Microelectronic Engineering 148 (2015) 122–128

Contents lists available at ScienceDirect

Microelectronic Engineering

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

Physical design automation of transistor networks

Adriel Mota Ziesemer Jr [a,b], Ricardo Reis [a,*]

[a] Universidade Federal do Rio Grande do Sul, PPGC/PGMicro, Porto Alegre, Brazil
[b] Instituto Federal do Rio Grande do Sul (IFRS), Canoas, Brazil

ARTICLE INFO

Article history:
Received 16 July 2015
Received in revised form 28 October 2015
Accepted 28 October 2015
Available online 31 October 2015

Keywords:
Physical design
Transistor network
EDA
Cell synthesis
Layout
Nanoelectronics

ABSTRACT

Integrated circuits implemented with traditional standard cell approaches use a limited set of cells available in a library, created in advance, to generate its layout. It breaks complexity but frequently generates circuits with more transistors (due to the reduced numbers of functions and sizes available), more area, higher delays and more power consumption (mainly due to static power consumption, which is proportional to the number of transistors) than its potential. Many approaches have been attempted to improve this scenario at layout level: cell synthesis tools (to speed up the turnaround time of new cells), library-free layout synthesis and full custom layouts. We present in this paper a review of the methodologies and algorithms used in prior works for transistor-level layout synthesis, and especially recent ones targeting technologies beyond 65 nm.

1. Introduction

Standard cell based automatic synthesis flows have been used in the industry and academia for decades. This technique is known to be very reliable and predictable since the same library of standard cells can be used in several different designs. However, the number and type of cells available in the library can limit the quality of the designed circuits. A solution is the use of a transistor network design automation tool [58,1] that can allow the automatic layout generation of any transistor network with any transistor sizing. This allows the reduction of the needed number of transistors to realize a logic function. And the reduction on the number of transistors also reduces leakage power, wire length and area [58]. A large and diverse cell set can be determinant to efficient designs. There are many studies on special purpose cells for specific problems like: asynchronous circuits [6,30,55], aging [31], NBTI [32], leakage [33,53], gate sizing [34], and SET [53]. These cell layouts are generally not available in cell libraries and are usually designed by hand, which increases the design time and limits the adoption and development of promising technologies. Moreover, the adoption of a new fabrication process may require each of the cells to be redesigned almost from scratch. It can take an experienced leaf cell designer several days to craft a large cell, and this amount of time is increasing as design rules become more complex. In this scenario, synthesis tools can increase cell layout design productivity by one order of magnitude [31]. It is also useful to generate initial layouts (to be further optimized by hand), cell size estimations, and to evaluate key aspects of the technology (like the height of the cells) in an early stage of development. In comparison to quality manual designs, cell synthesis tools present between −6% [45] and 3.6% [1] of area overhead, on average. But the main issue is that using an automatic layout generation tool we can generate the layout of any transistor network. This means that we can do a deep logical minimization allowing a significant reduction in the amount of transistors needed to realize an integrated circuit [58].

Because implementing every possible logic function for each cell size is intractable, and the inefficiency of the mapping can negate the advantages of having customized logic cells [52], some works focus on a library-free approach. CellTK [6] (45 nm) creates custom cells on demand, while Punch [41,53] employs an automatic full-custom layout generation strategy. Both methods result in a one-to-one mapping between circuit description and physical design, eliminating the disadvantages of being tied to a fixed library. However, circuit characterization is only performed in late design stages (which can be hard computationally and unreliable) and DRC errors are significantly more critical. These characteristics resemble those of full custom layouts, but with orders of magnitude time savings over expert human effort. Despite this fact, actual implementations have shown an area overhead of as much as 50% higher [6], probably due to conservative layout strategies for producing legal designs.

This article presents a review of the transistor-level layout synthesis problem. Prior works related to this subject are presented and discussed.

2. Layout synthesis

Lefebvre [2] classified the layout generation methods in three categories: procedural generators, re-compaction of existing libraries and

* Corresponding author.
E-mail address: reis@inf.ufrgs.br (R. Reis).

http://dx.doi.org/10.1016/j.mee.2015.10.018
0167-9317/

automatic cell synthesis. Procedural generators can be based on a language or symbolic, and each cell has its own description. Recompaction, as used in [11,13,16,17], makes use of a generic tool to migrate existing cell layouts to a new fabrication technology or to improve its characteristics (as for yield optimization, OPC and DFM). Cell synthesis tools [1,4,6,11,14,28,42,45,53] generate layouts from a transistor-level netlist description (as SPICE). It is considered the most flexible method regarding the cell architecture, and is the only one to require no previous layout information. It is also better suited to deal with non-linear technology scaling and new design rules.

Cell synthesis usually consists of mapping a cell netlist with individually sized transistors into a circuit layout, considering the design rules of a target fabrication technology. The circuit layout is typically defined by a template that provides a general strategy to design libraries with compatible cells, to embed expert's knowledge into the synthesis process and to reduce its complexity. The main aspects of a template are defined by a layout style. Some of its parameters can be changed in order to increase flexibility and cover characteristics that differ between cell libraries (as cell height, routing pitch, well height, supply width, TAPs, etc). The result of the synthesis process is usually produced after the solution of these two major subproblems: transistor placement and intracell routing. Other common subproblems are transistor folding, layout compaction, ports and TAPs placement.

