A Transistor-Level Placement Tool for Asynchronous Circuits

M. Salehi, H. Pedram, M. Saheb Zamani, M. Naderi, N. Araghi
Department of Computer Engineering, Amirkabir University of Technology
424, Hafez Ave, Tehran 15875, Iran
{salehi, pedram, szamani, naderi, araghi}@ce.aut.ac.ir

Abstract – Although asynchronous circuits are accepted as low-power, low-EMI and high-performance circuits, the roadblock to wide acceptance of asynchronous design methodology is poor CAD support, especially physical design tool. There are few academic design tools for asynchronous circuit design and synthesis, but there is neither a published tool nor a published document on physical design of these circuits. Since there are non-complementary CMOS circuits in the netlist synthesized using Caltech synthesis method, the commercial cell-based placement tools can't be used. In this paper we have presented a design flow for placement of asynchronous circuits at transistor-level considering their timing constraints.

Keywords: Asynchronous Circuit, Physical Design, Placement and Isochronic Fork

1- Introduction

One of the common timing models used in asynchronous circuit design is the QDI (quasi-delay insensitive) timing model proposed by Caltech research group [1]. In this model the circuit described in CHP (Communicating Hardware Process) is compiled to a set of PRs (Production Rule). In this set each PR is a pull-up or pull-down network of either a complementary or a non-complementary CMOS circuit in which the pull-up and pull-down networks are not dual. After circuit synthesis to the PR set, an operator reduction algorithm, is used to compile this set to a set of standard operators such as NAND, NOR and C-ELEMENT and a smaller set of PRs that otherwise couldn't be mapped to complementary standard operators. Due to these non-complementary characteristics in asynchronous circuits designed with the QDI timing model a cell-based layout synthesis method couldn't be used for these circuits and a transistor-level placement algorithm is required.

Although transistor-level placement algorithms are complicated and time consuming but transistor-level optimizations indicate that synthesized layouts using these algorithms are denser in area in contrast to cell-based designed layouts [2]. Another advantage of transistor-level layout synthesis method is

the dynamic layout library generated in the layout synthesis process. Having a dynamic library static pre-designed libraries are not required. Relying on the static libraries of devices has the following drawbacks [3]:

1. Limited range of options; not all possibilities can be covered in a finite set of modules. For example, the library can include only several variants of a device, among all possibilities.

2. Process technology change causes redevelopment of the whole library of devices.

In this paper we have presented a placement tool at transistor level for QDI asynchronous circuits that supports, non-complementary CMOS layout design using a hierarchical 2-D row-based method. Hierarchical methods such as [4] partition the circuit into sub-circuits or clusters, thereby decomposing the layout problem into two independent set problems: (a) generating a layout for each cluster and (b) finding a placement of the cluster layouts so that the overall cell width is minimized. Non-hierarchical approaches are efficient for small CMOS cells, however for larger cells, their run time increase rapidly. In the 1-D layout style single rows of P- and N-type diffusions are arranged in parallel, unfortunately this linear arrangement model has not been well accepted in practical cell design [3]. The main reason is that the area is not minimized in many cases.

The key innovations of this work are: automatic transistor chaining, transistor folding and diffusion merging and then clustering the circuit according to the merged-diffusion MOS transistors, building the layout library dynamically, placement considering isochronic forks and routability, and technology independent layout synthesis supporting design rule changes to current process

Section 2 describes the isochronic fork constraints and our contribution for satisfying this constraint. Section 3 presents a brief description of the current layout synthesis tools and their limitations. Section 4 presents our proposed design flow and in Section 5 and 6 the results and conclusion are presented respectively.

