Understanding and Addressing the Impact of Wiring Congestion During Technology Mapping

Davide Pandini†, Lawrence T. Pileggi‡, and Andrzej J. Strojwas‡
†STMicroelectronics, Central R&D, 20041 Agrate Brianza, Italy
‡Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, PA 15213 U.S.A.
email: davide.pandini@st.com, pileggi@ece.cmu.edu, ajs@ece.cmu.edu

Abstract*

Traditionally, interconnect effects are taken into account during logic synthesis via wireload models, but their ineffectiveness for DSM technologies has been demonstrated and various physical synthesis approaches have been spawned to address the problem. Of particular interest is that logic block size is no longer dictated exclusively by total cell area, yet synthesis optimization objectives are aimed specifically at minimizing the number and size of cells. Methodologies that incorporate congestion within the logic synthesis objective function have been proposed in [9][10][11] and [15]; however, as we will demonstrate, predicting the true congestion prior to layout is not possible, and the efficacy of any approach can only be evaluated after routing is completed within the fixed die size. In this paper we propose a practical, complete methodology which first performs congestion-aware technology mapping using a global weighting factor for the cost function [15], and then applies incremental localized unmapping and remapping on congested areas. This complete approach addresses the problem that one global factor is not ideally suited for all regions of the designs. Most importantly, through the application of this methodology to industrial examples we will show that any attempt at a purely top-down single-pass congestion-aware technology mapping is merely wishful thinking.

Categories and Subject Descriptors

B.6.3 [Logic Design]: Design Aids - Automatic synthesis.
B.7.2 [Integrated Circuits]: Design Aids - Layout, placement and routing.

General Terms

Algorithms, Performance, Design.

1 Introduction

It is well known that the physical interconnections between logic gates can have a dominant impact on the performance of modern DSM System-on-Chip (SoC) designs. For technologies of 0.25µm and below, wire capacitance dominates gate capacitance, thereby rapidly increasing the interconnect induced component of delay (as a percentage of the overall path delay) [1]. These interconnect effects, therefore, have to be carefully evaluated during all phases of a top-down ASIC design flow which, broadly speaking, consists of: logic synthesis and physical design (placement and routing). Prior to physical design, the interconnect models traditionally used in performance-driven tools are the wireloads. Wireload models are generated via statistics as a function of fanout loading and predefined net configurations. Although in [2] the analyses predicted that for block sizes within 50k gates, the typical DSM interconnect effects were not going to dominate performance for the foreseeable future, other researchers have shown contradictory results for the inaccuracy of wireload models at such block sizes [3][4]. While the problems of timing convergence due to these inaccuracies are well studied, of equal importance is the impact of wiring on defining the area of a block or IC, in particular for logic synthesis, which traditionally has attempted to minimize cell area in order to minimize block and chip area. Traditionally, optimization targeted towards the total cell area for a semi-custom design does not consider the interconnect area. However, since the DSM wirelength trends do not scale with feature sizes, cell area minimization can no longer guarantee overall block or chip area minimization. As a result, although the total gate area of the synthesized netlist can be made smaller than the block area, the design may not be routable within the assigned block.

In physical design, the required routing resources are captured in terms of the wiring congestion. During routing the cells are interconnected while trying to avoid the *congested* regions. This can often result in long wiring detours and increased overall net wirelength and delay. Placement and routing can sometimes fix, or avoid, potential congestion problems; however, attempting to solve such problems at the physical design stage is often too late. Ideally, area minimization which includes the impact of wiring via congestion models would occur during logic synthesis. But since there is generally very little or no physical information available at the synthesis stage of the design flow, commercial tools have attempted to combine logic synthesis and placement in an iterative procedure, or apply resynthesis during physical design. While these approaches can be effective, they do not completely address the impact of early synthesis decisions on the overall design routability, and without evaluating the necessary wiring resources, it is difficult to say anything about the true optimality of the synthesis results.

Since modeling congestion is more of an art than a science, and is a challenging problem even during physical design, reliably predicting the congestion during the synthesis phase of a top-down design flow is impractical without some form of iteration loops. However, we have demonstrated that some notion of congestion can be effectively incorporated into the synthesis cost function by systematically generating a population of synthesized netlists [15]. The results clearly show that congestion minimization during synthesis must be handled carefully in order to control the cell area penalty. Even so, such a methodology does not guarantee a completely congestion free design for a fixed block size, and local congestion regions can exist. Moreover, the true routability can only be determined after detailed routing is completed. Using the result with the best overall routability and area trade-offs, we apply a similar methodology to localized regions in order to incrementally resynthesize and eliminate congestion. It should be noted that some of

* This work was supported by the Central R&D of STMicroelectronics.

