Generic Integer Linear Programming Formulation for 3D IC Partitioning*

WAN-YU LEE, IRIS HUI-RU JIANG AND TSUNG-WAN MEI
Department of Electronics Engineering and Institute of Electronics
National Chiao Tung University
Hsinchu, 300 Taiwan

The success of 3D IC requires novel EDA techniques. Although many EDA techniques exist, this paper focuses on 3D IC partitioning, especially at the architectural level to maximize its benefits. First, logical formulations for 3D IC partitioning problems are derived and then the formulations are transformed into integer linear programs (ILPs). The ILP formulation can minimize the usage of vertical interconnects subject to the footprint and power consumption constraints. The flexibility of ILP formulation can be demonstrated by extending the generic ILP formulation to support designs with multiple supply voltages. This study proposes ILP reduction techniques to speed up the convergence. Experimental results based on the GSRC benchmark show that our approach converges efficiently. Moreover, our approach is flexible and can readily extend to the partitioning problems with variant objectives and constraints, and with different abstraction levels, for example, from the architectural level down to the physical level. This flexibility makes the ILP formulation a superior alternative to 3D IC partitioning problems.

Keywords: 3D IC, partitioning, integer linear program, through-silicon via, algorithms

1. INTRODUCTION

As technology advances into the nanometer era, the substantially high design complexity and the integration of heterogeneous designs in a chip require fresh design methodology. Three-dimensional (3D) integration, an emerging and promising technology, can meet this requirement. Compared with the two-dimensional (2D) implementation, a 3D IC stacks multiple dies and interconnects them vertically thus offering a smaller footprint, shorter interconnect delay/power, wider bandwidth, higher-density/larger-scale integration over heterogeneous technologies, potentially lowering cost and creating a shorter time-to-market [1-4]. As the optical lithography is approaching its natural limits as predicted by the International Technology Roadmap for Semiconductors (ITRS) [5], increasing the scale of integration is particularly attractive. Theoretically, a 3D IC may integrate and interconnect tens of dies, vertically. Fig. 1 illustrates a 3D IC using through-silicon vias (TSVs) to interconnect three sets of device and metal layers. One bonding layer is placed in between two adjacent sets. Although 3D integration is feasible and beneficial from many perspectives, the process variation on TSV, say 11.1%, is much larger than the variation of threshold voltage of the device, say 6.4% [3]. Hence, vertical interconnects (TSVs) should cleverly be used.

Received December 13, 2010; revised April 10, 2011; accepted June 14, 2011.
Communicated by Yao-Wen Chang.
* An earlier version of this paper was presented at SOCC-2009 [12]. This work was partially supported by Synopsys, Faraday, and NSC of Taiwan under Grant No. NSC 100-2220-E-009-047.

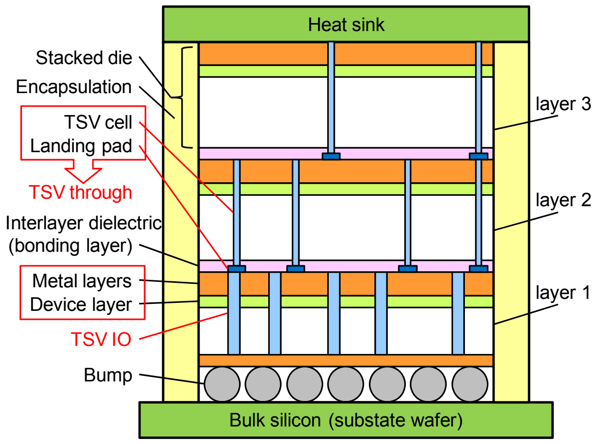


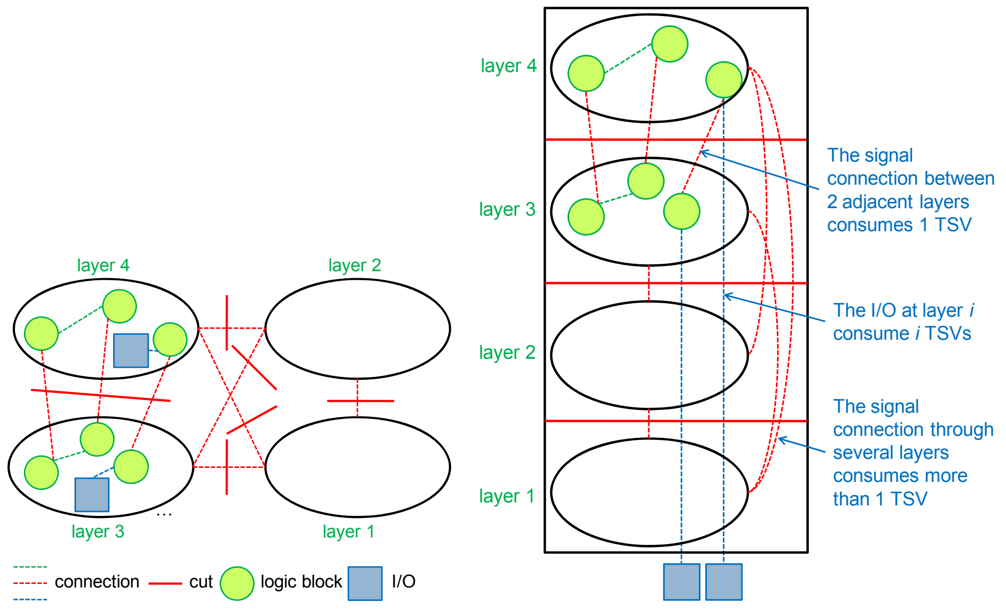
Fig. 1. A 3D IC with TSVs for vertical interconnects (via last, front to back integration) [3].