Following we present a compilation of the main aspects and methods used for layout synthesis.

3. Layout styles

Defines assumptions regarding how the layout should be produced. It is the most important aspect of the synthesis process because all the following algorithms are defined regarding this. Layout styles can be classified according to the position and orientation of the transistors inside the cell area (Fig. 1):

- 1-D — The most common in standard cell libraries, it was proposed first by Uehara and van Cleemput [3] and used in many other works including [1,4,5,6,21,28,45]. In this style, the transistors are drawn vertically in two horizontal diffusion rows: one for the P type and the other one for the N type transistors. It is better suited for complementary logic CMOS circuits since each P transistor can be paired to an N transistor, reducing the unused space inside the cell, but can support other logic families as well. A 1 1/2-D variation was proposed in [7] (and supported in [45]) which is flexible in the restriction of the transistor type in each row. This arrangement is better suited for non-dual circuits but has gotten little adoption until now.
- 2-D — Are better suited for analog cells, leaf cells in datapaths, large cells (with more than 30 transistors) and logic cells with non-dual transistor netlist. It usually allows transistors in different orientations (free-form layouts), as in [8], or to be placed in multiple rows (as an 1-D extension to support multiple height cells), as in [9,10,42,45].

Other things to be considered are: power structure, number and direction of the metal layers available for intracell routing, gridded design rules (GDRs), abutment scheme, position of the routing tracks, isolation transistors, multigate transistors, standard cell compatibility, regular layout fabric (prefabricated), TAPs, etc.

4. Transistor folding

Consists of breaking larger transistors into smaller parallel ones (called legs or fingers) in order to make it fit in the cell height, or in its respective P or N diffusion row. This arrangement allows diffusion sharing between its transistors, reducing its impact in area, but can also be optimized to maximize diffusion sharing with adjacent transistors. It can be done by choosing between breaking into an even or odd number of legs, as shown in Fig. 2. An exact method was proposed for the first time in [10] to treat this problem, using ILP (Integer Linear Programming). However, its applicability is restricted to static dual CMOS circuits and the legs are placed contiguously, without guarantee of a minimum-area layout. In [5], an area-optimal method without these restrictions was developed to cover new technologies with 1-D GDRs, where differently sized transistors cannot share diffusions. This design style is gaining ground for addressing the printability issues in subwavelength photolithography. While in most works the transistor-folding algorithm creates equal sized legs, as in [10,21], it is crucial to explore different sizes with GDRs [5].

Another option is to not address the folding problem directly, but instead, let the designer to define it manually before the cell synthesis process as in [4,6,45]. However, doing this step during the synthesis can increase convenience and achieve better optimization, especially when executed before (dynamic placement with static folding) [5,10,21,42] or along (dynamic placement and folding) the placement step [25]. When executed after, it usually achieve a lower quality result, as in the “2 legs” case of Fig. 2, where a diffusion break was inserted.

Transistors in series (called and-stacks) can be clustered to apply this method to the whole group simultaneously. It eliminates intermediate connections that are in the same electrical potential, reducing the total wire length, and also the number of diffusion-to-metal contacts (called straps). It allows the transistors to be placed closer together, as in [10], resulting in potentially smaller cell layouts with improved electrical properties. One side effect of this method is that the additional

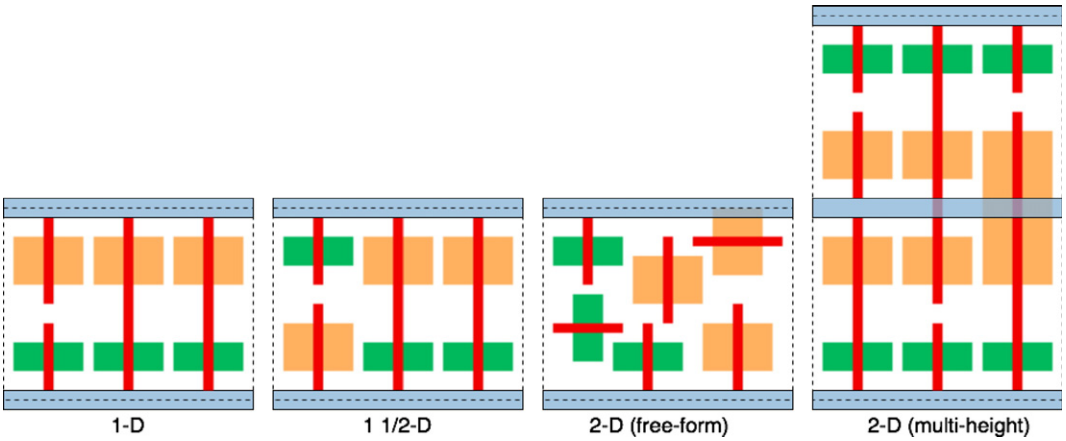

Fig. 1. Different types of layout styles.