ISPD'02, April 7-10, 2002, San Diego, California, USA.

this congestion may be due to the modified original cost function which produced the best *overall* routability, yet created localized congested regions. Following detailed routing, the local routing violations are identified and eliminated based on technology remapping and incremental placement. Most importantly, our results demonstrate that any fully predictive approach is going to fail in certain cases due to the difficulty in predicting congestion and routability at the synthesis stage.

This paper is organized as follows. In Section 2, previous work in layout-driven synthesis is overviewed. In Section 3, our methodology for congestion-aware logic synthesis is summarized. In Section 4, our incremental remapping and placement flow for congestion minimization is described, followed by Section 5, where results showing its effectiveness are discussed in details. Finally, Section 6 presents our conclusions.

2 Previous Work

Significant progress, both in academia and industry, has been made to combine logic synthesis and physical design. In [7], Pedram and Bhat integrated technology mapping with a companion placement to include wiring contribution in area and delay minimization. In the work presented in [8], technology mapping was performed concurrently with linear placement to generate trade-offs curves for area (and/or delay) minimization.

Various commercial tools iteratively integrate synthesis and layout to avoid the dependency on a wireload model or similar statistical approximation. Since these commercial tools must route the logic blocks, in general they attempt to address the impact of interconnects on both the delay *and* the block area. There have been other research results that describe attempts at modeling routability during: the technology independent phase of synthesis, the logic extraction stage, by using topological information such as the fanout range [10], or via a companion placement of the logic network nodes [9]. However, placement of the logic nodes (which represent the Boolean equations) does not correlate well with the physical placement of a mapped gate level netlist, and congestion is mostly considered only during physical design, either at the global placement level or during incremental placement updates within resynthesis procedures [12][13][14]. More recently, an approach which addresses the congestion problem during logic synthesis was presented in [11], where congestion was taken into account by means of the physical coordinates obtained from placing the technology independent netlist. However, this approach does not consider the global and local nature of congestion, and more importantly, it does not consider the suboptimality in terms of area (and/or delay) that results from introducing congestion into the synthesis optimization objectives. In this paper we will demonstrate that different layout regions can have very different routing demands. *Importantly, our results will also demonstrate that congestion across a chip cannot be effectively captured by means of a single number as in [11].*

3 Congestion-Aware Logic Synthesis

The objective of logic synthesis is generally to produce a circuit which implements a set of logic equations, occupies minimum Si area, and satisfies performance constraints such as timing and power consumption. Due to the size of industrial circuits and the complexity of the optimization problems, modern logic synthesis is divided into two phases: technology independent optimization and technology dependent optimization (technology mapping).

The negative effects on routability of excessive efforts on area minimization during logic synthesis have been discussed in [15]. In the traditional design flow, however, no physical information is available at the synthesis level to estimate the impact of congestion. It is well known that the changes which can be performed by layout tools are limited, and might not be able to solve hard routability and/or timing problems without introducing more routing resources and consequently increasing the total logic block area. Decisions regarding the structure of the gate level netlist and how it is mapped from a technology independent to technology dependent netlist can impact the global wirelength and routability. The methodology proposed in [15] can effectively reduce the global wirelength, thus significantly improving congestion across the chip. As a consequence, the total block area is also decreased due to the reduction in wiring demands.

The other important factor which must be taken into account during synthesis is timing. Congestion minimization has beneficial effects on delay, since by avoiding long detours around congested areas, the total wirelength is decreased. In [15] static timing analysis demonstrated the improvement on delay obtained by reducing congestion during synthesis.

3.1 Congestion-Aware Technology Mapping

An efficient solution to the technology mapping problem was implemented in DAGON [5] and MIS [6]. The main idea is to reduce technology mapping to DAG covering. Since the DAG covering problem was shown to be NP-complete, it was approximated by a sequence of tree coverings, which can be solved optimally using dynamic programming. Since the work in [5] and [6], technology mapping is usually divided into three phases: DAG partitioning, matching, and covering.