Successful 3D integration requires novel design and automation techniques. Partitioning is especially important because it has a great impact on the circuit performance across all abstraction levels. On the other hand, decisions and optimizations made at higher abstraction levels can gain more benefits than those done at lower levels. For example, functional partitioning at the architectural level has more benefits than circuit partitioning at the physical level, because functional partitioning can compromise functionality and connectivity between operations (logic blocks). Debugging a functionally partitioned design is significantly easier than debugging a circuit-wise partitioned one. In addition, the problem size at the architectural level is obviously smaller than that at the physical level, *e.g.*, tens of logic blocks, usually no more than one hundred, versus billions of cells. This paper focuses on architectural-level partitioning for 3D IC integration. This approach can readily extend to all abstraction levels. Here, one partition corresponds to one layer (stacked die), which is comprised of one device layer plus a set of metal layers. Previous works empirically showed that minimizing the cutsize can indirectly optimize wirelength [6, 7, 15]. Thus, good partitioning algorithms may result in better solutions for later stages.

Since all interactions with external signals can only be conducted through the bottom layer, the partitioning problem for 3D ICs is quite different from the conventional one. As shown in Fig. 2, the conventional multi-way partitioning algorithms, for example hMetis [8], cannot guarantee optimality for 3D integration. Considering a naïve extension based on the conventional multi-way partitioning, after each partition is assigned to a distinct set of device and metal layers in a 3D IC, one cut net or one input/output (I/O) may contribute more than one TSV when its related partitions are unfortunately scattered over several non-adjacent layers. Thus, the minimized cutsize during multi-way partitioning cannot reflect how well the usage of TSVs is optimized. Moreover, the arrangement of these partitions could be non-trivial.

This paper develops a partitioning approach to handle the impact on external interactions and the discrepancy with the conventional multi-way partitioning. We first derive logical formulations for 3D IC partitioning problems and then transform the formulations into integer linear programs (ILPs). The flexibility of ILP formulation can be demonstrated by extending the generic ILP formulation to support designs with multiple supply voltages. This study also proposes ILP reduction techniques to speed up the convergence by solving ILPs iteratively and performing pre-clustering. These techniques not only improve the efficiency of ILPs but also maintain a reasonably good solution quality. The ILPs can control the footprint, minimize the TSV count, and lower the power simultaneously. More importantly, this approach is very flexible and can readily extend to the partitioning problems with variant objectives and constraints, and with different abstract levels, such as the architectural, logic, or physical level. This flexibility makes the ILP formulation a superior alternative to the 3D IC partitioning problems.

The remainder of this paper is organized as follows. Section 2 introduces preliminaries of this work. Section 3 defines our problem, formulates ILPs, as well as proposes reduction techniques. Section 4 then extends the ILP formulations to handle designs with multiple supply voltages. Section 5 shows and analyzes experimental results. Finally, section 6 concludes this paper.


layer 4
layer 2
layer 3
layer 1
...
connection
cut
logic block
I/O
layer 4
layer 3
layer 2
layer 1
The signal connection between 2 adjacent layers consumes 1 TSV
The I/O at layer *i* consume *i* TSVs
The signal connection through several layers consumes more than 1 TSV

(a) hMetis multi-way partitioning. (b) Conversion to 3D integration.

Fig. 2. Conventional multi-way partitioning in (a) cannot guarantee optimality in 3D IC integration in (b).

2. PRELIMINARIES

This section introduces TSVs, multilevel multi-way partitioning, and multiple supply voltage.

2.1 Vertical Interconnects

The following TSV components are used for vertical interconnects in 3D ICs (see

Fig. 1).

1. A TSV cell connects a signal net between two internal neighboring layers.
2. Given two consecutive layers, a landing pad at the lower layer contacts the upper layer, and its size must be larger than a TSV cell for safe alignment. Since the landing pad is connected to the top metal layer, it is assumed not to affect the devices under it, and its area could be neglected.
3. A TSV through consists of a TSV cell and a landing pad.
4. A TSV IO connects an I/O terminal to one package pin. The diameter of a TSV IO is greater than that of a TSV cell. Please note that only the bottom set of device & metal layers (layer 1) contains the TSV IOs that can connect to the external package pins.

Hence, without loss of generality, this paper counts a TSV through or a TSV IO as one TSV.

2.2 Multilevel Multi-Way Partitioning

To produce a high-quality solution for large hypergraphs in a small amount of runtime, the multilevel partitioning adopts a bottom-up hierarchical approach based on recursive clustering; hMetis is one of the earliest and best multilevel hypergraph partitioning algorithms [8].

The multilevel scheme consists of three phases: coarsening (clustering), initial partitioning, and uncoarsening (disclustering) & refinement, as illustrated in Fig. 3. The initial bipartition is simply a random bipartition. The refinement on the uncoarsening hypergraphs is done by the classical Fiduccia and Mattheyses heuristic for bipartitioning on hypergraphs [9].

2.3 Multiple Supply Voltage (MSV)

Power consumption is a critical issue in modern VLSI design. Multiple supply voltage (MSV) is one of several low power techniques. MSV reduces system power by delivering a lower supply voltage to noncritical parts of a system, while maintaining system performance. Each logic block corresponds to a number of timing safe supply voltage values which ensure that the logic block works correctly. For example, the supply voltage of a logic block with timing safe supply voltages = {1.0V, 1.2V, 1.3V} is being decided. Designers deduce that less power consumption is better. Since power consumption is quadratically proportional to the supply voltage, 1.0V is assigned to the block. Considering the overall design targets, the fittest supply voltage value would be chosen as the operating voltage.

