A Predictive Distributed Congestion Metric With Application To Technology Mapping

Rupesh S. Shelar, *Member, IEEE*, Sachin S. Sapatnekar, *Fellow, IEEE*, Prashant Saxena, and Xinning Wang

***Abstract*— Due to increasing design complexities, routing congestion has become a critical problem in VLSI designs. This paper introduces a distributed metric to predict routing congestion and applies it to technology mapping that targets area and delay optimization. Our technology mapping algorithms are guided by a probabilistic congestion map for the subject graph to identify the congested regions, where congestion-optimal matches are favored. Experimental results on a set of benchmark circuits in a 90 nm technology show that congestion-aware mapping results in a reduction of 37%, on an average, in track overflows with marginal gate-area penalty as compared to conventional area-oriented technology mapping. For delay-oriented mapping, our algorithm improves track overflows by 20%, on an average, in addition to preserving or improving the delay, as compared to the conventional method.**

***Index Terms*— Congestion Estimation, Routing Congestion, Placement, Physical Design, Technology Mapping, Logic Synthesis**

I. Introduction

A. Motivation

Interconnect dominance is a daunting issue for sub-100 nm VLSI designs. It is a consequence of the rising design complexity: following Moore's law [1], the number of on-chip transistors (and therefore, according to Rent's rule [2], [3], the number of wires) is increasing exponentially every year. However, there is no proportionate increase in wiring resources, since die-sizes are expanding at a very slow pace [4], and upper metal layers that are offered with advances in technology are used mainly for routing global signals. As a result, even today's designs have regions where the unavailability of sufficient number of tracks to route the wires causes the circuits either to be unroutable or to violate the timing constraints due to long detours of wires. This is often referred to as the routing congestion problem. Although exact routing congestion information is known only after global routing, a failure to address congestion prior to this point implies that the designer is left with few degrees of freedom. Moving one step back, to placement, provides greater flexibilities in terms of global pin-density control and post-placement remedies (see, for instance, the congestion mitigation techniques described in [5]), but is still not always enough, and it is known that this does not remove the need for a number of design iterations. This is often due to the poor fidelity of congestion-unaware delay estimates, which cannot accurately capture the effect of long wire detours required for congestion reduction, or due to the unroutability of some designs where there may not be enough tracks available for routing.

Therefore, it is imperative to address congestion issues early in the design process to allow for more freedom to reduce congestion. Previous work on wire planning in logic synthesis [6], [7] at the technology-independent optimization stage is targeted towards wirelength estimation to consider the wire-delays, as opposed to our work which targets routing congestion. At this phase of logic optimization, it is not entirely clear which wires will be in the logic netlist, since these decisions are made during the technology mapping step. Technology mapping provides powerful capabilities for absorbing long interconnect wires into internal connections within complex gates, or for splitting complex gates into simpler gates, thus helping to alter the overall distribution of wires in the layout. Therefore, it is an ideal step, where the routing congestion problem may be attacked with relatively more freedom, albeit with relatively less information, than during placement and routing. Although several methods for integrating physical design with technology mapping have been proposed, there is little work on incorporating congestion considerations. Existing methods for this purpose, which are based on indirect metrics such as wirelength, are unsatisfactory, and the work presented in this paper is directed towards filling that void.

Manuscript received May 30, 2004; revised September 17, 2004. This work was supported in part by Semiconductor Research Consortium (SRC) under contract 2002-TJ-1092 and under award NSF CCR-0098117. It was carried out at the Strategic CAD Labs, Intel Corporation and at the Department of ECE, University of Minnesota. This paper was recommended by Guest Editor P. Groeneveld.

R. S. Shelar was with the Department of ECE, University of Minnesota, Minneapolis, USA. He is now with Intel Corporation, Hillsboro, OR, USA.

S. S. Sapatnekar is with the Department of ECE, University of Minnesota, Minneapolis, USA.

P. Saxena and X. Wang are with Intel Corporation, Hillsboro, OR, USA.

B. Previous Work

We review some of the previous works on congestion-aware technology mapping approaches in the literature. Stok *et al.* proposed a clustering of closely placed cells during technology mapping so that the matching choices covering distantly placed cells in the subject graph are ruled out [8]. This approach may result in long wires in the final netlist, and more importantly, may leave a significant portion of the design space unexplored. Pandini *et al.* proposed wirelength as a metric to be minimized during technology mapping in order to reduce the congestion [9]. Although large wirelength may be correlated with high congestion, the correlation is rather poor, and therefore, the mapping based on such a metric may not result in an effective optimization. This observation has been borne out by recent work by the same authors [10], who state that such a metric, when considered during technology mapping employing a traditional cost function may not result in decreased congestion. As pointed out by them, congestion is a local property that varies from bin to bin and is difficult to capture using a global metric like wirelength. This inference led them to the conclusion that congestion can only be targeted using iterative placement and technology mapping. However, such a conclusion is valid only when the congestion optimization is performed employing an indirect global metric in a traditional fashion and is not true in general.

C. Our Contributions

We present a technique for performing congestion-aware technology mapping. Instead of trying to absorb the congestion information into a single metric, we work with information about the distribution of congestion over the entire layout. The contributions of our work can be summarized as follows.

- Using empirical data on several benchmarks, employing different scripts and libraries and two different placement paradigms, we show the fidelity between the congestion maps for the subject graph and the mapped netlists, and exploit this fidelity during the technology mapping.

- Instead of applying an indirect metric such as wirelength [9], [11], we utilize probabilistic congestion estimates [12] to guide our technology mapping; these estimates are shown in [12] to have good fidelity with post-routing congestion measurements.
- Experimental results due to congestion-aware technology mapping algorithms on an industrial benchmark, ISCAS'85, and ITC'99 circuits show an improvement of 37% and 20%, on an average, in track overflows as compared to conventional mapping for area and delay minimization, respectively. These improvements come at the cost of 8% and 1% gate-area penalty, respectively, for area and delay minimization.

The organization of the rest of the paper is as follows. Section II introduces the terminology and problem definition, while Section III presents empirical and intuitive justifications for congestion fidelity for pre-mapped and mapped netlists. Sections IV and V illustrate congestion-aware technology mapping algorithms targeting area and delay, respectively, while Section VI discusses time complexity and possible extensions to these algorithms. Section VII presents experimental results and conclusions followed by the summary of the paper in Section VIII. A preliminary version of this paper was presented in [13].

II. Preliminaries

A. Terminology

The following terminology is used throughout this paper. A Boolean network is a directed acyclic graph (DAG), in which a node denotes a Boolean function, $f : B^n \rightarrow B$, where $B = \{0, 1\}$ and n is the number of inputs to the node. Traditional technology mapping is usually preceded by a decomposition of this abstract network into one that contains primitive gates, such as 2-input NAND's and inverters. The decomposed network is referred to as a subject graph or a premapped netlist. The subject graph is mapped on to a set of cells in the library during technology mapping; the resulting network is known as a mapped netlist, which is placed in a given block area and routed. The block area is divided into bins for congestion analysis purposes or for global routing. Each bin contains a limited number of horizontal and vertical tracks. The track overflow and congestion can be defined as follows.

Definition 2.1: The horizontal (vertical) track overflow for a given bin is defined as the difference between the number of horizontal (vertical) tracks required to route the nets through the bin and the available number of horizontal (vertical) tracks.

Definition 2.2: The horizontal (vertical) congestion for a given bin is the ratio of number of horizontal (vertical) tracks required to route the nets through the bin to the number of horizontal (vertical) tracks available.

A positive track overflow, or a congestion of more than 1.0, means that sufficient tracks are unavailable for the routing, while a negative value of the overflow, or a congestion smaller than 1.0, indicates the availability of tracks.

B. Problem Definition

Routing congestion depends on the following factors: the connectivity of the network, the placement of cells in the layout, and the routing of interconnects between the placed cells. Since there is relatively less freedom for attacking the routing congestion problem during the placement and routing stages, we concentrate on optimizing the first factor. The technology mapping step makes crucial decisions regarding the connectivity of the network, since the mapping of primitive gates to the library cells determines the set of wires that will be present in the circuit netlist. Traditionally, this has been carried out without any placement information. Although this has changed in recent physical synthesis vendor offerings, most approaches focus on the prediction of wirelength based on bounding box estimates that ignore congestion. The estimation of routing congestion without a placement for a network is, if not impossible, liable to be highly inaccurate, and one may have to rely on high level metrics such as adhesion [14]. However, several open questions about this metric remain unanswered: for example, whether it can be measured in a computationally efficient manner, and whether its fidelity is valid for mapped netlists. On the other hand, probabilistic congestion estimation [12] used after the placement of a mapped network has been demonstrated to correlate well with the congestion map generated after the routing, on both academic and industrial benchmark circuits. The estimation method divides the layout into bins and computes the congestion for a given bin under all possible routes for a given net. We employ the same method to guide our technology mapping algorithm. However, even such a method is difficult to adapt, since only the premapped netlist is available prior to technology mapping, and the level of correlation between the probabilistic congestion maps of the premapped netlist and the mapped one has not been studied in the past. One contribution of this work is to perform such a study. From empirical evidence obtained employing different logic synthesis scripts and placement algorithms on a variety of benchmarks, we show a good congestion correlation between premapped and mapped netlists. Once we establish the congestion correlation between these netlists, the problem of congestion-aware technology mapping can be defined as follows.

Problem definition 2.1: Given a subject graph of a network and a library of gates, synthesize a network optimizing area or delay such that the maximum (horizontal/vertical) congestion over all of the bins is less than the given threshold.

III. Congestion Fidelity

This section explores the level of fidelity between the congestion estimates before and after technology mapping for any circuit. For a given circuit, a premapped netlist contains primitive gates such as 2-input NAND's, while a mapped netlist contains a set of cells from a given library. Intuitively, the premapped and mapped netlists for a circuit share the same global connectivity, since the mapper absorbs some wires in the subject graph into the internal nodes of library cells, leaving other wires untouched. This points towards the possibility of good fidelity between congestion maps for premapped and mapped netlists. However, congestion also depends on the placement of elements (viz., primitive gates or gates in the library) in the netlist. Placement algorithms employed by commercial tools and in academia are typically based either on recursive multilevel bisectioning or force-directed quadratic programming. It would be useful to understand, even empirically, whether these placement algorithms react to the same global connectivity and block area constraints in a similar way. If so, there may be a good congestion correlation between premapped and mapped netlist. We explore this issue by performing a set of experiments using a variety of placers, logic synthesis scripts, libraries, and benchmarks.