A novel approach to placement-driven DAG partitioning and tree covering for congestion minimization was presented in [15]. Firstly, the technology independent netlist is placed in order to capture the connectivity on a chip layout image. Secondly, the network DAG is partitioned using the placement coordinates, and vertices placed in the same neighborhood are clustered in the same subject tree. Finally, the tree covering algorithm is significantly extended in order to include congestion minimization into the optimization objectives. Extensive efforts in gate area minimization yield high fanin gates, whose detrimental effect on routability stems from the increased interconnection length when the fanins of such gates are placed far apart. Therefore, in order to reduce interconnection length, fanin gates must be placed near their fanout. In [15], this target is achieved by including in the technology mapping objective function an additional term which represents the interconnect cost of a match. The wire cost is combined with the other optimization objectives (area and/or delay) into a single cost function by means of the congestion minimization factor K (for simplicity we consider here only area minimization):

$$COST(m, v) = AREA(m, v) + K \cdot WIRE(m, v), \qquad (1)$$

where $AREA(m, v)$ is the area cost of the match m at vertex v, and $WIRE(m, v)$ is the interconnect cost, which includes the wire cost of the match fanins and of their children, but does not include the wire cost of the transitive fanins [15]. The critical issue in this approach is the impact of congestion minimization into the cost function (1), which is controlled by the congestion minimization factor K. The wire contribution introduces a perturbation into the cost function, which results in area (and/or delay) penalty. By excessively focusing on congestion minimization, we may obtain a very suboptimal solution in terms of cell area (and/or delay), and consequently an unroutable solution within the assigned chip area. In contrast, if the impact of the wire cost in Eq. (1) is negligible, then the mapped netlist is structurally very similar to the minimum area solution ($K = 0.0$), which may be unroutable. Therefore, the interconnect impact into the technology mapping cost function must be carefully evaluated. However, this critical problem was not addressed in the work [11]. In contrast, the results reported in

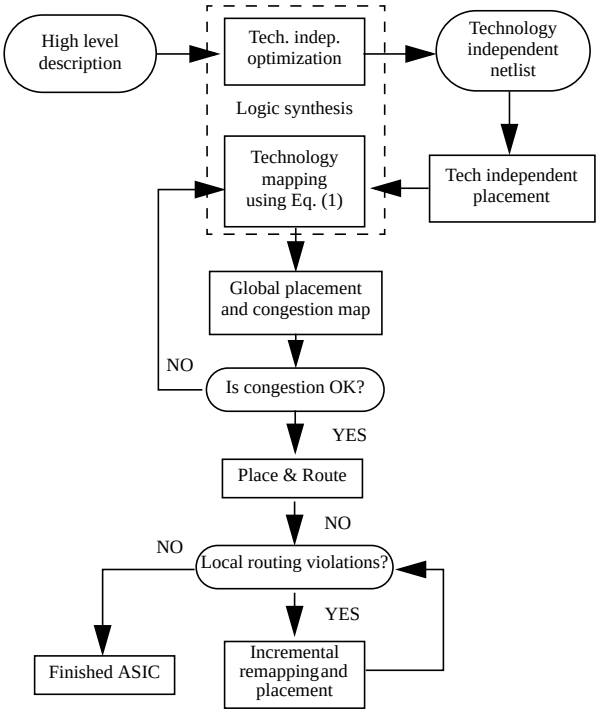

Figure 1. Modified ASIC design flow

[15] support the claim that excessive (or negligible) efforts on congestion minimization may yield unroutable netlists within the fixed die size. In conclusion, applying a purely predictive measure of congestion in the synthesis cost function can result in a significantly larger total cell area and an unroutable design.

3.2 Proposed Methodology

The congestion-aware technology mapping approach presented in [15] can be readily integrated into a traditional ASIC design flow, as shown in Figure 1. A technology independent netlist obtained via a standard logic synthesis tool is placed using a standard cell placement program. The first mapped netlist is generated by using Eq. (1) as the cost function for tech mapping and setting $K = 0.0$. Subsequently, this netlist is placed and congestion is evaluated. If the congestion map does not contain highly congested regions, the completion of the physical design can be attempted. On the other hand, if for $K = 0.0$ highly congested regions are still present, the congestion minimization factor K is increased in the tech mapping cost function. Using the tech independent placement information and a factor K that is greater than zero, a new mapped netlist is obtained, and congestion is re-evaluated. It is important to note that the technology independent netlist and its placement are generated only once. The computational complexity of technology mapping is linear with the size of the netlist, and by increasing K, structurally more routable netlists can be efficiently generated. If congestion does not improve with increases in K, however, the congestion begins to get worse for further increases in K as a result of the cell area penalty -- the circuit cannot be routed within the fixed die size and metal layers. Traditionally, at this point one would either relax the floorplan constraints or change the high level description and resynthesize the circuit. In contrast, if the congestion is contained in localized regions, we can apply this same congestion-aware synthesis via incremental technology remapping in the affected areas.

