Volume 47, Issue 2 March 2014 ISSN 0167-9260

ELSEVIER

INTEGRATION
the VLSI journal

CONTENTS

Research Articles

Cell-based interconnect migration by hierarchical optimization
E. Shaphir, R.Y. Pinter and S. Wimer 161

Fast and scalable parallel layout decomposition in double patterning lithography
W. Zhao, H. Yao, Y. Cai, S. Sinha and C. Chiang 175

Algorithms for TSV resource sharing and optimization in designing 3D stacked ICs
B. Lee and T. Kim 184

Nano-CMOS thermal sensor design optimization for efficient temperature measurement
O. Okobiah, S.P. Mohanty and E. Kougianos 195

High slew rate current mode transconductance error amplifier for low quiescent current output-capacitorless CMOS LDO regulator
R. Fathipour, A. Saberkari, H. Martinez and E. Alarcón 204

Analog circuit design space description based on ordered clustering of feature uniqueness and similarity
C. Ferent and A. Doboli 213

Design of a coarse-grained reconfigurable architecture with floating-point support and comparative study
M. Jo, D. Lee, K. Han and K. Choi 232

Simultaneous switching noise reduction by resonant clock distribution networks
B. Mesgarzadeh 242

[Continued on outside back cover

Available online at www.sciencedirect.com

ScienceDirect

INTEGRATION, the VLSI journal 47 (2014) 161–174

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Cell-based interconnect migration by hierarchical optimization

Eugene Shaphir [a], Ron Y. Pinter [b], Shmuel Wimer [c,d,*]

[a] Intel Corporation, Santa Clara, CA, USA
[b] Technion, CS Faculty, Haifa, Israel
[c] Bar-Ilan University, Engineering Faculty, Ramat-Gan, Israel
[d] Technion, EE Faculty, Haifa, Israel

ARTICLE INFO

Article history:
Received 14 May 2013
Received in revised form
18 October 2013
Accepted 22 October 2013
Available online 1 November 2013

Keywords:
VLSI design migration
Layout compaction
Interconnects
Design hierarchy
Cell-based design

ABSTRACT

Fueled by Moore's Law, VLSI market competition and economic considerations dictates the introduction of new processor's microarchitecture in a two-year cycle called "Tick-Tock" marketing strategy. A new processor is first manufactured in the most advanced stable process technology, followed in a one-year delay by introducing chips comprising same microarchitecture but manufactured in a newer scaled process technology, thus allowing higher production volumes, better performance and lower cost. Tick-Tock is enabled by the automation of chip's layout conversion from an older into a newer manufacturing process technology. This is a very challenging computational task, involving billions of polygons. We describe an algorithm of a hierarchy-driven optimization method for cell-based layout conversion used at Intel for already several product generations. It transforms the full conversion problem into successive problems of significantly smaller size, having feasible solutions if and only if the full-chip problem does. The proposed algorithm preserves the design intent, its uniformity and maintainability, a key for the success of large-scale projects.

1. Introduction

The design of high-end full-custom microprocessors such as those of Intel, AMD and IBM, is a very complex engineering task, involving hundreds of man-years efforts. Hierarchical design methodology is a key in achieving product specifications and time-to-market requirements, which otherwise could not be met. Fueled by Moore's Law [1], market competition and economic considerations dictate the introduction of new processor's micro-architecture in a two-year cycle, in the so-called "Tick-Tock" strategy [2] as illustrated in Fig. 1.

The Tick-Tock development strategy delivers every two year a new microarchitecture manufactured in the most advanced stable technology. This is called "Tock". It is then followed in about one-year delay by a "Tick" phase, delivering chips of the same microarchitecture as the recent Tock but in a new scaled manufacturing process technology, thus allowing higher production volumes, better performance and lower cost. An essential part of the Tick phase is the conversion of the underlying physical layout, comprising billions of polygons, into the new technology. Such conversion is known in VLSI jargon as *hard-IP reuse* [3]. An enabler for meeting this Tick-Tock interlacing is therefore the automation of chip's layout conversion from older into newer technology. Such automation is a very challenging computational task, involving billions of polygons that must satisfy complex geometric rules.

The polygons conversion is carried out by layout *compaction* algorithms. Those have been developed since the early days of VLSI electronic design automation (EDA) and a comprehensive description of various algorithms can be found in [4,5]. The compaction describes the positional relations of the polygons of the *source* layout aimed at conversion, by a directed graph, called the *constraints graph*. Its vertices represent edges of polygons and arcs represent left-to-right (bottom-to-top) adjacency and visibility relations. The arcs are assigned weights corresponding to the minimal sizes and spacing *design rules* of the new technology. The problem of sizing and positioning of the polygons in the new *target* layout is to find the smallest possible area into which the layout can legally fit.

The most general form of compaction involves moving the polygons of the layout in the x- and y-coordinates simultaneously, called two-dimensional (2D) compaction that was shown to be NP-complete [13,14]. Compactors therefore decompose the 2D problem into an alternating sequence of independent one-dimensional (1D) compaction steps, each changes only one set of coordinates. 1D compaction can be solved efficiently with longest path algorithms [4,5]. Polygons not on the critical paths are positioned such that some cost reflecting a design goal (e.g., performance, sensitivity for manufacturing defects, among a few others) is minimized. A heuristic solution of the 2D problem was

* Corresponding author at: Bar-Ilan University, Engineering Faculty, Ramat-Gan, Israel. Tel.: +972 3 5317208.
E-mail address: wimes@biu.ac.il (S. Wimer).

0167-9260/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.vlsi.2013.10.003

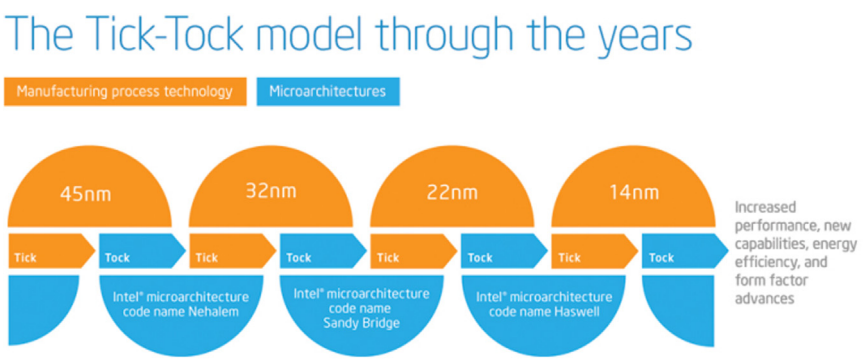

Fig. 1. "Tick-Tock" marketing strategy.

proposed in [15] where the problem was first solved by alternating 1D compactions. The layout was then relaxed by introducing extra jogs into the wires to enable further compression of the layout. In today's technologies which require high regularity and uniformity of the interconnecting wires, jog insertions are prohibited since that result in performance degradation and manufacturing yield loss. The work in [16] also showed the NP completeness of the 2D problem and proposed a branch-and-bound search algorithm. It is suitable for very small layouts, but impractical for the large-scale problems arising in full-chip compaction. The 2D problem was studied also in the context of *graph-drawing* [17,18], leading to similar consequences on its difficulty. To the best knowledge of the authors, the EDA industry-scale compactor in [10] is 1D. It allows the user to control the compaction iterations by specifying an appropriate parameter to be either *x-y-x* or *y-x-y*. Another EDA compaction tool described in [5] that was used for the layout migration of a full-scale microprocessor is also 1D.

Traditional compaction algorithms are flat, suited for relatively small layouts comprising up to a few tens of thousands of polygons. With the advancement of VLSI technology in the 90s to integration of few million transistors on a chip, design methodology moved towards more standardization, modularity and re-use, making chip structure hierarchical. In parallel, design rules became more complex, which altogether made layout migration a computation challenge (henceforth we use the terms compaction and migration interchangeably).

Algorithms and EDA vendor tools supporting hierarchy were proposed in [6–10]. Compaction creates unique blocks (also called *modules* or *cells*), which cannot be shared and re-used among different layouts. Therefore, although those compactors maintain layout hierarchy, the duplication and layout mutation same logic blocks is a major disadvantage that today's designs cannot afford. Thus, a new layout migration technology called *cell-based compaction* is in order. It uses a common, manually designed, standard-cell library, which is optimized regardless of its instances in the entire layout. Cell-based migration has the problem of creating a huge compaction constraints graph incorporating all the instances of all blocks, which is then translated into a huge optimization problem whose solution may take days or even weeks of computation time. This paper reduces the size of the compaction in one to two orders of magnitude.

The work of [6] was probably the first to address layout hierarchy. It ensured that the modularity of the target layout will stay similar to that of the source layout. It could handle efficiently small layout in the scale of tens to hundreds transistors. It did not take advantage of the repetitive instantiation of the same cells to reduce computation complexity, which our work does. This is a key to efficiently migrate layouts at chip scale.

The work in [7] handled larger blocks comprising hundreds to thousands transistors and was proven on real IBM design. Its main drawback is being tailored to control-logic, comprising two-levels of hierarchies: leaf-cells and the entire block. Moreover, same leaf-cells in different blocks were in-place compacted, resulting in various layout mutations of the same logic cell. This prohibits cell-level electrical characterization, a key for efficient timing analysis. Rather, timing analysis must take place at transistor-level, a big design effort overhead. Our work in contrast supports any hierarchy depth, making it useful for custom data-path and register-file design styles, comprising many levels of layout hierarchies. Moreover, our migration flow is cell-based, enabling the usage of standard-cell library with all the advantages of modular design and efficient timing analysis.

The authors of [8] claimed for cell-based layout compaction. Their work emphasized the pitch-matching of cells and heavily relied on the slicing structure of the layout. This effectively makes the algorithm useful for two-level place and route layout style as in [8], but inadequate for other layout styles mentioned above. As all the other hierarchical compactors, the cells are in-place compacted, prohibiting the advantages of real cell-based design.

The work in [9] took advantage of the special linear programming matrix form occurring in solving the layout constraints. It supports hierarchy, but as other works, the leaf-cells are compacted in-place, a drawback mentioned above. It was also proven on problems comprising only few thousands of variables and constraints, which is impractical for chip-scale problems.

The above works evolved later into large-scale hierarchical compaction tools availed by an EDA vendor [10], used successfully by the industry. Intel used such tools for several process generations [3,19], from 130 nm, through 90 and 65, to 45 and 32 nm, in the Tick-Tock cadence shown in Fig. 1. Unfortunately, the tool in [10] still in-place compacts leaf-cells, thus prohibiting the advantages of a real cell-based design. Moreover, while all past works worked on the entire flattened layout, thus solving huge problems, our algorithm is successively solving a series of far smaller problems, but still exploring the entire solution space.

Interconnect migration addressed in this paper nicely fits the 1D paradigm as illustrated in Fig. 2. Due to the uniform longitudinal and latitudinal nature of wires, which are the main subject of the compaction, there is not much optimality loss compared to 2D. The transformation applied to wires, whose target widths are determined prior to compaction. It is an *x*-shift of vertical layers and a *y*-shift of horizontal layers. Shifted wires are hooked by vias at their ends to perpendicular wires residing in an adjacent layer below and above. It is therefore straightforward to maintain connectivity after 1D iteration by stretching the perpendicular wires to the new coordinate of their ends.

The layout design rules imposed by modern VLSI process technologies become more and more complex and their number may reach a few hundreds. Fortunately, the majority of the increase occurs in the lower layers involving transistors and their interconnections used within logic cells, whose layouts are migrated manually.