A. Experimental Setup

To verify the fidelity between congestion estimates before and after technology mapping, we placed several premapped netlists, and the corresponding mapped netlists using the same block area and the same placement of input/output terminals. Two different placement algorithms were employed – a recursive bisectioning based algorithm in a publicly available tool, Capo [15], and a force-directed quadratic algorithm, Kraftwerk [16], implemented in a proprietary industrial placer. Different scripts, such as *rugged, boolean, algebraic, espresso,*

and *speedup* in SIS [17] were applied for preprocessing the netlists before technology mapping employing different libraries in SIS as well as an industrial library used for high-performance microprocessor designs. The following options were used for mapping and placement.

- Mapping was performed in SIS using the *map -s -n 0(1) -AFG -p* command that performs area and fanout optimization. No layout information was utilized to guide this technology mapping.
- Placement using Capo [15] or Kraftwerk [16] was performed with default options to minimize the total wirelength based on half perimeter bounding box estimates.

The premapped netlist is an abstract Boolean network containing primitive gates such as 2-input NAND's and inverters. For the placement of such a netlist, the primitive gates must be assigned areas. We assign the areas of the corresponding minimum-sized gates in the library to these primitive elements. Since the number of nodes in this netlist is large, the area of primitive gates must be scaled by a certain factor to present the same white space constraints as the mapped netlist for the placement. This factor is computed *a priori* as a ratio of the targeted gate-area to the area of premapped network using the following equation.

$$\text{Scaling factor} = \frac{\text{Block area} - \text{White space area}}{\text{Area of premapped network}} \quad (1)$$

Note that this factor is readily available given the block area, the white space specification (i.e., desired row utilization factor), the premapped netlist, and the cell library, and does not require any testcase-specific tuning.

B. Experimental Results

We show results for two representative benchmarks, C6288 and C7552, which differ vastly in their functionalities. Figures 1 (a) and (b) show congestion maps for the benchmark C6288 for the mapped and premapped netlists, respectively. The placement of both the networks is performed using Capo. In these plots, the XY plane shows the two dimensions of the layout area, while the Z-axis depicts the congestion. Visually, one can conclude that the distribution shown in Figure 1(b) is similar in nature to the congestion map shown in Figure 1(a). For most of the bins, the difference between the congestion in the premapped and mapped netlists is less than 10%. Similarly, the congestion maps for the benchmark circuit C7552 is shown in Figure 2; the netlists for this benchmark are placed using Kraftwerk [16]. The congestion map for the premapped netlist for C7552 shows characteristics similar to that of the corresponding mapped netlists. Observe that, unlike the usual pattern of the central area of a design being the most congested, these benchmarks exhibit a congestion hot-spot that is markedly off-center; in spite of this, the congestion maps for their premapped and mapped netlists correlate well. For a detailed treatment of the congestion correlation, please refer to [18].

Representative results for some ISCAS'85 benchmarks and the IDC circuit, an instruction decoder in a high-performance microprocessor design, employing different scripts, libraries, and placers, are shown in Table I, while similar results on more extensive set of benchmarks are presented in Table VII in the Appendix. Columns 2, 3, and 4 show the scripts used, the number of cells in the mapped netlists, and placement tools employed, respectively. Technology mapping in SIS [17] is performed using the area and fanout optimization option, employing either the lib2.genlib library in SIS or an industrial high performance library. It is worth noting that the mapped netlist is fanout-optimized, which possibly restructures the network after the mapping and may affect the global connectivity adversely. Columns 5 (6) and 7 (8) in the table show the average and maximum horizontal (vertical) congestion, respectively, while columns 9 and 10 show the statistical correlation between the congestion in premapped and mapped netlist. The correlation is defined as $\frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$, where $E[]$ is the expectation, μ is the mean, σ is the standard deviation; in our case, X and Y correspond to the congestion in the premapped and mapped netlists, respectively. A correlation value closer to 1 (-1) means that two random variables are strongly positively (negatively) correlated, while a value close to 0 means that variables are weakly correlated [19].

C. Justification Based on Experimental Results

In spite of fanout optimization that may affect the global connectivity and hence congestion fidelity, the congestion correlation between the subject graph and the mapped netlist is always greater than 0.6 for all the netlists. One may deduce the following based on these experimental results.

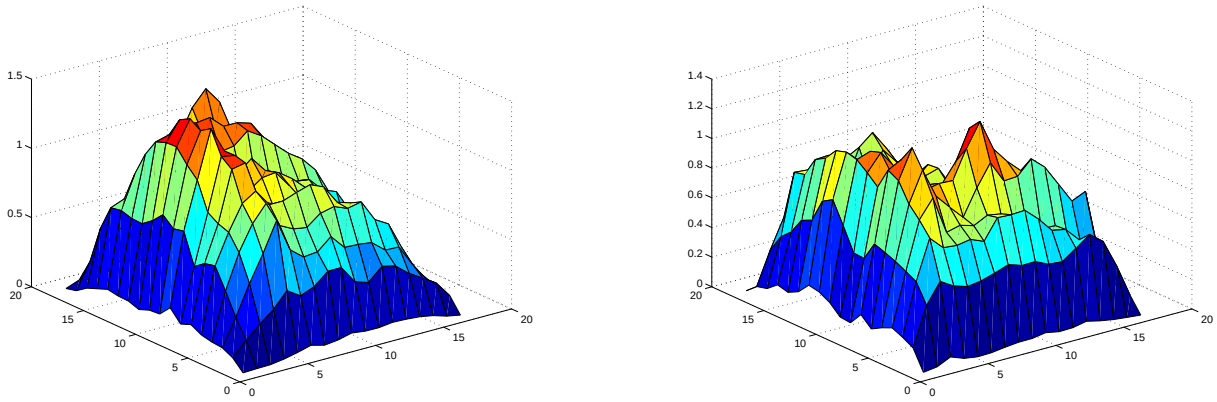
- Across different libraries, scripts, benchmarks, fanout optimization schemes, and placement algorithms, a good correlation exists between the congestion map for the subject graph and the congestion map for a mapped netlist.
- The reasons for the congestion correlation are likely to be the similarities in the global connectivity in the subject graph and the mapped netlist, the same block area and I/O terminal constraints, and the similar way in which any reasonable placement algorithm reacts to such resemblances in global connectivity and the block area constraints.
- The congestion correlation is smaller for Kraftwerk as compared to that for Capo. This can be partly attributed to the application of a cell-bloating technique to alleviate congestion for better pin-accessibility in our implementation of Kraftwerk, that actively modifies the congestion map for the mapped netlist. Furthermore, the cut-sizes that drive Capo perhaps correlate better to the congestion than do the force distributions that drive Kraftwerk (since the latter are influenced not only by the distribution of the nets but also the overlaps among the underlying cells).

IV. Congestion-aware Area-oriented Mapping

In this section, we focus on area optimization as an objective for technology mapping. For the purposes of congestion-aware mapping, the sparsely congested and densely congested regions must be identified. From the experiments in the previous section, which demonstrate the congestion correlation between a subject graph and its mapped netlist, we can conclude that the former netlist is accurate enough for this purpose. Since the primary objective of our congestion-aware technology mapper is area minimization, we employ a variation of a widely used dynamic programming-based technology mapping algorithm [20]. The technology mapping procedure involves the matching and covering phases: the former comprises storing the set of optimal matches at each node, while the latter involves constructing the network by selecting from the matches stored during the matching.

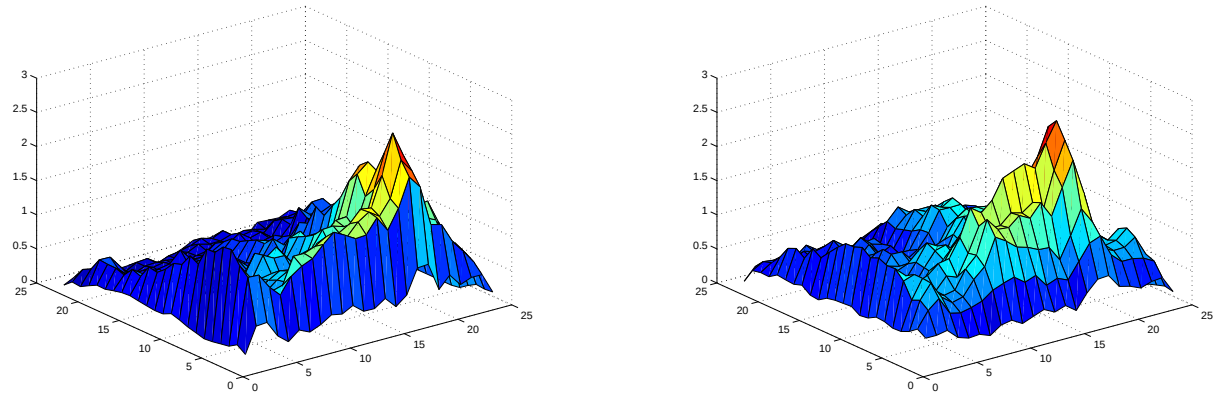
A. Example

A pure area/delay minimization objective during technology mapping can result in poor congestion, and Figure 3 illustrates a case where suboptimal matches may reduce congestion. Assume that all of the bins, shown as dashed squares in the figure, are congested and a match for the AOI33 function is considered. The inputs to the match enter through top and bottom bins on the left, while the output leaves from the middle bin on the right. Figure 3(a) shows one possible



(a) (b)

Fig. 1. Horizontal congestion for C6288 for (a) the area-oriented mapped netlist and (b) the premapped netlist: XY plane denotes the block area and Z axis shows the congestion. *script.rugged* is used for preprocessing the netlist and Capo [15] is employed for placement.



(a) (b)

Fig. 2. Horizontal congestion for C7552 for (a) the area-oriented mapped netlist and (b) the premapped netlist: XY plane denotes the block area and Z axis shows the congestion. *script.algebraic* is used for preprocessing the netlist and Kraftwerk [16] is employed for placement.

TABLE I

CONGESTION COMPARISON FOR THE NETLISTS BEFORE AND AFTER TECHNOLOGY MAPPING. MAX. (AVE.) CORRESPONDS TO MAXIMUM (AVERAGE), WHILE H (V) CORRESPONDS TO HORIZONTAL (VERTICAL).