Whether it is the initial mapping or localized remapping, it should be noted that the results presented in [15] showed that *a priori* estimation of the optimal K value is very difficult, since congestion not only depends on the structure of the netlist, but it also depends on the available routing resources.

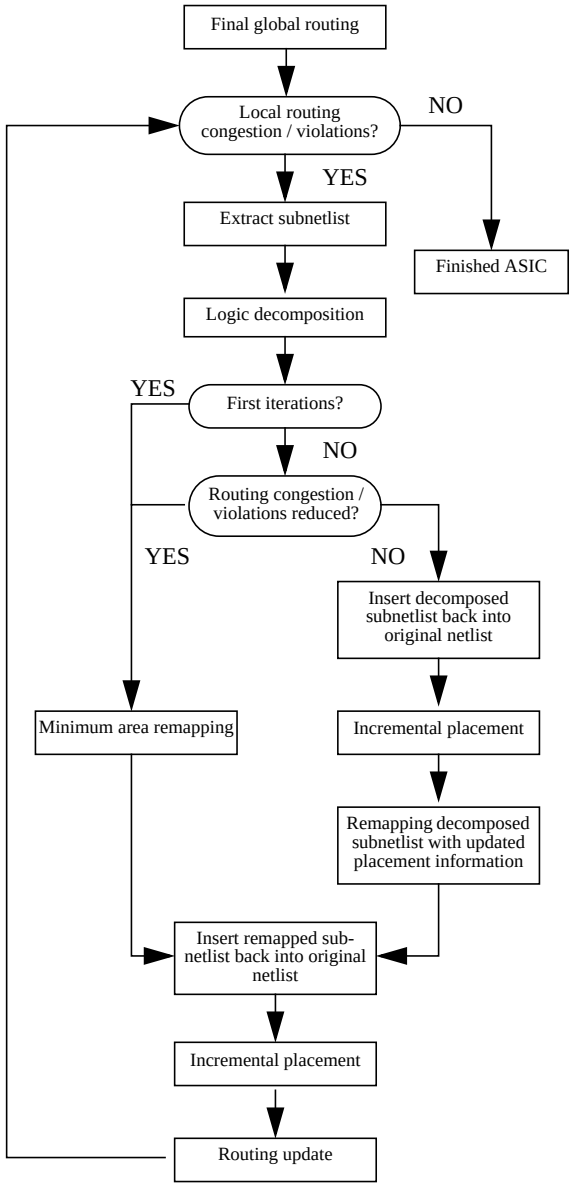

Figure 2. Incremental remapping for congestion minimization flow

4 Incremental Remapping for Congestion Minimization

Our approach for congestion minimization during technology mapping can be extended and implemented in a flow for incremental technology remapping and placement in order to progressively eliminate local routing violations after detailed routing, or localized congested areas after final global routing. It is important to note that as we increase K for the tech mapping, some areas get better in terms of congestion while some areas begin to suffer from an increasing area penalty. This is because the optimal value of K is not a constant over all regions of the chip. Choosing a single value for the initial mapping is important, but it may be equally important to subsequently revisit specific regions and recover some of the

area penalty. Our proposed methodology for placement-driven technology incremental remapping is shown in Figure 2. The flow has been implemented by integrating commercial tools, our technology mapper which includes congestion minimization capabilities, and several utility programs which have been developed in order to perform the necessary transformations on the gate level netlists and to interface the different tools in our flow. In general, this methodology can be implemented into any physical synthesis framework.

The first step in Figure 2 selects a congested area from the circuit layout where most of the potential routing violations are localized and extracts the subnetlist within this region. The external ports of this subnetlist can be either input/output ports of the original netlist, or nets which have cells both within and outside the layout region under observation. A further step may be necessary at this point to single out both sequential and multi-output combinational cells from the extracted subnetlist. Subsequently, the subnetlist consisting of only single-output combinational cells is decomposed into base functions. Since we lack analytical models, understanding the impact of congestion on routability is not so straightforward. The potential routing violations within a layout neighborhood may be determined either by local nets within the congested region, or by global nets which connect cells across the chip, both within and outside the region under observation.