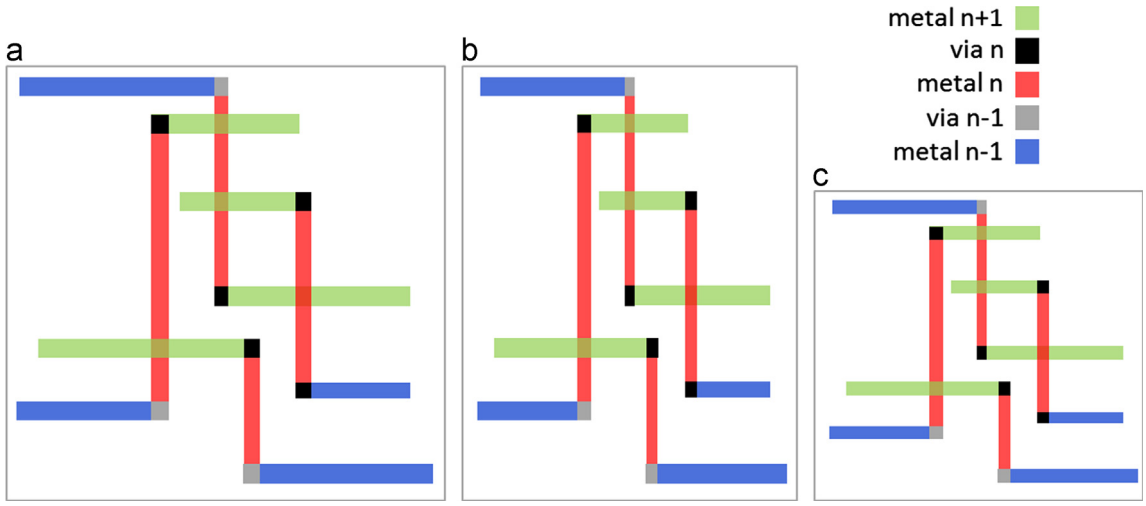

Fig. 2. A 2D compaction by successive 1D compactions. (a) original layout, (b) after *x* compaction, (c) after *y* compaction. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

The migration problem discussed in this paper does not touch those layers, but handles the higher metal layers used to interconnect logic cells. For the latter, the primary design rules are minimal width, minimal spacing, and the metal coverage of a via. While the target width of wires is set by performance considerations prior to applying compaction, their spacing is solved by the compaction algorithm. Vias are formed at the incidence of orthogonal wires as shown in Fig. 2. The orthogonal wire stretching which maintains signal connectivity, guarantees that vias can safely be landed. It may happen that the resulting layout still has some design rule violations. Those are manually fixed at a later stage of the design.

The main contribution of this paper is in presenting an algorithmic paradigm for cell-based compaction of complete VLSI chips, which has been proven in real industrial projects. Although compaction algorithms have been thoroughly studied and implemented in the past, their computational capabilities fall short compared to the demand of modern chips and design projects. The novelty of this work is in exploiting the hierarchical structure of VLSI chips to achieve orders of magnitude improvement in computation time. The correctness of this approach is rigorously proven. We present time and space efficient bottom-up and top-down migration algorithms with emphasis on wires (also called interconnections). This is the most difficult part of cell-based compaction. The rest of the paper is organized as follows. Section 2 describes the general framework of the migration algorithm. Section 3 presents the interconnect migration problem and outlines its solution by using a suitable graph-theoretic model. Section 4 provides the mathematical foundations of the compaction algorithm. Section 5 presents experimental results obtained for Intel's VLSI blocks designed in 45 nm technology. We conclude with some challenges for further research.

2. The placement-routing handshake

Due to their high complexity, VLSI chips are built hierarchically. The entire chip corresponds to the top of the hierarchy, while at the bottom there are the individual transistors. Transistors are then connected with each other in *standard cells*, implementing basic logic and memory functions. Those are connected together in more complex functional blocks such as adders, multipliers, and memory arrays, among others. Functional blocks are connected by wires in higher level functions such as Arithmetic-Logic Units (ALUs), control units, etc. The top of this hierarchy are DSP, networking, communication and sensor modules, where their connection constitutes the so called *System on Chip* (SoC), occupying the entire silicon die. Contemporary VLSI technologies may comprise a dozen metal layers carrying a huge number of interconnecting wires. Wires residing on even layers all have the same direction (e.g., vertical) while those residing on odd layers are orthogonal. Mixing both directions on the same layers is forbidden.

Fig. 3(a) illustrates a typical VLSI layout comprising several blocks placed within each other, thus constituting the hierarchy. Each block has IO ports through which it is connected by wires to other blocks (we use the terms wires and interconnections interchangeably). The wires connecting to each other child blocks placed within a parent, and to the IO ports of their parent, belong to the parent. The different colors of blocks' borders represent levels of the hierarchy as illustrated in Fig. 3(b). Notice that a block may be placed multiple times within different parents. The unique definition of a block is called a *master*, and its specific occurrence in the design is called an *instance* (we use the terms master and block interchangeably). The origin of a master is associated to its lower-left corner. Instances are placed within their parent at some *x* and *y offsets*. Instances place in a common parent cannot overlap.

The wires illustrated in Fig. 3(a) reside on two adjacent metal layers where wires are connected at their incidence point by a VIA. The solid wires interconnect black-border children within their green-border parent. The dotted wires are connecting orange-border children within their purple-border parent. Wires connecting orange-border blocks are not shown. Notice that both solid and dotted wires of the same color belong to the same physical layer and are therefore not allowed to touch each other as otherwise an electrical short occurs. Wires residing in the same layer must satisfy minimum width and spacing rules, as otherwise a malfunction at manufacturing may occur. Extra widths and spacing may be specified per wire to satisfy design goals as performance, noise immunity, small *IR* drop and reliability.

The progression from old to new technology is featuring a 0.7 average scale of all lateral dimensions, thus enabling to double the numbers of transistors per silicon area. This is the well-known Moore's Law, governing the VLSI evolution for already five decades [1]. Until late-90s, the 0.7 scaling equally applied to all lateral dimensions of the physical shapes over all layers. VLSI layouts could therefore be converted to newer technology by a purely linear transformation called *"optical shrink"*. The linear scaling has been broken in the last

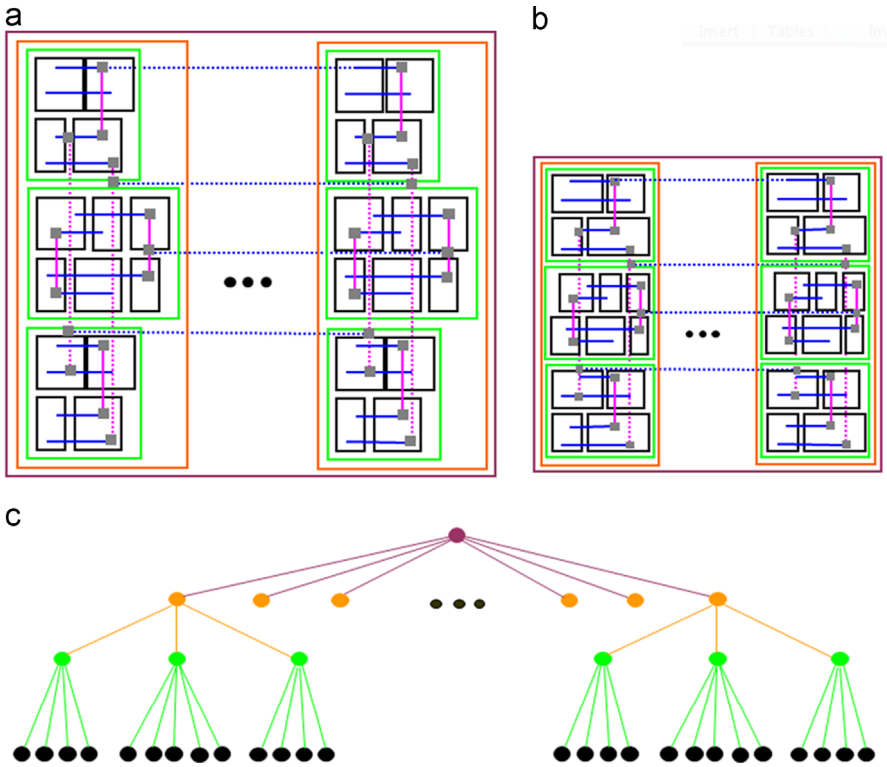

Fig. 3. A four-level hierarchical layout in old technology is shown in (a), its corresponding hierarchy tree in (b) and the target layout in new technology in (c). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

fifteen years due to many technology difficulties, which are not elaborated here. Layout conversion has therefore turned into a strongly non-linear problem. Performance requirements of specific electrical signals (called nets) may only worsen the nonlinearity by introducing extra geometric constraints.

3. Algorithm framework for migration

Due to its complexity, the generation of the physical layout of VLSI systems is traditionally performed in two steps: placement and routing [4]. Our migration algorithm follows the same pace: the placement of the instances of all the masters are migrated first, and then migration of all their underlying interconnections takes place. This paper is about the interconnections migration, but for the sake of completeness the placement migration is briefly described.

The target of the placement in Fig. 3(a) is shown in Fig. 3(c). Placement migration is aimed at three goals:

1. Producing a similar scaled layout, preserving all the left-to-right and bottom-to-top adjacency relations between its block instances.
2. Obtaining a small, compact layout, reflecting the 0.7 average Moore's Law scaling.
3. Target placement must accommodate the target width and spacing of the underlying wires whose migration will follow in a later step.

Placement migration takes place by reversely topologically ordering the masters according to the hierarchy tree. Masters whose instances reside at tree's leaves are migrated first, while the whole layout, corresponding to the top master, is migrated last. It follows by definition that once a master is addressed, all its descendants in the hierarchy tree have already been migrated, and their width and height are known. Setting the target dimensions of the currently migrated master is done by placing its child instances in relatively the same positions to each other as illustrated in Fig. 3(c). Note that the instances' masters are already migrated, thus having target dimensions defined.

To determine the target size of a block such that it will legally accommodate the migrated wires, its child blocks and its wire density in the source layout at each layer are taken into account. Wire densities are weighted by the scaling factor of their layer. The scale factor of blocks having high wire density in layers of poor scaling factor (larger than 0.7) may target 0.8 and higher. Blocks with high wire density in layers of good scaling factor (smaller than 0.7) may target 0.6 and lower. The locations of the child blocks within their parents are determined similarly by considering the wires residing between the blocks in the source layout. The resulting scaling factors of masters' dimensions may vary from 0.5 to 1.0, depending on their contents. This is shown by comparing Fig. 3(a) and (c). While some of the masters scaled down better than 0.7, a few others got worse than 0.7. We shall not further discuss the placement stage, which is beyond the scope of this paper, but rather address the delicate issues arising by wires migration discussed in the next sections.