Example	script/mapping	# Cells	Placer	congestion after/before mapping				Correlation	
				Max. H	Max. V	Ave. H	Ave. V	H	V
C432	rugged/area	257	Capo	1.27/1.45	1.46/1.99	0.41/0.47	0.48/0.65	0.91	0.90
C432	rugged/delay	328	Capo	1.17/1.45	1.51/1.99	0.39/0.47	0.46/0.65	0.96	0.95
C432	algebraic/area	237	Capo	1.22/1.15	1.38/1.6	0.37/0.35	0.44/0.51	0.97	0.96
C432	algebraic/delay	279	Capo	1.06/1.15	1.21/1.6	0.35/0.35	0.40/0.51	0.93	0.93
C432	boolean/area	375	Capo	1.04/1.55	1.42/1.68	0.43/0.45	0.51/0.67	0.95	0.94
C432	boolean/delay	501	Capo	1.47/1.41	1.46/1.51	0.54/0.50	0.63/0.70	0.93	0.93
C432	speedup/area	265	Capo	1.08/1.22	1.25/1.5	0.34/0.41	0.40/0.55	0.92	0.91
C432	speedup/delay	314	Capo	1.03/1.29	1.27/1.81	0.37/0.52	0.44/0.67	0.93	0.94
C6288	rugged/area	2311	Capo	1.73/1.34	1.88/2.00	0.69/0.57	0.81/0.82	0.85	0.86
C6288	rugged/delay	2383	Capo	1.45/1.34	1.75/2.00	0.61/0.57	0.71/0.82	0.86	0.87
C6288	algebraic/area	2275	Capo	1.37/1.79	1.55/2.20	0.50/0.73	0.60/0.98	0.76	0.78
C6288	algebraic/delay	2620	Capo	1.38/1.05	1.59/1.31	0.48/0.52	0.58/0.73	0.83	0.79
C6288	boolean/area	2329	Capo	0.89/0.85	1.05/1.32	0.40/0.40	0.48/0.66	0.75	0.71
C6288	boolean/delay	2605	Capo	1.38/1.23	1.53/1.72	0.47/0.48	0.56/0.70	0.79	0.79
C6288	speedup/area	4182	Capo	1.11/1.10	1.34/1.39	0.41/0.48	0.51/0.66	0.78	0.81
C6288	speedup/delay	4395	Capo	1.19/1.20	1.47/1.58	0.48/0.51	0.58/0.63	0.86	0.82
C7552	algebraic/area	1521	Kraftwerk	2.60/2.70	2.70/2.40	0.61/0.71	0.66/0.71	0.81	0.76
C7552	rugged/area	2060	Kraftwerk	2.04/2.05	2.27/2.26	0.65/0.69	0.71/0.79	0.64	0.68
C7552	boolean/area	1582	Kraftwerk	2.23/2.50	2.50/2.00	0.61/0.74	0.66/0.71	0.82	0.83
C7552	espresso/area	1457	Kraftwerk	1.68/2.10	1.85/2.20	0.64/0.69	0.69/0.79	0.73	0.65
C6288	algebraic/area	2528	Kraftwerk	1.60/1.48	1.05/1.35	0.52/0.61	0.58/0.64	0.77	0.76
C6288	rugged/area	2391	Kraftwerk	1.50/2.00	2.00/2.00	0.53/0.62	0.58/0.63	0.63	0.62
C6288	boolean/area	2583	Kraftwerk	1.49/1.79	1.61/1.82	0.47/0.54	0.53/0.57	0.64	0.70
C6288	espresso/area	2549	Kraftwerk	1.76/1.79	2.06/2.09	0.52/0.62	0.59/0.66	0.61	0.64
IDC	rugged/area	972	Kraftwerk	1.25/1.30	1.13/1.47	0.65/0.60	0.60/0.65	0.67	0.68
IDC	algebraic/area	800	Kraftwerk	2.09/1.67	2.06/1.80	0.50/0.47	0.53/0.45	0.70	0.61
IDC	boolean/area	1622	Kraftwerk	1.75/1.78	1.52/1.23	0.57/0.59	0.64/0.65	0.67	0.66
IDC	espresso/area	2233	Kraftwerk	1.89/1.93	2.17/2.24	0.51/0.55	0.56/0.55	0.75	0.74

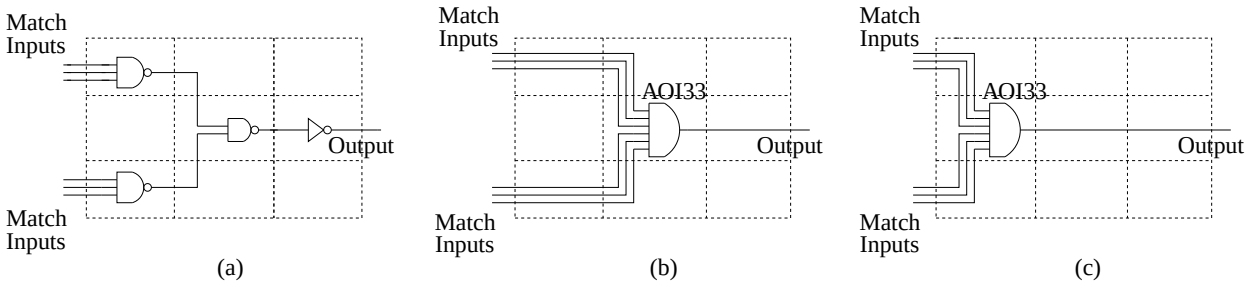


Fig. 3. Mapping choices: (a) Sub-optimal area and track requirement = 12. (b) Area-optimal and track requirement = 20. (c) Area-optimal and track requirement = 15.

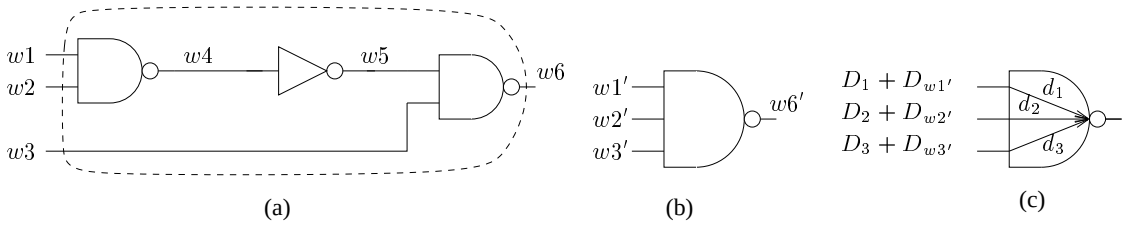


Fig. 4. Computing the congestion and delay cost of a match: (a) An example subject graph. (b) A match of 3-input NAND. (c) Delay computation.

match containing two three-input NAND's, a two-input NAND, and an inverter, while Figure 3(b) and Figure 3(c) show an alternative match, an AOI33, under two different placements. To simplify the computations, if we use the number of bin-boundary crossings as the congestion metric, instead of the probabilistic congestion metric, then the cost for the match in Figure 3(a) is 12, while that for the AOI33 matches in Figures 3(b) and (c) are 20 and 15, respectively. The latter also happens to be the minimum over all placements for the area-optimal AOI33 match. It is clear that the match in Figure 3(a) distributes the logic and therefore, creates lower congestion. This example also highlights limitations of the placement in alleviating congestion when area-optimal matches are chosen ignoring the costs of wires associated with them.

The cost of wires depends on the context: wires are inexpensive in sparsely congested regions, but are expensive in densely congested regions due to possible detours and hampered routability. One way to reduce this cost in densely congested zones without penalizing the design excessively is to account for their congestion contributions only in those zones. Our congestion-aware mapping heuristic serves this purpose well: in densely congested spots, it considers probabilistic routes based on the center-of-gravity locations for all possible matches and chooses the match that minimizes the congestion, while in sparsely congested spots, it chooses area-optimal matches.

B. Congestion Cost Computation

The congestion-aware mapping heuristic requires the assignment of a congestion cost, along with an area cost, to each match. The congestion cost depends on the total congestion caused due to the nets subsumed by a match, its fanin nets and its fanout nets. Specifically, it is given by,

$$cost^C_{Match} = \Sigma\, cost^C_{net\_created} - \Sigma\, cost^C_{net\_subsumed} \quad (2)$$

where, $cost^C_{Match}$ is the congestion cost of the match, $cost^C_{net\_created}$ ($cost^C_{net\_subsumed}$) is the congestion cost of the nets created (subsumed) by the match. For example, for a 3-input NAND match shown in Figure 4(b) corresponding to the subject graph shown in Figure 4(a), the congestion cost is as follows:

$$\begin{aligned} cost^C_{Nand3} &= C_{w1'} + C_{w2'} + C_{w3'} + C_{w6'} - (C_{w1} \\ &\quad + C_{w2} + C_{w3} + C_{w4} + C_{w5} + C_{w6}) \end{aligned} \quad (3)$$

The nets $w1'$, $w2'$, $w3'$, and $w6'$ correspond to the new location of the match and the fanins and fanouts of the match; we compute the new location of a match as the center of gravity of the locations of its fanin and fanout gates. Multi-terminal nets are modeled using cliques for the congestion computation, and congestion contribution of each edge is scaled by a factor of $2/n$, where n is the number of edges.

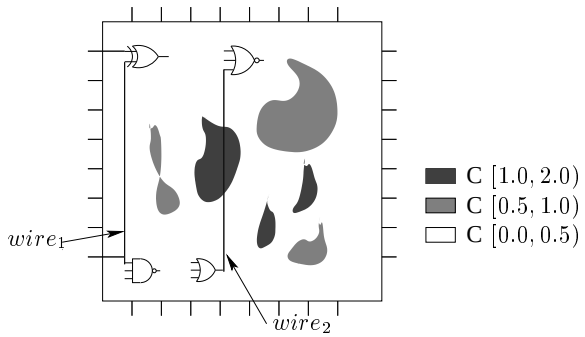


Fig. 5. Context-dependent congestion cost for the wires. In the figure, 'C' refers to the horizontal or vertical routing congestion.

