Automatic Layout Synthesis with ASTRAN Applied to Asynchronous Cells

Adriel Ziesemer Jr.*, Ricardo Reis*, Matheus T. Moreira†, Michel E. Arendt†, Ney L. V. Calazans†
*Universidade Federal do Rio Grande do Sul (UFRGS), PGMicro
†Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), PPGCC
Email: {amziesemerj, reis}@inf.ufrgs.br, {matheus.moreira, ney.calazans}@pucrs.br, michel.arendt@acad.pucrs.br

***Abstract*—This work presents ASTRAN, a tool for automatic layout generation of cell libraries, and the use of this tool in the production of a cell library for asynchronous logic components called ASCEnD. In this context, ASTRAN is able to achieve orders of magnitude savings in cell generation time if compared to manual design. ASTRAN supports technologies down to 65nm and simultaneous two-dimensional cell layout compaction. It can deal with non-complementary logic cells, and allows producing any type of transistor network. The comparison of the generated layouts to those of the hand designed ASCEnD library revealed that ASTRAN achieves an average of 26% less area, about 50% less total parasitic capacitance and worst case input capacitance, and 23% lower delay.**

I. Introduction

Standard cell-based automatic synthesis of application specific integrated circuits (ASICs) has been used in industry and academia for a long time. This design method is very reliable and predictable since a same library of standard cells can be used in several different designs and mature tool flows are available to support. However, the quality of the designed circuits can be limited by the number and type of cells available in the library. A large and varied cell set is needed to enable efficient designs. Moreover, there are many studies on ways to optimize them for specific problems: asynchronous circuits, leakage, low-power, NBTI, etc. Standard cell libraries usually count with handmade layouts, which also limit the fast adoption of new technologies. Furthermore, each new fabrication process requires the redesign of all cells practically from scratch, if no effective CAD tools are available.

II. Overview

The academic netlist-to-layout tool ASTRAN [1] was developed to synthesize cell layouts in technologies down to 65nm. The tool generates the cells' layout under a linear (1-D) layout style and supports: unrestricted circuit structures, continuous transistor sizing, folding, poly and over-the-cell metal 1 routing, and tightening the spacing rules for DFM. ASTRAN features a transistor placement algorithm for width reduction and an intra-cell router. Mixed-Integer Linear Programming (MILP) is used for compaction; it produces the final layout according to the results provided by the placement and routing steps taking into account the technology design rules.

The input of ASTRAN is a transistor level description of a circuit in SPICE format. Each circuit, called ".subckt", can be synthesized into a standard cell-level layout. Technology

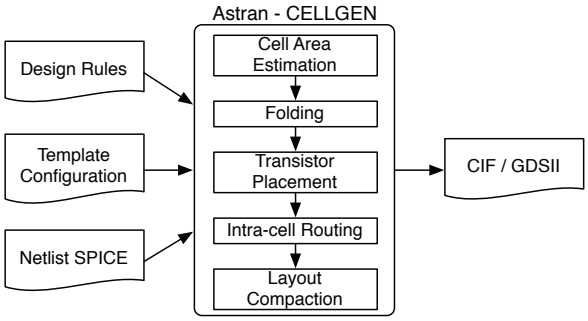

Fig. 1. ASTRAN cell synthesis flow.

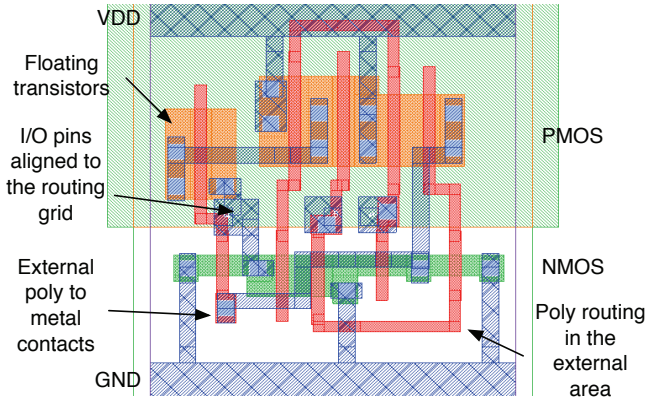

Fig. 2. ASTRAN cell example, showing typical elements of its layout style.

rules are set according to the values defined by the foundry. ASTRAN supports the rules defined by most CMOS processes down to 65nm. The cell topology (height, routing grid, wells/power rails position, and other library specific aspects) are defined according to the target library.

