Congestion-Aware Logic Synthesis

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*

In this era of Deep Sub-Micron (DSM) technologies, the impact of interconnects is becoming increasingly important as it relates to integrated circuit (IC) functionality and performance. In the traditional top-down IC design flow, interconnect effects are first taken into account during logic synthesis by way of wireload models. However, for technologies of 0.25µm and below, the wiring capacitance dominates the gate capacitance and the delay estimation based on fanout and design legacy statistics can be highly inaccurate. In addition, logic block size is no longer dictated solely by total cell area, and is often limited by wiring area resources. For these reasons, wiring congestion is an extremely important design factor, and should be taken into consideration at the earliest possible stages of the design flow. In this paper we propose a novel methodology to incorporate congestion minimization within logic synthesis, and present results for industrial circuits that validate our approach.

1 Introduction

In DSM technologies interconnects play a crucial role in the overall performance of VLSI systems. For technologies of 0.25µm and below, interconnect capacitance becomes dominant with respect to gate capacitance, thus rapidly increasing the interconnect induced delay (as a percentage of the overall path delay). Therefore, the impact of interconnects on performances has to be carefully evaluated in order to satisfy the design constraints during all phases of the traditional ASIC top-down design flow, which broadly speaking consists of: logic synthesis and physical design (placement and routing). Constraints on timing are defined together with the logic description of the circuit and must be considered during all stages of the top-down flow to ensure timing closure.

For DSM technologies, a significant portion of the critical path delay is directly associated with interconnects [1]. Therefore, in order to meet performance constraints after layout, interconnect effects must be considered both in logic synthesis and physical design. The interconnect models used in timing-driven layout tools were essentially based on fanout loading and predefined net configurations. However, a fanout based model for delay estimation can be highly inaccurate for modeling the actual interconnect delay prior to layout, since by not considering the actual topology of the wires, it cannot accurately predict the distributed RC effects. In a performance-driven methodology, optimization based on wireload models often yields results which are significantly different from the actual post-layout values. As a consequence, many iterations between logic synthesis and physical design are usually necessary to achieve the timing closure. Unfortunately, this iterative process does not have any guarantee of convergence, and significant changes to the high-level description of the circuit may be necessary, thus introducing a critical bottleneck in achieving time-to-market targets.

A second, and related top-down design flow problem that is created by DSM interconnects is area estimation. Typically, optimization is targeted towards the total cell area for a semi-custom design, and the interconnection area is not considered. However, since the DSM wirelength trends do not scale with feature sizes, it is no longer practical to assume that cell area minimization will also minimize the overall size of the logic block. As a result, although the total gate area of the synthesized netlist can be quite small, it may not fit into the assigned die or block area once it is fully routed.

In physical design, the required routing resources are captured in terms of the routing congestion. During physical design and routing in particular, nets are routed one-by-one in an attempt to bypass 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, typically attempting to solve such problems at the physical design stage is too late.

Ideally, area minimization which includes the impact of wiring via congestion models would occur during logic synthesis; however, there is generally very little or no physical information available at the synthesis stage of the design flow. For this reason, commercial approaches from EDA vendors 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 wireability.

During the technology independent phase of synthesis, the Boolean equations representing the logic network are subject to logic optimization for minimizing the number of literals, which have been shown to correlate well with total cell area [2][3]. Unrestrained factorization based on kernel extraction yields gates with a high fanout count, which in turn can increase the overall wire meandering, thus augmenting the path delays and the total block area. In fact, a gate of small size shared between several functions may increase the wiring area to an extent that far exceeds the area saved by not duplicating the gate functionality. In contrast, uncontrolled minimization for the gate area during the technology dependent phase of synthesis, produces gates with a high number of fanins that also can potentially require more routing resources and wiring area. It is apparent that without some understanding of the impact of the necessary wiring resources, it is difficult to say anything about the true optimality of the synthesis for modern wiring limited technologies.

It is important to stress, however, that simply focusing on congestion minimization during the synthesis technology dependent

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

step would also yield very sub-optimal solutions in terms of both cell area and delay. When the cell area becomes very large, the congestion will also ultimately become poor and the circuit will not be routable within the assigned die area constraints and metal layers. *A priori* estimation at the synthesis level of the efforts on congestion minimization, along with the traditional synthesis optimization objectives such as area and/or delay, is a very difficult problem, since not only does congestion depend on the structure of the gate level netlist, but it also depends on the amount of routing resources available (i.e., the die area and the number of metal layers).

Therefore, we focus on first addressing the problem of congestion minimization in logic synthesis during technology mapping. It is only at this stage that it is possible to exploit placement information to capture the connectivity of the technology independent representation of the circuit consisting of base functions, and then explore the trade-offs between area (and/or delay) and congestion minimization when a fixed amount of routing resources are available. The objective of our methodology is to synthesize structurally routable netlists which satisfy the constraints on total block area and metal layers, and meet the other performance constraints such as timing.