2- Problem Definition

In asynchronous circuits, designed with the QDI timing model, there is no timing constraint on the circuit except isochronic fork constraints. Each isochronic fork is a fork in which the differences in delays between its two branches, is shorter than the delays of the operators to which the fork is an input. In this model, the forks that are not acknowledged in all branches on both positive and negative transitions are considered as isochronic fork. Therefore when a transition on a branch is acknowledged, it is guaranteed that the transition is sensed on all other branches. In Figure1, delays of branches 1 and 2 are d1 and d2, respectively. Suppose that branches 1 and 2 are only acknowledged on the positive and negative transitions, respectively. So it is required that:

$$|d1 - d2| < \min(g1\,\text{delay}, g2\,\text{delay})$$

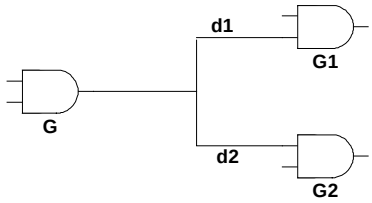


Figure1. Isochronic fork

In another case, assume that branch 1 is acknowledged on both transitions and branch 2 is just acknowledged on one of the transitions. In this situation, the

isochronic fork constraint can be simplified as:

$$d2 - d1 < \min(g1\,delay, g2\,delay)$$

Due to the considerable delay of interconnects compared to the gate delays in the deep submicron technology, isochronic fork constraint is becoming a critical point in the layout design of asynchronous circuits, and cannot be satisfied using a commercial synchronous layout synthesis tool in which there is no assumption for this constraint. In our design tool (PERSIA) in order to satisfy this constraint, isochronic fork is considered as a high weighted parameter in cost function of the placement.

3- Related Works

Many papers have been published in the area of transistor-level layout synthesis such as [2, 3, 4, 5, 6, 7], and most of them have focused on specific fundamental problems related to transistor placement, routing and compaction and have described layout synthesis systems developed mainly to demonstrate their specific innovation. Some of these design tools are designed for complementary CMOS circuits [4, 5, 6] hence cannot be used in QDI circuit layout design. In order to use existing design tools [2, 3, 7] in layout design of QDI circuits they must have the capability of distinguishing and considering isochronic forks.

In this paper we have presented a placement tool at transistor-level for QDI asynchronous circuits that supports non-complementary CMOS layout design, and satisfies the isochronic fork constraint.

4- Design Flow

In this section, we present our placement design flow from a transistor-level netlist to a design rule correct placed layout. The diagram in Figure2 is adopted as our top-level framework.

The inputs to our design tool consist of a process file, which contains a description of design rules and a netlist file containing a list of sized transistors and their interconnections. The design tool is process transparent and the process file is just updated for process changes. In the first step of our placement process, the input netlist is transformed into a cluster netlist. At this step, diffusion-merged transistor chains and clusters are identified, transistors with large widths are folded, transistor chain layout and then finally the layout of cluster is designed and added to the library. As a result of this part we have a dynamic layout library of dynamically diffusion-merged clusters and also a cluster netlist. This netlist is used in a simulated annealing-based placement algorithm [8]. In the reminder of this paper we discuss each step of our design flow in details.

