5th International Conference on Control Engineering&Information Technology (CEIT-2017)
Proceeding of Engineering and Technology –PET
Vol.34 pp.13-16

Routability-Driven Timing optimization

Safia HAKIM*, Mr Hassan BERBIA**
* Mentor Graphics Corporation/IC Design Solutions Division – Rabat, Morocco *
** ENSIAS/ Al Jazari Team, University Mohammed V – Rabat, Morocco
E-mail safia_hakim@mentor.com
h_berbia@yahoo.com

***Abstract*—The use of post-route optimization on a detailed routed design is an effective way to improve the timing without harshly impacting the routability of the design. With the increase of design complexity, one of the major challenges faced with newer technologies in VLSI (very large scale integration) design is routability. It can be difficult to successfully route a design during routing stage, and it becomes even harder to reroute successfully and with no violations (timing and physical) after disturbing it during post-route.**

In this paper, we reduce the optimization impact on the routing topology during the post-route flow stage. This is done by using fine-grained interleaved calls from optimization to detail routing with design rule awareness, instead of coarse iterations between routing and optimization after many targets have been optimized.

Keywords— *Vlsi; Asics;Routing; Optimization; DRC; LVS*

Introduction

With today's newer fabrication technology improvements following Moore's law, integrated circuits (ICs) are continually shrinking [1], creating many new challenges for the place and route flow.

We face a lot of issues in converging during the place and route flow. Each flow stage impacts downstream steps in the flow, making later stages like routing and post-route more challenging, it is hard to successfully route a design during the routing stage, and it has become even harder to reroute after disturbing it during the post-route stage. That's why we have to control and minimize the optimization disturbance.

Traditionally, post-route optimizations would be limited to footprint compatible cell swaps, including the restriction of the same pin locations to avoid impacting routing, or on-route buffering to minimize rerouting [2]. However in today's technologies with more complicated physical design rules, additional metal layers, and so forth, it is difficult to achieve accurate estimates before detail routing. This is exacerbated by problems predicting capacitive coupling. This necessitates further post-route optimization causing larger perturbations, making design closure difficult.

Due to the complexity of Application Specific Integrated Circuit (**ASIC)** designs, and the very large numbers of interconnections between logic cells in them, the routing is performed in two stages: global routing, followed by detail routing. Global routing is done to provide an estimate of wire capacitance and resistance for timing analysis earlier in the design flow. Global routing congestion needs to be accounted for and can be mitigated. The detailed routing is the task of assigning rectilinear wiring interconnects – the actual metal shapes – to the nets [3]. In this paper, we address the detailed routing, especially when used as an ECO (engineering change order) router after post-route optimization.

The remainder of this work is organized as follows. Section 2 presents some basic concepts of routing and optimization, and it describes how the flow engine works. Section 3 explains the difference between the existing approach and our new approach. Section 4 provides a case study where we show the benefits of merging the routing and optimization engines. Finally, Section 5 draws the conclusion.

I. Terms and Definitions:

A simplified view of the ASIC physical implementation flow consists of the steps shown in Figure 1.

In this paper, we address the problem of rerouting the designs after appropriate post-route optimization changes have been performed. In the post-route optimization stage, we address timing violations exposed by more accurate wiring parasitic extraction, after detailed routing (physical layer correct extraction, vias, etc.).

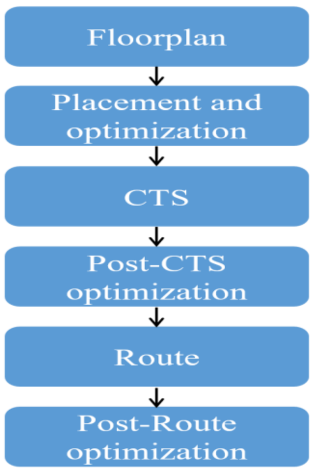

Fig. 1. Place and route flow

ISSN 2356-5608

A. Post-route optimization:

The post-route optimization (PRO) operates on detail routed designs. The PRO engine is designed to minimize disturbances in the existing wires and quickly bring the design to closure. PRO fixes both setup and hold timing violations.

Cells are swapped and sized with other cells that have equivalent footprint and logic. PRO takes advantage of wiring information to try to insert buffers "on-route" to minimize the disturbance to existing routing.