This paper is organized as follows. In Section 2, previous work on the limitations of the wireload model in layout-driven synthesis is overviewed, along with other motivation for this work. In Section 3, our approach for congestion-aware technology mapping is presented, followed by Section 4 results showing its effectiveness. In Section 5 we propose a methodology for congestion minimization during technology mapping. Finally, Section 6 summarizes our conclusions.

2 Previous Work and Motivation

2.1 Wireload Models

In [4], Sylvester and Keutzer forecasted that for block sizes within 50k gates, the typical DSM interconnect effects were not going to dominate performance, even for feature size scaling well into the nano technologies. According to their results [4], the traditional design flow could be still used for blocks of 50k gate size with a sufficient cell sizing capability. A major limitation of this result lies in the cost of overdesigning caused by a pessimistic increase of the cell driving strengths, which might be unacceptable, since power is rapidly becoming one of the most important constraints in a variety of modern SoC and wireless applications.

In contrast, in [5] Hojat and Villarrubia demonstrated that for the synthesis of a block of full custom logic without large driving capabilities, even at 20k gate block size, the interconnect effects can be significant, and the wireload models ineffective. In [6], Pileggi further speculated that in a hierarchical block-based design style, the number of blocks and the amount of global interconnect, i.e., the inter-block wires, necessary to connect these blocks, is bound to increase as DSM technologies advance. Consequently, the global wirelength distribution spread will also increase, and delay budgeting techniques across the design hierarchy will become more difficult. More recently, Gopalakrishnan et al. [7] confirmed that the inherent wireload model inaccuracy can have a strong impact on predicting the lengths and delays of local nets in industrial designs. For this reason the problem of predicting the impact of interconnect on delay and area prior to physical design remains an unsolved problem.

2.2 Previous Work

Significant progress, both in academia and industry, has been made to provide logic synthesis with physical information in order to overcome the limitations of the wireload model. In [9], Pedram and Bhat integrated technology mapping with a companion placement to include wiring contribution in area and delay minimization. In the work presented in [10], 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 or similar statistical approximation. Since these commercial tools must route the logic blocks, in general they attempt to address the impact of interconnect on both the delay *and* the block area. There have been other research results that describe attempts at modeling routability in the synthesis technology independent phase, during logic extraction, either by using topological information such as the fanout range [16], or a companion placement of the logic network nodes [15]. However, placement of the logic nodes (which represent the Boolean equations) does not correlate well with the physical placement of a structural gate level netlist, and congestion is currently considered only during physical design, either at the global placement level or during incremental placement updates within resynthesis procedures [17][18][19].

2.3 Getting to the Problem Earlier in the Flow

In the traditional design flow, integration between logic synthesis and physical design is quite difficult, since no layout information is available. However, 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 (consisting of base functions like two-input NANDs and inverters) to technology dependent netlist can impact the global wirelength and congestion. Consider the unbound netlist shown in Figure 1, where two possi-

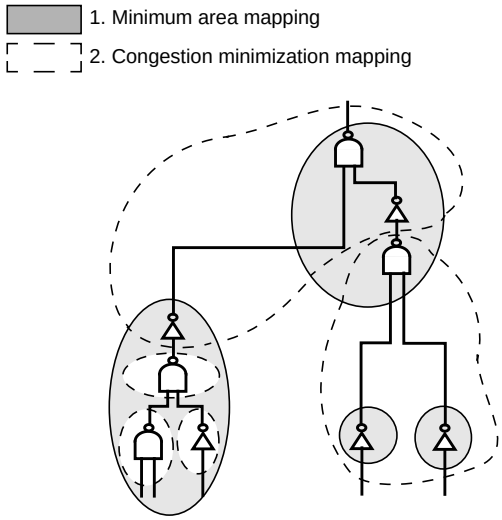

Figure 1. Minimum area vs. congestion mapping

ble mappings on the CORELIB8DHS 2.0, a proprietary library in 0.18µm technology of STMicroelectronics, Inc., are shown. The first solution represents the minimum area mapping and consists of one three-input NAND, one AOI21, and two inverters, for a total cell area of 53.248µm^2.

Suppose that Figure 1 represents a subregion extracted from a layout image of the uncommitted netlist initial placement, such that each gate depicted in the picture reflects its relative geometrical location on layout. A different mapping can be obtained considering not only the minimum cell area, but also trying to place the fanin

gates near their fanout. The second solution shown in Figure 1 includes a couple of two-input OR cells, a couple of two-input NANDs, and one inverter, for a total cell area of 65.536μm^2. As shown in the picture, the minimum area netlist has a longer global wirelength, since the fanin gates are placed far apart from their fanout. On the other hand, the second mapped netlist has a larger cell area, but the fanin gates are placed near their fanout, thus reducing the wirelength and congestion, and potentially improving routability.