3. GENERIC ILP FORMULATION FOR 3D IC PARTITIONING

This section details our generic ILP formulation as well as the speed-up generic ILP formulation.

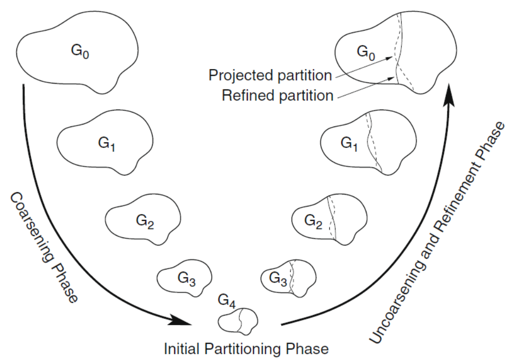


Fig. 3. The three phases of hMetis [8]: coarsening, initial partitioning, uncoarsening phases.

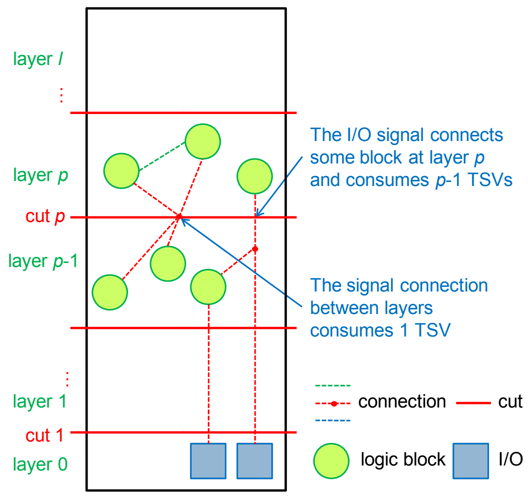

layer l
layer p
cut p
layer p-1
layer 1
cut 1
layer 0
The I/O signal connects some block at layer p and consumes p-1 TSVs
The signal connection between layers consumes 1 TSV
connection
cut
logic block
I/O

Fig. 4. 3D IC partitioning.

3.1 Problem Definition

As depicted in Fig. 4, the input design contains logic blocks, I/O terminals, as well as the connections between them. Implementing this design using an *l*-layer 3D IC is equivalent to partitioning it into *l* layers with the minimum TSV usage and the minimum footprint. Hence, the *l*-layer 3D IC partitioning problem is formulated as follows.

Problem: *l*-layer 3D IC Partitioning
Given the number of layers *l* of a 3D IC and a design with logic blocks, I/O terminals, the connectivity between them, find *l* partitions with the minimum TSV usage and with the best area-balance.

3.2 Hypergraph Modeling

The input design is modeled as a hypergraph, $G = (V, E)$. A vertex in V represents either an I/O terminal or a logic block, while a hyperedge in E denotes a net connecting two or more vertices. An l-layer 3D IC can be achieved by a multi-way partitioning on the input hypergraph. The TSV usage is counted by the number of TSVs used, including both TSV throughs and TSV IOs. The footprint is contributed by the largest area over l layers, so the minimum footprint can be achieved by minimizing the largest area or by balancing the area of each layer in the 3D IC.

3.3 ILP Formulation

Table 1 lists the notations used for ILP formulation and the basic constraints. Based on Table 1, the α-bounded area limits the footprint. Therefore, the objective function only counts the number of TSVs used. The TSV usage can be given by

$$\sum_{ei \in E} \sum_{p = 1 \ldots l} y_{i,p}, \tag{1}$$

Table 1. Notations used in our generic ILP formulation.

Category	Notation	Description
General	l	The number of layers l in a 3D IC. The external world is viewed as layer 0.
	p	Cut p is the cut between layers $p - 1$ and p
	G	Input design is a hypergraph $G = (V, E)$
	v_i	Vertex $v_i \in V$ is a logic block or an I/O terminal.
	e_i	Hyperedge $e_i = \{v_{i1}, v_{i2}, \ldots, v_{iq}\} \in E$ is a q-pin net.
Layer assignment	$x_{i,j}$	$x_{i,j}$ is a 0-1 integer variable associated with vertex v_i. $x_{i,j} = 1$ if v_i is assigned to layer j; otherwise, $x_{i,j} = 0$. For an I/O terminal v_i, $x_{i,0} = 1$, while for a logic block v_j, $x_{j,0} = 0$.
TSV	$y_{i,p}$	$y_{i,p}$ is a 0-1 integer variable associated with net e_i. $y_{i,p} = 1$ if net e_i introduces a TSV to cut p, between layers $p - 1$ and p, $1 \le p \le l$; otherwise, $y_{i,p} = 0$.
Area	s_i	s_i is the area of vertex v_i. $s_i = 0$ if v_i is an I/O terminal.
	t	t is the area of a single TSV.
	A	$A = \sum_{vi \in V} s_i$, total area of all vertices.
	α	α is the perfectly balanced area for each layer, $\alpha = A/l$.
	ε	ε is the α-bounded ratio of the deviation from the perfectly balanced area, $0 < \varepsilon < 1$. The α-bounded area for each layer is subject to the range $[(1 - \varepsilon)\alpha, (1 + \varepsilon)\alpha]$.