The congestion cost of a wire depends on the route and the congestion in the bins that the route passes through. Probabilistically, all of the routes in the bounding box of the net are assumed to be equally possible[1] [12]. If the congestion in a bin in the bounding box of the net is small, say 0.4, as compared to the threshold congestion (say 1.0, for instance), then the congestion contribution of that net for that bin is assumed to be 0. This is because a small value of the congestion metric corresponds to the availability of numerous tracks, and the routability of the net through the bin is unaffected. However, if the bin is congested, then the probabilistic congestion contribution of the net to that bin must be considered, as its routability is hampered. In case of Figure 5, $wire_1$ and $wire_2$ will have different congestion costs even though the shortest routes in both the cases may

[1]This assumption may not always be true. Typically, routers try to minimize vias and therefore, for two terminal nets, L and Z routes are considered first. Such information can be taken into account while generating the congestion map as in [21].

have the same length; the congestion cost of the former will be zero, while that of the latter will have a positive value as its bounding box contains congested bins. The following equation captures this causality relation between routability and congestion while computing the congestion cost of a net, $cost^C_{net}$,

$$cost^C_{net} = \Sigma_{\{Bin \in BoundingBox(net): C(Bin) > C_{max}\}} C^{Bin}_{net} \quad (4)$$

where $C(Bin)$ is the congestion in a bin, C_{max} is the threshold congestion, and C^{Bin}_{net} is the congestion due to the specific net within the bin. One can observe that this definition filters out the contributions of uncongested bins from the congestion cost. The

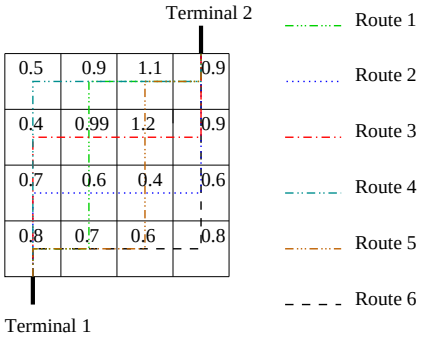


Fig. 6. Computing the congestion cost of a wire probabilistically as in [12].

bounding box for a two-terminal net is shown in Figure 6. It contains 16 bins, and the congestion value associated with each bin is shown in the figure. For the net connecting terminals 1 and 2, six possible L- and Z-shaped routes are shown for the purpose of illustration[2]. To compute the congestion cost, if the threshold value of congestion (C_{max}) is set to 1.0, then we consider only the congested bins for which congestion value is greater than 1.0, i.e., bins for which the congestion metric is 1.1 and 1.2. Three routes (route 1, 4, and 5) pass through the bin with congestion 1.1, while two routes (route 3 and 5) pass through the bin with congestion 1.2. Assuming all the routes to be equally possible, the demand (the ratio of number of paths passing through the bin to the total number of paths) for tracks in the latter bin is $\frac{2}{6}$. Similarly, the demand for tracks in the former bin is $\frac{3}{6}$. Employing the definition 2.2, congestion contribution of the net for these bins can be computed by dividing the demands by the number of available tracks (N_{Tracks}). Thus, the congestion cost of the net is given by

$$cost^C_{net} = \frac{1}{N_{Tracks}} \times (\frac{2}{6} + \frac{3}{6}) \quad (5)$$

The congestion cost for a match can be calculated from that of its incident nets using Equation 2. A positive cost for the match implies that it may increase the congestion beyond the threshold value in some bins, while a negative cost implies that it may decrease the congestion in some of the bins where congestion exceeds the threshold value.

C. Algorithm for Congestion-aware Area-Oriented Mapping

Algorithm 4.1 shows the pseudo-code for choosing the best match at a node during the matching phase of the technology mapping. The triplet (C_i, A_i, D_i) associated with a match M_i denotes the congestion, area, and delay cost associated with the match. The algorithm is called for every match at a node during the matching phase to decide the best one to be stored at the node. Although the

[2]In practice, we use probabilistic congestion estimates that consider river routes as well.

Algorithm 4.1 Select the best match considering the congestion

Input: Match $M_1(C_1, A_1, D_1)$ and match $M_2(C_2, A_2, D_2)$
Output: The best match between the M_1 and M_2

```
if (C1 == C2) then
    if (A1 < A2)||((A1 == A2)&&(D1 < D2)) then
        return M1;
    else
        return M2;
    end if
end if
if (C1 < C2) then
    return M1;
else
    return M2;
end if
```

congestion cost is given priority over the area and delay, area-optimal matches will be chosen for the nodes in the sparsely congested regions, as stated by the following proposition.

Proposition 4.1: If bins in bounding boxes of all of the nets, corresponding to all of the matches at a node, have congestion values that are smaller than the threshold congestion, then an area-optimal match will be stored as the best match at that node.

This is a direct consequence of the fact that the congestion cost for all nets corresponding to all of the matches in such a case is zero from Equation (4), and the pseudocode shows that under this scenario, the area-optimal match is always chosen. The above proposition is important for congestion-aware mapping, since previous work in [10] has shown that the traditional way of considering the cost, ($K_1 \times$ Area $+ K_2 \times$ Wirelength) during technology mapping requires different values of K_2 in the different regions in the circuit as a single value of K_2 fails to capture the importance of congestion in different regions. Choosing a single value of K_2 is equivalent to assuming that the entire circuit is uniformly congested with a single congestion value. In reality, the congestion in the circuit may vary continuously from 0 to 1, or may even be greater than 1, while the routability changes in a discrete manner: in case of a bin with congestion value greater than 1, at least some nets are detoured or are unroutable, while the routability of all the nets passing through a bin is unaffected when the congestion for the bin is less than 1. Assigning the congestion cost to the nets in the congested bins accounts for this discrete nature of routability and also allows the mapper to select area-optimal matches in the sparsely congested regions. Both of these purposes are critical and are served by our algorithm, while previous approaches [8], [9] have not addressed these.

V. Congestion-aware Delay-oriented Mapping

The congestion-aware area-oriented mapping framework presented in the previous section can be extended to delay-oriented technology mapping, which typically employs one of the following two classes of delay models: load- or gain-based. In this section, we focus only on delay-oriented mapping based on the former, since an extension based on the latter is similar. Technology mapping targeting delays involves storing piece-wise linear load-delay curves, $\{(l_1, D_1), (l_2, D_2), \cdots\}$, during the matching phase, where l_i and D_i denote load and delay co-ordinates, respectively, of an end-point of a piece-wise linear segment. At each node, a set of matches that are delay-optimal for certain load ranges are stored on these curves. During the covering phase, when loads are known, delay-optimal matches are chosen from the curves. SIS [17] contains an implementation of a delay-oriented mapper based on this scheme, which uses a fanout-based wire-load model,

ignores wire-delays and therefore, may lead to suboptimal results. To perform delay-oriented mapping accurately, it is necessary to consider wire-delays during delay computation.

The delay computation, which considers the effects of wires, involves accounting for wire-loads as well as wire-delays. This can be done by modeling the wires using an RC π model. Assume that a gate g_1 drives a gate g_2 through a wire w. Then the delay from the inputs of the gate g_1 to the input of g_2 through the wire w is given by the following equation

$$D = D_{g1} + D_w \tag{6}$$

where D_{g1} and D_w are the delays of the gate g_1 and the wire w, respectively. Employing the Elmore delay model[3] [23], the gate delay D_{g1} is given by

$$D_{g1} = D_{internal} + R_d \times (C_w + C_{g2}) \tag{7}$$

where $D_{internal}$ is the internal delay of the gate, R_d is the effective resistance of the gate, C_w is the capacitance of the wire w, and C_{g2} is the input capacitance of the gate g_2. Similarly, the wire-delay D_w is given by

$$D_w = R_w \times (\frac{C_w}{2} + C_{g2}) \tag{8}$$

where R_w is the resistance of the wire.

In general, the resistance (and capacitance) of a wire is a function of its length and a choice of metal layers. Since the resistivity of the upper metal layers is smaller because of the higher width and thickness as compared to lower metal layers, these metal layers are used to route the long wires. For the short wires, lower metal layers are utilized, since reliability and resistance of the vias along with subsequent congestion does not justify the use of upper metal layers. The range of wirelengths and choice of metal layers can be determined empirically for a given process technology as in [24], and this can be used to compute wire-delays during technology mapping. We employ such a scheme to account for the wire-loads and delays during congestion-aware delay-oriented mapping.

A. Slack-aware Congestion Cost Penalty Heuristic

To reduce congestion during delay-optimal technology mapping, one can choose solutions that leads to a smaller congestion cost at the expense of increased gate delay off the critical paths. Figure 5 shows an example where track requirement can be reduced by replacing a complex gate with a set of smaller gates. The small gate implementation increases the routing flexibility, but potentially at the expense of increased gate delay. To conduct congestion-aware delay-optimal mapping, congestion-aware choices should be stored during the matching phase. These choices, however, must not affect the delay adversely. To achieve this, the matching stage should have the following properties.

1) It should store delay-optimal choices on critical paths.
2) It should consider congestion-aware choices on non-critical paths and ensure that these paths will not become critical ones.
3) On different non-critical paths, it should weigh congestion-aware matches differently, i. e., on paths with a large slack, it should confer higher priority to matches that reduce congestion, while on paths with a small slack, it should treat congestion-aware choices with a relatively smaller preference.

To store congestion-aware choices on non-critical paths with a varied degree of importance, solutions that increase or decrease the congestion should be, respectively, penalized or favored, in proportion to the slack available at that node. This can be achieved by adding the congestion cost of a match weighed by the slack available at a given node to the delay because of the match. The congestion cost for a match depends on the corresponding cost of fanin and fanout nets, and nets that are subsumed by the match, as given by Equation 2, which is reproduced below for the sake of readability

$$cost^C_{Match} = \Sigma\, cost^C_{net\_created} - \Sigma\, cost^C_{net\_subsumed} \tag{9}$$

where, $cost^C_{Match}$ is the congestion cost of the match, $cost^C_{net\_created}$ ($cost^C_{net\_subsumed}$) is the congestion cost of the nets created (subsumed) by the match. To penalize the matches that cause congestion and to favor those that reduce it, a penalty is added to the delay due to a match before storing it on load-delay curve. The penalty corresponding to the congestion cost of a match is given by the following equation

$$D_{penalty} = k \times slack_{node} \times cost^C_{Match} \tag{10}$$

where, k is a user-defined constant and $slack_{node}$ is the positive slack available at the given node (or 0 if the slack is negative). Note that $k = 0$ corresponds to conventional delay-oriented technology mapping. In practice, k should be small enough such that it does not worsen the delay on non-critical paths so much that they become critical. Too small a value of k, however, reduces the effectiveness of the congestion-aware mapper producing results closely similar to a conventional mapper from a congestion (as well as delay) stand-point. In practice, an effective value of k can be obtained by experiments starting with a value of 0 and increasing to a point where the delays due to the congestion-aware mapper start getting worse than those due to a conventional one. For all of our experiments in Section VII-B , we use $k = 0.03$ obtained by such an empirical procedure. The slack associated with each node is computed by the first pass of conventional delay-oriented matching, as explained in the following subsection. The heuristic can be explained using the following example. Figure 4 shows a match of 3-input NAND, which subsumes wires $w1$, $w2$, $w3$, $w4$, $w5$, and $w6$, as shown in Figure 4(a), while it creates wires $w1'$, $w2'$, $w3'$, and $w6'$, as shown in Figure 4(b); we assume that the match is placed at the center of gravity of its fanins and fanouts, as in case of area-oriented mapping. Figure 4(c) shows the delay computation, where $D_i + D_{wi'}$, $i = 1, 2, 3$, are arrival times, including the corresponding wire-delays, at the inputs of the match, while d_i are internal delays for the corresponding pins. The delay of the match is given by the following equation

$$\begin{aligned} D \;=\; & max(D_1 + D_{w1'} + d_1, D_2 + D_{w2'} + d_2, \\ & D_3 + D_{w3'} + d_3) \end{aligned} \tag{11}$$