This example introduces the trade-offs that must be considered when logic synthesis is extended in order to include congestion among the optimization objectives. Attempting to minimize congestion concurrently with cell area (and/or delay) yields a sub-optimal solution in terms of cell area (and/or delay), but the netlist obtained can potentially reduce the wirelength. This improvement can be significant enough that the total block area is decreased due to the reduction in routing resources.

Furthermore, traditional synthesis does not address routability. For an actual example, consider one of the circuits taken from the International Workshop on Logic Synthesis 1993 (IWLS93) benchmark suite. The benchmark chosen here is called TOO_LARGE, and consists of 27,977 base gates (two-input NANDs and inverters). The Register-Transfer-Level (RTL) description of the circuit was synthesized by the logic synthesis tool SIS [8] and then mapped onto the CORELIB8DHS 2.0. A technology independent representation of the logic network was also generated using SIS. This technology independent netlist was mapped for minimum area using the technology mapping algorithm DAGON [11]. Both the mapped gate level netlists (obtained with SIS and DAGON) were placed and routed using the same floorplan constraints, with an identical die size of 153915μm^2 and three metal layers, and the *place&route* tool Silicon Ensemble^{TM}, by Cadence Design Systems, Inc.. The results are reported in Table 1.

	Cell Area (μm^2)	No. of Rows	Area Utilization%	No. of Routing violations
SIS	126394	61	82.12	3673
DAGON	129851	61	84.37	0

Table 1. TOO_LARGE routing results

Obviously, SIS yields the best result in terms of area minimization because it can apply synthesis technology independent optimization techniques directly to the RTL description of the circuit. On the other hand, DAGON which is a technology mapping program, is limited by the initial structure of the technology independent netlist. As shown in Table 1, even though the netlist obtained with SIS has less area utilization (and consequently more routing resources available) with respect to the netlist obtained with DAGON, it is unroutable. In contrast, the netlist mapped with DAGON can be routed successfully in the same block area, even though it has a higher cell utilization of that area. This example demonstrates that excessive efforts in area minimization during logic synthesis can result in higher congestion, hence larger block area.

3 Our Approach: Congestion-Aware Technology Mapping

The objective of logic synthesis is 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 systems divide this task into two phases: technology independent optimization and technology mapping. The first phase is based on logic transformations applied to the Boolean equations describing the logic network, and the objective is to find a representation with the minimum number of literals in the factored form [3].

The technology dependent phase, also called technology mapping, is the task of transforming a technology independent logic network of base functions, into a technology dependent gate level netlist, i.e., an interconnections of components which are cell instances of a given library, such that the area of the final implementation is minimized and timing constraints are satisfied.

A common approach to technology mapping replaces subnetworks of the original technology independent network with instances of the cell library, minimizing cell area (and/or delay). An efficient solution to the technology mapping problem was proposed by Keutzer and implemented in DAGON [11] and MIS [12]. 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 of Keutzer, technology mapping is usually divided into three separated phases: DAG partitioning, matching, and covering. In the first phase, the network DAG is partitioned into a forest of trees. Subsequently, for each tree, a matching algorithm identifies all possible matches, corresponding to instances of a cell library, for each subnetwork. Finally, during the covering phase an optimal choice, according to some cost factor, among the matches is selected.

In our approach, we include physical information in the DAG partitioning and in the covering step. This is accomplished by first placing the technology independent network to capture the connectivity on a chip layout image. Floorplan constraints such as pin assignment and die size can be generated and used during this tech independent placement. A pair of geometrical coordinates is assigned to each vertex of the unbound network DAG after this placement step.

3.1 Placement-Driven DAG Partitioning

Two different schemes have been proposed for DAG partitioning. In the first approach, implemented in DAGON [11], the DAG of the decomposed network is partitioned at each multi-fanout node. The limitation of this approach is due to avoiding optimization across multi-fanout nodes. The second approach implemented in MIS [12], identifies cones of logic starting from the circuit primary outputs and following all the paths along the transitive fanins to the circuit primary inputs. Although this approach tends to partition the circuit into larger blocks, thus potentially increasing optimization opportunities, it suffers from two drawbacks. The first problem lies in the logic duplication across the logic cones boundaries which could be necessary to obtain a legal circuit. Secondly, its performance depends on the order in which the primary outputs are chosen to generate the cones of logic.