Figure 1 illustrates the flow employed by ASTRAN. The flow steps description is the subject of next sections. Given a network, the design flow objective is to place and route its transistors using the proposed layout style, to minimize cell width and interconnection length. ASTRAN then compacts the circuit abstract representation to produce an error-free layout in CIF or GDS2 formats.

III. Layout Style

The 1-D layout style consists of two rows of P and N transistors with vertical gates [2].

978-1-4799-2507-0/14/$31.00 ©2014 IEEE

TABLE I
OUR LAYOUT STYLE

1. Support to unrestricted transistor structure and individually sized transistors. No re-ordering occurs;
2. Transistors placed separately in horizontal rows for PMOS and NMOS;
3. Intra-cell routing with polysilicon, metal 1 and diffusion (stretching);
4. Contacts between polysilicon and metal 1 everywhere, except over transistors;
5. Transistors can be placed anywhere inside P and N regions for routing. They can move freely ("floating") during compaction;
6. Tracks over the P and N diffusions for metal routing;
7. Supply rows are in the top and bottom of the cell, resp.;
8. TAPs can be placed under the supply rows in the cell boundary, according to the cell template;
9. Input and output ports are aligned to the routing grid. They are also allowed over transistors;
10. Jogs (dog legs) may occur in metal and poly routing (this can be disabled);
11. Fixed cell height and n-well positions. Cell width is multiple of the routing grid horizontal pitch.

Table I lists ASTRAN layout style assumptions and Figure 2 illustrates these. To enhance routing flexibility, several techniques are available, including: diffusion stretching, poly routing and vertically moving transistors.

IV. CELL GENERATION FLOW

We refer here to the cell flow generation steps represented inside the round corner rectangle in the center of Figure 1.

A. Cell Area Estimation

Area of PMOS and NMOS transistors depend on the cell template, height, N-well position and design rules. The minimum pitch between metal lines define how to compute tracks position. Intra-cell routing requires reserving a minimum of one horizontal track between PMOS and NMOS transistors.

B. Folding

Transistor sizing is essential to produce high performance circuits. Layouts produced in the 1-D layout style with different sized transistors tend to waste area, since the height of each diffusion row is adjusted according to its tallest transistor. To solve this problem, one of the most used methods is transistor folding, which consists in building bigger transistors as a set of parallel interconnected rectangles. This keeps the cell height small, at the expense of an increase in cell width.

This works addresses the dynamic placement with the static folding problem. Given the diffusion rows height limits, transistor folding occurs by directly modifying the cell netlist, creating new transistors in parallel before executing the placement. This gives more freedom to the placement algorithm, and it can then achieve better results than folding transistors after placement.

C. Transistor Placement

The purpose of the placement step is to find out a transistor ordering that leads to a better design. Cellerity [3] was one of the first tools to successfully solve this problem considering not only diffusion gaps and gate mismatches, but also channel density, using simulated annealing. Other methods find the optimal transistor placement [4] but without considering channel occupation or wire length.