When the apparent routing violations are caused by global nets, the most effective strategy is to remap the decomposed subnetlist for minimum area. The cell area within the congested region can thus be reduced, and consequently more routing resources are available to relieve congestion. On the other hand, if this approach is ineffective, the decomposed subnetlist can be incrementally replaced and remapped using the updated placement information, as in Section 3.

It is not known *a priori*, however, whether the routing violations will be caused by global or local nets. Therefore, following our methodology that is outlined in Figure 2, we first attempt to increase the routing resources within the congested region by remapping the decomposed subnetlist for minimum area. Note that this subnetlist may not be minimum area because of the global congestion factor *K*, that was used for the initial mapping. When projected routing violations are not reduced via $K = 0.0$, then the decomposed subnetlist is remapped for congestion minimization, using the incrementally updated placement informations, thus placing the fanins close to their fanout. The area penalty introduced with this second approach must be carefully controlled within a few percent of the minimum area solution. Although we cannot guarantee that all local violations are completely removed, our methodology is guaranteed to consistently reduce them. If after two/three incremental remapping iterations the violations are not significantly reduced, the circuit is probably unroutable within the floorplan constraints.

Our incremental remapping and placement methodology can be integrated into the ASIC top-down flow as shown in Figure 1. It provides a practical, robust methodology which addresses both global and local congestion. The key observation is that not all the regions of the chip have similar routing demands, and an accurate estimation of congestion at the synthesis level remains a very difficult problem since routability not only depends on the network structure, but also on the die size and metal layers. Although we can choose what seemingly is the best value of the congestion minimization factor *K* for an effective global congestion reduction, some local congested areas may be present after final routing. The work presented in [11] does not address this issue at all. In contrast, our methodology considers congestion both at the global and local level, and proposes a complete solution approach.

5 Results

The incremental flow presented in Section 4 has been implemented using commercial tools such as the logic synthesizer Design Compiler™, by Synopsys, Inc., the *place&route* tool Silicon Ensemble™, by Cadence Design Systems, Inc., and our technology mapper described in Section 3. The cell library used is the CORELIB8DHS 2.1, a proprietary library in 0.18µm of STMicroelectronics, Inc., and all the experimental results were obtained by restricting the block routing to three metal layers. Our incremental approach has been tested on different combinational circuits. The first set of results are for the circuit PDC, from the International Workshop on Logic Synthesis 1993 (IWLS93) benchmark suite. The technology independent representation (two-input nands and inverters) consists of 23,058 gates, and the die size was fixed at 229786µm^2, corresponding to 74 standard cell rows. The minimum area mapping yields a total cell area of 128438µm^2, and with such floorplan constraints the circuit was unroutable (5447 routing violations after detailed routing).

K	Cell Area (µm^2)	No. of Cells	Area Utilization%	No. of Routing violations
0.0	128438	7070	55.89	5447
0.00075	131477	7134	57.22	3673
0.001	132514	7268	57.67	0
0.0025	140161	8094	61.0	28
0.005	147714	8780	64.28	0
0.0075	151769	9201	66.05	0
0.05	163103	10617	70.98	158
0.5	178975	12274	77.89	6270
1.0	180330	12417	78.48	7770

Table 1. Circuit PDC - Congestion minimization results

The results reported in Table 1 are shown for several values of *K*, among which, our approach successfully relieved congestion globally and generated structurally routable netlists within the same fixed die size. Note that it is difficult, if not impossible, to determine *a priori* which values of the congestion minimization factor *K* will produce routable netlists. Therefore, when we obtain a solution with only a few local violations after detailed routing, is it better to continue searching for an optimal value of *K*, or is it preferable to apply our incremental remapping methodology? To demonstrate the efficacy of the incremental flow described in Section 4 we consider the netlist obtained with $K = 0.0025$ (28 local routing violations after detailed routing) from the table above. The results obtained with the incremental remapping and placement approach are reported in Table 2. Note that this result, based on a

K	Cell Area (µm^2)	No. of Cells	Area Utilization%	No. of Routing violations
0.0	139727	8078	60.81	0
0.01	139846	8092	60.86	0