where $y_{i,p}$ is a 0-1 variable computed by the contribution from a q-pin net $e_i = \{v_{i1}, v_{i2}, \ldots, v_{iq}\} \in E$ to cut p, and the computation can be fulfilled by applying logical operations as follows,

$$\begin{aligned} y_{i,p} = {} & [(x_{i1,1} \textbf{ OR } x_{i1,2} \textbf{ OR} \ldots \textbf{OR } x_{i1,p\text{-}1}) \textbf{ OR} \\ & (x_{i2,1} \textbf{ OR } x_{i2,2} \textbf{ OR} \ldots \textbf{OR } x_{i2,p\text{-}1}) \textbf{ OR} \ldots \textbf{OR} \\ & (x_{iq,1} \textbf{ OR } x_{iq,2} \textbf{ OR} \ldots \textbf{OR } x_{iq,p\text{-}1})] \textbf{ AND} \\ & [(x_{i1,p} \textbf{ OR } x_{i1,p+1} \textbf{ OR} \ldots \textbf{OR } x_{i1,l}) \textbf{ OR} \\ & (x_{i2,p} \textbf{ OR } x_{i2,p+1} \textbf{ OR} \ldots \textbf{OR } x_{i2,l})) \textbf{ OR} \ldots \textbf{OR} \\ & (x_{iq,p} \textbf{ OR } x_{iq,p+1} \textbf{ OR} \ldots \textbf{OR } x_{iq,l})],\ 1 \le p \le l. \end{aligned} \tag{2}$$

Eq. (2) means $y_{i,p} = 1$ if some logic blocks of net e_i are located below cut p and some are above. Moreover, Table 2 details how to convert logical expressions [10], *e.g.*, logical AND and logical OR, into mathematical constraints.

Therefore, combining Tables 1 and 2 together allows us to successfully formulate the 3D IC partitioning problem as the following ILP:

$$\begin{array}{ll} \text{minimize} & \sum_{ei\in E} \sum_{p=1..l} y_{i,p} \\ \text{subject to} & \sum_{j=0..l} x_{i,j} = 1,\ v_i \in V; \\ & (1-\varepsilon)\alpha \le \sum_{vi\in V} s_i\, x_{i,j} + t\sum_{ei\in E} y_{i,j} \le (1+\varepsilon)\alpha,\ 1 \le j \le l. \end{array}$$

Some detailed equations and formulations in Tables 1 and 2 are not shown here for simplicity. (Note that they should be fed into the ILP solver for the completeness of the formulation.) It can be seen that our formulation is very flexible because the objective function, TSV usage, and the main constraint, area balance, can be mutually exchanged.

Table 2. Logical expressions vs. mathematical constraints.

Logical Expression	Mathematical Constraints
$C = A$ AND B	$C \le A$
	$C \le B$
	$C \ge A + B - 1$
$C = A$ OR B	$C \ge A$
	$C \ge B$
	$C \le A + B$

3.4 ILP Reduction

The state-of-the-art ILP solvers, *e.g.*, LINGO and CPLEX, are capable of finding optimal/feasible solutions according to our formulation. However, the runtime of ILP solvers might be lengthy. For a circuit with hundreds of logic blocks, the runtime can be unreasonable due to numerous constraints and variables. This study proposes two reduction techniques to speed up ILP.

First of all, we solve ILP iteratively to speed up the convergence. Fig. 5 lists the iterative ILP algorithm. For each iteration, the UNBALANCED-2LAYER-ILP function partitions the circuit into *two* layers. After partitioning is finished, the logic blocks on the top layer (*i.e.*, layer two) are fixed. Thus, in line 4 of ITERATIVE-ILP, these blocks are excluded from the future iteration by taking the difference of G and the set of vertices placed on the top layer. Please notice that when $l = 1$, only one layer is left implying that the un-fixed blocks will be placed in layer one. Hence, no more partitioning is needed.

```
ITERATIVE-ILP($G$, $l$)
1. $m = 0$; // #TSVs
2. while $l > 0$ do
3.     $m = m$ + UNBALANCED-2LAYER-ILP($G$, $l$);
4.     $G = G - \{v_i | x_{i,2} = 1\}$;
5.     $l = l - 1$;
```

Fig. 5. The iterative ILP algorithm. UNBALANCED-2LAYER-ILP means 2-way partitioning with an unbalanced area constraint.

Except the area constraint, the UNBALANCED-2LAYER-ILP function is basically the same with the generic ILP formulation mentioned in section 3.3. In the top layer, the constraints still remain while they are modified for dealing with the unbalance area in the bottom layer. Since the bottom layer is actually mapping to layers 1, …, $l - 1$, the area at bottom layer should be $l - 1$ times of that at the top layer. As the result, the UNBALANCED-2LAYER-ILP can be formulated as follows.

$$\begin{array}{ll} \text{minimize} & \sum_{ei \in E} \sum_{p=1,2} y_{i,p} \\ \text{subject to} & \sum_{j=0..2} x_{i,j} = 1,\ v_i \in V; \\ & (1 - \varepsilon)\alpha \leq \sum_{vi \in V} s_i\, x_{i,2} + t\sum_{ei \in E} y_{i,2} \leq (1 + \varepsilon)\alpha; \\ & (1 - \varepsilon)\alpha(l - 1) \leq \sum_{vi \in V} s_i\, x_{i,1} + t\sum_{ei \in E} y_{i,1} \leq (1 + \varepsilon)\alpha(l - 1). \end{array}$$

After termination, the number of TSVs and the layer of each logic block are obtained. Since UNBALANCED-2LAYER-ILP always returns a partial solution that is either optimal or feasible, the summation of these partial solutions is always feasible. In fact, our experimental results will show that iterative ILP trades a reasonable number of TSVs with a dramatic runtime reduction.