The congestion cost of the match, repeated from Equation 3, is given by the following equation

$$\begin{aligned} cost^C_{Nand3} \;=\; & C_{w1'} + C_{w2'} + C_{w3'} + C_{w6'} - (C_{w1} \\ & + C_{w2} + C_{w3} + C_{w4} + C_{w5} + C_{w6}) \end{aligned} \tag{12}$$

To make the match congestion-aware, a delay penalty proportional to the above cost is added to the delay. Therefore, the delay of the match is now given by

$$D_{congestion-aware} = D + k \times slack_{node} \times cost^C_{Nand3} \tag{13}$$

It is obvious that matches with a positive congestion cost are penalized, while those with a negative cost are favored. As desired, for matches on non-critical paths with a large slack, the congestion cost is weighed more than that for their counterparts on paths with small slacks. In sparsely congested regions, where enough tracks

[3]More accurate delay models, such as asymptotic waveform evaluation (AWE) [22], can also be employed while keeping the rest of the algorithmic framework intact.

are available, and on critical paths, no delay penalty is added and therefore, delay-optimal matches are still chosen in those cases.

B. Algorithm for Congestion-aware Delay-oriented Mapping

Algorithm 5.1 Congestion-aware delay oriented technology mapping

Input: N = A subject graph, L = A cell library, PO = A set of primary outputs, n = A node, o = A primary output
Output: Mapped netlist

```
for ∀ n ∈ N, using library L, in topological order do
    Perform conventional delay-oriented matching
end for
Compute delay ∀o ∈ PO due to conventional matching
Compute slack ∀o ∈ PO
for ∀ n ∈ N, in reverse topological order do
    Slack_n ← minimum{Slack_fanout(n) ∀ fanout(n)}
end for
for ∀ n ∈ N, using libray L, in topological order do
    Perform congestion-aware matching using Algorithm 5.2
end for
for ∀ n ∈ N, in reverse topological order do
    Perform covering with choices stored during congestion-aware
    matching stage
end for
```

Algorithm 5.2 Congestion-aware matching: compute load-delay curve for a node

Input: n = A node, M = A set of matches at the node
Output: Load-delay curve for n

```
for m ∈ M do
    for i ← 1, ··· , |inputs_m| do
        D ← max(D_i + D_wi + d_i)
    end for
    if Slack_n > Slack_minimum then
        Cost_m^C ← ComputeCongestionCostOfMatch(m)
        D ← D + k × Slack_n × Cost_m^C
    end if
    UpdateLoadDelayCurve(n, m, D)
end for
```

Algorithm 5.1 shows pseudo-code for the proposed technology mapping algorithm, which involves two matching phases as opposed to one in conventional mapping. The first matching phase is same as that of conventional matching and stores load-delay curves without considering any congestion effects. At the end of this phase, delays at the primary outputs are computed using solutions stored at those nodes followed by slack estimation, where slack is defined as the difference between required and actual arrival times. Once slacks are computed for all outputs, they are propagated towards inputs employing reverse topological traversal. The reverse traversal ensures that slacks are assigned to all fan-outs of a given node before the assignment of the slack, which is the minimum among slacks of all the fan-outs, to that node. The second round of matching, which is congestion-aware, utilizes the slack and congestion cost information to assign a penalty to the matches, as shown in Algorithm 5.2, while the covering phase proceeds in a traditional manner to choose matches that are optimal for given loads.

Algorithm 5.2 shows the pseudo-code for computation of the load-delay curve at a node. For each match m from the set M, the delay, D, is computed considering the arrival times of the inputs D_i, wire-delay D_{wi}, and internal delay, d_i, of the gate corresponding to the match. The congestion cost of the match, $Cost_m^C$, is then computed considering the corresponding costs of subsumed and created wires, and an appropriate penalty employing Equation 10 is added to the delay of the match if the available slack at node ($Slack_n$) is greater than certain minimum threshold ($Slack_{minimum}$). For all of our experiments in Section VII-B, $Slack_{minimum}$ is chosen to be 25 ps, which is a fanout-of-4 delay of a typical inverter in the library used there. The load-delay curve is then updated to store the match if it is optimal for some range of loads.

VI. Complexity, Limitations, and Extensions to the Algorithms

The asymptotic time complexity of our congestion-aware technology mapping algorithms is almost unchanged from that of a conventional technology mapping. The congestion cost computation of a match takes $O(|Nets_{Match}| \times N_{Bins})$, where $|Nets_{Match}|$ is the number of nets associated with a match and N_{Bins} is the number of bins over entire layout; N_{Bins} is a constant for a given layout, although it may be large as compared to other constants subsumed by $O()$. Therefore, congestion cost computation takes $O(|Nets_{Match}|)$ time, which is same as that of structural matching employed in the mapper [17]. Our delay oriented mapper involves two rounds of matching, which asymptotically does not change the computational complexity, but affects the run-time in practice marginally, as seen from the experimental results.

Since this technology mapping procedure is applied to tree structures after the initial subject graph generation and the decomposition of DAG's into trees, the algorithm does not have any control over high fanout nets, or over the fanout nets created due to matches at the roots[4] of the trees. The congestion due to these high fanout nets is controlled by the structure of initial network and fanout optimization. The effectiveness of the congestion-aware mapper proposed here is influenced by the scripts used for technology independent optimization, technology decomposition, and fanout optimization after technology mapping.

Pre-routed blockages in the design can be incorporated into our congestion cost by reducing the appropriate number of tracks in the corresponding bins. Most placers are adequate at handling blockages. Therefore, subject graph nodes or mapped cells are not placed in blocked areas. While long wires may require repeaters that are not visible in the subject graph, observe that these buffers do not change the congestion cost.

In the current implementation, we do not update the congestion map dynamically during technology mapping. However, this update can be carried out during the covering phase, thus allowing a more accurate selection of the best match stored at a node. In case of area-oriented mapping, multiple congestion-aware choices may be stored during the matching phase in addition to the area-optimal one, in order to enable the selection of a good congestion-aware solution with the updated congestion map available during covering.

The strong correlation between the congestion maps of the unmapped and mapped netlists suggests that the underlying placements are also correlated. Therefore, placement of the unmapped netlist can provide regioning directives that can not only help speed up the subsequent placement of the mapped netlist, but also make the eventual placement (and therefore, the wire loads) more predictable.

VII. Experimental Results

The experimental flow used in our experiments is compared with a typical modern conventional mapping and layout flow in Figure 7. Although early conventional flows did not involve a subject graph

[4]All of the nodes in the tree have a fanout of 1 except for the root.

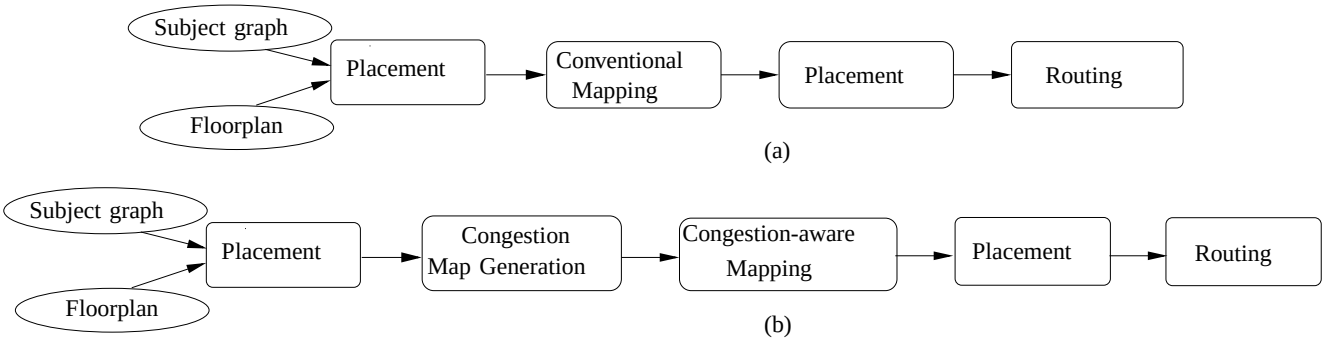


Fig. 7. Design flows for (a) conventional and (b) congestion-aware mapping.

placement step, most design flows today incorporate the generation of some placement information prior to or during mapping in order to make wire loads more realistic (see, for example, [25]). Thus, the subject graph placement step in our proposed flow no longer constitutes as much of an overhead as it would have done in the past.

The probabilistic congestion estimation algorithm from [12] and the congestion-aware technology mapping algorithms presented in sections IV and V were implemented in C/C++ and incorporated in SIS [17]. The subject graphs were created by running *script.rugged* followed by *tech_decomp -o 2* in SIS [17]. For area-oriented technology mapping, we present a set of experimental results obtained using a force-directed quadratic placer, Kraftwerk [16], and a proprietary industrial router, while for delay-oriented mapping, we show results generated employing a recursive bipartitioning placer Capo [15] and a global router [26]. For congestion-aware mapping, a subject graph was first created. It was placed using Kraftwerk for area-oriented mapping, while Capo was employed for the placement in case of delay-oriented mapping. The congestion map for the subject graph was then generated and used in our congestion-aware mapper. After area-oriented technology mapping, the circuits were placed employing Kraftwerk followed by global routing using a proprietary industrial router. For delay-oriented mapping, Capo and a global router [26] were employed, respectively, for placement and routing. In all of our experiments, a bin-size of 4.8×4.8 μm^2 was used. We present the results due to area-oriented mapping followed by the same for delay-oriented mapping.