As shown in Fig. 3(a), wires of the same metal layer are distributed among different levels of the hierarchy. It therefore happens that physically adjacent wires are “blind to each other” at some step of the bottom-up migration algorithm. Since the setting of masters' target sizes takes place in reversed topological order, at the time the size of the child master is being set, it is not aware of other overlapping wires belonging to its ancestors. More severely, instances of the same master may interact differently with wires of ancestors. Such blindness is resolved by the migration algorithm discussed below, provided that the dimensions of the master are

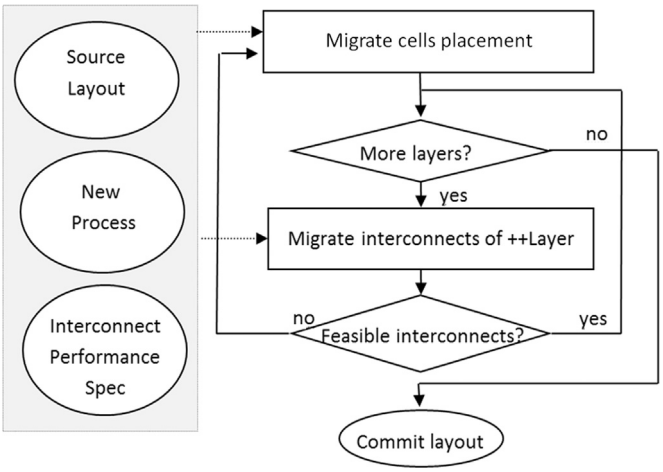

Fig. 4. Complete layout migration flow.

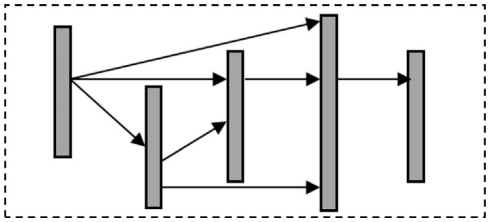

Fig. 5. Wires positioned within a block and their visibility graph.

set sufficiently large to legally accommodate all the overlaying wires incurring across all its instances in the layout.

Fig. 4 shows the entire layout migration flow and its placement-routing handshake. It mixes optimization algorithms as those described in the paper, with manual artwork made by layout design experts. The placements of masters are migrated first as described above. Then routing migration is invoked, where the wire contents of the physical metal layers are migrated layer by layer, taking into consideration delay, power and other performance constraints. For instance, the wires of a delay critical net are enforced to a certain width and spacing to reduce their resistance and capacitance. For noise sensitive nets, spaces to adjacent wires are enforced, to reduce interference. Given a layer, the wires of each master are migrated in a reversed topological order. If it happens that the dimensions of a master could not accommodate the constraints imposed by its internal wires, non-feasibility is alerted. To mitigate few options exist. The dimensions of the problematic master and the corresponding ancestors are relaxed, the width and spacing constraints of the problematic wires are modified, or the violations (e.g. wire shorts) are manually fixed.

3.1. Modeling interconnects by a visibility graph

The optimization problem of interconnect migration is handled layer by layer. Two vertical wires are said to be visible if a horizontal line can be used to connect them without being intersected by any other wire. The adjacency relations between the wires are described by a directed graph $G=(V,E)$. Its vertices represent wires, and an arc connects two vertices if the corresponding wires are visible to each other, directed from the wire with the smaller abscissa to the wire with the larger abscissa. Fig. 5 illustrates several wires and their visibility graph. Once the widths of the wires and their spacing are specified, their abscissa can be computed by the longest path in G [4]. A longest path algorithm is used later by the compaction algorithm to detect positive cycles resulting from non-feasibility.

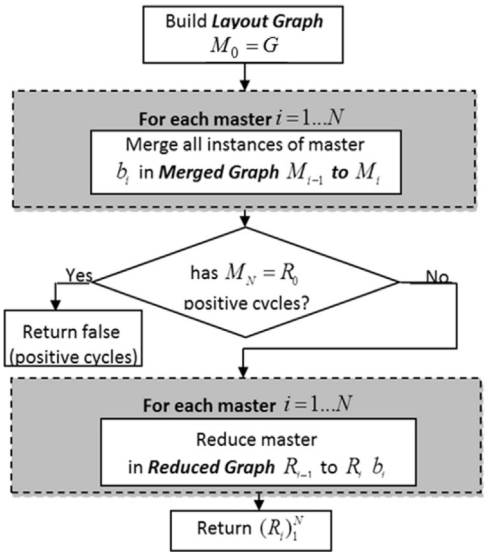

Fig. 6. Finding of the feasibility ranges of wires' abscissae.

3.2. Description of the algorithm

The input of the compaction algorithm consists of:

1. The layout hierarchy and relative block positions as defined by the source (old) layout.
2. The visibility graph G defined by the wires in the source layout and their width and minimum spacing specifications in the target (new) layout.
3. The sizes of the blocks in the target layout as determined by the placement.

The compaction algorithm is aimed at setting the abscissae of the wires within their blocks in the target layout to satisfy the following constraints.

1. The layout of a master must be identical across all its instances in the target layout, hence uniquely defined.
2. The left-to-right order of the instances is preserved across the entire layout hierarchy.
3. The widths of the blocks in the target layout must accommodate their descendant blocks and wires.
4. The visibility graphs are preserved.
5. Satisfy the spacing requirements in the target layout.

Fig. 6 shows the steps of the compaction algorithm.

1. Construction of the visibility graph of the entire layout. Multiple instances of the same master within a parent block are registered.
2. *Contracting* the visibility graph by *merging* the multiple instances of a wire into a single object. *If the merged graph M is free of positive cycles then a feasible compaction solution exists.*
3. Defining a series of concise (called *reduced*) graphs $(R_i)_1^N$ which capture all the essential information of the fully expanded graph. The reduced graphs are then solved successively. A solution is a setting of nonnegative weights to the graph's arcs such that the above constraints are satisfied.
4. The solutions obtained in step 3 satisfy the constraints within each block, thus representing a family of feasible solutions. Those are floated to their parents and are used to find the family of feasible solutions for positioning the wires of the parent master.

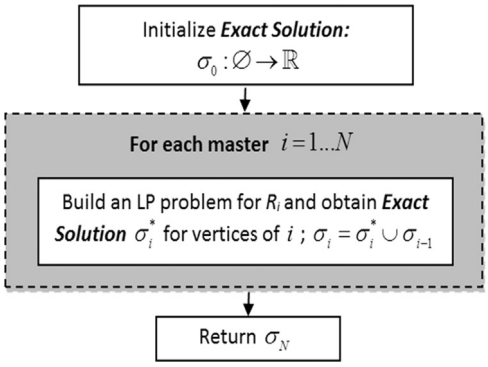

Fig. 7. Committing for wires' abscissae.

A feasible solution of wires' abscissae within a master is a set of closed intervals obtained by the above flow shown in Fig. 6. Each wire can be positioned anywhere in its corresponding interval and the resulting layout is guaranteed to be legal. The final abscissae of the wires are determined by solving a linear program (LP). Its constraints are the above intervals. Applying this process progressively to each master from top to bottom (in reversed topological order) as shown in Fig. 7, yields the entire layout.

The LP's objective function is defined as a weighted sum of wire spacing in an attempt to maintain the delays of the critical nets sufficiently small. It is well known that the coupling capacitance between adjacent wires is a predominant delay, power and noise factor, and it is inversely proportional to the space between wires [11]. Each net is assigned with a positive coefficient, reflecting its relative delay criticality. The LP's objective function can also minimize the power consumption by assigning to each wire a coefficient reflecting its switching activity. The weighted sum of spaces is then proportional to the power consumed by interconnect switching. Nets sensitive to noise can be similarly treated. Combinations of the above design considerations were discussed in [11].

3.3. Construction of the flat layout visibility graph

Let V be the wires and borders of a layout L. For $v \in V$, $I(v)$ denotes the instance to which v belongs, $m(v)$ denotes the corresponding object (wire or border) of the master, $T(I)$ is the instance's master, and $D(T(I))$ is its width in target technology. The parent master of an instance I is $P(I)$.

A *Flat Layout Visibility Graph* (FLVG) $G = (V, E)$ is defined and satisfies the properties below.

1. The vertices V are all the wire and border objects in L. We subsequently use the term vertex and object interchangeably.
2. A pair of wires $u, v \in V$ visible to each other implies an arc $(u, v) \in E$ with a weight $W(u, v)$ defined by the sum of three terms: half target width of u, the minimum wire-to-wire spacing of the new technology and half target width of v. $W(u, v)$ can be increased if due to design considerations.
3. Each pair of vertices $u, v \in V$, where u is a left border of I and v is a wire of I, implies an arc $(u, v) \in E$ with a weight $W(u, v)$ defined by the sum of half wire's target width and half of the minimum wire-to-wire spacing of the new technology. This ensures that minimum spacing design rule is satisfied for adjacent wires belonging to two abutting instances.
4. Similar to 3 but u is a wire and v is a right border.
5. Right and left borders u and v, respectively, visible to each other, satisfying $P(I(u)) = P(I(v))$ and $I(u) \neq I(v)$, impose an arc $(u, v) \in E$ with a weight $W(u, v) = 0$. The role of which is to preserve the left-to-right instances order avoid their overlap. The role of the next property is to enforce the size of an instance to the size of its master.
6. Left border u and right border v, satisfying $I(u) = I(v)$, imply two oppositely directed arcs $(u, v) \in E$ and $(v, u) \in E$ with weights $W(u, v) = D(T(I(u)))$ and $W(v, u) = -D(T(I(u)))$. The role of the next two properties is to enforce an instance to entirely reside within the area of its parent.
7. Two left borders u and v satisfying $T(I(u)) = P(I(v))$ imply an arc $(u, v) \in E$ with a weight $W(u, v) = 0$.
8. Two right borders u and v satisfying $P(I(u)) = T(I(v))$ imply an arc $(u, v) \in E$ with a weight $W(u, v) = 0$.

Figs. 8 and 9 describe pseudo codes to construct a FLVG.

Fig. 10(a) illustrates a parent master A comprising two child instances of the same master B. The wires belonging to master A are colored in green while those of master B are colored in blue. From A's perspective the left-to-right order of all the wires (both blue and green) must be preserved. Also, the green wires cannot extend beyond A's border, whose target size has already been determined at placement phase. Similar requirements hold for the blue wires in. Furthermore, relative position of the blue wires in B must be identical in both instances, since B is a unique master. Fig. 10(b) is the corresponding FLVG. The vertical borders are

```
1. FlatGraph G=(V,E)
2. V ← {layout wires} ∪ { vertical borders of masters }
3. E ← ∅
4. for each u∈V
5.    for each v∈V visible to u on the right
6.       if u,v are wires OR
              u is a border v is a wire of same instance OR
              v is a border u is a wire of same instance
7.                E ← E∪{(u,v)}
8.                W(u,v) ← spacing requirement
```

Fig. 8. Introduction of wire-to-wire and wire-to-border arcs into FLVG.

```
1. for each instance in layout
   Let u,v∈V correspond to the instances left, right borders, resp.
2.    E ← E∪{(u,v)}∪{(v,u)}
3.    W(u,v) ← target width of the instance
4.    W(v,u) ← -W(u,v)
```

Fig. 9. Introduction of border-to-border arcs into FLVG.