Second, logic blocks are pre-clustered to further reduce variables used in ILP. We use FirstChoice proposed in [14] as our pre-clustering method. FirstChoice clusters logic blocks based on the strength of connectivity by net weights, which is the inverse of the number of blocks connected by this net minus one. FirstChoice performs coarsening in descending order of net weights. The area of this cluster is the summation of logic blocks belonging to it. For area balancing, we set an area upper bound for a cluster. In addition, I/O terminals will not be coarsened. Before applying ILP, logic blocks with strong connections are clustered together. Then, the iterative-ILP algorithm described above is applied. During ILP, the logic blocks in the same cluster are regarded as one vertex.

4. AN EXTENSION TO MSV DESIGN

For demonstrating the flexibility of our generic ILP, this study extends ILP formulation proposed in section 3 to MSV design. For each block, several timing safe supply voltages are given. One and only one timing safe voltage will be chosen as the operating voltage. The power constraint is added to minimize the overall power consumption. The extension is detailed in the following subsections.

4.1 Problem Definition

For MSV design, the targets of ILP include minimizing TSV usage, footprint, or power. Hence, the extended ILP problem with MSVs can be defined as follows:

Problem: *l*-layer 3D IC Partitioning with MSVs
Given the number of layers l in a 3D IC and a design with logic blocks, I/O terminals, the connectivity between them, and the timing safe supply voltages of each logic block, find l partitions with the minimum TSV usage, subject to area-balance as well as power-balance.

Please note that due to the flexibility of the ILP formulation, the objective and the constraints can be exchanged (*i.e.*, minimize power consumption subject to the limit of TSV usage).

4.2 Extended ILP Formulation for MSV Design

Table 3 presents the notations for extended ILP formulation with MSVs and their constraints. Compared to Table 1, which lists the variables used in generic ILP formulation, Table 3 includes notations for voltage assignment and power constraints. The formulation of extended ILP with MSVs is presented as follows,

$$
\begin{array}{ll}
\text{minimize} & \sum_{ei\in E} \sum_{p=1..l} y_{i,p} \\
\text{subject to} & \sum_{j=0..l} x_{i,j} = 1,\ \sum_{j=1..r} z_{i,j} = 1, v_i \in V; \\
 & (1-\varepsilon)\alpha \le \sum_{vi\in V} s_i\, x_{i,j} + t\sum_{ei\in E} y_{i,j} \le (1+\varepsilon)\alpha,\ 1 \le j \le l. \\
 & (1-\varepsilon)\beta \le \sum_{vi\in V} \boldsymbol{c_i} x_{i,j}(\sum_{k=1..r} z_{i,k} U_k^2) + w\sum_{ei\in E} y_{i,j} \le (1+\varepsilon)\beta,\ 1 \le j \le l.
\end{array}
$$

Compared to the formulation in section 3.3, a constraint related to power (dynamic power) is added. If the power density of a layer is much higher than others, the heat generated from this layer will flow to the other layers and affect them. To avoid this situation, the constraint of balancing power for each layer is considered. Please note that β here is not a constant (see Table 3). Moreover, the power is consumed not only by logic blocks but also by TSVs. The power dissipation of a logic block is directly proportional to the capacitance of this block, and quadratically proportional to its operating voltage. The power consumed by a TSV w is set as a constant for simplicity. Each used supply voltage is modeled as an I/O terminal (placed in layer 0); the logic blocks operating at the same voltage level are also viewed as being connected by a net–the common supply voltage.

Table 3. Notations used in our extended ILP formulation for MSV design.

Category	Notation	Description
General	L	The number of layers l in a 3D IC. The external world is viewed as layer 0.
	p	Cut p is the cut between layers $p - 1$ and p
	G	Input design is a hypergraph $G = (V, E)$
	v_i	Vertex $v_i \in V$ is a logic block or an I/O terminal. A supply voltage is viewed as an I/O terminal for MSV design
	e_i	Hyperedge $e_i = \{v_{i1}, v_{i2}, \ldots, v_{iq}\} \in E$ is a q-pin net. The set of logic blocks operating at the same voltage U_j forms a hyperedge. $U_j \cup \{v_i : z_{ij} = 1\}$
Layer assignment	$x_{i,j}$	$x_{i,j}$ is a 0-1 integer variable associated with vertex v_i. $x_{i,j} = 1$ if v_i is assigned to layer j; otherwise, $x_{i,j} = 0$. For an I/O terminal v_i, $x_{i,0} = 1$, while for a logic block v_j, $x_{j,0} = 0$.
TSV	$y_{i,p}$	$y_{i,p}$ is a 0-1 integer variable associated with net e_i. $y_{i,p} = 1$ if net e_i introduces a TSV to cut p, between layers p-1 and p, $1 \le p \le l$; otherwise, $y_{i,p} = 0$.
Area	s_i	s_i is the area of vertex v_i. $s_i = 0$ if v_i is an I/O terminal.
	t	t is the area of a single TSV.
	A	$A = \sum_{vi \in V} s_i$, total area of all vertices.
	α	α is the perfectly balanced area for each layer, $\alpha = A/l$.
	ε	ε is the α-bounded ratio of the deviation from the perfectly balanced area and power, $0 < \varepsilon < 1$. The α-bounded area for each layer is subject to the range $[(1 - \varepsilon)\alpha, (1+\varepsilon)\alpha]$, while the α-bounded power for each layer is subject to the range $[(1 - \varepsilon)\beta, (1+\varepsilon)\beta]$.
Voltage assignment	$z_{i,j}$	$z_{i,j}$ is a 0-1 integer variable associated with vertex v_i. $z_{i,j} = 1$ if v_i is assigned to voltage U_j; otherwise, $z_{i,j} = 0$. For an I/O terminal v_i, $z_{i,j} = 0$. For a logic block, a subset of timing safe supply voltages from $U=\{U_j, j = 1..r\}$ is given.
Power	$\sum_{j=1..r} z_{i,j} U_j$	The operating voltage of logic block v_i.
	c_i	The capacitance of logic block v_i.
	w	w is the power of a single TSV.
	P	$P = \sum_{vi \in V} c_i(\sum_{k=1..r} z_{i,k} U_k^2) + w\sum_{ei \in E}\sum_{p=1..l} y_{i,p}$, total power.
	P_j	$P_j = \sum_{vi \in V} c_i x_{i,j}(\sum_{k=1..r} z_{i,k} U_k^2) + w\sum_{ei \in E} y_{i,j}$, total power for layer j. For simplicity, the TSV power for cut j is included at layer j.
	β	β is the perfectly balanced logic power for each layer, $\beta = P/l = (\sum_{vi \in V} c_i(\sum_{k=1..r} z_{i,k} U_k^2) + w\sum_{ei \in E}\sum_{p=1..l} y_{i,p})/l$.