A. Results due to Area-oriented Mapping

Table II shows the post-routing results obtained using the Kraftwerk placer and proprietary router for conventionally mapped and congestion-aware netlists. Technology mapping is performed employing a proprietary industrial cell library used in high-performance microprocessor designs. Our experiments employ a 90 nm technology and allow the router to use 4 metal layers[5]: metal 1 with no preferred direction, metals 2 and 4 for the horizontal direction, and metal 3 for the vertical direction. The entries of the form 'a / b' in the Columns 3 through 7 mean that 'a' ('b') corresponds to conventionally (congestion-aware) mapped netlist. The block area shown in Column 2 is used for both of these netlists for the benchmarks shown in Column 1. Since the same block area is used for both the netlists, there is no block area penalty. Columns 3, 4, and 5 show the average row utilization, the total track overflow over all the bins after global routing, and the number of bins with congestion more than 1.0, respectively, while Columns 6 and 7 show the maximum and average congestion, respectively. For small benchmarks such as C1355, C432, and C880, a small number of bins is congested in the conventionally mapped netlists while none of the bins is congested in the congestion-aware mapped netlists. C499 and C1908 show zero routing track overflows, while other small benchmarks have only a few congested bins, indicating that routing congestion is not an important issue for designs up to a few hundred cells. As the design size grows beyond a thousand cells, routing congestion starts becoming a critical problem, as indicated by increased track overflows for benchmarks such as IDC, C6288, and C7552. In these cases, the congestion-aware mapped netlists have been able to reduce the track overflows by 87%, 43%, and 29% while the number of congested bins has decreased by 81%, 65%, and 25%, respectively. Based on the increase in average congestion for all of the benchmarks, accompanied by a reduction in the number of congested bins and the number of track overflows, we see that congestion-aware mapping tends to map the logic so as to distribute the congestion from densely congested regions to the sparsely congested regions. The improvement in congestion comes at the cost of an increase in gate-area, which is reflected in higher row utilization in the case of the congestion-aware netlist for all the benchmarks.

TABLE II
COMPARISON OF CONVENTIONAL AREA-ORIENTED MAPPING WITH CONGESTION-AWARE AREA-ORIENTED MAPPING. PLACEMENT AND ROUTING IS PERFORMED USING KRAFTWERK AND A PROPRIETARY ROUTER, RESPECTIVELY, FOR A 90 NM TECHNOLOGY. COLUMNS WITH LABEL 'RU' AND 'CB' DENOTE THE AVERAGE ROW UTILIZATION AND THE NUMBER OF CONGESTED BINS IN A CIRCUIT, RESPECTIVELY.

Circuit	Area	RU	Overflow	CB	Congestion	
	μm^2	%		#	Maximum	Average
C1355	2380	68 / 79	2 / 0	1 / 0	1.3 / 0.9	0.35 / 0.43
C1908	2457	68 / 78	0 / 0	0 / 0	0.8 / 0.9	0.34 / 0.40
C432	1728	66 / 69	1 / 0	1 / 0	1.1 / 0.9	0.35 / 0.37
C499	2618	64 / 73	0 / 0	0 / 0	0.9 / 1.0	0.34 / 0.40
C6288	16920	61 / 68	32 / 18	20 / 7	1.3 / 1.3	0.49 / 0.52
C7552	17633	61 / 67	655 / 461	258 / 193	1.3 / 1.3	0.65 / 0.69
C880	2534	71 / 82	4 / 1	2 / 1	1.3 / 1.2	0.42 / 0.48
IDC	6919	63 / 70	83 / 10	32 / 6	1.3 / 1.2	0.53 / 0.60
Average		65 / 73	97 / 61	39 / 25	1.16 / 1.08	0.43 / 0.48

B. Results due to Delay-oriented Mapping

Table III show *post-routing* results for delay-oriented mapping obtained using the recursive bisectioning based placer Capo [15] and a global router that can optimize both congestion and delay [26]. Technology mapping was performed employing lib2.genlib library in SIS [17]. Up to 4 different strengths were added for each cell in the

[5]While 90 nm and subsequent process generations have a large number of metal layers, the upper layers are usually reserved for global signals, clock and power distributions, leaving block synthesis to operate in the lower layers [27].

TABLE III
COMPARISON OF CONVENTIONAL DELAY-ORIENTED MAPPING AND CONGESTION-AWARE DELAY-ORIENTED MAPPING. PLACEMENT AND ROUTING IS PERFORMED EMPLOYING A PUBLICLY AVAILABLE PLACER CAPO [15] AND A ROUTER [26], RESPECTIVELY, FOR 100 NM TECHNOLOGY [28]. COLUMNS WITH LABELS 'RU' AND 'MC' DENOTE THE AVERAGE ROW UTILIZATION AND THE MAXIMUM ROUTING CONGESTION IN A CIRCUIT, RESPECTIVELY.

Example	Area	RU	Delay	MC	Overflow
	μ^2	%	ps		
b14	57600	75 / 75	2919 / 2839	2.60 / 2.20	1703 / 1625
b15	80202	75 / 77	2830 / 2821	3.40 / 2.60	5171 / 4440
b20	129600	70 / 70	3377 / 3294	4.10 / 3.20	8667 / 6871
C1355	3439	80 / 82	789 / 774	1.70 / 1.50	227 / 173
C1908	3616	80 / 80	1059 / 1065	1.70 / 1.10	323 / 87
C432	1962	80 / 80	854 / 840	1.40 / 1.40	66 / 54
C499	3550	80 / 80	823 / 791	1.60 / 1.20	262 / 188
C6288	21379	80 / 80	4771 / 4682	1.70 / 1.40	515 / 474
C7552	22350	75 / 75	1497 / 1255	2.40 / 1.80	1383 / 743
C880	3944	80 / 83	890 / 867	1.70 / 1.40	378 / 270
Average		78 / 79	1980 / 1922	2.22 / 1.78	1869 / 1492

library, which was then characterized for 100 nm technology [28]. Column 1 in the table shows a benchmark circuit, while Column 2 shows the block area. Columns 3, 4, 5, and 6 show the average row utilization percentage, the circuit delay in ps, the maximum congestion, and the total overflow, respectively. The entries of the form 'a / b' in these columns have the same meaning as before. Congestion-aware netlists tend to have, on the average, slightly larger gate-area, which is reflected in overall 1% increase in the average row utilization percentage. The delays in congestion-aware netlists, on the average, are smaller as compared to the corresponding conventional netlists in most cases because of the application of congestion-aware mode only on non-critical paths. Track overflows and maximum congestion have improved consistently for all the benchmarks due to congestion-aware mapping; the average improvement over conventional mapping is 20% and 19%, respectively. Results on large benchmark circuits, such as b14, b15, and b20, from ITC99 benchmark suite [29] show similar improvements. This shows that our heuristic is effective in alleviating the routing congestion without penalizing the delay values. However, there is a limitation: in case of benchmarks such as C6288 and b14, where the number of nodes on non-critical paths is small because of their cone-like structure, the congestion-aware mapper has limited flexibility, and this is reflected in relatively smaller improvements in track overflows. In case of circuits with a relatively large number of nodes on non-critical paths, such as C7552, congestion-aware mapper has not only improved track overflows by 46% but also the delay by 16%.

TABLE IV
CONGESTION-AWARE MAPPING RESULTS OBTAINED WITH DIFFERENT THRESHOLDS. COLUMNS WITH LABELS'RU' AND 'MC' DENOTE THE AVERAGE ROW UTILIZATION AND THE MAXIMUM ROUTING CONGESTION IN A CIRCUIT, RESPECTIVELY.

Benchmark	C_{max}	RU (%)	Delay (ps)	MC	Overflow
C1908	0.9	80	1030	1.10	75
C1908	1.0	80	1065	1.10	87
C1908	1.1	80	1055	1.10	88
C7552	0.9	75	1297	2.10	1090
C7552	1.0	75	1255	1.80	743
C7552	1.1	75	1307	2.20	1120

Table IV shows the effect of varying threshold congestion (C_{max}) on congestion-aware mapping. It can be seen that overflow/delay results are somewhat insensitive to threshold congestion in case of C1908, while they have a somewhat larger variation for C7552, but are still better than conventional mapping. Table V shows a comparison of run-times in seconds for conventional and congestion-aware mapping based flows. The mapping and placement is performed on Sun Ultra Sparc 60 machine with 400MHz clock speed, while routing is carried out on Intel Pentium Pro machine with 3.2GHz clock speed. The run-time of congestion-aware mapping is slightly worse than conventional mapping, as expected, because of the two rounds of matching and congestion cost computation; it is 17% larger than its conventional counterpart, on an average. It can be observed that placement of subject graph requires substantial run-time, but is required for both conventional and congestion-aware mapping. (For conventional mapping, it is employed to compute wire-loads and wire-delays based on locations of the subject graph nodes, instead of using the relatively inaccurate fanout-based wireload model, which often ignores wire-delays).

TABLE V
COMPARISON OF RUN-TIMES FOR CONVENTIONAL DELAY-ORIENTED MAPPING AND CONGESTION-AWARE DELAY-ORIENTED MAPPING. PLACEMENT AND ROUTING IS PERFORMED EMPLOYING A PUBLICLY AVAILABLE PLACER CAPO [15] AND A ROUTER [26], RESPECTIVELY, FOR 100 NM TECHNOLOGY [28]. MAPPING AND PLACEMENT IS PERFORMED ON SUN ULTRA SPARC^{TM} 60 MACHINE WITH 400MHZ CLOCK SPEED, WHILE ROUTING IS CARRIED OUT ON INTEL PENTIUM PRO^{TM} PROCESSOR-BASED MACHINE WITH 3.2GHZ CLOCK SPEED. IN THE TABLE, 'SG' REFERS TO THE SUBJECT GRAPH FOR A PARTICULAR CIRCUIT.