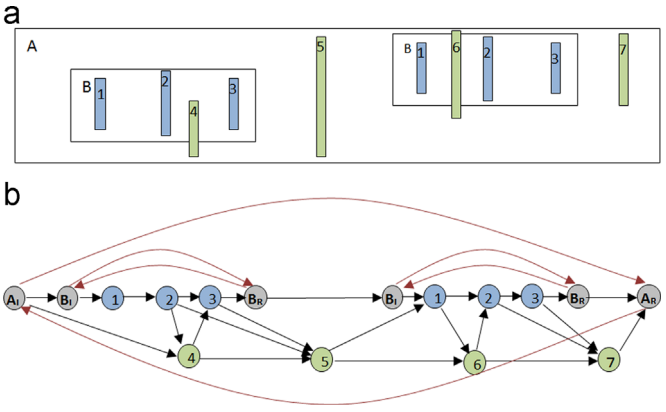

Fig. 10. Two-level hierarchical layout (a) and its corresponding FLVG in (b). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

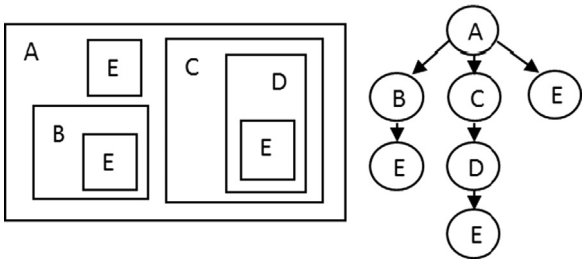

Fig. 11. Hierarchical layout and its corresponding tree.

```
proc mergeMasterInstances(graph G , master t)
   Let G = (V, E)
   Let G_t = (V_t, E_t) represent flat graph of master t
1.  V ← V ∪ V_t
2.  for each instance i of master t in the layout
       Let x be the offset of instance i
3.     for each v_t ∈ V_t
          Let v ∈ V represent v_t in instance i
4.        for each u ∈ V so that (u, v) ∈ E
5.              E ← E ∪ {(u, v_t)}
6.              W(u, v_t) ← W(u, v) − x
7.              E ← E \ {(u, v)}
8.        for each u ∈ V so that (v, u) ∈ E
9.              E ← E ∪ {(v_t, u)}
10.             W(v_t, u) ← W(v, u) + x
11.             E ← E \ {(v, u)}
12.       V = V \ {v}
```

Fig. 12. Pseudo code for vertex merging.

represented by gray vertices, designated by a master name with index L or R for the left and right borders, respectively. The oppositely directed parallel red arcs enforce the master size in target layout.

3.4. Merging block instances

The second step of the compaction algorithm transforms the FLVG G into a simpler graph M obtained by merging vertices corresponding to the same wire in its various instances across the entire hierarchy. This step also adds arcs taking care of the similarity constraints to ensure uniqueness of the master in the target layout. Assuming that the layout comprises N masters, M is successively constructed in N steps. We say that two masters A and B satisfy the partial order relation $A \prec B$, called *hierarchical order*, if A is a descendant of B. The hierarchical layout is then partially ordered and the order can be represented by tree. Fig. 11 illustrates a hierarchical layout and its corresponding tree.

3.5. Graph merging

The construction of M starts from the hierarchy tree leaves. Th e masters $b_i, 1 \le i \le N$, are indexed by their partial order $\prec$. A corresponding sequence of merged graphs $M_i, 1 \le i \le N$, is implied, where $M_0 \triangleq G$. M_i is generated from M_{i-1} by merging all the vertices corresponding to the instances of the same wire or

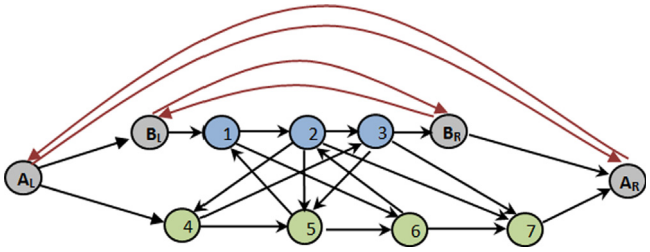

Fig. 13. The contracted graph resulting after merging the graph in Fig. 10. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

border of b_i into a single vertex to ensures uniqueness of the target master. The weights of the incoming and outgoing arcs of that vertex are updated according to the offset of the instance to which the vertex belongs. This merging significantly reduces the number of vertices while capturing the complete spacing and size constraints.

Fig. 12 presents the merging pseudo code. The procedure is working iteratively in a reversed topological order (bottom-up) of the masters (see Fig. 6). Let $G_t = (V_t, E_t)$ denote the flattened graph of a master t. All the vertices $v \in V$ induced by t's instances across the entire layout are replaced by a vertex $v_t \in V_t$ of the master t. Each incoming arc (u, v) (outgoing arc (v, u)) is replaced by an arc $(u, v_t)((v_t, u))$ and its weight is modified by subtracting (adding) the offset of the instance to which v belongs. We denote by M the graph resulting by the iterative merging transformations. It is subsequently proven that G is free of positive cycles, namely, the compaction has a feasible solution, if and only if M does. Note that the number of vertices in M is significantly reduced compared to G by a factor related to the average number of master's instances. Fig. 13 illustrates the resulting graph after the merging is applied to the graph in Fig. 10. The time complexity of the merge phase is $O(|V|+|E)$, as each vertex and arc in the graph is treated exactly once.

3.6. Graph reduction

So far the number of vertices has been cut. In the following we reduce the number of arcs. Similar to merging, it is done by employing successive transformations to M in reversed topological order (bottom-up) of the masters. Iteration eliminates the vertices corresponding to the current master. Let the masters $b_i, 1 \le i \le N$, be topologically ordered by $\prec$. A corresponding sequence of reduced graphs $R_i, 1 \le i \le N$, is produced, where $R_0 \triangleq M$. The graph R_i is obtained from R_{i-1}. Let v be a vertex corresponding to a wire or the vertical border of b_i. An incoming arc (u, v) and an outgoing arc (v, w) are replaced by an arc (u, w) satisfying $W(u, w) = W(u, v) + W(v, w)$, thus eliminating the vertex v. If an arc $(u, w)'$ already exists in R_i the weight of the new arc (u, w) is set to $W(u, w) = \max\{W(u, v) + W(v, w), W(u, v)'\}$. A pseudo code of the graph reduction is shown in Fig. 14. Fig. 15 illustrates the reduced graph obtained from the merged graph in Fig. 13. It follows from the planarity of the FVLG that the worst-case time complexity of the reduction phase is $O(|V|)$.

3.7. Deriving an exact solution

The compaction problem has a feasible solution if its corresponding constraints graph has no positive cycle. The positive cycle is reported to the placement program and a relaxation of master's children placement takes place (see Fig. 4). It was also claimed that the FLVG G has no positive cycle if and only if the corresponding merged graph M has no positive cycle (proven in Section 4). Assuming that M is such, we subsequently describe

```
proc reduceMasterInstances(graph G, master t) {
  Let G=(V,E)
  Let G_t=(V_t,E_t) represent flat graph of master t
1.  V'←V
2.  E'←E
3.    for each v_t∈V_t
        Let v represent v_t in G
4.      for each u∈V so that (u,v)∈E
5.        for each w∈V so that (v,w)∈E
6.          E'←E'∪{(u,w)}
7.          W(u,w)←max{W(u,v)+W(v,w),W(u,w)}
8.      for each u∈V so that (u,v)∈E
9.        E'←E'\{(u,v)}
10.     for each w∈V so that (v,w)∈E
11.       E'←E'\{(v,w)}
12.     V'←V'\{v}
13.   return (V',E')
```

Fig. 14. A pseudo code of graph reduction.

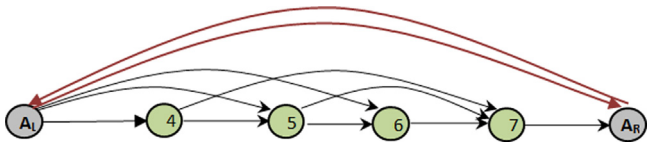

Fig. 15. The graph obtained by reducing the graph in Fig. 13. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

```
Let G=(V,E)  R_i=(V_i,E_i)  R_{i-1}=(V_{i-1},E_{i-1})
Let σ:V→ℝ
1. for each (u,v)∈E_{i-1}\E_i
     Let x_u,x_v represent the variables of LP problem
   corresponding to u,v respectively
2.   if u,v∈V_{i-1}\V_i
3.     Add LP constraint: x_v−x_u≥W(u,v)
4.   else if u∈V_{i-1}\V_i
5.     Add LP constraint: σ(v)−x_u≥W(u,v)
6.   else
7.     Add LP constraint: x_v−σ(u)≥W(u,v)
8. Invoke LP solver to solve the problem
9. for each v∈V_{i-1}\V_i
10.    σ(v)←LPsolverSolution(v)
```

Fig. 16. A linear program to locate the wires of a master.

how the exact locations of the wires can be determined by successively solving a small LP for each of the masters. Unlike the merging and reduction phases which took place in reversed topological order (bottom-up) of the masters, the commitment of exact locations of wires takes place in direct topological order (top-down). Here is where the major advantage of our algorithm lays; while past hierarchical compactors worked on the entire flattened layout, thus solving a huge LP, our algorithm is successively solving a series of far smaller LPs, exploring exactly the same solution space as the "flattened" LP does (proven in Section 4).

The LP solutions take place in the topological order of $b_i, 1 \le i \le N$, from b_N (root) down to b_1 (leaf). After the LP problem for b_{N-j} has been solved means the exact location of the wires (values of LP variables) in each of the masters $b_N, b_{N-1}, \ldots, b_{N-j}$ is determined. Recall that b_{N-j-1} implied a corresponding reduced graph R_{N-j-1}. Therefore, only those variables (wires locations) related to b_{N-j-1} are left to be determined, where a feasible solution is guaranteed. Appropriate pseudo code is shown in Fig. 16. For the special case of deriving a solution for the root master b_N we define a reduced graph $R_{N+1}=(V_{N+1},E_{N+1})$, where V_{N+1} is the set of vertices that represent the borders of the master b_N and $E_{N+1}=\varnothing$. Note that $\sigma(v)$ is determined for V_{N+1} since the borders of b_N are known.

The locations of the wires represented by V_i in the pseudo code in Fig. 16 have already been determined and the code determines the locations of the wires of $V_{i-1} \setminus V_i$. An arc of R_{i-1} implies a constraint of the LP problem. Since the exact locations for the vertices of V_i have already been determined, only the arcs of $E_{i-1} \setminus E_i$ are of interest. An arc (u,v) implies a constraint of the LP in line 3 if the exact location of both vertices have not yet been determined (i.e. both vertices are in $V_{i-1} \setminus V_i$). In this case each vertex is represented by a variable of an LP problem. If the exact location of one vertex v has already been determined, there is no corresponding variable in the LP, but the exact location $\sigma(v)$ is used (lines 5 and 7).

Fig. 17 illustrates the relation between a reduced graph and its corresponding LP. The green vertices represent the wires of master A, whose locations have already been determined upon the solution of the LP corresponding to R_A. Their committed locations are specified next to the vertices. The positions of the gray vertices, representing vertical borders, have also been determined, since their vertices and the parallel arcs enforcing A's size (as determined by placement), also exist in R_A. The reduced graph R_B is illustrated in (a), for which the LP variables are those corresponding to the wires of master B. The constraints imposed on arc lengths are translated into the inequalities in (b).

4. Correctness of the migration algorithm

We subsequently show that if there is a legal positioning of the wires satisfying all the constraints mentioned above, the algorithm described in Section 3 will find one by applying LP. This section states few propositions leading to the equivalence between a layout and its various graph representations. The proofs can be found in the appendix. The reader is referred to the FLVG definitions in 3.3.