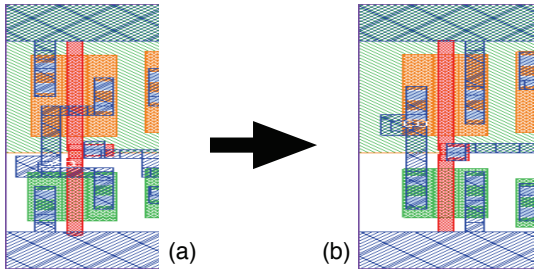

Fig. 3. Local routing density enhancement example for an inverter within a full adder. In (a) the local channel density is 0-1-2 (I/O pins are not evaluated) while in (b) the density is 1-1-1.

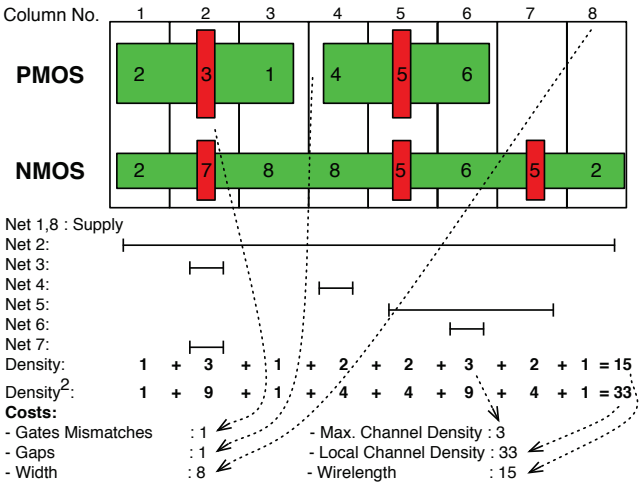

Fig. 4. Placement cost evaluation.

This paper's approach first implemented transistor placement with the *threshold accepting* heuristic from Hentschke [5] in the graphical CAD framework for physical synthesis ICPD [6]. It was possible to find optimal or near optimal results in most test cases.

ASTRAN improved the previous results by including the maximum channel density and local channel density in the cost function. The maximum channel density cost is important to reduce the number of horizontal connections in the most congested column, allowing for routing more complex cells. The local channel density cost helps flipping transistors, which reduces the number of connections that cross in each column, as Figure 3 shows. Therefore, its cost is evaluated using the square of column density values, as it is better to spread connection points than to have local congestion spots. Figure 4 presents all metrics evaluated during this step.

D. Intra-cell Routing

The transistor ordering resulting from placement translates to a routing graph that expresses the number and position of the tracks calculated in the area estimation step and the transistors width. This graph is illustrated in Figure 5. Nodes represent intersections of connections and edges are a con-

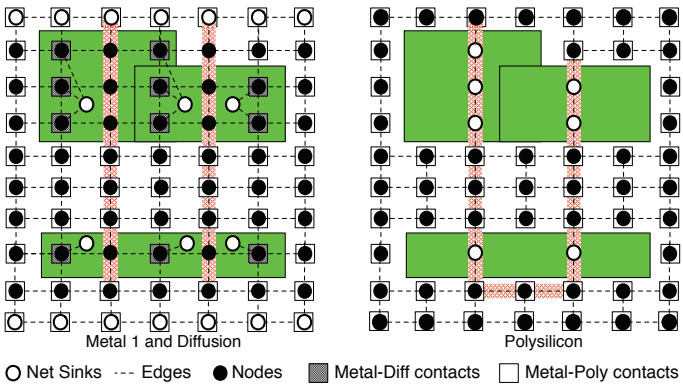

Fig. 5. Intra-cell routing graph.

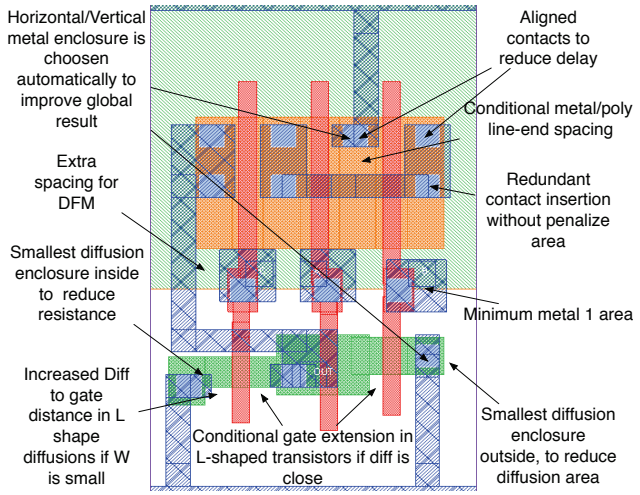

Fig. 6. Design rules supported by ASTRAN.