Table 2. Circuit PDC - Incremental flow (one iteration)

methodology which employs an initial congestion-aware technology mapping followed by an incremental pass on the localized congestion regions, the area utilization is 60.81% versus an area of 64.28% for $K = 0.005$. The subnetlist within the congested region was decomposed with Design Compiler™, remapped for minimum area ($K = 0.0$) and recombined with the original netlist. After final routing, all the routing violations were eliminat-

ed and the circuit was routable within the assigned die size. Alternatively, also incrementally replacing, and remapping the decomposed subnetlist for congestion minimization ($K = 0.01$) with updated placement coordinates produced a fully routable solution, but with a slightly higher cell area utilization. When the original netlist was obtained with $K > 0.0$, i.e., was not minimum area, by incrementally decomposing and remapping subnetlists within congested areas, we can incrementally reduce the overall cell area.

5.1 Industrial test case

Our congestion minimization methodology was applied to a commercial IC from STMicroelectronics which implements a face recognition circuit [16]. The size of the random logic portion of the circuit was about 250k gates, yielding a gate level netlist of about 45k standard cells after logic synthesis. It is worth pointing out that this industrial design was a very challenging test case for our methodology, since about 40% of the standard cells are either sequential or combinational with multiple outputs, like adders, decoders, etc., which cannot directly benefit from our approach since they cannot be decomposed and remapped for congestion minimization. The subnetlist with only single-output combinational cells was decomposed in base functions using Design Compiler™. The technology independent subnetlist was recombined with the sequential and multi-output combinational cells, and this new netlist was placed with Silicon Ensemble™, in order to generate the initial placement for the congestion-aware technology mapping approach presented in [15], and summarized in Section 3. Subsequently, the technology independent subnetlist was remapped both for minimum area ($K = 0.0$), and for congestion minimization ($K > 0.0$). Routing was carried out under tight floorplan constraints with three metal layers, and the die size fixed to 2131133μm^2, corresponding to 228 standard cell rows. The minimum area netlist ($K = 0.0$) was unroutable, while by increasing K our approach dramatically improved the routability, even with increased area utilization. Although global congestion was successfully relieved, some local congested areas were not removed, as reported in Table 3. For an industrial design of this complexity, the designer may attempt to fix by hand this limited amount of violations, but this task may result in a long trial-and-error process.

K	Cell Area (μm^2)	No. of Cells	Area Utilization%	No. of Routing violations
0.0	1270309	41916	59.61	22466
0.00075	1273250	42382	59.75	29
0.001	1275687	42731	59.86	44

Table 3. Industrial circuit - Congestion minimization results

Alternatively, our incremental methodology for congestion minimization can be applied to remove the local violations. The final layout of the mapped netlist obtained with $K = 0.001$, where detailed routing terminated with 44 routing violations, is shown in Figure 3. A subnetlist within a rectangular region surrounding the congested area was extracted, decomposed and remapped for minimum area during the first iteration. By incrementally remapping for minimum area the technology independent subnetlist, more routing resources were made available in the congested region. The results of our incremental approach are reported in Table 4. After the first iteration, the violations were reduced to 25, and the corresponding layout is shown in Figure 4. Note that the incrementally remapped and replaced region is highlighted in this figure. The original violations localized within the congested region were removed after the first iteration. However, some new violations were generated near the borders of the congested region under consideration. By relieving congestion in one region, we may introduce some perturbations in the surrounding area, especially when the wiring and even the utilization in the neighboring area is changed by remapping and replacement within the initial congested region. This observation is consistent with all the experiments carried out on different circuits. Namely, when one iteration did not suffice in removing all the routing violations, the remaining violations where reduced in number and were localized in close proximity of the congested area. During the second iteration all the routing viola-

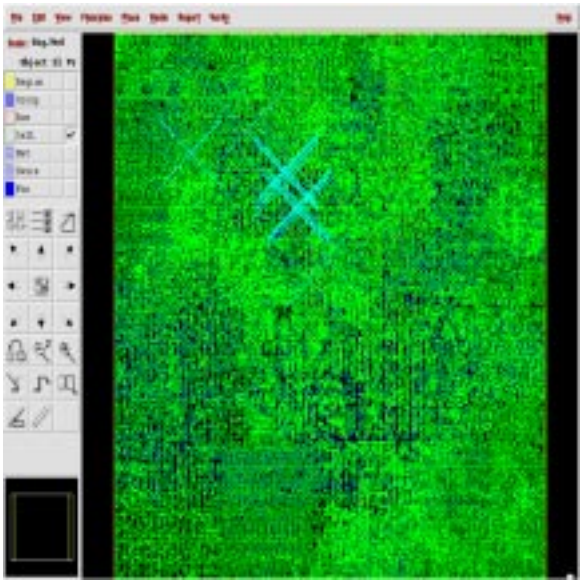

Figure 3. Final routing (44 routing violations)

