An Educational Tool for Design Automation of CMOS Cells

Adriel Ziesemer, Cristiano Lazzari, Ricardo Reis
Instituto de Informatica
Universidade Federal do Rio Grande do Sul (UFRGS)
{amziesemerj, clazz, reis}@inf.ufrgs.br

Abstract

This paper presents a didactic tool that makes possible the automatic generation of full layouts of CMOS cells from its transistor level netlist in SPICE format. The tool allows the creation or modification of the netlist of a CMOS cell, including transistor sizing. It also let the user to quickly see the layout resulting after each modification. The tool does generate the cells in the linear matrix layout style and can automatically apply folding in the transistors when it is needed. Additionally, it is included a layout editor to allow the visualization of the generated layout.

1. Introduction

With the constant increase in the complexity of VLSI circuits, Computer Assisted Learning (CAL) tools that help the student to understand and to experiment all synthesis details are welcome. When looking the physical design steps, the student can experiment a Standard Cell method or a Full Custom approach. The first one don’t allow the student to experiment the whole set of variables and possibilities that we can have when designing a full layout. The use of a Full Custom approach is an excellent experience for a student, but as it takes a long time to test different transistor sizes, shapes and placement, the student don’t have time to experiment a nice set of possibilities. So, we believe that it is important to have also a tool to allow a fast and automatic full layout generation where the student can experiment a large set of options related to the layout. So, we present a new cell synthesis tool capable to produce cell layouts on the fly, from their transistor level netlist description.

2. The Cell Synthesis Tool

The tool can quickly create the layout of CMOS cells described as a transistor network using SPICE format and the generated layout is saved in the CIF format. The input is a file with the netlist of the cells that contains a list of cells, their input/output pins, the size of the transistors and how they are connected. Additionally, a configuration file with the generation parameters should be provided. This file defines the technology rules to be used, the desired height of the cells, the pitch size of the grid and the width of the supply signals.

The layout of the cells is always generated using a style similar to a linear matrix [1] style. The area of the cell is divided in two regions, one for the P transistors and other for the N transistors. There are intra-cell routing tracks between the diffusions. The input/output pins are placed over these tracks aligned to the grid. The body/well ties are placed under the supply rails.

The tool first read the netlist of the cells and executes a folding algorithm. This algorithm breaks the transistors with gates width larger than the gate of the standard transistor. The width of the standard transistor is defined by the width of the related diffusion strip. The folding modifies the netlist of the cell by creating two or more parallel transistor segments such that the sum of their sizes has the original transistor width (see Figure 1). After this, a transistor placement procedure is called to order the P and N transistors in their respectively diffusion regions. A simulated annealing based algorithm is used to perform this operation. The cost function aims to minimize the number of diffusion breaks and the routing complexity of the cell.

Figure 1. Example of a register automatically generated without folding (a) and with folding (b)
A graph representing the final position of the transistors and its internal nets are then created and routed. The algorithm used is a negotiation-based router similar to PathFinder [2]. At last, a layout compactor reads the routing result and creates a set of equations based on the technology rules used. A Linear Programming solver is executed to find a feasible solution that minimize some constrains - like the width of the cell - and return the values of the variables. These variables represent the positions of rectangles in each layer. The layout is then saved in the CIF format.

The cell generator is not restricted to the generation of cells with dual transistors. An example is shown in Figure 1 were a register was automatically generated by the tool. While in (a) the generator was configured to produce the cell with height of 8 times the pitch, in (b), the height was reduced to 7 times to force the algorithm to apply folding in the P transistors (in the bottom). It can also produce the layout of circuits without a complementary logic.

Large cells with more than thirty transistors were successfully generated.

3. Layout Visualization

To visualize the layout obtained by the synthesis tool, it is used a layout editor developed by our group called Ya-CIF. The editor was entirely developed using Tcl/Tk programming language what makes it easily portable to many platforms. It features most of the basic operations like: add, delete, select, resize and move rectangles, zoom, ruler, copy and paste, etc. These operations are enough to allows the student to visualize the layout and to make simple editions. The student can also use the editor to create their own layouts from scratch and compare to the automatically generated ones. Figure 2 show the interface of the editor and the layout of a transmission gate cell automatically generated by the tool.

4. Conclusion

This paper presented a tool set to allow VLSI students quickly create and visualize layout of CMOS cells. The student can experiment the constraints involved in the design process of a cell layout like design rules, transistor sizing, transistor placement, via placement, etc. The students can also use the editor features to create their own layouts. They can experience the design automation options and the full custom design, comparison the results and possibilities. This is a good exercise to learn about the constraints and possibilities that we have when designing a CMOS Cell.

REFERENCES