nection between such nodes (which translates to: metal/poly tracks, contacts or vias). Next, an optimisation step is executed to insert Steiner Nodes and improve the routing result.

E. Layout Compaction

This is the process of translating the abstract cell representation produced by previous steps into a cell layout. A previous tool [6] was able to generate layouts for processes up to 350nm and compaction was performed almost exclusively horizontally. ASTRAN manages not only to support 65nm (which has to deal with conditional rules), but also simultaneously compact the layout in two dimensions. This was possible by using mixed integer linear programming (MILP). Binary variables represent whether a rule should be applied or not. Using linear equations over these variables it is possible to model mutually exclusive and co-occurring rules. With the Gurobi tool [7] used as solver it was possible to find the optimal solution for most models in reasonable time.

Figure 6 shows the rules supported by ASTRAN and some of the constraints we try to optimize.

V. Case Study and Experimental Results

To evaluate the quality of layouts generated by ASTRAN, a subset of cells available in the ASCEnD library [8], [9]

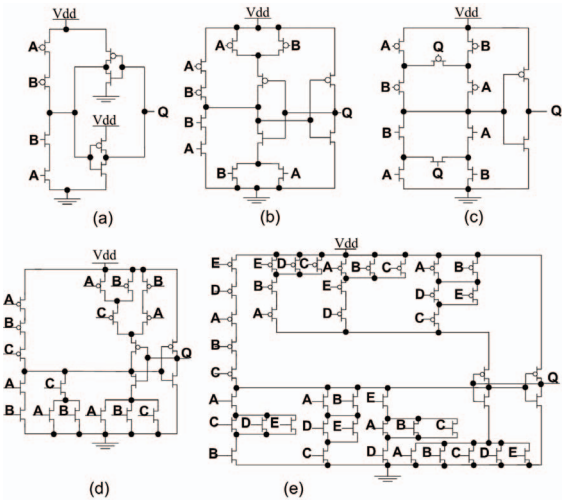

Fig. 7. Examples of asynchronous cells in ASCEnD: (a) MAC2, (b) SUC2, (c) VBC2, (d) NCL23 and (e) NCL35.

served as case study. This is a library that contains special gates used in the design of asynchronous circuits, like C-elements and NCL gates, which complements traditional logic gate libraries. ASCEnD targets the STMicroelectronics 65nm CMOS technology. It contains over 600 handmade cell layouts. This allows a fair comparison, of ASTRAN with handmade layouts.

The case study set includes 3 different 2-input C-element topologies, with two driving strengths available for each of these. Topologies are: Martin (MAC2), Sutherland (SUC2) and van Berkel (VBC2), and strengths are X4 and X18. The different degrees of complexity among the topologies motivated their choice as Figure 7 illustrates. Martin is the simplest, with only 8 transistors. Sutherland and van Berkel topologies require 12 transistors each, and the latter has a more intricate routing (see Figure 7(b)-(c)). Two Null Convention Logic (NCL) gates, NCL23 and NCL35 complete the case study set. These employ more transistors and have more complex topologies, which enables evaluating the quality of ASTRANs transistors placement and nodes routing. As Figure 7(d)-(e) shows, NCL23 has 20 transistors and NCL35, 44. Further details can be found in [8] and [10].