Our placement-driven DAG partitioning algorithm is outlined in Figure 2 and is based on a Depth First Search (DFS) traversal from the circuit primary outputs (roots) to the primary inputs (leaves), similar to the algorithm described in [12]. However, it differentiates from that approach since partitioning at multi-fanout vertices is carried out by taking into account the physical location of the corresponding base gates obtained from the tech independent placement. Our partitioning is based on the following property: *the father of every internal vertex is always the nearest vertex on the chip layout image according to some distance metric.*

For each circuit DAG root, the procedure *DAG_Partitioning()* calls the sub-procedure *Placement-Driven Partitioning* (*PDP*) which begins the DAG traversal starting from a root. The input to the *PDP* sub-procedure consists of the circuit DAG, one vertex (which is one of the roots when the procedure is initiated), and an array containing the placement coordinates for each vertex. The DAG traversal is carried out by means of a DFS. When a multi-

```
procedure DAG_Partitioning ( graph DAG, array COORD )
    for_each v in DAG do
        v.father = nil;
    od;
    for_each v in DAG.roots() do
        PDP ( DAG, v, COORD );
    od;

procedure PDP ( graph DAG, vertex v, array COORD )
    v.visited = true;
    for_each e in DAG.outedges(v) do
        w = DAG.target(e);
        if ( not w.visited ) then
            dist = INFINITY;
            for_each f in DAG.inedges(w) do
                u = DAG.source(f);
                this_dist = distance( COORD[u], COORD[w] );
                if ( this_dist < dist ) then
                    dist = this_dist;
                    w.father = u;
                fi;
            od;
            PDP ( DAG, w, COORD );
        fi;
    od;
```

Figure 2. Placement-Driven DAG Partitioning Algorithm

fanout vertex is encountered, a loop on his parents is executed. The function *distance()* uses the placement information to compute the geometric distance between two adjacent vertices. The closest parent is marked as father of the vertex under consideration, and once the procedure *DAG_Partitioning()* has terminated, a father has been assigned to each vertex. In the case of a multi-fanout vertex, every in-edge adjacent to a parent different from the father is detached, and a new vertex is connected to this edge which becomes a new leaf of a tree. While the amount of logic duplication of our DAG partitioning algorithm is comparable with the approach described in [12], its performance does not depend on the order the DAG roots are processed, but it depends only on the physical location of the base gates on the layout image. Moreover, subject trees which cluster vertices placed in the same neighborhood are obtained by means of this DAG partitioning algorithm.

3.2 Placement-Driven Tree Covering for Congestion Minimization

The tree covering step can be optimally solved in linear time with the size of the tree by means of a dynamic programming algorithm. This approach, originally proposed by Keutzer [11] for minimum area mapping, was extended by Rudell [13] for minimum delay mapping under both constant and non-constant load, and by Rudell [13] and Touati [14] for minimum area mapping under delay constraints. However, the basic structure of the covering algorithm does not significantly change, and the main difference lies in the optimization objective expressed by the cost function. As a consequence, by modifying the cost function, different optimization targets can be addressed.

In our approach, while maintaining the structure of the tree covering algorithm described in [11], in order to minimize congestion we expand the optimization objectives by including the wirelength contribution into the cost function. Extensive efforts towards gate area minimization yield a netlist with high fanin gates. The detrimental effect of high fanin gates on congestion stems from the increased interconnection length, when the fanins of such gates are placed far apart from their fanout gate. Therefore, in order to reduce interconnection length, fanin gates must be placed near their fanout. This target can be achieved by including in the technology mapping optimization objective an additional term which represents the interconnect cost of a match. Our optimization objective consists of two figures of merit: the first one takes into account area (and/or delay), while the second one considers the wirelength between the matching gate and its fanins. To simplify our explanation, we refer here only to area minimization. The area-related cost is given by the area of the library cell associated to the match under consideration, plus the area value of the coverings of the subtrees rooted at the match fanins. Such coverings are the minimum area solutions at these rooted subtrees, according to the dynamic programming principle of optimality [20]. The area cost at vertex *v* for match *m*, and the wire cost of its fanin interconnections are respectively expressed by:

$$AREA(m, v) = area(m) + \sum_{v_i} areaCost(v_i), \qquad (1)$$

$$WIRE1(m, v) = \sum_{v_i} dist(pos(m, v), pos(match(v_i), v_i)), \qquad (2)$$

where $v_i \in fanins(m, v)$. The function *area(m)* returns the area value of the library cell corresponding to match *m*, while the function *fanins(m,v)* yields all the tree vertices corresponding to the leaves of the pattern graph of match *m*. The function *areaCost(v)* returns the cost of the minimum area cover of the subtree rooted at vertex *v*, which was previously computed during the bottom-up traversal. The function *pos(m,v)* yields the position corresponding to the center of mass on the chip layout image of the base gates covered by match *m*, while the function *match(v)* returns the match at vertex *v*, and the function $dist(p_1, p_2)$ computes the distance between two points, i.e., the centers of mass, in the chip layout image.

The computation of the center of mass is always based on the physical locations obtained from the initial placement of the technology independent netlist. Once a match has been selected for a vertex cover, all the positions of the vertices covered by the match are replaced with the coordinates of their center of mass. This procedure allows for incremental updates in the placement of the technology independent netlist, when the base gates are covered by a pattern graph and replaced with the matching library cell. A similar approach for incremental updates of the companion placement of the unbound netlist was proposed in [9].