4.1. Layouts and graphs

Let V be the wires and borders of a layout L and $G=(V^G,E^G)$ be the corresponding FLVG. A *Solution* σ of L is a function$\sigma: V \to \mathbb{R}$. We denote by $Z(u,v)$ the spacing imposed in the target layout between $u,v \in V$(to be distinguished from $W(u,v)$defined in the FLVG). A solution σ implies a *feasible* layout if for each $u,v \in V$ the condition $\sigma(v)-\sigma(u) \ge Z(u,v)$ holds. The *coordinate* $\sigma(p)$ of the wire p is the abscissa of its center. The offset $\lambda(I)$ of a master instance I is the abscissa of its left border. A feasible solution σ preserves the uniqueness of masters in the layout if for each $v_1, v_2 \in V$ satisfying $T(I(v_1))=T(I(v_2))$ and $m(v_1)=m(v_2)$, the condition $\sigma(v_2)-\sigma(v_1)=\lambda(I(v_2))-\lambda(I(v_1))$ holds.

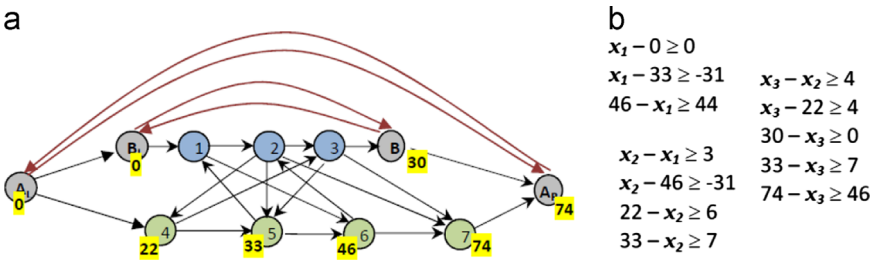

Fig. 17. A reduced graph is in (a) and the corresponding LP in (b). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

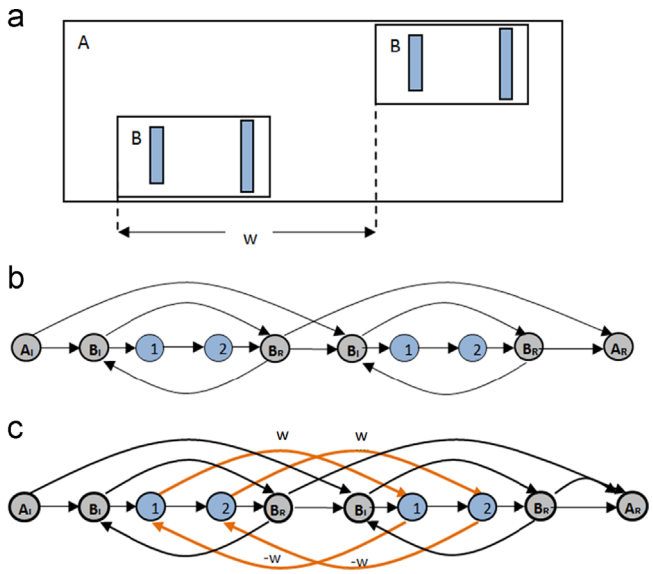

Fig. 18. A layout in (a), its corresponding FLVG in (b) and HCG in (c). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

A *Hierarchy Constraint Graph* (HCG) $H=(V^H,E^H)$ of a layout L is derived from its FLVG $G=(V^G,E^G)$ by setting $V^H=V^G$. Its arcs are obtained by augmenting E^G with *similarity* arcs, defined by $\{(u,v)|\ m(u)=m(v)\}$ with weights $W(u,v)=\sigma(I(v))-\sigma(I(u))$.

A path from u to v is denoted by $u\rightarrow v$ and its total sum of arc weights is $S(u\rightarrow v)$. The length of the longest path connecting u with v is $C(u\rightarrow v)$. The solution σ of H is *feasible* if for each $(u,v)\in E^H$ there exists $\sigma(v)-\sigma(u)\geq C(u\rightarrow v)$. It follows from the correspondence between layout objects (wires, borders) and graph's vertices, and the correspondence between spacing requirements and arc weight, that a feasible layout and a feasible graph are just two views of the same thing. Fig. 18 illustrates a layout (a) and its corresponding FLVG (b) and HCG (c). Master B contains two wires (colored blue) and is instantiated twice in master A. Borders of A and B represented in the FLVG and HCG by the gray vertices, while the wires by blue vertices. Similarity arcs are brown.

Propositions 1–3 below summarizes the equality between the properties of layouts and the associated graphs.

Proposition 1. A solution σ of a layout L is feasible if and only if σ is a feasible solution of its corresponding FLVG.

Proposition 2. A solution σ of a layout L is feasible and preserving its hierarchy and uniqueness of masters if and only if σ is a feasible solution of its corresponding HCG.

Proposition 3. Given a HCG H of the layout L, H has a feasible solution if and only if it has no positive cycles.

4.2. The algorithm's invariants

The hierarchical migration is carried out by a series of graph transformations, simplifying the layout representation. It is therefore necessary to show that the solution space is invariant under those simplifications.

An *i-Hierarchy Constraint Graph* $H_i=(V_i,E_i)$ of a layout L is derived from FLVG $G=(V^G,E^G)$ by setting $V_i=V^G$ and augmenting E^G with similarity arcs defined by $\{(u,v)|\ m(u)=m(v),1\leq m(u)\leq i\}$, with weights $W(u,v)=\lambda(I(v))-\lambda(I(u))$. This defines a series of N transformations corresponding to the partial order $\prec$ of the masters, gradually converting the representation of the layout from $H_0=G$ to $H_N=H$. It is subsequently shown that the simplified merged graphs $M_i, 0\leq i\leq N$, used in the migration algorithm (see Fig. 6), are equivalent to $H_i, 0\leq i\leq N$, in such a way that the length of longest path between any two vertices is preserved, while M_i significantly diluted its arcs.

Proposition 4. Let $H=(V^H,E^H)$ be an HCG derived from $G=(V^G,E^G)$ by adding similarity arcs of a master t and $M=(V^M,E^M)$ generated from G by merging all the instances of t. Then for every $u,v\in V^H$ satisfying $I(u)=I(v)$ and $T(u)=T(v)=t$, there exists $C_M(m(u),m(v))=C_H(u,v)$.

Proposition 5. Let $H=(V^H,E^H)$ be an HCG generated from $G=(V^G,E^G)$ by adding similarity arcs of a master t and $M=(V^M,E^M)$ generated from G by merging all the instances of t. Then for every $u,v\in V^H$ satisfying $I(u)=I(v)$, $T(u)\neq t$ and $T(v)\neq t$ there exists $C_M(u,v)=C_H(u,v)$.

Proposition 6. Let $H=(V^H,E^H)$ be an HCG generated from $G=(V^G,E^G)$ and $M=(V^M,E^M)$ generated from G by the merging phase of the algorithm as shown in Fig. 12. Then for every $u,v\in V^H$ satisfying $I(u)=I(v)$, $T(u)\neq t$ and $T(v)\neq t$, there exists $C_M(u,v)=C_H(u,v)$.

Corollary 1. A layout L has a feasible solution preserving hierarchy if and only if its merged graph M has no positive cycles.

Proof. Follows from Propositions 2, 3 and 6. ■

Proposition 7 below captures the invariance of the reduction phase: the longest path weight between a pair of vertices in the graph remains unchanged under reduction.

Proposition 7. Let $R=(V^R,E^R)$ be a reduced graph generated from G by a single invocation of *reduceMasterInstances* procedure shown in Fig. 14. Then for every $u,v\in V^R$ there exists $C_R(u,v)=C_G(u,v)$.

Proposition 8. Let $R=(V^R,E^R)$ be a reduced graph generated from $M=(V^M,E^M)$. Then for every $u,v\in V^R$ there exists $C_R(u,v)=C_M(u,v)$.

Next comes the phase of deriving of a specific (committed) solution as shown in Fig. 16. This phase works top-down, where iteration treats a single master. We define an *i–partial solution* for a graph $G=(V^G,E^G)$ as a function$\sigma_i: V_i\rightarrow\mathbb{R}, 0\leq i\leq N$, where $V_i\subseteq V^G$ is all the vertices which have been solved after the i-th iterations

(their position have already been decided). The set V_i is all the vertices representing the wires of the masters $b_1, \ldots, b_i$ in the $\prec$ order. This phase starts with σ_0 for V_0, including all the borders of the master instances, and it ends with the entire layout. Proposition 9 below relates to a single iteration.

Proposition 9. For $0 \le i \le N$, the i–partial solution of the merged graph M of layout L, obtained from the exact solution derivation phase, is feasible.

It follows immediately from Proposition 9 that the N–partial solution completely solves the problem. Proposition 10 below constitutes the successive derivation of feasible solutions.

Proposition 10. Let M be the merged graph of a layout L which has no positive cycles. Let σ_{i-1} be a feasible $(i-1)$-partial solution of M. There exists a feasible i-partial solution σ_i of M which is obtained at iteration i.

The following concludes the proofs of the correctness of the series of transformations.

Lemma 1. *If there exists a feasible solution preserving the hierarchy of a layout L, the flow in Fig. 16 will return a feasible solution σ for the HCG, H of L.*

Lemma 2. *For any feasible solution σ preserving the hierarchy for a given layout L, there is an objective function F, such that the flow will obtain σ.*

Theorem. *The flow satisfies the requirements 1–5 stated in the description of the algorithm.*

5. Experimental Results

The following results have been obtained for Intel's 65 nm process technology (Tock) microprocessors, branded as Core 2 Duo, where the interconnecting wires have been migrated into 45 nm process technology (Tick). The examples below incorporate the results for several blocks, each comprising thousands of nets. The quality of such migration is determined by the performance of the underlying circuits in the target layout, measured by simulations. The migration set the goal of achieving delay reduction of 0.7, and hence speeding up the clock frequency by a factor of 1.4, which is Moore's Law premise.

Wire widths and line-to-line space specifications have been derived from the Elmore delay model [12], based on the electrical parameters of the 45 nm technology. The migration described in this paper was applied. The positive cycles discovered by the algorithm have been resolved by reducing wire widths and spaces corresponding to those vertices and arcs occurring on the positive cycles. Although such relaxations resolved the problem and legalized the positions of wires, delay violations due to resistance and coupling (line-to-line) capacitance increase may occur. Those are resolved by VLSI design techniques such as using stronger circuit drive or timing tuning, which are beyond the scope of this paper.

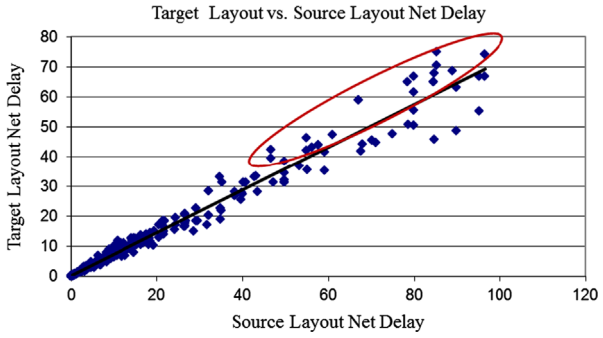

Fig. 19. Delay results of net in target layout versus source layout.