4.3 ILP Reduction for MSV Design

For MSV design, iterative ILP is also used to speed up. By adding the power constraints, the formulation of UNBALANCED-2LAYER-ILP is changed as follows,

$$\begin{array}{ll}
\text{minimize} & \sum_{ei\in E} \sum_{p=1,2} y_{i,p} \\
\text{subject to} & \sum_{j=0..2} x_{i,j} = 1,\ \sum_{j=1..r} z_{i,j} = 1, v_i \in V; \\
& (1-\varepsilon)\alpha \le \sum_{vi\in V} s_i\, x_{i,2} + t\sum_{ei\in E} y_{i,2} \le (1+\varepsilon)\alpha, \\
& (1-\varepsilon)\alpha(l-1) \le \sum_{vi\in V} s_i\, x_{i,1} + t\sum_{ei\in E} y_{i,1} \le (1+\varepsilon)\alpha(l-1); \\
& (1-\varepsilon)\beta \le \sum_{vi\in V} c_i x_{i,2}(\sum_{k=1..r} z_{i,k}U_k^2) + w\sum_{ei\in E} y_{i,2} \le (1+\varepsilon)\beta, \\
& (1-\varepsilon)\beta(l-1) \le \sum_{vi\in V} c_i x_{i,1}(\sum_{k=1..r} z_{i,k}U_k^2) + w\sum_{ei\in E} y_{i,1} \le (1+\varepsilon)\beta(l-1).
\end{array}$$

After termination, the number of TSVs, the layer of each logic block, and the operating voltage of each block assigned are obtained. We also apply pre-clustering to reduce variables in ILP for MSV design. During ILP, the logic blocks in the same cluster are regarded as one vertex. The area of this cluster is the summation of logic blocks belonging to it, while the set of timing safe supply voltages of this cluster is the intersection of those of the logic blocks.

5. EXPERIMENTAL RESULTS

This study has conducted experiments on the GSRC floorplanning benchmarks [11, 16]. This set of testcases can be used for both the architectural and physical levels. The statistics of the testcases are listed in Table 4. In addition, the postfix attached in each testcase name indicates the number of logic blocks, for example, n10 has 10 logic blocks and 69 I/O terminals. The ε value is set based on hMetis [8] and enlarged as the number of logic blocks in a testcase increasing [12]. For MSV design, the set of available supply voltages is {1.0, 1.1, 1.2, 1.3, 1.4, 1.5}; we set timing safe voltages for each block the same as [16] in each testcase. The results are obtained based on the IBM ILOG CPLEX Optimizer [13].

Table 4. Statistics of the testcases.

testcase	#vertices	#hyperegdes	l	ε (%)	
				hMetis [12]	Ours
n10	79	118	2	2.0	2.0
n30	242	349	3	5.5	4.0
n50	259	485	4	9.0	6.0
n100	434	885	5	9.5	15.0
n200	764	1585	6	NA	18.0
n300	869	1893	7	NA	20.0

To show the effectiveness of the pre-clustering technique, greedy pre-clustering is provided as the baseline. For each cluster, greedy pre-clustering iteratively adds the vertex with the least number of pins outside the cluster. When the current cluster is saturated, meaning the vertices inside it reach the pre-specified bound, the next cluster will be chosen. Greedy pre-clustering terminates when the number of vertices becomes manageable.

5.1 Generic ILP vs. Iterative ILP

Table 5 summarizes the results of the generic ILP and iterative ILP. The second and third columns present the number of TSVs and runtime of the generic ILP formulation. The rest of columns list the number of TSVs and runtime of the iterative ILP formulation without pre-clustering, iterative ILP formulation with greedy pre-clustering and iterative ILP formulation with FirstChoice pre-clustering.

Table 5. Results on GSRC benchmark with single voltage.

test-case	Generic ILP		Iterative ILP					
			w/o pre-clustering		w/ greedy pre-clustering		w/ FirstChoice pre-clustering	
	#TSVs	runtime (sec)	#TSVs	runtime (sec)	#TSVs	runtime (sec)	#TSVs	runtime (sec)
n10	113	0.10	132	0.36				
n30	481	19.41	539	2.21				
n50	798***	4000.00	726	45.69				
n100	1807***	10000.00	1333	129.90	1437	68.51	1268	15.80
n200	NA*	NA*	3011	4523.02	3401	1804.39	2857	2019.33
n300	NA*	NA*	3711	8210.66	4320	5862.55	3346	4090.65
Average	1.000**	1.000**	1.037**	0.010**	--	--	--	--

* NA represents the ILP formulation is terminated by CPLEX.
** The average of the first three cases.
*** This is a feasible solution instead of an optimal solution. Timeout is set as the runtime, respectively.