During each pass of the dynamic programming algorithm, for every vertex which is a descendant of the vertex under consideration, an optimal match has already been found. Hence, the corresponding center of mass for this match has already been computed and stored. As a consequence, the function *wireCost(v)* returns the interconnect contribution between the optimal match of vertex *v* and the optimal matches of its children, and is called only for vertices already visited during the bottom-up traversal of the covering algorithm. Once the optimal match for the vertex under consideration has been selected, and the optimal matches along with their center of mass for all its fanins are known from previous passes, the wire cost for this match is computed and stored by: first, computing its center of mass, and then by using Eq. (2) to compute all wire contributions between this optimal match and its fanins. Subsequently, this value can be retrieved in constant time during the next passes of the algorithm. Therefore, when function *wireCost()* is called for each fanin of the match under observation, all the wirelength contributions for these matches have already been computed and stored, and do not have to be evaluated every time the function *wireCost()* is called. The wire cost of the children is thus expressed by:

$$WIRE2(m, v) = \sum_{v_i \in fanins(m, v)} wireCost(v_i). \qquad (3)$$

The overall wire contribution for match *m* at vertex *v* is obtained by the function *WIRE(m,v)*, represented by the following ex-

pression:

$$WIRE(m, v) = WIRE1(m, v) + WIRE2(m, v), \quad (4)$$

where the first term includes the wirelength of the interconnections between match *m* and its fanins, while the second term takes into account the wirelength contribution of each of these fanins and its children. Functions *AREA(m,v)* and *WIRE(m,v)* yield two independent figures of merit for the global minimization objective of the tree covering algorithm, i.e., the cell area and interconnection length respectively. Consequently, Eq. (1) and Eq. (4) can be combined into a single cost function by means of the congestion minimization factor *K*:

$$COST(m, v) = AREA(m, v) + K \cdot WIRE(m, v), \quad (5)$$

where K can be set by the user in order to control the impact of congestion minimization with respect to cell area minimization.

3.3 Multi-objective Optimization Trade-offs

The example shown in Figure 1 compares area minimization and area plus congestion minimization, and provides insight into the trade-offs which need to be considered in our congestion-aware technology mapping methodology. The tree covering algorithm proposed by Keutzer is optimal for minimum area mapping. The optimality is due to the dynamic programming algorithm which performs its choices based upon a cost function which minimizes area. Any perturbation introduced into this basic cost function, like the inclusion of interconnection length for congestion minimization, yields sub-optimal solutions with respect to minimum cell area. The solution obtained by including congestion minimization into the optimization objectives has an area penalty, but since the fanin gates are placed near their fanout, it has less wirelength and is likely to improve routability. However, if the interconnect contribution in the cost function given by Eq. (5) becomes large, the area penalty might impair the beneficial effects on congestion of our approach, since with a fixed die size, less resources are available to the router. Consequently, the circuit might only be successfully wired by increasing the routing space. More on this later when we analyze our results in Section 4.

It should be noted that our approach has significant differences with respect to the work presented in [9]. Firstly, we focus on the problem of congestion minimization, while the approach in [9] considers the interconnect contribution to the total area and delay minimization. However, considering the interconnection length to minimize the total chip area is questionable, especially when the interconnect factor in the cost function is not carefully evaluated and controlled. Our results in Section 4 will show that in such cases the block area can increase significantly. Therefore, proposing to minimize the overall layout area with a methodology where the cell area can rapidly grow out of control seems to be dubious.

The impact of wires in the cost function depends directly on their length, which in turn depends on initial floorplan constraints, such as chip or block size. For our proposed methodology it is important that the size of the layout image for the technology independent netlist placement changes as a function of the actual block and floorplan constraints. When the size of the layout image changes, the absolute length of the interconnections also changes, since the technology independent netlist only consists of base gates which essentially have the same size, i.e., inverters and two-input NANDs. Therefore, for different chip size values, the impact of the wire cost during technology mapping will vary. When the layout image size for initial placement is large, then the absolute values of the interconnection lengths may introduce a large perturbation in the basic cost function for area minimization, thus causing a large cell area trade-off penalty.

Even though estimation of layout area with a predictor may yield a more accurate value for the initial layout size, in [9] there is no correlation between the cell area and the wire cost terms of the cost function. Therefore, the methodology in [9] could produce a solution close to the minimum cell area netlist, or a solution with a large cell area penalty, with little chance of predicting *a priori* which one will occur. Hence, it is not possible to estimate -- *prior to obtaining the mapped netlist* -- whether the chosen cost function will yield a result that will be routable within the fixed die size constraints. Furthermore, in [9] the wire cost includes all wirelengths required to connect all gates from the fanins of the match under consideration, along the transitive fanins, to the primary inputs. The amount of the wire cost can thus rapidly outgrow the area values of the library cells, thus introducing a large perturbation in the area minimization objective of the cost function, and consequently a large cell area penalty. Moreover, the wire contribution in the cost function is not uniform, since the amount of transitive fanins changes dramatically when the match is near the leaves, or in proximity of the root of the subject tree. Finally, when the sum of wirelengths required to connect all gates from primary inputs to the matching gate becomes large with respect to the sum of the cell areas, then it is no longer clear what is the optimization objective of technology mapping.