Fig. 19 illustrates the signal delay reduction (speedup). The linear line represents 0.7 delay reduction so all the dots under it represent properly sped up nets. As shown, some of the nets did not meet that goal. Fortunately, the delays of the majority of those are far smaller than the target clock cycle of the processor, so they do not impose any problem. Only those encircled required further treatment in the design to avoid critical signals exceeding the target clock cycle. Here lays the major advantage of automatic layout migration; it delivers satisfactory performance for the majority of the interconnects, leaving a relatively small percentage (less than 10%) for further fix-up by engineering effort.

The experimental results obtained by the hierarchical compaction algorithm are demonstrated for a set of nine blocks and are summarized in Table 1. For each block the number of vertices and arcs in its corresponding FVLG are specified. Next are shown the number of vertices and arcs in the merged graph. The computational efficiency is demonstrated by the column reporting the largest LP problem incurred by the series of the graph reductions. Recall that the variables of an LP are the vertices of a reduced graph. The LP column presents reductions of the problem sizes by one to two orders of magnitude compared to the flattened layout. The computational efficiency premise is also shown by the two last columns of runtime. To grasp the runtime that could be obtained by using the algorithms in [6–10], the contraction of the all the instances of a master into a single cell were ignored. This effectively turns the algorithm to behave similarly to those mentioned in the introduction. A runtime speedup of one to two orders of magnitude is shown.

The area scaling obtained for the top-level block is specified for each experiment. It follows from the 0.7 lateral and longitudinal average scaling factors that the area should ideally scale to 0.49, but as mentioned in Section 3 the scaling factors may vary within a wide range. To further demonstrate the area scaling variance, Fig. 20 illustrates the area scaling distribution of all the 601 master cells constructing the block of experiment 9.

6. Conclusions

This work showed how true hierarchical migration of large layouts can be performed by efficiently exploring the entire solution space. The proposed algorithm takes full advantage of the inherent hierarchy built into VLSI designs. By applying a series of transformations, the underlying computational problems have been reduced by two orders of magnitude, making the solution of large problems feasible.

The proposed bottom-up feasibility step is useful in 22 nm and below to find the feasibility range of each wire, but the non-existence of positive cycles turns from a sufficient condition for feasible solution into a necessary one. The top-down phase which decides the exact location of the wires within their feasible range requires ILP rather than LP. Another possible solution is to apply a dynamic programming, as done in [20] for flat layouts. This still requires further development to support hierarchy. Both approaches are beyond the scope of this paper and a matter for further research. There may also be wires that will have no solution, so those must be left for later manual fixes.

Acknowledgments

The authors are thankful for the useful reviewers' comments, which helped improving the manuscript.

Table 1
Tests statistics – size of graphs, area scaling and runtime comparisons.

Test no.	FLVG vertices	FLVG arcs	Merged graph vertices	Merged graph arcs	Variables in largest LP problem	Reduced to FLVG ratio	Area scaling	Runtime hierarchy [s]	Runtime flattened [s]
1	1151	5755	499	2249	302	0.2	0.44	7	68
2	2064	8456	1687	7130	598	0.29	0.49	11	120
3	4457	16,890	1253	6101	434	0.098	0.46	313	2509
4	12,928	55,892	2535	19,524	386	0.03	0.50	244	2547
5	23,020	106,554	4683	40,334	1561	0.0687	0.55	1167	15,644
6	29,714	109,212	4205	16,255	1173	0.0395	0.47	676	14,139
7	45,964	177,946	5963	24,928	1308	0.0284	0.50	1532	45,337
8	62,944	248,915	7850	44,963	1277	0.0202	0.58	2099	42,835
9	73,005	294,479	9455	63,737	1420	0.0194	0.54	2560	78,158

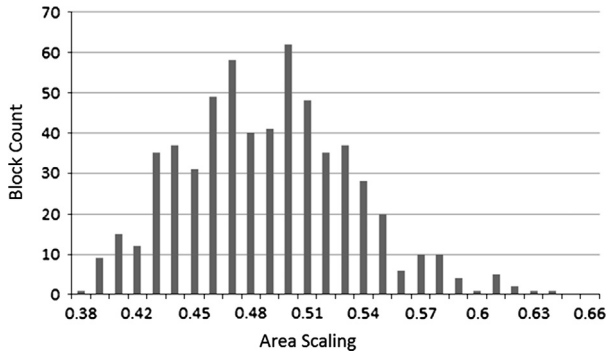

Fig. 20. Area scaling distribution of master cells.

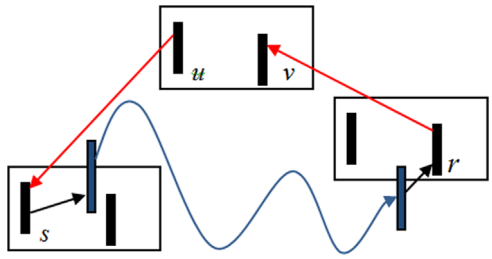

Fig. 21. Paths in H_t. In red are similarity arcs. Blue represents $x_1 \to x_2 \to \ldots \to x_n$.(For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Appendix

Propositions 4a and 4b are supplementary for the forthcoming proof of proposition 4.

Proposition 4a. Let $H=(V^H,E^H)$ be an HCG generated from $G=(V^G,E^G)$ by adding similarity arcs of a master t, and $M=(V^M,E^M)$ be generated from G by merging all the instances of t. Then for every $u,v\in V^H$ satisfying $I(u)=I(v)$ and $T(u)=T(v)=t$, there exists $C_M(m(u),m(v))\le C_H(u,v)$.

Proof. Let $\pi: m(u)\to x_1\to x_2\to\ldots\to x_n\to m(v)$ be the longest path in M. We will construct the path π' in H and show by induction on number k of t's vertices in the path that $S(\pi')\ge S(\pi)$. For $k=0$ none of the vertices belongs to t. There are two cases: $x_1=m(v)$ and $x_1\neq m(v)$. If $x_1=m(v)$ then $(u,v)\in E^G$. The weight of the arc was modified twice by the procedure *mergeMasterInstances*, once by adding the offset of instance $I(u)$ (line 10 in Fig. 12) since the arc is outgoing of u. It was then modified by subtracting the offset of same instance $I(v)$ (line 6 in Fig. 12) since it is an incoming arc of v; Since $I(v)=I(u)$, $W_M(m(u),m(v))=W_G(u,v)$. Since (u,v) is not a similarity arc.

Consider the case $x_1\neq m(v)$. By construction of M, there exists $s\in V^G$, $(s,x_1)\in E^G$, $m(s)=m(u)$ and the weight of the arc changed exactly once by adding the offset of $I(s)$ (line 10 in Fig. 12), namely, $W_M(m(u),x_1)=W_G(s,x_1)+\lambda(I(s))$. By construction of M, there exists $r\in V^G$, $(x_n,r)\in E^G$, $m(r)=m(v)$, and the weight of the arc has been modified exactly once by subtracting the offset of instance $I(r)$ (line 6 in Fig. 12), namely, $W_M(x_n,m(v))=W_G(x_n,r)-\lambda(I(r))$.

If $T(x_j)\neq t, 1\le j\le n$ then the path $x_1\to\cdots\to x_n$ exists in G, and all the arcs in the path are unchanged by *mergeMasterInstances*. The HCG H contains the similarity arcs of t, thus $(u,s)\in H$, and $(r,v)\in H$. Therefore, $W_H(u,s)=\lambda(I(s))-\lambda(I(u))$ and $W_H(r,v)=\lambda(I(v))-\lambda(I(r))$. Let us calculate the weight of path $u\to s\to x_1\to\cdots\to x_n\to r\to v$ in H as illustrated on Fig. 21. There exists

$$S_H(u\to s\to x_1\to\cdots\to x_n\to r\to v)=W_H(u,s)+W_H(s,x_1)+C_H(x_1\to x_n)$$
$$+W_H(x_n,r)+W_H(r,v)\ge W_H(u,s)+W_G(s,x_1)+C_G(x_1\to x_n)$$
$$+W_G(x_n,r)+W_H(r,v)=[\lambda(I(s))-\lambda(I(u))]+[W_M(m(u),x_1)-\lambda(I(s))]$$
$$+C_M(x_1\to x_n)+[W_M(x_n,m(r))+\lambda(I(r))]+[\lambda(I(v))-\lambda(I(r))]$$

Since $I(u)=I(v)$, there exists $S_H(u\to\cdots\to v)\ge W_M(m(u),x_1)+C_M(x_1\to x_n)+W_M(x_n,m(v))=C_M(m(u),m(v))$, which completes the basis of the induction.

Assume by induction that the claim holds for every path having $k-1$ vertices of $t$at most, and let π be a longest path in M having $k$vertices of t. Since $k\ge 1$, the path $\pi: m(u)\to x_1\to\cdots\to m(w)\to\cdots\to x_n\to m(v)$ includes a mid-vertex $w\in V^G$, $m(w)=t$. Clearly the sub-path $\pi_1: m(u)\to x_1\to\cdots\to m(w)$ is also a longest one and similarly holds for $\pi_2: m(w)\to\cdots\to x_n\to m(v)$. Thus induction hypothesis holds for both paths π_1 and π_2, and $C_M(m(u),m(w))\le C_H(u,w)$, $C_M(m(w),m(v))\le C_H(w,v)$. In conclusion we showed that$C_M(m(u),m(v))=C_H(u,w)+C_H(w,v)\le C_H(u,v)$ □

Proposition 4b. Let $H=(V^H,E^H)$ be an HCG generated from $G=(V^G,E^G)$ by adding similarity arcs of a master t and $M=(V^M,E^M)$ generated from G by merging all the instances of t. Then for every $u,v\in V^H$ satisfying $I(u)=I(v)$ and $T(u)=T(v)=t$, there exists $C_M(m(u),m(v))\ge C_H(u,v)$.

Proof. Let $\pi: u\to x_1\to x_2\to\ldots\to x_n\to v$ be the longest path in H. We will construct the path π' in M and show $S(\pi')\ge S(\pi)$. The correctness is shown by induction on number k of t's vertices in the path. For $k=0$ none of the vertices belongs to T. There are two cases $x_1=m(v)$ and $x_1\neq m(v)$. The case $x_1=m(v)$ is identical to Proposition 4a. Consider the case $x_1\neq m(v)$. By construction of M, the weight of arc $(u,x_1)\in E^G$ changed exactly once by adding the offset of instance $I(u)$ (line 10 in Fig. 12), $W_M(m(u),x_1)=W_G(u,x_1)+\lambda(I(u))$. By construction of M, the weight of the arc $(x_n,v)\in E^G$ changed exactly once by subtracting the offset of instance $I(v)$ (line 6 in Fig. 12), and hence $W_M(x_n,m(v))=W_G(x_n,v)-\lambda(I(v))$.