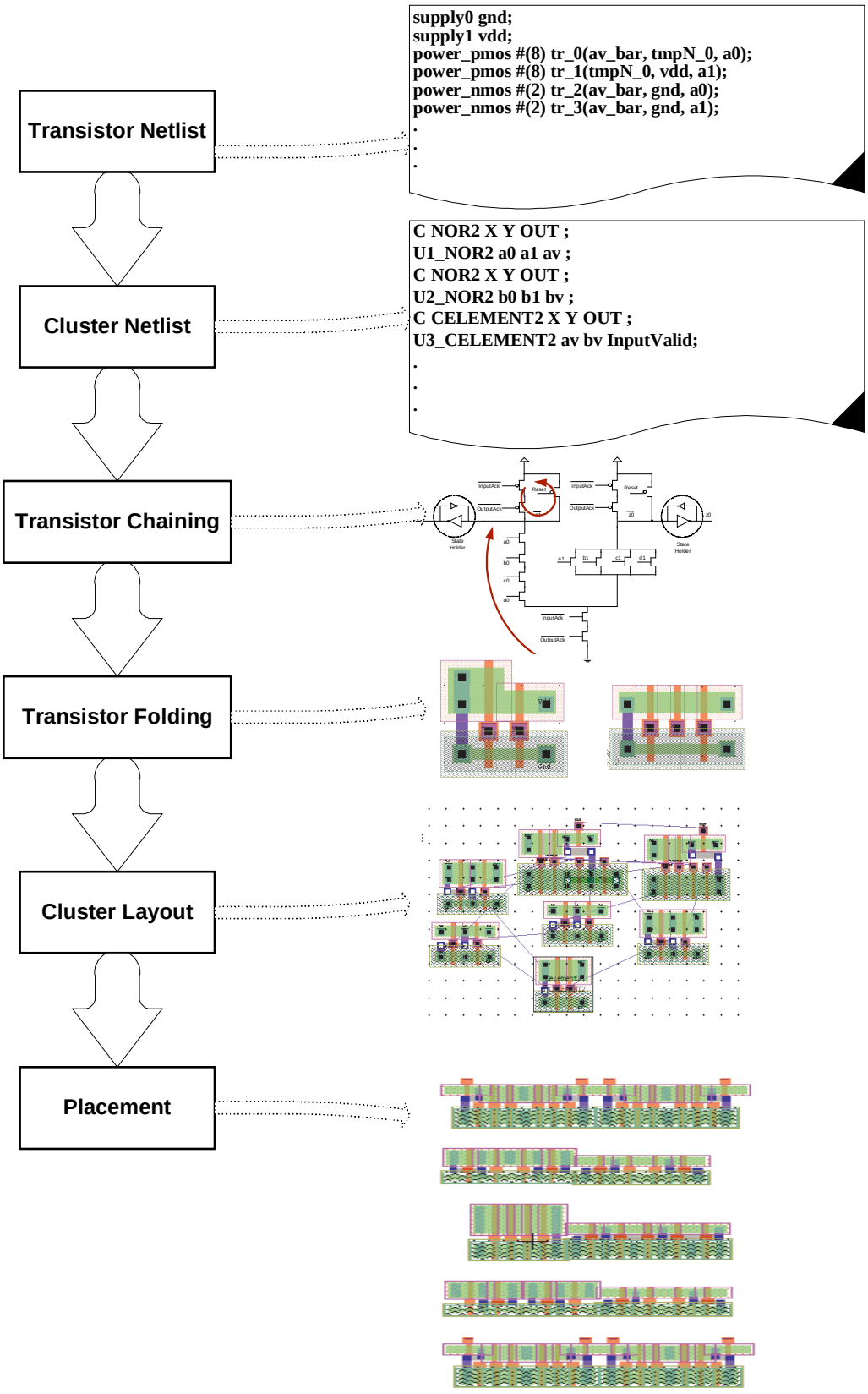


Figure2. Design flow

4.1. Clustering and Pattern Matching

This step, as the entry point of our design flow, focuses on reading the netlist and transforming it into clusters that are our building blocks from this point on. A cluster is a set of connected transistor chains that produces an intermediate signal of the circuit. Since non-hierarchical approaches are efficient for a small netlist, Clustering and hierarchical approaches are usual techniques that are used in most of the recent works [2, 4, 6, 9]. Using a hierarchical approach and clustering, transistor netlist is converted to a cluster netlist. As an advantage of this method when a cluster layout is once designed, layout design of the same clusters in the cluster netlist faced later on is avoided for, it is already added to the library, therefore an optimization in the algorithm run time. Due to the fact that there are considerable amount of similar complementary clusters synthesized from Caltech synthesis method such as (NAND, NOR, C-Element for completion detection circuits) the layout synthesis method is simplified to two stages:

1. Layout design of complementary clusters once, as soon as they are faced.
2. Layout design of non-complementary clusters whenever faced.

Clustering is done in three stages as follow:

- Transistor Chaining: In order to minimize the diffusion area of series-parallel MOS circuits, we have to find a Eulerian trail in the diffusion graph. Given a cluster netlist, a modified diffusion graph G is first generated. This graph incorporates the symmetry constraints among transistors. Next a trail cover on G, which satisfies the symmetry constraints in the circuit, is found. Two transistor chains are shown in Figure3.