It can be seen that the iterative ILP improves the runtime dramatically. On average, our iterative ILP speeds up generic ILP by 100 times while suffering only 3.7% penalty on TSVs for the first three cases. For the bigger cases, n100, n200 and n300, iterative ILP with two different pre-clustering techniques is also performed. Greedy pre-clustering trades 14% more #TSVs with a 1.67X speedup; FirstChoice pre-clusterung even saves 7% #TSVs with a 2X speedup. It implies that with a sophisticated pre-clustering method, iterative ILP can perform very well. FirstChoice pre-clustering greatly reduces the variables used in ILP, so the ILP can converge faster and probably find better solutions compared with generic ILP. For example, generic ILP just generates a feasible solution for n100, while iterative ILP with FirstChoice pre-clustering obtains a better feasible solution in a short runtime. As a result, it can be concluded that this iterative ILP not only improves runtime of generic ILP but also provides reasonably good solutions.

5.2 ILP with MSVs

Table 6 summarizes the results of the generic ILP and iterative ILP with MSVs. It can be seen that iterative ILP achieves a 7X speedup with 0.3% TSV overhead, which also indicates that a huge benefit in runtime comes along with small TSV overhead. For testcases from n100 to n300, iterative ILP with FirstChoice pre-clustering not only runs 23% faster than generic ILP but also saves 3% #TSVs, while iterative ILP with greedy pre-clustering uses average 9% more TSVs.

Table 6. Results on GSRC benchmark with MSVs.

test-case	Generic ILP		Iterative ILP					
			w/o pre-clustering		w/ greedy pre-clustering		w/ FirstChoice pre-clustering	
	#TSVs	runtime (sec)	#TSVs	runtime (sec)	#TSVs	runtime (sec)	#TSVs	runtime (sec)
n10	124	2.60	135	2.28				
n30	497	2639.00	545	27.54				
n50	833	4000.00	778	902.00				
n100	NA[*]	NA[*]	1344	2399.95	1450	1846.70	1571	1815.59
n200	NA[*]	NA[*]	3231	4999.83	3493	4009.57	3066	4013.70
n300	NA[*]	NA[*]	4051	7799.61	4482	6632.15	3763	6508.97
Average	1.000[**]	1.000[**]	1.003[**]	0.140[**]	--	--	--	--

[*] NA represents the ILP formulation is terminated by CPLEX.
[**] The average of the first three cases.

Table 7. Results of area and power. (Generic ILP vs. Iterative-ILP w/o pre-clustering).

test-case	Single voltage		MSV			
	Area[*]		Area[*]		Power[**]	
	Generic ILP	Iterative ILP	Generic ILP	Iterative ILP	Generic ILP	Iterative ILP
n10	1.01	1.01	1.01	1.01	0.72	0.85
n30	1.03	1.03	1.03	1.03	0.69	0.83
n50	1.04	1.06	1.05	1.05	0.71	0.80
n100	1.10	1.11	--	1.12	--	0.75
n200	--	1.37	--	1.39	--	0.82
n300	--	1.20	--	1.19	--	0.76

[*] Area=footprint_area/*A*/*l*.
[**] Power=*P*/ the power dissipation if all logic blocks are using the highest timing safe voltage as operating voltage.

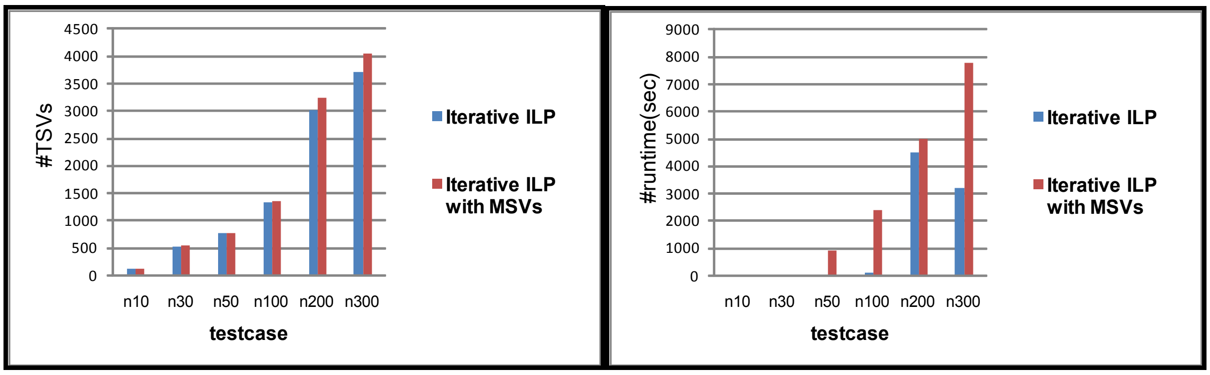


Fig. 6. Comparisons of (a) #TSVs and (b) runtime between iterative ILP (without pre-clustering) with and without MSVs.