After PRO, any open nets need to be routed, such as between connected pins of newly placed cells that need a wire routed between them, or for portions of a net that have been removed due to deletion of buffers. In some cases, a net may need to be completely rerouted. In addition, any physical design rule violation (DRV) must be fixed. Thus the need to perform detail routing after optimization.

B. Detailed routing :

After completion of standard cell placement and optimization, the next phase is to detail route the ASIC design. Then extraction of routing and parasitic parameters is done for the purpose of static timing analysis.

The task for detail routing is to connect the pins of each net by wires and vias. The wires of different nets must be disjoint, that is, respective wires of different nets must satisfy certain minimum distance constraints. Additionally, wiring layout is limited by constraints specified in the design rules [4]. To make sure the design is clean after routing, layout versus schematic (LVS) and design rule checking (DRC) are performed.

C. Physical Verification:

Design rules checks (DRC):

Design rule checks are physical checks of metal width, pitch, and spacing requirements for the different layers. These rules depend on the technology. We need to clean up the DRC to ensure correct logical connections of various components, otherwise the fabricated chip may be functionally incorrect [5].

The most common design rules used for IC layout are [6]:
- Minimum spacing
- Minimum width
- Minimum edge length
- Encapsulation of layer
- Enclosure violation

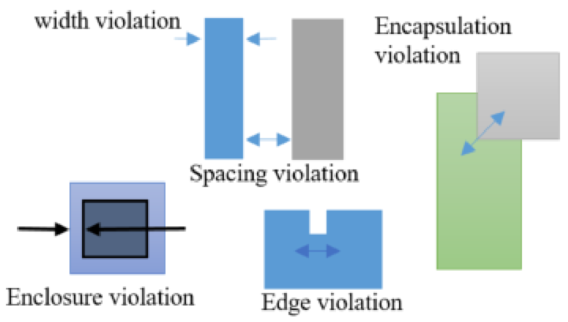

Fig. 2. Example showing design rules violations

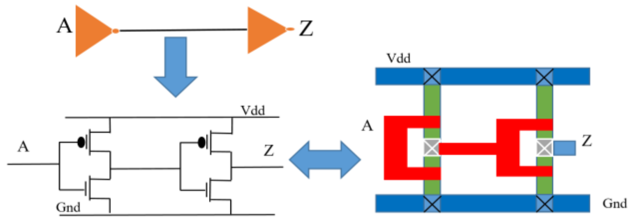

Fig. 3. Example showing Layout Versus schematic matching

Layout versus schematic (LVS):

LVS is another major check in the physical verification stage. It verifies that the layout you have created is functionally the same as the schematic/netlist of the design, i.e., you have correctly transferred into geometries your intent while creating the design [7].

Some of the LVS errors are:
- Shorts: wires that should not be connected are overlapping.
- Opens: connections are not complete for certain nets.

II. ROUTING AND OPTIMIZATION MERGE (CASE STUDY)

In this section, we describe the routability driven optimization problem and our approach.

A. Problem formulation

The optimizer doesn't update detail routing under the hood, so the routing topology can become inaccurate during optimization. Global routing (GR) estimates are used, which can be inaccurate. Figure 4 illustrates this problem.

While performing timing optimization, the optimizer adds, resizes, and removes cells. This includes removal, replacement, and breaking of some connections, creating DRVs (open net, disconnected wires, etc.) which require detail routing to fix. Also while performing routing, the router takes some detours and moves some wire, along with routing new nets added by the optimizer. This disturbs the timing, requiring another round of optimization.

We stop the back and forth between the router and the optimizer once we clean all DRVs and meet timing requirements. The DRV and the timing violations must be fixed for the design to function correctly. These iterations increase the flow runtime, and make design closure difficult, especially for newer technology nodes.

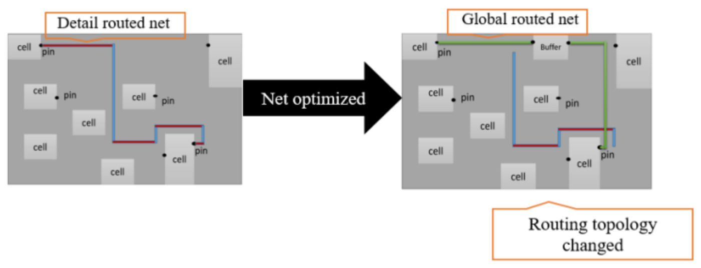

Fig. 4. Example showing the current behvior. Since the optimizer calls the GR server under the hood, the new buffered net is now globally routed.