Our methodology attempts to control the cell area penalty by means of the congestion minimization factor K that can be defined by the user to drive a search for the optimal result, namely the minimum cell area that is routable within the specified block or chip area. In this way, solutions with an area penalty within a few percent with respect to minimum cell area can be obtained. Furthermore, wire cost is limited to only the fanins of the match under consideration and their children, instead of all the transitive fanins, from the matching gate, to the primary inputs. Since our objective is to have the fanin gates placed near their fanout, considering all the transitive fanins will impair the effectiveness of our approach, as it becomes ambiguous which of the fanins taken into account during the evaluation of the wire contribution impact the technology mapping cost function.

4 Results

The algorithms presented in Section 3 have been implemented into a prototype technology mapping tool, which addresses all the typical optimization objectives of the synthesis technology dependent phase, like area and/or delay. Additionally, this tool includes the congestion minimization objective described in Section 3.2, which is user-controlled by means of the congestion minimization factor K. The combinational circuits selected from the IWLS93 benchmark suite are: SPLA and PDC, whose size in terms of base gates (NAND2s and inverters) is 22,834 and 23,058 respectively. The unbound netlists are within the block size limit of 50,000 gates, reported by Sylvester and Keutzer [4], as the limit where logic synthesis does not have to be drastically changed even in DSM technologies. Our results show that even within this block size limit, wire congestion cannot be neglected. In all the experiments we used the CORELIB8DHS 2.0 cell library, the *place&route* tool Silicon Ensemble^{TM}, and wiring was limited to three metal layers.

A fair evaluation of our method requires that the synthesis technology independent phase must be clearly separated from technology mapping. Therefore, we always compare our technology mapping algorithm with DAGON (K = 0.0), which is based on heuristics that in general perform quite well, and always yield results comparable with other technology mapping algorithms.

For the circuit SPLA the die size was fixed to 207062μm^2 with an aspect ratio of one, corresponding to 71 standard cell rows. Within such die dimensions, the netlist obtained with DAGON was unroutable (4794 routing violations reported by Silicon Ensemble^{TM}). Afterwards, the impact of congestion on routability was evaluated by assigning K values in the range from 0.0001 up to 1.0.

The results are reported in Table 2. Note that as K increases, the

K	Cell Area (μm^2)	No. of Cells	Area Utilization%	No. of Routing violations
0.0	126521	7184	61.1	4794
0.0001	128205	6998	61.92	4737
0.00025	128184	7014	61.91	5307
0.0005	128356	7061	61.99	0
0.00075	128766	7135	62.19	0
0.001	129257	7203	62.42	0
0.0025	134717	7727	65.06	0
0.005	143081	8346	69.1	4805
0.0075	147435	8774	71.2	4958
0.01	149577	9017	72.24	4869
0.05	158097	10047	76.35	5867
0.1	162861	10547	78.65	7865
0.5	175346	11875	84.68	6777
1.0	176984	12060	85.47	8893

Table 2. SPLA congestion minimization vs. *place&route* results

perturbation introduced into the cost function (5) becomes more and more important, area minimization becomes less effective and the amount of active cell area increases accordingly, along with a consistent increment in the number of library cells. Consequently, the augmented area utilization limits the amount of available wiring resources. The routability range is divided into three different regions, as shown in Table 2. When the value of K is sufficiently small, the impact of the congestion optimization objective in Eq. (5) is negligible with respect to the area minimization target. Hence, the mapped netlist does not significantly differ from the minimum area solution, and remains unroutable. When K increases, the contribution of the congestion minimization term becomes more relevant and significantly alters the structure of the mapped netlist. As reported in Table 2, the number of routing violations drops to zero and the circuit becomes fully routable within the fixed die size. After further increasing K, the impact of congestion minimization dominates the optimization objectives, the cell area penalty increases and yields an unroutable netlist. This benchmark was also synthesized with SIS and mapped for minimum area, starting from the RTL description of the circuit, in order to combine the technology independent optimizations outlined in Section 1, with technology mapping. The netlist obtained with SIS was structurally unroutable even for larger chip area values. In fact, with a die size of $223691\mu m^2$, corresponding to 73 rows and area utilization of about 58%, the circuit still presented 3884 routing violations, while the netlist obtained with DAGON, was fully routable with 72 rows.