ASTRAN automatically generated layouts for all case study cells. Six characteristics allowed comparing handmade and ASTRAN-generated ASCEnD cells: total area, total parasitic capacitance, worst case input capacitance, average delay, average energy per transition (EPT) and average leakage. Total area was extracted from layouts. Layout extraction assuming worst case RC allowed measuring total parasitic capacitance and worst case input capacitance. The former corresponds to the sum of all parasitic capacitances and the latter is the worst case input capacitance from all cell inputs. SPICE simulation of the extracted netlists using ".measure" commands produced the remaining measurements. Simulations included all transition arcs for each input and output pair and exercising all static states for each cell. Simulation scenarios employed typical fabrication and operating condition models and all cells had

TABLE II
COMPARISON BETWEEN LAYOUTS GENERATED USING ASTRAN AND EQUIVALENT MANUALLY DESIGNED LAYOUTS FROM ASCEND.

	# T.	Run-time	Area (μm^2)		Tot. Parasitics (fF)		Worst In. Cap. (fF)		Avg. Delay (ps)		Avg. EPT (fJ)		Avg. Leakage (nW)	
		ASTRAN (s)	ASTRAN	ASCEnD	ASTRAN	ASCEnD	ASTRAN	ASCEnD	ASTRAN	ASCEnD	ASTRAN	ASCEnD	ASTRAN	ASCEnD
MAC2X4	8	4	3.64	4.68	1.0644	1.7913	0.1725	0.2756	36.16	38.81	4.67	5.07	39.00	39.00
MAC2X18	8	14	7.28	9.36	2.2195	3.3034	0.3064	0.5387	50.69	52.92	19.36	19.97	127.194	127.17
SUC2X4	12	12	6.76	6.24	1.6208	3.6566	0.3971	0.8298	64.09	71.82	5.47	5.95	32.79	32.90
SUC2X18	12	32	7.28	8.32	2.4449	3.9845	0.6842	0.8567	51.97	57.01	16.82	17.90	117	117.11
VBC2X4	12	40	6.76	7.28	2.9553	3.3623	0.4437	0.6034	47.32	48.85	5.56	5.70	35.84	35.78
VBC2X18	12	52	7.8	9.36	3.0028	4.0721	0.4677	0.6481	49.53	52.05	16.54	17.07	116.68	116.63
NCL23X4	20	50	8.84	11.96	3.5175	7.2529	0.6562	1.1929	72.38	88.22	3.76	4.47	32.13	32.70
NCL35X4	44	* 43200	17.16	31.2	9.4512	23.824	1.2526	2.8690	133.32	246.45	5.26	9.71	32.63	33.20

FO4 equivalent loads in their outputs. The average delay was measured as the average between the propagation delays measured for all arcs. Average EPT is the average between the energy consumed for switching the output, for each transition arc. Finally, average leakage is the average power measured during static states, as caused by leakage currents.

Table II summarizes the obtained results. It also presents transistor count and ASTRAN run-time for layout generation. The latter is the clock wall time using an Intel Xeon W3540 2.93GHz workstation. All cells were generated in less than 60s except NCL35X4, which was interrupted after 12 hours, resulting in sub-optimal layout compaction. This is due to the elevated transistor count of this cell (see Figure 8). In the authors experience low complexity gates (up to 20 transistors) requires a whole work day of a designer to produce up to two cell layouts. ASTRAN can generate such a layout in less than one minute. A more complex design, such as the example NCL gates, can take several days from a designer. As for the layout area, ASTRAN managed to generate more compact layouts than handmade layouts except for the SUC2X4 cell. For this cell the tool produced a layout with an area overhead of roughly 8%. However, in the other cases it provided area reductions ranging from 7% to 45%.