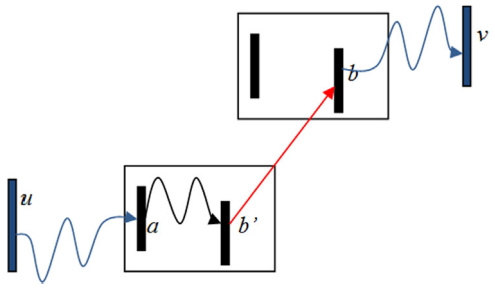

Fig. 22. The path in H_t. In red is a similarity arc. Blue represent the paths $x_1 \to \ldots \to x_n$ and $y_1 \to \ldots \to y_m$.(For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Since $T(x_j) \neq t, 1 \leq j \leq n$, there is no similarity arcs in the path $u \to x_1 \to \cdots \to x_n \to v$, thus it fully exists in G and belongs to the same instance. By the construction of M there exists $W_M(x_j, x_{j+1}) = W_H(x_j, x_{j+1})$. Let us calculate the weight of path $m(u) \to s \to x_1 \to \cdots \to x_n \to t \to m(v)$ in M. There exists

$$S_M(m(u) \to x_1 \to \cdots \to x_n \to m(v)) = W_M(m(u), x_1) + C_M(x_1 \to x_n) + W_M(x_n, m(v)) = [W_G(u, x_1) + \lambda(I(u))] + C_H(x_1 \to x_n) + [W_G(x_n, v) - \lambda(I(v))]$$

Since $I(u) = I(v)$, there exists $S_M(m(u) \to x_1 \to \cdots \to x_n \to m(v)) = W_H(u, x_1) + C_H(x_1 \to x_n) + W_H(x_n, v) = S_H(u \to x_1 \to \cdots \to x_n \to v)$, which completes the basis of the induction.

Assume by induction that the claim holds for every path having $k-1$ vertices of t at most, and let π be a longest path in H having k vertices of t. Let (x_k, x_{k+1}) be the first similarity arc in π. Clearly, $m(x_k) = m(x_{k+1})$ and $T(I(x_k)) = T(I(x_{k+1})) = t$. If $I(x_k) \neq I(u)$ and $I(x_{k+1}) \neq I(u)$ we will introduce a new path $\pi' = u \to x_1 \to \cdots \to x_k \to w \to x_{k+1} \to \cdots \to x_n \to v$ such that $I(w) = I(u)$ in H and show that $S(\pi') \geq S(\pi)$There exists

$$S_H(x_k \to w \to x_{k+1}) = W_H(x_k, w) + W_H(w, x_{k+1}) = \lambda(I(w)) - \lambda(I(x_k)) + \lambda(I(x_{k+1})) - \lambda(I(w)) = W_H(x_k, x_{k+1}).$$

Clearly, the sub-path $\pi_1 : m(u) \to x_1 \to \cdots \to m(w)$ is a longest path in M, similarly it holds for $\pi_2 : m(w) \to \cdots \to x_n \to m(v)$. Since induction hypothesis holds for both paths π_1 and π_2, there exists $C_M(m(u), m(w)) \geq C_M(u, w)$, and $C_M(m(w), m(v)) \geq C_M(m(w), m(v)) \geq C_H(w, v)$. Hence $C_M(m(u), m(v)) \geq C_M(m(u), m(w)) + C_M(m(w), m(v)) \geq C_H(u, v)$. □

Proof of Proposition 4. Follows from Propositions 4a and 4b. ■

Propositions 5a and 5b are supplementary for the forthcoming proof of Proposition 5.

Proposition 5a. Let $H = (V^H, E^H)$ be an HCG generated from $G = (V^G, E^G)$ by adding similarity arcs of a master t, and $M = (V^M, E^M)$ be generated from G by merging all the instances of t. Then for every $u, v \in V^H$ satisfying $I(u) = I(v)$, $T(u) \neq t$ and $T(v) \neq t$, there exists $C_M(u, v) \leq C_H(u, v)$.

Proof. Let $\pi : u \to x_1 \to x_2 \to \ldots \to x_k \to v$ be the longest path in M. There are two cases. If $T(I(x_i)) \neq T, 1 \leq i \leq k$, then the same path exists in G and all the arcs in the path are unchanged by *mergeMasterInstances*. Consider the case $T(I(x_j)) = T$ for some j. Let $m(a)$ be the first vertex in π that belongs to T, $m(b)$ be the last one, so the path is $\pi : u \to x_1 \to \ldots$, where $m(I(x_i)) \neq T$ and $m(I(y_i)) \neq T \ldots \to x_n \to m(a) \to \ldots \to m(b) \to y_1 \to \ldots \to y_m \to v$, as illustrated in Fig. 22.

By the construction of M, there exists $a \in V^G$ such that $(x_n, a) \in E^G$. The weight of the (x_n, a) changed exactly once by subtracting the offset of instance $I(a)$ (line 6 in Fig. 12), $W_M(x_n, m(a)) = W_G(x_n, a) - \lambda(I(a))$. Similarly there exists $b \in V^G$ such that $(b, y_1) \in E^G$. The weight of the arc (b, y_1) changed exactly once by adding the offset of instance $I(b)$ (line 10 of in Fig. 12), $W_M(m(b), y_1) = W_G(b, y_1) + \lambda(I(b))$. By Proposition 4a there exists a path $a \to \ldots \to b'$ in H such that $I(b') = I(a)$, $m(b') = m(b)$ and $C_M(m(a), m(b)) \leq C_H(a, b')$. Therefore,

$$C_H(u, v) = C_H(u, x_n) + W_H(x_n, a) + W_H(a, b') + W_H(b', b) + W_H(b, y_1) + W_H(y_1, v) \geq C_M(u, x_n) + [W_M(x_n, m(a)) + \lambda(I(a))] + W_M(m(a), m(b)) + [\lambda(I(b)) - \lambda(I(b'))] + [W_M(m(b), y_1) - \lambda(I(b))] + W_M(y_1, v) = C_M(u, v),$$

which concludes the proof. □

Proposition 5b. Let $H = (V^H, E^H)$ be an HCG generated from $G = (V^G, E^G)$ by adding similarity arcs of a master T, and $M = (V^M, E^M)$ be generated from G by merging all the instances of T. Then for every $u, v \in V^H$ satisfying $I(u) = I(v)$, $m(u) \neq T$ and $m(v) \neq T$, there exists $C_M(u, v) \geq C_H(u, v)$.

Proof. Let $\pi : u \to x_1 \to x_2 \to \ldots \to x_k \to v$ be the longest path in H. There are two cases. If $m(I(x_i)) \neq T, 1 \leq i \leq k$, then the same path exists in G and all the arcs in the path are unchanged by *mergeMasterInstances*. Consider the case $m(I(x_j)) = T$for some j. Let a be the first vertex in π that belong to T let b be the last one. The path is $\pi : u \to x_1 \to \ldots \to x_n \to a \to \ldots \to b \to y_1 \to \ldots \to y_m \to v$, where $m(I(x_i)) \neq T$ and $m(I(y_i)) \neq T$, as illustrated in Fig. 23.

By definition of H there is $b' \in E^H$ such that $m(I(b')) = m(I(a))$ and similarity arcs $(b, b'), (b', b) \in E^H$, $W(b, b') = -W(b', b)$. Thus there is a new path π' in H, $\pi' : u \to x_1 \to \ldots \to x_n \to a \to \ldots \to b \to b' \to b \to y_1 \to \ldots \to y_m \to v$, with the same weight. By the construction of M $(x_n, m(a)) \in E^M$. The weight of (x_n, a) changed exactly once by subtracting the offset of instance $I(a)$ (line 6 in Fig. 12), $W_M(x_n, m(a)) = W_G(x_n, a) - \lambda(I(a))$. Similarly for the weight of $(m(b), y_1) \in E^G$. The weight of the arc (b, y_1) changed exactly once by adding the offset of instance $I(b)$ (line 7 of in Fig. 12), $W_M(m(b), y_1) = W_G(b, y_1) + \lambda(I(b))$. By Proposition 4a there exists path $a \to \ldots \to b'$ in H such that $I(b') = I(a)$, $m(b') = m(b)$ and $C_M(m(a), m(b)) \geq C_H(a, b')$. Therefore,

$$C_H(u, v) = C_H(u, x_n) + W_H(x_n, a) + W_H(a, b') + W_H(b', b) + W_H(b, y_1) + W_H(y_1, v) \leq C_M(u, x_n) + [W_M(x_n, m(a)) + \lambda(I(a))] + W_M(m(a), m(b)) + [\lambda(I(b)) - \lambda(I(b'))] + [W_M(m(b), y_1) - \lambda(I(b))] + W_M(y_1, v) = C_M(u, v)$$

which concludes the proof. □

Proof of Proposition 5. Follows immediately from Propositions 5a and 5b. □

Proof of Proposition 6. Follows Immediately from Propositions 4 and 5 by induction on the number of ***merging iterations.*** ■

Proof of Proposition 7. By induction on number iterations of loop 4-1 of the reduction phase (Fig. 14). Denote by R_i the graph after i-th iteration of the loop. At an iteration exactly one vertex (with all it arcs) is removed and some other arcs are added to the graph.

The basis R_0 of the induction equals G since nothing was changed. Assume by induction that the claim holds for R_{i-1}. Let π be the longest path in R_{i-1} between pair of u, w and let v be the vertex that is reduced at the step i. Clearly if v does not belong to the path, the path is unchanged. Otherwise, consider the path $\pi : u \to \ldots \to s \to v \to t \to \ldots \to w$. Assume w.l.o.g that π has no loop. As in line 7 of *reduceMasterInstances*, for the arcs (s, v) and (v, t) the new arc (s, t) added to R_i. Since $W_{R_i}(s, t) = W_{R_{i-1}}(s, v) + W_{R_{i-1}}(v, t)$, there is a path $\pi' : u \to \ldots \to s \to t \to \ldots \to w$ in R_i satisfying $W_{R_i}(\pi') = W_{R_{i-1}}(\pi)$, and hence $C_{R_i}(u, w) \geq W_{R_{i-1}}(u, w)$. It is left to show that $C_{R_i}(u, w) \leq W_{R_{i-1}}(u, w)$, which is similar to the above case. ■

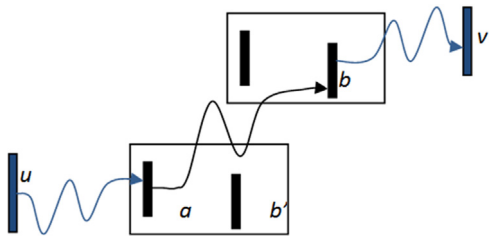

Fig. 23. The path π in H_t. Blue represents the paths $x_1 \to \ldots \to x_n$ and $y_1 \to \ldots \to y_m$.(For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Proof of Proposition 8. Follows from Proposition 7 with an induction on number of calls to *reduceMasterInstances*. ■

Proposition 9a is supplementary for the forthcoming proof of proposition 9.

Proposition 9a. Let $G=(V^G,E^G)$ be a directed graph with weights $W: E^G \to \mathbb{R}^+$ and let $f: V^G \to \mathbb{R}^+$ be a non-negative function satisfying $W(u,v) \le f(v)-f(u)$. Then for any path $\pi: s \to \ldots \to t$ in G there exists $S(\pi) \le f(t)-f(s)$.