The other important performance constraint which must be taken into account during synthesis is the delay. Congestion minimization has beneficial effects on timing, since by avoiding long detours around congested regions, it also decreases the total wirelength. Therefore, Static Timing Analysis (STA) was performed to verify the impact of our approach on delay. In all the experiments, the mapped netlists were routed without violations with Silicon Ensemble^{TM}, the timing constraints were extracted from the completed layout and the static timing analyzer Prime Time^{TM}, by Synopsys, Inc., was used. The results for the circuit SPLA are shown in Table 3. The critical path of the netlist corresponding to K = 0.001, which was fully routable with 71 rows as reported in Table 2, has

K	Critical Path Arrival Time (nsec)	Comparison with critical path K = 0.0	Chip Area (μm^2) No. of rows
0.0	iJ0J(in) oJ23J(out) 17.85	iJ0J(in) oJ23J(out) 17.85	217678 72
0.001	iJ0J(in) oJ16J(out) 17.43	iJ0J(in) oJ23J(out) 17.06	207062 71
SIS	iJ0J(in) oJ44J(out) 18.57	iJ0J(in) oJ23J(out) 17.98	231015 75

Table 3. SPLA static timing analysis results

a smaller arrival time with respect to the netlists obtained both with DAGON (K = 0.0) and SIS. Moreover, the same path as the critical one in the netlist corresponding to K = 0.0 has even a substantial smaller arrival time. Therefore, while improving congestion and routability, our approach also improves timing. This is not unexpected, since timing will generally improve, and be more predictable, when there is less meandering of the routing. It is also worth pointing out that the circuit obtained with our approach is routable within a smaller total chip area, and by minimizing congestion we also significantly reduce the total chip or logic block size, even with a sub-optimal cell area.

For the benchmark PDC the results are reported in Table 4. The die size was fixed at $229786\mu m^2$, corresponding to 74 standard cell rows. The impact of congestion minimization on PDC is similar to SPLA. Within the range from K = 0.00025 to K = 0.0075 the mapped netlist is basically routable, since both 2 and 9 routing violations can be fixed during post-routing. However, when K = 0.00075, the netlist was unroutable. In this case the impact of congestion minimization within the cost function did not suffice at improving congestion, but it should be noted that the floorplan constraints are very tight and neither the solution obtained with DAGON, nor the solution generated with SIS were routable. The same experiments on the netlist synthesized with SIS were carried out also on PDC, and even in this case, to fully route the circuit, the rows had to be increased up to 77, corresponding to a chip area of $248562\mu m^2$. The STA results are reported in Table 5. Although the critical path for K = 0.0 is slightly smaller with respect to the critical path obtained with K = 0.001, this netlist has not suffered any performance degradation. In fact, in order to route the netlist mapped with DAGON, an extra row was necessary, thus augmenting the total chip area. However, the arrival time on the critical path for K = 0.0 has improved in the netlist corresponding to K = 0.001. Furthermore, the structurally congested netlist synthesized with SIS has the worst performance, both in terms of routability and delay.

5 Proposed Methodology

Our approach for congestion-aware technology mapping can be integrated into the traditional design flow, as shown in Figure 3. A technology independent netlist and its initial placement can be obtained with the tools of logic synthesis and placement. The first mapped netlist is generated by setting K = 0.0. Subsequently, this netlist is placed and congestion is evaluated. If the congestion map is acceptable, then physical design can be carried out and the synthesis is complete. If, however, for K = 0.0 the congestion map still has highly congested regions, the congestion minimization factor K is increased, a new mapped netlist is obtained, and congestion is re-evaluated. It is important to note that in our approach the technology independent netlist and its placement are generated only once. The computational complexity of the technology mapping algorithm described in Section 3 is linear with the size of the technology independent netlist. Hence, as discussed in Section 4 and reported in Table 2 and Table 4, by increasing K it is possible to ef-

K	Cell Area (μm^2)	No. of Cells	Area Utilization%	No. of Routing violations
0.0	128438	7070	55.89	5447
0.0001	129905	6882	56.53	3592
0.00025	130023	6912	56.58	2
0.0005	130630	7021	56.85	0
0.00075	131477	7134	57.22	3673
0.001	132514	7268	57.67	0
0.0025	140161	8094	61.0	9
0.005	147714	8780	64.28	0
0.0075	151769	9201	66.05	0
0.01	154141	9453	67.08	86
0.05	163103	10617	70.98	158
0.1	167485	11064	72.89	37
0.5	178975	12274	77.89	6270
1.0	180330	12417	78.48	7770

Table 4. PDC congestion minimization vs. *place&route* results

K	Critical Path Arrival Time (nsec)	Comparison with critical path K = 0.0	Chip Area (μm^2) No. of rows
0.0	iJ12J(in) oJ30J(out) 21.48	iJ12J(in) oJ30J(out) 21.48	233482 75
0.001	iJ9J(in) oJ24J(out) 21.79	iJ12J(in) oJ30J(out) 21.07	229786 74
SIS	iJ9J(in) oJ7J(out) 23.26	oJ12J(in) oJ30J(out) 22.55	248562 77