Example	SG placement	Mapping	Placement	Routing
	Run-time (s)	Runtime (s)	Runtime (s)	Runtime (s)
b14	721	297 / 348	193 / 185	258.14 / 266.21
b15	967	486 / 584	248 / 294	283.70 / 295.00
b20	1483	885 / 1026	407 / 412	717.04 / 718.83
C1355	50	11 / 13	15 / 15	1.47 / 1.59
C1908	47	12 / 14	14 / 14	1.68 / 1.70
C432	26	7 / 8	8 / 8	1.15 / 1.05
C499	44	11 / 12	14 / 15	2.05 / 2.00
C6288	277	88 / 117	86 / 99	90.60 / 86.48
C7552	324	190 / 203	85 / 106	111.28 / 120.77
C880	52	12 / 17	17 / 16	2.81 / 2.07
Average		199 / 234	108 / 117	146.99 / 149.57

C. Wirelength and Detour Distributions

For large benchmarks, the wiring distributions obtained after global routing showed significant improvements as a result of our congestion-aware area-oriented technology mapping flow. The improvement in the wiring distribution is best exemplified by a reduction in the incidence of detours on the routes, where we define the detour of a route as the difference between its actual length and the total size of its minimum spanning tree (MST[6]). These results are summarized in Table VI, and are further explained using histograms and scatter plots of netlengths and detours for benchmark circuit IDC. In the table, entries of the form 'a / b' have the same meaning as before. Figure 8 shows a plot of the number of nets vs. their detour for the benchmark IDC. Similar wire distribution plots that are obtained for benchmarks C6288 and C7552 can be found in [18]. In the figure, the log-scale Y-axis shows the number of nets, while the X-axis shows the detour, in μm, for all the nets on a linear scale. The height of a

[6]Because of the canonicity of MST's, MST estimates are used to compute the detours even though they tend to be overestimates as compared to minimum Steiner estimates.

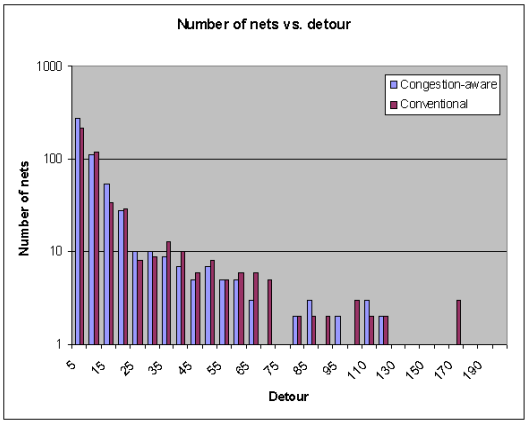


Fig. 8. Number of nets vs. detour length (μm) for the IDC circuit. The placements of the conventionally mapped netlists and those of the premapped and mapped netlists, in case of congestion-aware mapping, are performed using Kraftwerk.

TABLE VI
SUMMARY OF TOTAL WIRELENGTH AND DETOUR AFTER GLOBAL AND DETAILED ROUTING FOR CONGESTION-AWARE AREA-ORIENTED TECHNOLOGY MAPPING. PLACEMENT AND ROUTING IS PERFORMED USING KRAFTWERK AND A PROPRIETARY ROUTER, RESPECTIVELY, FOR A 90 NM TECHNOLOGY.

Example	Global routing	Detailed routing	Detailed routing
	Total wirelength (μm)	Total wirelength (μm)	Detour length (μm)
IDC	25278 / 24320	27360 / 26940	5747 / 4809
C7552	58717 / 57464	62412 / 60862	9922 / 5965
C6288	53155 / 50944	58361 / 55236	7816 / 7016

brown (purple) bar[7] in the figure represents the number of nets in the conventional (congestion-aware) netlist for a given detour range. It can be observed that for shorter detour ranges, the number of nets in the congestion-aware netlist dominates their conventional counterpart, while as the detour length increases, the number of nets from the conventional netlist dominates that in the congestion-aware netlist. Although the total number of wires increases in the congestion-aware case, most of this increase occurs at short wire lengths, as can be seen from the figure.

Figure 9 shows a plot of netlength vs. detour length for the nets whose length is greater than 100 μm in congestion-aware and conventionally mapped netlist for IDC. These are the nets that are usually responsible for the routing problems. In the figure, the symbols '+' and '$\times$' indicate the actual length, in μm, of a net belonging to the corresponding detour range, in μm, specified on the X-axis, for the congestion-aware and conventionally mapped netlist, respectively. Thus, for instance, a '$\times$' corresponding to 230 μm on the Y-axis and in the column for 50 μm on the X-axis implies that there is a net of length 230 μm whose detour length lies between 47.5 to 52.5 μm in the conventional netlist. One can observe that the congestion-aware technology mapping not only tends to reduce the length of the long wires, but also tends to route them with smaller detours, and hence, makes them more predictable prior to the routing. For small detour ranges, the cumulative wirelength in the congestion-aware mapped netlist dominates the corresponding wirelength in the conventionally mapped netlist as implied by Figure 8. This is not only because more wires tend to have short detours, but also due to an increase in the number of short wires. As the detour length

[7]In grayscale print, brown and purple bars are dark-colored and light-colored bars, respectively.

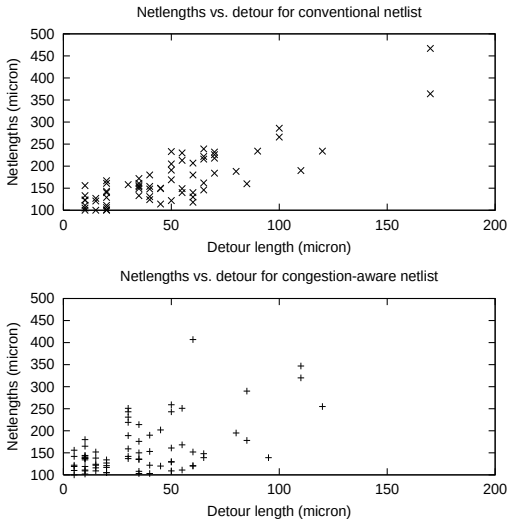


Fig. 9. Scatter plots of netlengths vs. detour length (μm) for nets with length greater than 100 (μm) in the IDC circuit. In these plots, 'x' and '+' denote a net in conventional and congestion-aware netlist, respectively.

increases, the cumulative wirelength for the conventionally mapped netlist dominates its counterpart for congestion-aware netlist, since there are more long wires with larger detours in the conventional netlist than in the congestion-aware netlist. Thus, congestion-aware mapping tends to reduce large detours by increasing the number of short wires and allowing more wires to have smaller detours. It also improves the total wirelength, although marginally, as seen from Table VI. Furthermore, the reduction in the detours of the wires under congestion-aware mapping also improves the predictability of their length, delay, load, and repeater requirements during mapping and placement.

D. Summary of Experimental Deductions

The following conclusions may be drawn from the experimental results.

1) The congestion-aware algorithms for area and delay-oriented technology mapping show consistent improvements in track overflows over conventional mapping methods. The improvement is significant: 37% in the case of area-oriented mapping, and 20% in the case of delay-oriented mapping while preserving or improving delay. These results indicate that technology mapping is indeed effective in handling routing congestion.
2) The consistency in the results also indicate that our heuristics are effective. More importantly, it justifies the use of congestion map prediction employing premapped netlists to guide the mapping process.
3) As compared to conventionally mapped netlists, congestion-aware netlists tend to have better wirelength distribution: typically, the length of the longest wire is reduced, the number of nets with long detours (which are usually hard to model accurately during mapping) is smaller, the total wirelength is slightly smaller, and there is an increase in the number of short wires.

VIII. CONCLUSION

In this paper, we have proposed technology mapping algorithms for alleviating the routing congestion. These algorithms employ a predictive congestion map based on the premapped netlists. Using empirical data, we have shown that there exists a strong correlation between

the predictive congestion map based on a premapped netlist and the congestion map of the corresponding mapped netlist. This empirical evidence is utilized to justify the use of predictive congestion maps to guide the technology mapping algorithms. These algorithms employ congestion cost functions such that in sparsely congested regions, area- or delay-optimal matches are always chosen and hence, the corresponding penalty is minimized. Experimental results due to these algorithms show, on the average, improvements of 37% and 20% in track overflows, respectively, for area and delay-oriented mappings, over conventional methods with marginal gate-area and delay penalty. Moreover, congestion-aware netlists tend to have better wirelength distributions as compared to their conventional counterparts.

IX. Appendix

Table VII shows congestion correlation for a set of ISCAS'85 and MCNC'91 benchmarks. Column 1 shows the name of the circuit, while columns 2 and 3 show the scripts with mapping objective and the number of cells in the mapped netlists, respectively. The lib2.genlib library in SIS [17], which is characterized for 100 nm [28] technology, is employed for technology mapping that targets area and delay minimization followed by fanout optimization. The library contains up to four instances of different sizes for each cell in the library. The fanout optimization of the mapped netlists, which possibly restructures the network, may affect the global connectivity adversely. The circuits are placed using Capo [15]. Columns 4 and 5 in the table show the statistical correlation, respectively, for horizontal and vertical congestion between the congestion in premapped and mapped netlist. The maximum and average congestion for these netlists can be found in [18]. It can be observed that a strong congestion correlation exists between premapped and mapped netlist in spite of the fanout optimization.

TABLE VII

CONGESTION COMPARISON FOR THE NETLISTS BEFORE AND AFTER TECHNOLOGY MAPPING. ALL THE NETLISTS ARE PLACED USING CAPO [15]. FOR NETLISTS OF CIRCUITS FROM ISCAS'85 AND MCNC'91 SUITE, OBTAINED USING DIFFERENT SCRIPTS AND MAPPING OPTIONS, THIS TABLE SHOWS CONGESTION CORRELATION BETWEEN MAPPED AND CORRESPONDING PREMAPPED NETLISTS. THE SYMBOLS H AND V CORRESPOND TO HORIZONTAL AND VERTICAL CONGESTION IN THE CIRCUIT, RESPECTIVELY.