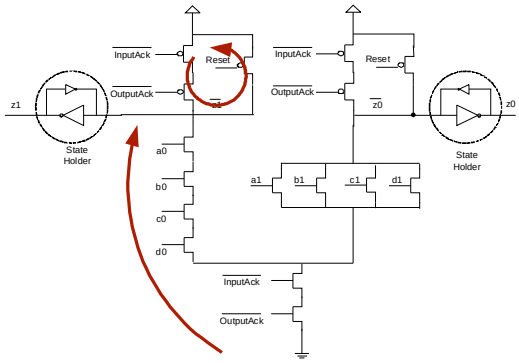
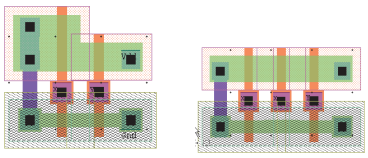


Figure3. Transistor chaining

The transistor chaining method used here is a variation of the technique used by Basaran [9].

- Transistor Folding: Given a maximum size for PMOS and NMOS transistors and transistor netlist of each cluster, according to method presented by Kim [10], cells are synthesized using different folding combinations with different transistor sizes to determine the minimum width cluster that meets the specified height. As a simple example, in Figure4 the PMOS transistor of height 12 is folded to two transistors of height 6, and therefore height minimization an area optimization about 100 λ^2.



(a) (b)

Figure4. Transistor Folding, (a) before folding, (b) after folding

- Layout design of the cluster according to the diffusion-merged folded transistors.

Area saving in diffusion-merge may seem negligible when a few transistors are considered. For example as indicated in Figure5 the optimized area between two minimum size merged MOS transistors is about $400\lambda^2$ ($16\mu m^2$ in 0.35μ technology)

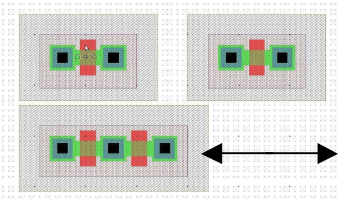


Figure5. Diffusion merging

However, when the transistor count reaches thousands of transistors, it will be a considerable area. In our design flow, in order to take advantage of this optimization, we are to identify transistor chains of shared source/drain diffusions. Then considering the size of each transistor, chains and cluster layouts are designed. According to the port positions, there may be a variety of layouts for a cluster that are all added to the library and used in the placement for intra-cluster optimization. In another word the positions of the ports are specified dynamically in the placement for intra-cluster optimization.

4.2. Global Placement

Using the cluster netlist and layout library from the pervious stage and based on a simulated annealing placement algorithm, we have generated a placement of the circuit considering a cost function containing maximum wire length, total area and isochronic fork constraint as cost parameters. Simulated annealing is probably the most well developed method available for module placement today. Although it is very time consuming but yields excellent results. This algorithm starts with a random placement and in each step using a perturb function either displace, mirror a module or interchange two modules with each other and according to the cost function accepts all moves that result in a reduction in cost. Moves that result in a cost increase are accepted with a probability that decreases with the increase in cost. A parameter T, called the temperature, is used to control the acceptance probability of the moves that result in cost increase. Higher values of T cause more such moves to be accepted. Such as most implementations of this algorithm our acceptance probability function is given by $\exp(-\Delta C/T)$ where ΔC is the cost increase. In the beginning, we set the temperature to a very high value so most of the moves are accepted. Then the temperature is gradually decreased so the cost increasing moves have less chance of being accepted. Ultimately the temperature reduces to a very low value so that only moves causing a cost reduction are accepted, and the algorithm converges to a low cost configuration.

Our cost function is a function of maximum wire-length, total area, overlap and row length control penalty, and the isochronic fork constrain.