Fig. 6 illustrates the comparison between the number of TSVs (see Fig. 6 (a)) and runtime (see Fig. 6 (b)) of iterative ILP (the one without pre-clustering) with and without MSVs, respectively. With MSVs, the number of TSVs used for supplying different voltages must be considered. Thus, #TSVs of iterative ILP with MSVs is naturally more than that without MSVs. Experimental results show that there is nearly 6% more TSVs used in iterative ILP with MSVs. Fig. 6 (b) demonstrates the runtime of these two ILPs. Since the variables of voltage assignment constraints of power balance are deliberated, the complexity of ILP is increased, resulting in slower convergence. The runtime overhead of iterative ILP with MSVs is 25%, compared to iterative ILP. However, considering that the runtime of iterative ILP is much faster than that of generic ILP, this runtime penalty is relatively tolerable.

Table 7 shows the area and power results of generic ILP as well as iterative ILP without pre-clustering. For each testcase, the area is the ratio of footprint area over A/l, while the power represents the ratio of total power dissipation P over the power dissipation if all logic blocks are using the highest timing safe voltage as operating voltage. With MSV design, the power saving is 20% on average. Although iterative ILP does not balance area very well, the results are still acceptable. The footprint area overhead of n200 is not within the ε ratio is because that the iterative ILP may lose global information during iterations.

6. CONCLUSION

In this paper, we derive integer linear programs for 3D IC partitioning problems that minimize one of the three objectives: the overall TSV usage, the footprint, and the power. Extending the generic ILP formulation to support designs with MSVs demonstrates that the proposed approach is very flexible and can easily extend to the partitioning problem with variant objectives and constraints. Last but not least, this study speeds up the ILP formulation by iteratively solving it and pre-clustering. Experimental results show that the speed-up approach reduces runtime substantially while maintains a reasonably good solution quality.

ACKNOWLEDGEMENT

The authors would like to thank the reviewers for their insightful and valuable comments.

REFERENCES

1. W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer, and P. D. Franzon, "Demystifying 3D ICs: The pros and cons of going vertical," *IEEE Design and Test Magazine*, 2005, pp. 498-510.
2. W.-L. Hung, G. Link, Y. Xie, N. Vijaykrishnan, and M. J. Irwin, "Interconnect and thermal-aware floorplanning for 3D microprocessors," in *Proceedings of International Symposium on Quality Electronic Design*, 2006, pp. 98-104.
3. C. Ferri, S. Reda, and R. I. Bahar, "Parametric yield management for 3D ICs: Models and strategies for improvement," *ACM Journal on Emerging Technologies in Computing Systems*, Vol. 4, 2008, pp. 19:1-19:22.
4. X. Dong and Y. Xie, "System-level cost analysis and design exploration for three-

dimensional integrated circuits," in *Proceedings of ACM Asia and South Pacific Design Automation Conference*, 2009, pp. 234-241.

5. The International Technology Roadmap for Semiconductors (ITRS), 2009, http://www.itrs.net/.
6. Z. Yi, X. Hong, Q. Zhou, Y. Cai, J. Bian, H. Yang, P. Saxena, and V. Pitchumani, "A divide-and-conquer 2.5-D floorplanning algorithm based on statistical wirelength estimation," in *Proceedings of IEEE International Symposium on Circuits and Systems*, 2005, pp. 6230-6233.
7. T. Yan, Q. Dong, Y. Takashima, and Y. Kajitani, "How does partitioning matter for 3D floorplanning?" in *Proceedings of ACM Great Lakes Symposium on VLSI*, 2006, pp. 73-78.
8. G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Application in VLSI domain," *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 7, 1999, pp. 69-79.
9. C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Proceedings of ACM/IEEE Design Automation Conference*, 1982, pp. 175-181.
10. L. Schrage, *Optimization Modeling with LINGO*, 6th ed., Lindo Systems Inc., 2008.
11. GSRC benchmark.
12. I. H.-R. Jiang, "Generic integer linear programming formulation for 3D IC partitioning," in *Proceedings of IEEE International SoC Conference*, 2009, pp. 321-324.
13. IBM ILOG CPLEX Optimizer, http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.
14. G. Karypis and V. Kumar, "Multilevel K-way hypergraph partitioning," Technical Report No. TR 98-036, Department of Computer Science, University of Minnesota, 1998.
15. H.-S. Ye, M. C. Chi, and S.-H. Huang, "A design partitioning algorithm for three dimensional integrated circuits," in *Proceedings of IEEE International Symposium on Computer Communication Control and Automation*, 2010, pp. 229-232.
16. Q. Ma and E. F. Y. Young, "Voltage island-driven floorplanning," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 2007, pp. 644-649.

Wan-Yu Lee (李婉毓) received the B.S. degree in Electrical Engineering from National Chung Hsing University, Taichung, Taiwan, in 2005 and the M.S. degree in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2007. She is currently working toward the Ph.D. degree at the Institute of Electronics, National Chiao Tung University. Her research interests include power management and EDA of 3D IC designs.

Iris Hui-Ru Jiang (江蕙如) received the B.S. and Ph.D. degrees in electronics engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 1995 and 2002, respectively. She has been with VIA Technologies, Inc. from 2002 to 2005. She is currently an Associate Professor with the Department of Electronics Engineering and the Institute of Electronics, NCTU. Her current research interests include VLSI physical design and interaction between logic synthesis and physical design. Dr. Jiang is a member of IEEE, ACM and Phi Tau Phi.

Tsung-Wan Mei (梅宗菀) received the B.S. degree in computer science in 2008 and the M.S. degree in electronics engineering in 2011, both from National Chiao Tung University, Hsinchu, Taiwan. She is now with Silicon Integrated Systems Corp., Hsinchu, Taiwan.