Example	script/mapping	# Cells	Correlation	
			H	V
9sym	rugged/area	314	0.79	0.82
9sym	rugged/delay	422	0.84	0.87
9sym	algebraic/area	283	0.83	0.88
9sym	algebraic/delay	341	0.81	0.84
9sym	boolean/area	284	0.83	0.88
9sym	boolean/delay	397	0.78	0.80
rd84	rugged/area	406	0.76	0.82
rd84	rugged/delay	459	0.81	0.84
rd84	algebraic/area	672	0.79	0.86
rd84	algebraic/delay	680	0.79	0.78
rd84	boolean/area	559	0.81	0.81
rd84	boolean/delay	629	0.75	0.73
alu2	rugged/area	353	0.78	0.84
alu2	rugged/delay	454	0.76	0.75
alu2	algebraic/area	405	0.83	0.89
alu2	algebraic/delay	441	0.76	0.79
alu2	boolean/area	457	0.68	0.70
alu2	boolean/delay	579	0.71	0.67
C1355	rugged/area	356	0.82	0.86
C1355	rugged/delay	422	0.70	0.72
C1355	algebraic/area	602	0.76	0.78
C1355	algebraic/delay	638	0.79	0.85
C1355	boolean/area	601	0.76	0.74
C1355	boolean/delay	654	0.74	0.83
C1908	rugged/area	386	0.83	0.84
C1908	rugged/delay	479	0.79	0.80
C1908	algebraic/area	342	0.88	0.90
C1908	algebraic/delay	580	0.81	0.82
C1908	boolean/area	569	0.82	0.84
C1908	boolean/delay	620	0.82	0.82
C499	rugged/area	391	0.80	0.78
C499	rugged/delay	402	0.63	0.64
C499	algebraic/area	593	0.74	0.82
C499	algebraic/delay	641	0.76	0.79
C499	boolean/area	611	0.77	0.77
C499	boolean/delay	642	0.69	0.69
C880	rugged/area	328	0.85	0.85
C880	rugged/delay	557	0.68	0.66
C880	algebraic/area	409	0.82	0.85
C880	algebraic/delay	410	0.81	0.82
C880	boolean/area	448	0.79	0.74
C880	boolean/delay	597	0.72	0.72
C7552	rugged/area	1930	0.66	0.64
C7552	rugged/delay	2688	0.63	0.66
C7552	algebraic/area	2378	0.77	0.78
C7552	algebraic/delay	2279	0.68	0.68
C7552	boolean/area	2321	0.71	0.70
C7552	boolean/delay	2735	0.64	0.65

Acknowledgment

The authors would like to thank the anonymous reviewers for their thoughtful comments, which helped improve the manuscript. The first author would like to thank Cristinel Ababei from the University of Minnesota and the members of PlatoCBD™ and Quasar™ teams at Intel Corporation for their invaluable help on the design flow.

References

[1] G. E. Moore, "Cramming more components onto integrated circuits," in *Electronics Magazine*, vol. 38, pp. 114–117, Apr. 1965.

[2] B. S. Landman and R. L. Russo, "On a pin versus block relationship for partitions of logic graphs," *IEEE Transactions on Computers*, vol. C-20, pp. 1469–1479, 1971.

[3] P. Christie and D. Stroobandt, "The interpretation and application of Rent's rule," *IEEE Transactions on Very Large Scale Integrated Systems*, vol. 8, pp. 639–648, Dec. 2000.

[4] S. Borkar, "Obeying Moore's law beyond 0.18 micron," in *Proceedings of the IEEE International ASIC/SOC Conference*, pp. 26–31, Sept. 2000.

[5] A. B. Kahng, P. Rodman, and P. Villarubia, "Physical chip implementation methodology: Hot spots and best practices." Tutorial at ACM/IEEE Design Automation Conference, June 2002. Available at `http://vlsicad.ucsd.edu/~abk/TALKS/`.

[6] W. Gosti, A. Narayan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Wireplanning in logic synthesis," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 26–33, Nov. 1998.

[7] W. Gosti, S. R. Khatri, and A. L. Sangiovanni-Vincentelli, "Addressing timing closure problem by integrating logic optimization and placement," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 224–231, Nov. 2001.

[8] L. Stok and T. Kutzschebauch, "Congestion aware layout driven logic synthesis," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, pp. 216–223, Nov. 2001.

[9] D. Pandini, L. T. Pileggi, and A. J. Strojwas, "Congestion aware logic synthesis," in *Proceedings of Design Automation and Test in Europe*, pp. 664–671, Mar. 2002.

[10] D. Pandini, L. T. Pileggi, and A. J. Strojwas, "Global and local congestion optimization in technology mapping," *IEEE Transactions on Computer-Aided Design of Integrated Circuits aand Systems*, vol. 22, pp. 498–505, Apr. 2003.

[11] L. Stok, M. A. Iyer, and A. Sullivan, "Wavefront technology mapping," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, pp. 531–536, Nov. 1999.

[12] J. Lou, S. Thakur, S. Krishnamoorthy, and H. S. Sheng, "Estimating routing congestion using probabilistic analysis," *IEEE Transactions on Computer-Aided Design of Integrated circuits and Systems*, vol. 21, pp. 32–41, Jan. 2002.

[13] R. S. Shelar, S. S. Sapatnekar, P. Saxena, and X. Wang, "A predictive

distributed congestion metric and its application to technology mapping," in *Proceedings of the ACM International Symposium on Physical Design*, pp. 210–217, Apr. 2004.
[14] P. Kudva, A. Sullivan, and W. Dougherty, "Metrics for structural logic synthesis," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 551–556, Nov. 2002.
[15] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Can recursive bisection alone produce routable placements?," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 477–482, June 2000.
[16] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 269–274, June 1998.
[17] E. M. Sentovich, "SIS: A system for sequential circuit synthesis." Memorandum No. UCB/ERL M92/41, May 1992.
[18] R. S. Shelar, *Synthesis for Nanometer Technologies*. PhD thesis, University of Minnesota, Minneapolis, May 2004.
[19] M. H. DeGroot and M. J. Schervish, *Probability and Statistics*. Addison Wesley, Boston, MA, 3rd ed., 2001.
[20] K. Keutzer, "DAGON: Technology Binding and Local Optimization by DAG Matching," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 341–347, June 1987.
[21] J. Westra, C. Bartels, and P. Groenveld, "Probabilistic congestion prediction," in *Proceedings of the ACM International Symposium on Physical Design*, pp. 204–209, Apr. 2004.
[22] L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Transactions on Computer-Aided Design of Integrated circuits and Systems*, vol. 9, pp. 352–366, Apr. 1990.
[23] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *Journal of Applied Physics*, vol. 19, pp. 55–63, January 1948.
[24] P. Saxena and B. Halpin, "Modeling repeaters explicitly within analytical placement," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 699–704, June 2004.
[25] M. Pedram, "Logical-physical co-design for deep sub-micron circuits: Challenges and solutions," in *Proceedings of the Asia South Pacific Design Automation Conference*, pp. 137–142, Feb. 1998.
[26] J. Hu and S. Sapatnekar, "A timing-constrained simultaneous global routing algorithm," *IEEE Transactions on Computer-Aided Design of Integrated circuits and Systems*, vol. 21, pp. 1025–1036, Sept. 2002.
[27] P. Saxena, N. Menezes, P. Cocchini, and D. A. Kirkpatrick, "Repeater scaling and its impact on CAD," *IEEE Transactions on Computer-Aided Design of Integrated circuits and Systems*, vol. 23, pp. 451–463, Apr. 2004.
[28] "Berkeley predictive technology model." `http://www-device.eecs.berkeley.edu/~ptm/download.html`.
[29] F. Corno, M. S. Reorda, and G. Squillero, "RT-level ITC 99 benchmarks and first ATPG results," *IEEE Design & Test of Computers*, pp. 44–53, July 2000.

Rupesh S. Shelar received the B. E. degree in instrumentation engineering from the Marathwada University, Aurangabad, India in 1997, the M. Tech. degree in electrical engineering with specialization in microelectronics from the Indian Institute of Technology, Mumbai in 1999, and the Ph. D. degree in electrical engineering from the University of Minnesota, Minneapolis in 2004. He was a software engineer with Silicon Automation Systems, India from 1999 to 2000. He spent the summers of 2002 and 2003 at Strategic CAD Labs, Intel researching congestion-aware logic synthesis. He is currently a senior component design engineer in the desktop products group at Intel, where he works on the backend design methodology for a 65 nm Pentium 4 design. His research interests include logic synthesis and physical design.

Sachin S. Sapatnekar received the B.Tech. degree from the Indian Institute of Technology, Bombay in 1987, the M.S. degree from Syracuse University in 1989, and the Ph.D. degree from the University of Illinois at Urbana-Champaign in 1992. From 1992 to 1997, he was an assistant professor in the Department of Electrical and Computer Engineering at Iowa State University. He is currently a Professor in the Department of Electrical and Computer Engineering at the University of Minnesota. He has coauthored two books, "Timing Analysis and Optimization of Sequential Circuits," and "Design Automation for Timing-Driven Layout Synthesis," and is a co-editor of a volume, "Layout Optimizations in VLSI Designs," all published by Kluwer. He has been an Associate Editor for the IEEE Transactions on VLSI Systems, the IEEE Transactions on CAD, and the IEEE Transactions on Circuits and Systems II, has served on the Technical Program Committee for various conferences, as Technical Program and General Chair for the Tau workshop and the International Symposium on Physical Design. He is currently a Distinguished Visitor for the IEEE Computer Society and a Distinguished Lecturer for the IEEE Circuits and Systems Society. He is a recipient of the NSF Career Award and SRC Technical Excellence Award, and received the best paper awards at DAC 1997,2001, and 2003, and at ICCD 1998.

Prashant Saxena received the B.E.(Hons.) degree in electrical and electronic engineering and the M.Sc.(Tech.) degree in computer science, both from the Birla Institute of Technology and Science, Pilani, India, in 1991, and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, Urbana, IL, in 1998.

He has been with the Strategic CAD Labs, Intel Corporation, Hillsboro, OR since 1998, where he is currently a Staff Engineer. He co-developed the layout methodology for domino logic synthesis used on the 180 nm Pentium 4 design, and pioneered the noise convergence methodology for the standard cell based portions of the 90 nm Pentium 4 design. He has authored or co-authored more than 20 papers in refereed conferences and journals, one book chapter, and the specification for the Virtual Socket Interface Alliance (VSIA) signal integrity standard. His current research interests are in physical synthesis and layout, signal integrity, and process scaling issues.

Dr. Saxena has served on the technical program committee for the ISPD, ISCAS and IWSOC conferences, as well as on the several National Science Foundation (NSF) panels and Semiconductor Research Corporation (SRC) task forces. As an invited speaker (and subsequently a panelist) at ISPD, he helped spotlight the upcoming buffer explosion problem. He was a recipient of Intel Architecture Group Trailblazer Award in 2000 and a Best Paper Award at Intel's internal technical conference in 2003.

Xinning Wang received her M.S. and Ph.D. in Computer Engineering from Syracuse University (Syracuse, New York) in 1989 and 1992, respectively. She has been with Intel Corporation since 1992 and involved in R&D of a wide range of CAD tools used in high performance microprocessor design and verification. Currently she is a senior staff CAD researcher in Strategic CAD Labs and leads Intel's in-house logic synthesis research effort. Her research interests include design methodologies and frameworks for unified design and verification, integrated/interactive logic and physical synthesis, synthesis and optimization techniques for high-performance and structured circuits.