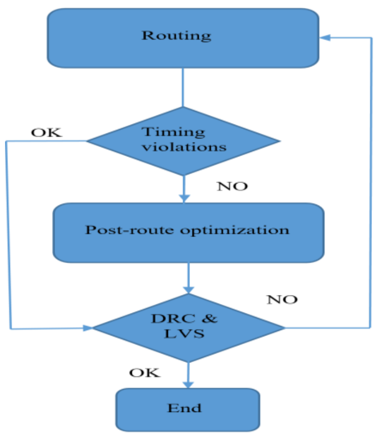

Fig. 5. Back and forth between router and optimizer for design closure.

B. Our approch

The objective is to preserve the routing topology, by calling the ECO router after each optimization target, instead of calling the routing engine on the whole design after the optimization engine has finished optimizing many targets.

The routing changes needed after optimization are just a part of the routing algorithm. So instead of calling each engine separately, increasing run time, we suggest making the post-route optimizer directly call the needed router function.

The basic idea is to make the optimizer aware of physical DRV by modeling some of the design rules to preserve routability along with improving timing. The purpose is to make the optimization engine use those models during the optimization process for better design closure.

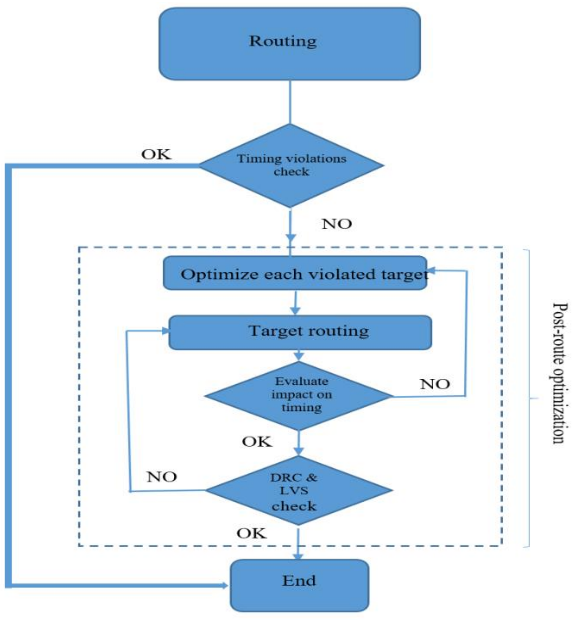

Fig. 6. New optimization approach to avoid iterations between the router and the optimization engine.

Algorithm 1: default Approach

```
Input:
 Set optimization mode and objectives.
 Get list of pins to optimize from the bottleneck engine.
 Get congestion map from the GR server.
 Get arrival and required times from the timer.
Output:
 Improvement of a specified objective for a specific mode
Optimization:
foreach pin [pin_to_optimize]
     foreach mode [optimization mode]
            foreach transform [optimization transform list]
                 Apply the chosen transform on the pin
           end
      end
end
Routing:
Call detailed router on the whole design
```

Fig. 7. Existing post-route algorithm.

Algorithm 2: New Approach

```
Input:
 Set optimization mode and objectives.
 Get list of pins to optimize from the bottleneck engine.
 Get congestion map from the GR server.
 Get arrival and required times from the timer.
Output:
 Improvement of a specified objective for a specific mode
Optimization:
foreach pin [pin_to_optimize]
        foreach mode [optimization mode]
                foreach transform [optimization transform list]
                    Apply the chosen transform on the pin
                    Detail route the change as needed
               end
        end
end
```

Fig. 8. New post-route optimization approch Algorithm Tcl wise

C. Algorithms:

Our Algorithm is not yet implemented at the C++ level in the Mentor Graphics Nitro-SoC^{TM} place-and-route tool. To prove the benefits of our approach, we present our algorithm (Algorithm2) in a way that can be implemented in the Tcl programming language. All our experiments were performed using Nitro-SoC.