As Table II shows, layouts generated by ASTRAN presented lower parasitics and lower worst case input capacitances for all cells. For the former, reductions ranged from 12% to up to 60%. For the worst case input capacitance the range is from 20% to 56%. Reductions in parasitics of the cell are quite relevant, as they lead to improvements in performance. In fact, results point to slightly reduced average delays for all C-Elements and to substantial reductions for the NCL gates, roughly 46% for NCL35X4. This indicates the suitability of ASTRAN for coping with high complexity cells. EPT was also reduced in all cases, a consequence of the reduction of parasitics. In fact, in best and worst cases EPT reductions were of 46% and 2%. Finally, the average leakage was roughly the same for layouts generated with either ASTRAN or handmade ASCEnD. This is due to the fact that leakage currents are directly related to the transistor dimensions, which is the same for both cases, and it is minimally affected by layout structure. Finally, besides the reported gate-level optimizations, improvements are also expected at the circuit level, as the input capacitance is considerably smaller in the automatically generated cells.

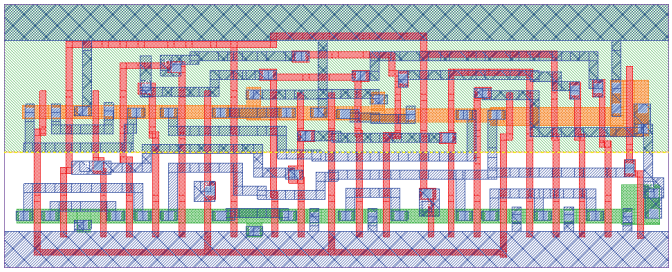

Fig. 8. Layout of the NCL35X4 cell generated by ASTRAN.

VI. CONCLUSIONS

We developed a transistor netlist to cell layout synthesis tool called ASTRAN. We compared it to manually designed cells from the original ASCEnD library. ASTRAN was able to produce cell layouts with less area, capacitance, delay and dynamic energy consumption in most cases. The authors believe that ASTRAN can significantly reduce the time-to-market of cell libraries and can also be useful for applications such as on-demand cell synthesis.

REFERENCES

[1] Astran - automatic synthesis of transistor networks. [Online]. Available: https://code.google.com/p/gme-ufrgs/

[2] T. Uehara and W. M. vanCleemput, "Optimal layout of cmos functional arrays," *IEEE Transactions on Computers*, vol. 30, no. 5, pp. 305–312, 1981.

[3] M. Guruswamy *et al.*, "CELLERITY: A fully automatic layout synthesis system for standard cell libraries," in *Proceedings of the 34th Design Automation Conference, DAC*. Anaheim, California, United States: New York: ACM, 1997, pp. 327–332.

[4] T. Iizuka, "Optimal layout synthesis of standard cells in large scale integration," Ph.D. dissertation, Department of Electronic Engineering, Graduate School of Engineering, The University of Tokyo, Tokyo, Japo, 2007.

[5] R. Hentschke, "Algorithms for wire length improvement of vlsi circuits with concern to critical paths," Ph.D. dissertation, PPGC, UFRGS, Porto Alegre, 2007.

[6] A. Ziesemer, C. Lazzari, and R. Reis, "Transistor level automatic layout generator for non-complementary cmos cells," in *Very Large Scale Integration, 2007. VLSI - SoC 2007. IFIP International Conference on*. USA: IEEE, oct. 2007, pp. 116 –121.

[7] Gurobi optimizer. [Online]. Available: http://www.gurobi.com/

[8] M. T. Moreira, B. Oliveira, J. Pontes, and N. Calazans, "A 65nm standard cell set and flow dedicated to automated asynchronous circuits design," in *IEEE International SOC Conference*. USA: IEEE, 2011, pp. 99–103.

[9] M. T. Moreira, C. Oliveira, R. Porto, and N. Calazans, "Design of ncl gates with the ascend flow," in *Latin American Symposium on Circuits and Systems*. USA: IEEE, 2013.

[10] M. T. Moreira, B. Oliveira, F. Moraes, and N. Calazans, "Impact of c-elements in asynchronous circuits," in *International Symposium on Quality Electronic Design*. USA: IEEE, 2012, pp. 437–441.