Iteration	Cell Area (μm^2)	No. of Cells	Area Utilization%	No. of Routing violations
1st	1273225	42418	59.74	25
2nd	1272697	42326	59.72	0

Table 4. Industrial circuit - Incremental flow results

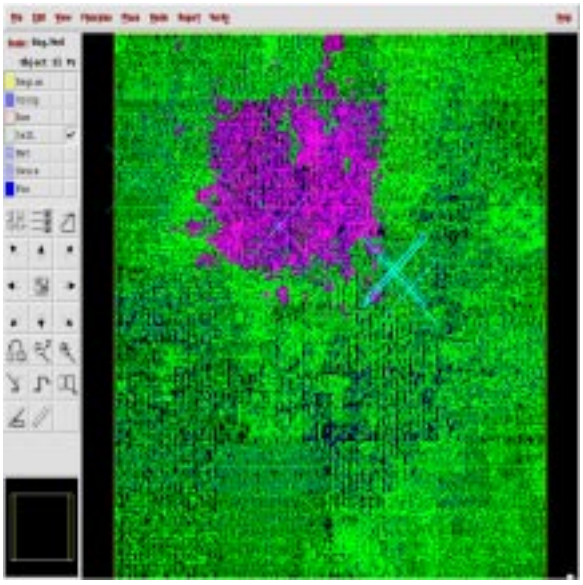

Figure 4. Incremental flow - 1st iteration (25 violations)

tions were removed and we obtained a fully routable netlist as shown in Figure 5 and reported in Table 4. It is worth pointing out that the minimum area netlist ($K = 0.0$) was unroutable within the fixed floorplan constraints, as reported in Table 3. Our approach for congestion minimization during technology mapping [15] significantly improved the global routability, by reducing the routing violations from 22,466 down to only a few ones, given the same die size and metal layers. Subsequently, our incremental remapping and placement methodology removed all the local violations from the design. This methodology provides the designer with a robust approach to relieving congestion before taking more time-consuming steps to either modify the RTL description and resynthesize the entire circuit, or determine the proper amount of routing resources without wasting Si area. Although this approach cannot guarantee to remove all the violations, the incremental flow effectively reduces them during the iterative process. However, it is not possible to estimate how many iterations will be necessary to remove all the violations. Empirically we have found that three to four iterations generally remove all the violations, or reduce them to a small number that can be fixed by hand.

It must be stressed once again that there is no single value of the congestion minimization factor *K* which in general can guarantee a congestion free design under fixed die size constraints. Moreover, by including congestion minimization among the synthesis optimization objectives, very suboptimal solutions in terms of area (and/or delay) may be obtained, thus eventually impairing congestion and routability. Therefore, we believe that with only a first pass of congestion-aware synthesis followed by a second pass of incremental remapping we can effectively address the global and local congestion during synthesis.

6 Conclusions

In this paper we have proposed a complete methodology for congestion minimization, both at the synthesis and layout stage of the design flow. The wiring congestion is a very important design factor which can directly impact the time-to-market, and must be considered both globally in logic synthesis, and locally in physical design. While global congestion can be effectively addressed during the technology dependent phase of logic synthesis, the suboptimality obtained by including congestion into the synthesis optimization objectives must be carefully evaluated. Hence, we believe that other attempts at controlling congestion at the synthesis level would prove successful only in a random, unpredictable way, due to the nature of models for congestion, and the behavior of multi-objective optimization.

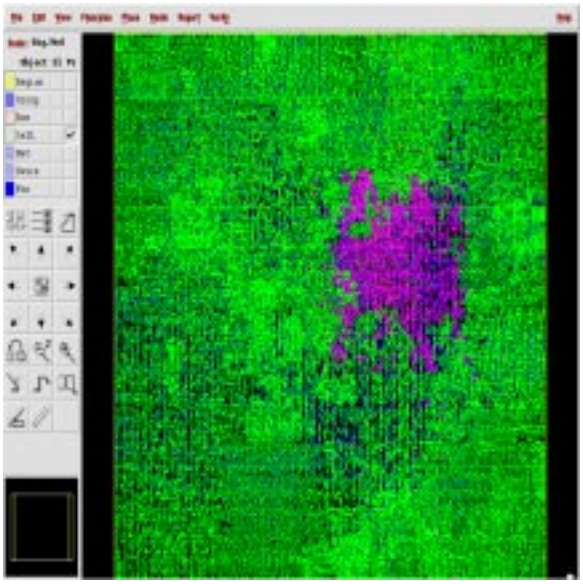

Figure 5. Incremental flow - 2nd iteration (no violations)