Figure 7 presents the default algorithm (Algorithm 1) where the optimizer and the router are called separately. Figure 8 shows the new Algorithm (Algorithm2) where the two engines are tightly interleaved with each other.

III. Experimental Results

To evaluate our proposed approach, we conducted experiments on three detail routed designs, detailed in Table 1.

Circuit	#Cells	#nets	#pins	Utilization	#DRC	#LVS	slack
Circuit 1	443679	729722	3046185	72.19 %	632	514	-193ns
Circuit 2	2815	5343	18109	36.82 %	32	0	-1311ns
Circuit 3	57654	57518	283414	62.54%	0	68	-155ns

Table .1.Experimental Designs Caracteristiques

We applied Algorithm 1 and Algorithm 2 on our design. In Table 2, we report the DRC and LVS violations, the total negative slack for the timing violations, and design utilization.

Circuits	Algorithm 1				Algorithm2			
	#DRC	#LVS	slack	Utilization	#DRC	#LVS	slack	Utilization
Circuit 1	715	453	-173ns	72.24 %	708	452	-171ns	72.24%
Circuit 2	39	0	-1309ns	36.84%	30	0	1310ns	36.82 %
Circuit 3	0	83	-103ns	64.74%	0	74	-103ns	64.15%

Table .2. DRC and LVS comparison between default and new approach.

Using initial values from Table 1 and final values from Table 2 we calculated the improvement:

Circuits	Algorithm 1		Algorithm 2		Gain DRC	Gain LVS
	DRC	LVS	DRC	LVS		
Circuit 1	13.1%	-11.9%	12.0%	-12.1%	1.1%	0.2%
Circuit 2	21.0%	0.0%	-6.2%	0.0%	27.2%	0.0%
Circuit 3	0.0%	22.0%	0.0%	8.9%	0.0%	13.1%

Table .3. DRC and LVS comparison between default and new approach.

The reduction in DRC violations is up to 27.2%, and the reduction in LVS violations is up to 13.1%. This is a very encouraging improvement, especially in such a competitive domain where even the smallest gain can make a difference.

IV. Conclusion

The objective of this work is to prove the need for a smart engine combining both optimization and some of the routing features. We show that by adopting the new approach, the improvement was up to a 27.2% reduction in DRC violations and 13.1% reduction in LVS violations. This motivates implementing this approach in C++.

The C++ implementation will provide more accurate results. It will help us avoid the overheads for a separate optimizer call on each target, which was not possible in Tcl, thus saving runtime.

Acknowledgments

This paper was supported by Mentor Graphics Corporation. We thank our colleagues from the IC Design Solutions division, who provided insight and expertise that greatly assisted the research.

We thank Dr. Hazem El Tahawy (Mentor Graphics, Managing Director MENA Region) for supporting this work. From the Place-and-Route Solutions group in the ICDS division, we thank David Chinnery (Architect, Optimization), Sarvesh Bhardwaj (Group Architect, Optimization), for help and guidance through this research; and Jay Yanamandala (Architect, Product Validation) for the opportunity to work on this topic.

References

[1] Hiroshi Iwai, “End of the scaling theory and Moore's law”, International Workshop on Junction Technology (IWJT), 2016.

[2] David Chinnery, Leon Stok, David Hathaway, and Kurt Keutzer, “Design Flows”, chapter 1 in EDA for IC Implementation, Circuit Design, and Process Technology, Second Edition, CRC Press, 2016.

[3] Tim Nieberg, “Gridless pin access in detailed routing”, Design Automation Conference, 2011.

[4] Dian Zhou, Modern ASIC Design, Science Press, 2011.

[5] Vipul Patel, Design Rule Checks (DRC) - A Practical View for 28nm Technology, Design & Reuse, 2017.
http://www.design-reuse.com/articles/41504/design-rule-checks-drc-a-practical-view-for-28nm-technology.html

[6] Sini Mukundan, Physical Design Flow V: Physical Verification, VLSI Pro. http://vlsi.pro/physical-design-flow-v-physical-verification/

[7] Jeremy Espenshade, and Michael Romero, “CUDA Independent Study Final Paper”, Halogenica, 2008. http://halogenica.net/wp-content/uploads/2010/12/CUDA_DRC_Paper.pdf