is trivial that at most L level shifters will be inserted on any path of the DAG after voltage assignment. This indicates that after level shifter insertion, the delay of the whole circuit will be increased by $L \cdot d_{ls}$ units in the worst case where d_{ls} denotes the delay of a level shifter. Therefore, in order to guarantee that the timing constraint can still be satisfied after level shifter insertion, we simply put $T_{cycle} - L \cdot d_{ls}$ as the timing constraint of the circuit while solving the voltage assignment problem.

The comparisons are displayed in Table I. The column *Max Power (MaxP)* denotes the total power consumption when each

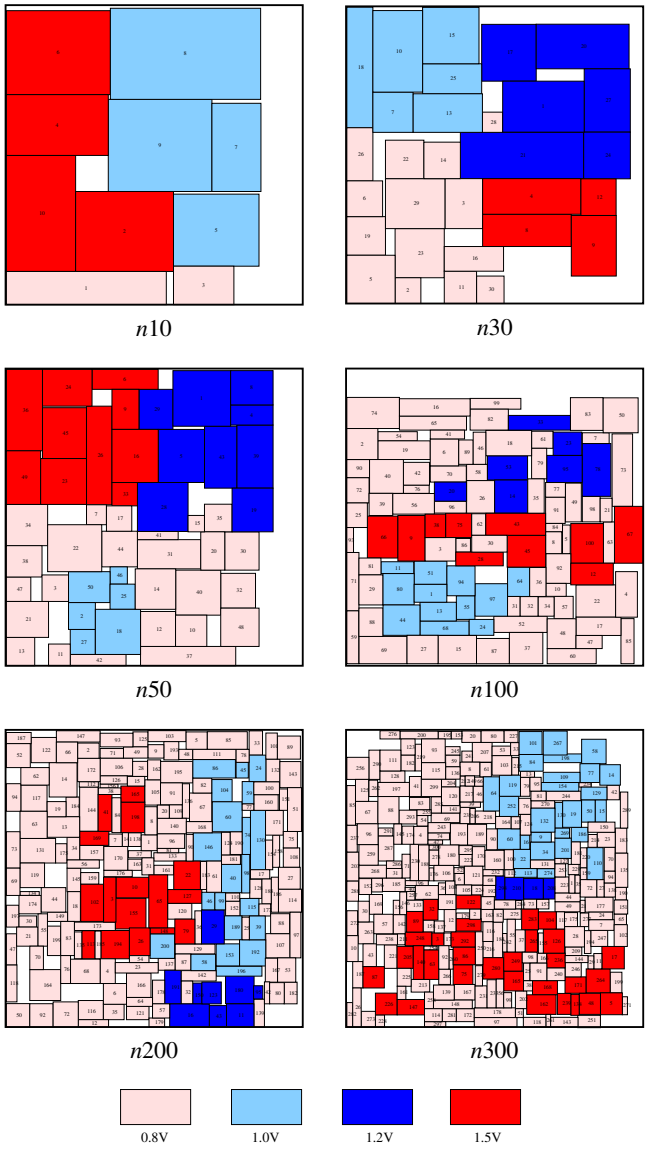

Fig. 9. Resultant Floorplans with Each Module Having Four Legal Working Voltage Levels

module of the circuit works at its highest legal voltage level, and the column *Power Cost with LS (P)* shows the sum of the actual power consumption of each module plus the power consumption of all the level shifters inserted. The power saving is calculated by $(MaxP - P)/MaxP$. We can see that more power saving can be achieved in much less running time by using our approach. An average of 24.61% (v.s. 6.68% by [9]) power saving can be achieved in an average running time of $273s$ (v.s. $1911s$ by [9]) with our approach. Besides, we further demonstrated the effectiveness of our approach by performing another set of experiments in which the same set of circuits is used but the number of legal working voltage levels for each module is augmented to four. For this second set of experiments, the resultant floorplans of all the six circuits are displayed in Fig 9, and the detailed results are listed in Table II, from which we can see that an average of 28.62% power saving can be achieved. Our algorithm was implemented in the C programming language and all the experiments were performed on a Linux machine with a 3.20 GHz CPU and 2 GB RAM.

TABLE I

COMPARISONS WITH PREVIOUS WORK [9]

Data Set	Max Power (MaxP)	Power Cost with LS (P)		Power Saving (%)		Power Network Resource		LS Number		Dead Space (%)		Wire Length		Run Time (s)	
		ours	[9]	ours	[9]	ours	[9]	ours	[9]	ours	[9]	ours	[9]	ours	[9]
n10	216841	189942	216841	12.40	0	1516	965	4	0	2.32	4.87	1693	1729	1.09	6.001
n30	205650	153526	190717	25.35	7.26	1738	1369	26	57	7.74	9.03	8517	8202	9.31	115.069
n50	195140	152684	172884	21.76	11.40	1589	1514	34	119	8.96	21.10	15726	16395	25.34	569.360
n100	180022	120450	179876	33.09	0.10	1612	1671	77	92	7.53	34.07	17381	18734	121.61	1768
n200	177633	135250	174818	23.86	1.58	1619	2040	129	399	7.19	46.52	18597	20128	454.60	4212
n300	273499	188113	219492	31.22	19.75	1844	2147	151	452	18.11	44.10	23145	27026	1026.68	4800
Average	-	156661	192438	24.61	6.68	1653	1617.7	70	186	8.64	26.61	14176	15369	273.10	1911.74
Difference	-	1	1.23	1	0.27	1	0.98	1	2.66	1	3.08	1	1.08	1	7