Proof. Immediate by induction on length of π. □

Proof of Proposition 9. Let σ_i be a i-partial solution for $M=(V^M,E^M)$. The correctness is shown by induction on i. The basis of the induction is $i=0$ and $V_0=M$. Clearly σ_0 is a feasible solution for M. We assume that the claim holds for all $i<k$ and prove it for $i=k$. Using Proposition 9a it is sufficient to show that for every two vertices u,v such that $(u,v)\in E^M$ there exists $\sigma_k(v)-\sigma_k(u) \ge W_M(u,v)$. If $v,u \in V_{k-1}$, the inequality holds by the induction hypothesis. Otherwise, assume w.l.o.g that $v \notin V_{k-1}$ and consider two cases: $u \in V_{k-1}$ and $u \notin V_{k-1}$. If $u \in V_{k-1}$, the constraint $\sigma_k(v)-\sigma_{k-1}(u) \ge W_M(u,v)$ added to LP problem (line 7, Fig. 16). Since $\sigma_k(u)=\sigma_{k-1}(u)$ and σ_k satisfies the constraints, $\sigma_k(v)-\sigma_k(u) \ge W_M(u,v)$. The case $u \notin V_{k-1}$ is similar, and the constraint that is added is $\sigma_k(v)-\sigma_k(u) \ge W_M(u,v)$ (line 3, Fig. 15) which makes σ_k feasible. □

Proof of Proposition 10. The existence of solution is shown by construction. At iteration i, we define the solution $\sigma_i(v)$ for any vertex $v \in V_i$ as follows:$\sigma_i(v) = \max\{C_{R_i}(u,v)+\sigma_{i-1}(u) | u \in V_{i-1}\}$ if $v \notin V_{i-1}$ and $\sigma_i(v)=\sigma_{i-1}(v)$ otherwise. Since M does not contain positive cycles by Proposition 8, R_i does not contain positive cycles. Hence $C_{R_i}(u,v)<\infty$ and σ_i is well defined. It is left to show that σ_i is feasible. By Proposition 9a it is enough to show that for any pair of vertices u,v there exists$\sigma_i(v)-\sigma_i(u) \ge W_M(u,v)$. Let$u,v \in V_i$. If $u,v \in V_{i-1}$ then $\sigma_i(v)=\sigma_{i-1}(v)$ and $\sigma_i(u)=\sigma_{i-1}(u)$. Since σ_{i-1} is feasible, we have $\sigma_i(v)-\sigma_i(u) \ge W_M(u,v)$. If $u \in V_{i-1}$ and $v \notin V_{i-1}$, it follows from Proposition 8 that for any $w \in V_{i-1}$ $C_{R_i}(w,v)=C_M(w,v)$, thus $\sigma_i(v) = \max\{C_M(w,v)+\sigma_{i-1}(w) | w \in V_{i-1}\}$ and therefore $\sigma_i(v)-\sigma_{i-1}(u) \ge C_M(u,v)$. Since $\sigma_i(u)=\sigma_{i-1}(u)$ we obtain $\sigma_i(v)-\sigma_i(u) < W_M(u,v)$. If $u,v \notin V_{i-1}$ then by Proposition 8 $C_{R_i}(u,v)=C_M(u,v)$, and hence $\sigma_i(u) = \max\{C_M(w,u)+\sigma_{i-1}(w) | w \in V_{i-1}\}$. Let $s \to \ldots \to u$ be the longest path such that $s \in V_{i-1}$. Since $s \to \ldots \to u \to v$ is a path in R_i there exists $\sigma_i(v) \ge C_M(s,v)+\sigma_{i-1}(s) \ge C_M(s,u) + W(u,v)+\sigma_{i-1}(s)=\sigma_i(u)+W(u,v)$and thus$\sigma_i(v)-\sigma_i(u) \ge W_M(u,v)$. The case $u \notin V_{i-1}$and $v \in V_{i-1}$ is similar. □

Proof of Lemma 1. Since L has feasible solution that preserves hierarchy, by the Corollary 1 its merged graph M does not contain positive cycles. By Proposition 9, the solution σ is feasible for M, and by Proposition 6 it is feasible for$H$. Since Mdoes not contain positive cycles by Proposition 10, σwill be returned by the flow. □

Proof of Lemma 2. Let σ be a feasible solution for L. Define for a solution φ, an objective function $F(\varphi)=\sum_{v \in V} \Delta_v$ where $\Delta_v=|\sigma(v)-\varphi(v)|$. Clearly F is linear and Δ can be written as a pair of LP constraints $\Delta_v \ge \sigma(v)-\varphi(v)$ and $\Delta_v \ge \varphi(v)-\sigma(v)$. The function $F$is non-negative since $\Delta_v \ge 0$. Showing that the flow returns a solution φ such that $F(\varphi)=0$ will imply $\varphi=\sigma$ and will complete the proof. This is shown by induction on the solution construction steps.

The basis of induction is the 0-partial solution φ_0, since proposition 2 ensures that σ is a feasible solution for HCG of L, comprising all the vertices of V_0 (borders). Assume by induction that the $(i-1)$-partial solution φ_{i-1} is equal to σ for all the vertices of V_{i-1}. Define $\varphi^*: V_i \to \mathbb{R}$ to be the partial solution for the vertices that are derived at iteration i, where for every $v \in V_i$ there exists $\varphi^*(v)=\sigma(v)$. Clearly φ^* is feasible since σ is feasible and thus satisfies all the LP constraints. Moreover, φ^* is optimal since$F(\varphi^*)=0$. Thus if solution φ_i returned by the LP solver differs from φ^*, φ_i would be sub-optimal, contradicting the LP solver optimality. □

Proof of Theorem. By Lemma 1 the solution σ returned by the flow is feasible for the HCG of L and thus σ is a feasible solution for L, preserving the hierarchy by Proposition 2. The solution σ is feasible and hence satisfies $W_G(u,v) \le \sigma(v)-\sigma(u)$. In particular σ preserves the visibility relation (wires order). Similarly, for the any instance I and its borders u,v the solution satisfies $D(m(I)) \le \sigma(v)-\sigma(u)$ and $-D(m(I)) \le \sigma(u)-\sigma(v)$, implying $D(m(I)) = \sigma(v)-\sigma(u)$, since σ satisfies the target placement. By construction of FVLG, every spacing specification is represented by an arc (u,v). Being a feasible solution σ satisfies all specifications. □

References

[1] T.S. Perry, Gordon Moore's next act, IEEE Spectrum on-line, May 2008.
[2] Intel's Tick-Tock Model, ⟨http://www.intel.com/technology/tick-tock/index.htm⟩.
[3] R. Nitzan and S. Wimer, AMPS and SiClone integration for implementing 0.18–0.13 μm design migration, in: Proceedings of the Synoposys Users Group (SNUG) Conference, San Jose, CA, March 2002.
[4] T. Lengaur, Combinatorial Algorithms for Integrated Circuit Layout, Chapter 10: Compaction, John Wiley (1990) 579–643.
[5] M. Reinhardt, Automatic Layout Modification: Including Design Reuse of the Alpha CPU in 0.13 μm SOI Technology, Kluwer Academic Publishers, 2002.
[6] J.L. Burns, A.R. Newton, Efficient constraint generation for hierarchical compaction, in: Proceedings of the IEEE International Conference on Computer Design, October 1987, pp. 197–200.
[7] J.L. Burns, J.A. Feldman, C5M—A control-logic layout synthesis system for high-performance microprocessors, IEEE Trans. CAD of Integr. Circ. Syst. 17 (1) (1998) 14–23.
[8] S.-Z. Yao, C.-K. Cheng, D. Dutt, S. Nahar, C.-Y. Lo, Cell-based hierarchical pitchmatching compaction using minimal LP, in: Proceedings of the 30th Design Automation Conference, 1993, pp. 395–400.
[9] L.-Y. Wang, Y.-T Lai, Graph-theory-based simplex algorithm for VLSI layout spacing problems with multiple variables constraints, IEEE Trans. CAD Integr. Circ. Syst. 20 (8) (2001) 967–979.
[10] Simultaneous, n-level hierarchical layout compaction, Process migration and Physical Optimization. ⟨http://www.sagantec.com/SiClone.pdf⟩.
[11] K. Moiseev, A. Kolodny and S. Wimer, Power-Delay optimization in VLSI microprocessors by wire spacing, ACM Transactions on Design Automation of Electronic Systems, Vol. 14, No. 4, August 2009, Article no. 55.
[12] A. Kanhng, K. Masuko, S. Andmuddu, Analytical delay models for VLSI interconnects under ramp input. In Proceedings of the IEEE International Conference on Computer-Aided Design ICCAD, 1996, pp. 30–36.
[13] S. Sastry A. Parker, The complexity of two dimensional compaction of VLSI layouts, in Proceedings of the International Conference on Circuits and Computers, 1982, pp. 402–406.
[14] T. Lengauer, On the solution of inequality systems relevant to IC-layout, J. Algorithms 5 (No. 3) (1984) 408–421.
[15] H. Shin, A.L. Sangiovanni-Vincentelli C.H. Siquin, Two-dimensional compaction by zone refining, Proceeding of the 23rd ACM/IEEE Design Automation Conference, 1986, pp. 115–122.
[16] M. Schlag, Y.Z. Liao, C.K. Wong, An algorithm for optimal two-dimensional compaction of VLSI layouts, Integr. VLSI J. 1 (2–3) (1983) 179–209.
[17] G. Klau, P. Mutzel, Optimal compaction of orthogonal grid drawings, Lecture Notes in Computer Science: Integer Programming and Combinatorial Optimization, 1610, Springer (1999) 304–319.
[18] M. Patrignani, On the complexity of orthogonal compaction, Comput. Geometry 19 (1) (2001) 47–67.

[19] S. Wimer, Planar CMOS to multi-gate layout conversion for maximal fin utilization, Integr. VLSI J. (2013) http://dx.doi.org/10.1016/j.vlsi.2013.03.004.
[20] K. Moiseev, A. Kolodny, S. Wimer, Interconnect bundle sizing under discrete design rules, IEEE Trans.CAD Integr. Circ. Syst. 29 (10) (2010) 1650–1654.

Eugene Shaphir received his B.A. in Mathematics and M.Sc. in Computer Science from Technion, Israel Institute of Technology, Haifa, Israel. Since 2001 he has been with Intel, working on EDA tools including signal integrity, timing and power. Since 2013 he is with Google, working on search engine. His interests include graph algorithms, heuristic algorithms, distributed algorithms and cryptography.

Pinter is a Professor of Computer Science (and Medicine by courtesy) at the Technion – Israel Institute of Technology. He received his B.Sc. Summa Cum Laude in Computer Science from the Technion in 1975, and his M.Sc. and Ph.D. in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology in 1980 and 1982. He held various research staff positions at Bell Labs and IBM where he was eventually the program manager for Computer Science at the IBM Research Lab in Haifa, Israel, and was also a member of the IBM Academy of Technology. He served as the VP for Research and Development at Compugen Ltd. Tel-Aviv, Israel.

He joined the Technion Faculty in the Department of Computer Science in 2001 and was promoted to Full Professor in 2010. His current research interest is primarily Computational Systems Biology, but he maintains his activities in areas in which he was engaged earlier in his career e.g. VLSI layout algorithms, high performance computing and code optimization in compilers. Prof. Pinter has published extensively in various archival journals and refereed conference proceedings. He also spent sabbatical leaves at Yale University, the IBM Almaden Research Center and Princeton University.

Shmuel Wimer (M'10) received the B.Sc. and M.Sc. degrees in mathematics from Tel-Aviv University, Tel-Aviv, Israel, and the D.Sc. degree in electrical engineering from the Technion-Israel Institute of Technology, Haifa, Israel, in 1978, 1981 and 1988, respectively.

He worked for 32 years at industry in R&D, engineering and managerial positions, for Intel Corporation from 1999 to 2009, and prior to that for Sagantec, microCAD, IBM, National Semiconductor and Israeli Aerospace Industry. He is presently an Associate Professor with the Engineering Faculty, Bar-Ilan University, Ramat-Gan, Israel, and an Associate Visiting Professor with the Electrical Engineering Faculty, Technion. His current interests include VLSI circuits and systems design optimization and combinatorial optimization.