Moreover, since different layout regions have different routing demands, congestion must be taken into account also at the local level. The novel incremental remapping and placement approach proposed in this work successfully relieves local congestion, and results demonstrating the effectiveness of our methodology on industrial circuits have been presented.

Acknowledgments

The authors are grateful to Roberto De Checchi and Auro Lazzini, of Cadence Design Systems, Inc., Rozzano, Italy, for several enlightening discussions on physical design and continuous support with tools.

References

[1] H. B. Bakoglu. "*Circuits, Interconnections and Packaging*". Addison Wesley, Reading, MA, 1990.

[2] D. Sylvester, and K. Keutzer. "*Getting to the Bottom of Deep Submicron*". In Proc. ACM/IEEE Intl. Conf. on Computer-Aided Design, pages 203-211, Nov. 1998.

[3] S. Hojat, and P. Villarrubia. "*An Integrated Placement and Synthesis Approach for Timing Closure of PowerPC Microprocessors*". In Proc. IEEE Intl. Conf. on Computer Design, pages 206-210, October 1997.

[4] P. Gopalakrishnan, A. Odabasioglu, L. T. Pileggi, and S. Raje. "*Overcoming Wireload Model Uncertainty During Physical Design*". In Proc. Intl. Symp. on Physical Design, April 2001.

[5] K. Keutzer. "*DAGON: Technology Binding and Local Optimization by DAG Matching*". In Proc. ACM/IEEE Design Automation Conf., pages 341-347, June 1987.

[6] E. Detjens, G. Gannot, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang. "*Technology Mapping in MIS*". In Proc. ACM/IEEE Intl. Conf. on Computer-Aided Design, pages 116-119, Nov. 1987.

[7] M. Pedram, and N. Bhat. "*Layout Driven Technology Mapping*". In Proc. ACM/IEEE Design Automation Conf., pages 99-105, June 1991.

[8] A. H. Salek, J. Lou, and M. Pedram. "*An Integrated Logical and Physical Design Flow for Deep Submicron Circuits*". IEEE Transactions on CAD, vol. 18, no. 9, pages 1305-1315, Sept. 1999.

[9] M. Pedram, and N. Bhat. "*Layout Driven Logic Restructuring/ Decomposition*". In Proc. ACM/IEEE Intl. Conf. on Computer-Aided Design, pages 134-137, Nov. 1991.

[10] H. Vaishnav, and M. Pedram. "*Minimizing the Routing Cost During Logic Extraction*". In Proc. ACM/IEEE Design Automation Conf., pages 70-75, June 1995.

[11] T. Kutzschebauch, and L. Stok. "Congestion Aware Layout Driven Logic Synthesis". In Proc. ACM/IEEE Intl. Conf. on Computer-Aided Design, Nov. 2001.

[12] G. Meixner, and U. Lauther. "*Congestion-Driven Placement Using a New Multi-Partitioning Heuristic*". In Proc. ACM/IEEE Intl. Conf. on Comp. Aided Design, pages 332-335, Nov. 1990.

[13] P. N. Parakh, R. B. Brown, and K. A. Sakallah. "*Congestion Driven Quadratic Placement*". In Proc. ACM/IEEE 35th Design Automation Conf., pages 275-278, June 1998.

[14] L. N. Kannan, P. R. Suaris, and H. G. Fang. "*A Methodology and Algorithms for Post-Placement Delay Optimization*". In Proc. ACM/ IEEE Design Automation Conf., pages 327-332, June 1994.

[15] D. Pandini, L. T. Pileggi, and A. J. Strojwas. "*Congestion-Aware Logic Synthesis*". In Proc. DATE, Mar. 2002.

[16] M. Borgatti, F. Lertora, B. Foiret, and L. Cali'. "*A Reconfigurable System featuring Dynamically Extensible Embedded Microprocessor, FPGA and Customisable I/O*". In Proc. IEEE CICC, May 2002.