TABLE II

RESULTS WITH MORE LEGAL WORKING VOLTAGE LEVELS

Data Set	Power Cost with LS	Power Saving (%)	Power Network Resource	LS Number	Dead Space (%)	Wire Length	Run Time (s)
n10	167012	22.98	1818	9	4.08	1839	1.46
n30	142717	30.60	2012	37	16.80	9132	13.79
n50	143562	26.43	1945	43	10.66	15438	43.44
n100	124428	30.88	2377	108	8.98	16277	195.85
n200	133360	24.93	2382	176	11.16	17968	780.74
n300	175281	35.91	2413	188	12.83	22348	1812.37
Average	147726	28.62	2157	94	10.75	13834	474.61

V. CONCLUSION

In this paper, we proposed a framework for power optimization under timing constraints in floorplanning. We show that the voltage assignment problem under timing constraint can be formulated as a dual network flow problem that can be transformed into a primal minimum cost network flow problem using the Lagrangian relaxation technique and can subsequently be solved optimally by a cost-scaling algorithm in polynomial time when the delay choices of each module are continuous in the real or integer domain. We can make use of this approach to obtain a feasible voltage assignment solution in the general cases with power consumption approximating the minimum one. Globally, a simulated annealing based FAST-SP floorplanner is employed to pack the modules providing physical information to the voltage assignment engine. In each iteration, the cost-scaling solver is invoked to assign voltage levels, after updating the wire delays. Experimental results show that our approach is very effective and efficient in reducing power consumption and power network resource. In comparison with previous work [9], our framework achieves an additional 18% power saving on average, in much less running time.

REFERENCES

[1] 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.

[2] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, "Network Flows: Theory, Algorithms, and Application," Prentice Hall, 1993.

[3] R.L.S. Ching and E.F.Y. Young, "Post-placement Voltage Island Generation," *Proceedings of International Conference on Computer-Aided Design,* pp.641-646, 2006.

[4] A.V. Goldberg, "An efficient implementation of a scaling minimum-cost flow algorithm," *Journal of Algorithms,* 22:1-29, 1997.

[5] A.V. Goldberg and R.E. Tarjan, "Solving minimum cost flow problem by successive approximation," *Proceedings of the 19th ACM Symposium on the Theory of Computing,* pp.7-18, 1987.

[6] 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.

[7] 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 Computer Design,* pp.689-696, 2004.

[8] D.E. Lackey, P.S. Zuchowski, T.R. Bednar, D.W. Stout, S.W. Gould, and J.M. Cohn, "Managing Power and Performance for System-on-Chip Designs Using Voltage Islands," *Proceedings of International Conference on Computer-Aided Design,* pp.195-202, 2002.

[9] W.-P. Lee, H.-Y. Liu, and Y.-W. Chang, "Voltage Island Aware Floorplanning for Power and Timing Optimization," *Proceedings of International Conference on Computer-Aided Design,* pp.389-394, 2006.

[10] 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,* pp.650-655, 2007.

[11] B. Liu, Y. Cai, Q. Zhou, and X. Hong, "Power Driven Placement with Layout Aware Supply Voltage Assignment for Voltage Island Generation in Dual-Vdd Designs," *Proceedings of Asia and South Pacific Design Automation Conference,* pp.582-587, 2006.

[12] Q. Ma and E.F.Y. Young, "Voltage Island-Driven Floorplanning," *Proceedings of International Conference on Computer-Aided Design,* pp.644-648, 2007.

[13] 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,* pp.798-803, 2007.

[14] H. Murata, K. Fujiyoushi, S. Nakatake, and Y. Kajitani, "Rectangle-Packing-Based Module Placement," *Proceedings of International Conference on Computer-Aided Design*, pp.472-479, 1995.

[15] X. Tang and D.F. Wong, "FAST-SP: a fast algorithm for block placement based on sequence pair," *Proceedings of Asia and South Pacific Design Automation Conference,* pp.521-526, 2001.

[16] K. Usami and M. Horowitz, "Clustered Voltage Scaling Technique for Low-Power Design," *Proceedings of International Symposium on Low Power Electronics and Design,* pp.3-8, 1995.

[17] H. Wu, I.M. Liu, D.F. Wong, and Y. Wang, "Post-Placement Voltage Island Generation under Performance Requirement," *Proceedings of International Conference on Computer-Aided Design,* pp.309-316, 2005.

[18] 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.