5- Results

To compare the quality of the layouts generated by our design tool (PERSIA), we chose 5 manually handcrafted PCHB layouts as our benchmarks. These designs include a PCHB-AND2/AND4 (2/4-bit ANDs), PCHB-NOR2/NOR4 (2/4-bit NORs), and PCHB-BUF4 (4-bit buffer). All of these benchmarks have

been designed using PCHB-based design methodology [11, 12]. The first version of our design tool has been applied to these benchmarks. The Results are summarized in table1.

Table 1. Experimental Results

Cell	Transistors	Nets	Clusters	Manual Layout Area (λ^2)	PERSIA Layout Area (λ^2)	Improve
PCHB-NAND2	38	12	7	20336	21420	-5%
PCHB-NOR2	38	12	7	19795	20916	-5.6%
PCHB-NAND4	58	24	9	32965	33535	-1.7%
PCHB-NOR4	58	24	9	31278	31902	-2%
PCHB-BUF4	128	49	14	49144	50204	-2.1

Handcrafted layouts are hand-optimized and therefore 1.7% to 5.6% denser than automatic layouts designed with our design tool, but as it is shown when the number of transistors increases the automatic layout is comparable to the hand-optimized one.

6- Conclusion

In this paper we have presented a fully automatic transistor-level placement tool for QDI circuits. Layout synthesis style is based on a hierarchical 2-D row-based method. It is flexible to handle many process technologies, along with the capabilities of (a) satisfying isochronic fork constraints, (b) dynamic diffusion merging for cell width minimization and (c) dynamic transistor folding for cell height minimization. Although this design tool is designed for non-complementary QDI asynchronous circuits in which the isochronic fork constraint must be satisfied, it efficiently handles any netlist of transistors of any width and length, ignoring the isochronic fork constraint if not required.

References

[1] A. J. Martin, “Synthesis of Asynchronous VLSI Circuits”, CS-TR-93-28, 1991.

[2] P. Gopalakrishnan, R. A. Rutenbar, “Direct Transistor-Level Layout for Digital Blocks,” *ICCAD* 2001.

[3] T. Serdar , C. Sechen, "AKORD: Transistor Level and Mixed Transistor/Gate Level Placement Tool for Digital Data Paths", *International Conference on CAD*, Nov. 1999, pp. 91-97.

[4] A. Gupta and J. P. Hayes, “A Hierarchical Technique for Minimum-Width Layout of Two-Dimensional CMOS Cells” *Proc. Int’l Conf. on VLSI Design*, pp. 15-20, Jan. 1997.

[5] M. Guruswamy, Robert L. Maziasz, Daniel Dulitz, Srilata Raman, Venkat Chiluvuri, Andrea Fernandez, and Larry G. Jones, “CELLERITY: A Fully Automatic Layout Synthesis System for Standard Cell Libraries”, *Proceedings of 34th Design automation conference*, 1997.

[6] A. Gupta and J. P. Hayes, “CLIP: An Optimizing Layout Generator for Two-Dimensional CMOS Cells”, *Proceedings of 34th Design automation conference*, 1997.

[7] J. M. Cohn, D. J. Garrod, R. A. Rutenbar, L. R. Carley, "KOAN/ANAGRAMII: New Tools for Device-Level Analog Placement and Routing", *IEEE Journal of Solid-State Circuits*, vol.26, no.3, March 1991, pp. 330-342.

[8] N. sherwani, "*Algorithms for VLSI physical design Automation,"* 3rd edition, Kluwer Academic Publishers, 1999.

[9] B. Basaran and R. A. Rutenbar, "An O(n) Algorithm for Transistor Stacking with Performance constraints", *Proceedings of 33th Design automation conference*, 1996.

[10] J. Kim, S. M. Kang, "An Efficient Transistor Folding Algorithm for Row-Based CMOS Layout Design", *Proceedings of 34th Design automation conference*, 1997.

[11] A. Lines. Pipelined Asynchronous Circuits. *M.Sc. Thesis, California Institute of Technology*, June 1995, revised 1998

[12] K. Saleh, "Asynchronous Design Using Pre-Synthesized Templates", *Technical Report*, Amirkabir University of Technology, September 2003.