Table 5. PDC static timing analysis results

ficiently generate netlists which are structurally more routable. When the congestion map indicates that a netlist is within the routability region, such netlist is placed and routed. If after the physical design step the circuit is still unroutable, then a new mapped netlist can be efficiently obtained by further increasing K. However, if by increasing K the congestion does not improve, but eventually it begins to get worse as a result of the cell area penalty,

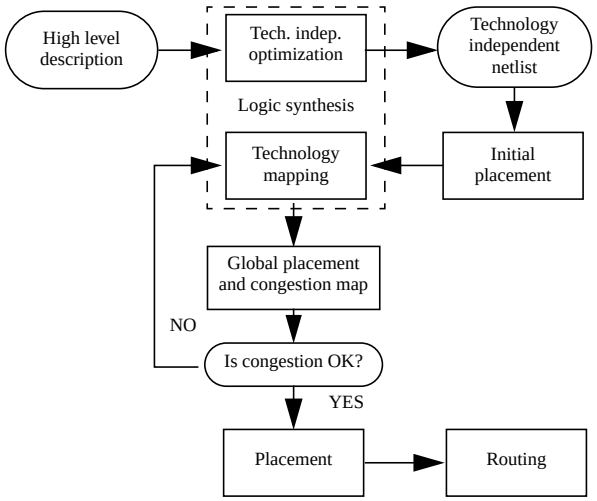

Figure 3. Modified ASIC design flow

the circuit is not routable within the fixed die size and metal layers, and further steps of detailed *place&route*, which are always computationally very expensive, can thus be avoided. At this point, either floorplan constraints are relaxed, by introducing more wiring resources, or the high level description is changed and the circuit resynthesized.

Once the impact of congestion minimization within the cost function (5) has been understood, since the computational complexity of technology mapping is far less then synthesizing a circuit from its high level description, designers may use our methodology, and by increasing K, efficiently generate solutions which are potentially less congested. Afterwards, congestion can be quickly evaluated, since obtaining a congestion map based on global placement and coarse routing is less costly then executing the detailed *place&route* step. When the designer is satisfied with the congestion map, and congestion is uniformly distributed across the chip, then the final placement and routing can be executed.

From the results of this work we conclude that *a prior*i estimation of a range of K values, which yield routable netlists, it is not possible, since congestion not only depends on the structure of the netlist, but it also depends on the available routing resources, i.e., the die size and the metal layers available. From all the experiments we have carried out, in order to effectively exploit our approach, empirical results show that the area penalty obtained by increasing K should be kept within a few percent of the minimum area solution.

6 Conclusions

In this paper we have studied the impact of congestion on block size for logic synthesis, and have proposed a new approach for congestion minimization during technology mapping. Our technology mapping algorithm has been integrated in a novel methodology for congestion-aware logic synthesis which expands the traditional design flow. We have shown that traditional logic synthesis focused on area minimization may yield structurally unroutable circuits and hence produce larger logic block areas due to routing congestion. We have further demonstrated that by minimizing congestion we also obtain an improvement on timing. The trade-offs between cell area and congestion minimization have been discussed in detail. While the proposed methodology is of a somewhat pragmatic flavor, we believe that other attempts at controlling congestion during synthesis would prove to be successful only in a random, unpredictable way, due to the nature of models for congestion, and the behavior of multi-objective cost functions during synthesis.

References

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

[2] M. Lightner, and W. Wolf. "*Experiments in Logic Optimization*". In Proc. ACM/IEEE Intl. Conf. on Comp. Aided Design, pages 286-289, Nov. 1988.

[3] R. Brayton, G. Hatchel, and A. L. Sangiovanni-Vincentelli. "*Multi-level Logic Synthesis*". IEEE Proceedings, Vol. 78, No. 2, pages 264-300, Feb. 1990.

[4] 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.

[5] 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.

[6] L. T. Pileggi. "*Achieving Timing Closure for Giga-Scale IC Designs*". In Proc. Intl. Symp. on Timing Issues, pages 25-28, March 1999.

[7] 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.

[8] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. "*SIS: A System for Sequential Circuit Synthesis*". Electronics Research Laboratory, Memorandum No. UCB/ERL M92/41, University of California at Berkeley, May 1992.

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

[10] 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.

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

[12] 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.

[13] R. Rudell. "*Logic Synthesis for VLSI Design*". Memorandum UCB/ERL M89/49, Ph.D. Dissertation, University of California, Berkeley, April 1989.

[14] H. J. Touati. "*Performance-Oriented Technology Mapping*". Technical Report UCB/ERL M90/109, University of California, Berkeley, Nov. 1990.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] E. Horowitz, and S. Sahni. "*Fundamentals of Computer Algorithms*". Computer Science Press, New York, NY, 1978.