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

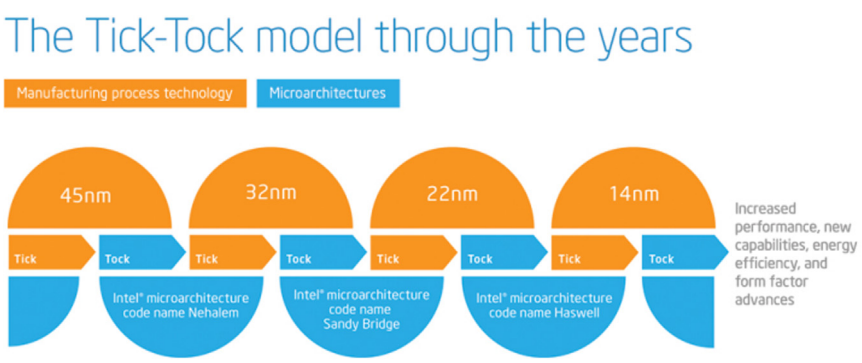


**Fig. 1.** "Tick-Tock" marketing strategy.

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 reuse, 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.