Predictable Routing

Ryan Kastner, Elaheh Bozorgzadeh and Majid Sarrafzadeh
Department of Electrical and Computer Engineering
Northwestern University, Evanston, IL 60208
kastner,elib,majid@ece.northwestern.edu

Abstract

Predictable routing is the concept of using prespecified patterns to route a net. By doing this, we allow an more accurate prediction mechanism for metrics such as congestion and wirelength earlier in the design flow. Additionally, we can better plan the routes, insert buffers and perform wire sizing earlier. With comparable routing quality, we show that we can predictably route up to 80% of a selected subset of nets. Also, we introduce methods for finding a group of nets which can be predictably routed.

1 Introduction

The process of routing can be divided into two subproblems, global and detailed routing. Global routing decomposes the routing problem into smaller, manageable routings for the detailed router. Specifically, the global router finds a rough path for each net while trying to reduce the chip size, shortening the wire length and distributing the congestion across the routing area, among other things [5, 7, 11]. Detailed routing uses the results of global routing to find an exact realization of the interconnections in VLSI circuits. The focus of this paper is on the global routing problem.

Predictable routing is the idea of using prespecified patterns to route a net. This is particularly useful for CAD tools preceding global routing in the design flow. For example, most placement tools use quick routing metrics to find congestion and wirelength information. In this paper, we develop quick routing methods that do not affect the quality of the routing solution. Since we know these metrics will not affect the routing, congestion and wirelength are more accurately modeled earlier in the design flow. Also, since we know the route of a net, we can start wire sizing, wire planning and optimally add buffers[1] without going through the time consuming process of routing. As fabrication technology moves into deep submicron (DSM) device sizes, local interconnect effects have an increasingly dominant role [2]. These effects include increasing interconnect capacitance and resistance. Logic and behavioral synthesis tools have no accurate way of modeling interconnect. Predictable routing provides these tools with an accurate interconnect prediction mechanism. These ideas are further explained in Section 3.1.

In this paper, we propose modifications to the global routing algorithm. These modifications allow nets to be predictably routed with little or no loss in the quality of the global routing solution. In Section 2, we discuss the idea of congestion and briefly review maze routing. Section 3 introduces the idea of predictable routing through pattern routing. We discuss the 1-density routing problem which uses the pattern routing concept. In Section 4, we present methods for finding a subset of nets which can be predictably routed. We conclude in Section 5.

2 Preliminaries

A *grid graph* is a graph G(V,E) such that each vertex corresponds to a point in a plane. See Figure 1 for further explanation. A *net* = $\{(x_1, y_1,), (x_2, y_2), (x_3, y_3), ..., (x_n, y_n)\}$ is an unordered set of points on a grid graph. A single point of the net is referred to as a *terminal*. A *routing* of a net is a set of grid edges such that the terminals are fully connected. The *route edges* of a net are the set of edges used in the routing of that net.

A *global bin* is a rectangular partition of the chip. By partitioning the chip into many rectangular regions and placing the cells into these regions, we have a placement using global bins. The boundaries of the global bins are *global bin edges*.

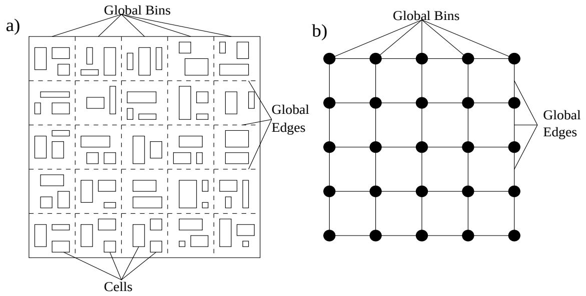

Figure 1: **(a)** Placement of cells into global bins. **(b)** The corresponding grid graph.

In this paper, we assume that a global placement of cells and their interconnections are given by some placement engine (our experiments used Dragon [6] which is comparable in quality to commercial version of Timberwolf [13]). The cells are placed into global bins and each cell is assumed to be placed in the center of the global bin. Looking at Figure 1, it is easy to see that the global bins and edges can be transformed into a grid graph. The interconnections between the cells can be modeled by nets.

2.1 Congestion

Congestion in a layout means that there are too many nets routed in a local area. This causes difficulty for the detailed router as it may not find a feasible routing solution. We want to evenly distribute the routing across the total chip area.

The congestion of an edge is the number of nets routed over a global bin edge. The *capacity* (also referred to as supply) of edge e is c_e. It is the maximum number of nets that can be routed over e. c_e is a fixed value that is based on the length of the edge and the technology used in creating the chip. The routing demand of e, specified as d_e, is defined as the number of route edges crossing e. Similarly, the demand of a vertex v is d_v. Here the demand corresponds to the number of routes that pass though the vertex v (equivalently the global bin v). An edge is overflown if and only if the $d_e > c_e$. Formally, the overflow of an edge is:

$$overflow_e = \begin{cases} d_e - c_e - t & \text{if } d_e > c_e \\ 0 & \text{otherwise} \end{cases}$$

t is a threshold value which allows d_e to go above c_e without an overflow penalty. t is included because you can often route up to t nets though neighboring bins without affecting the congestion of those bins. t is

[1]If we know the net topology the complexity of buffer insertion for delay becomes polynomial time solvable [4].

usually a small constant (approx. 2-5). Using the global bin and global edge notation, the total overflow of a routing is:

$$overflow = \sum_{e=1}^{n} overflow_e$$

where n is the number of bin edges. The total overflow reflects the shortage of routing resources for a particular set of edge capacities. A routing with a minimized total overflow is one of the objectives of our global router. Our industrial experience shows that total overflow is a good measure of congestion.

2.2 Global Maze Routing

We implemented a global maze router. The maze router takes every net and routes them one at a time according to a cost function.

$$overflow_{route} = \sum_{e \in RouteEdges} overflow_e$$

$$length_{route} = \mid RouteEdges \mid$$

$$cost_{route} = \alpha \times overflow_{route} + length_{route}$$

$$cost_{total} = \sum_{allnets} cost_{route}$$

There is a tradeoff between minimizing overflow and minimizing wire length. Ideally, you could minimize both concurrently. Most often this is not possible. Our cost function can solely minimize wire length (set $\alpha = 0$). Likewise you can minimize overflow by setting $\alpha \gg 1$. We found that varying α from 10 to 100 minimizes the total overflow while keeping the wire length minimal.

For nets with more than two-terminals, we use Steiner trees to partition the net into a set of two-terminal nets. There is much research on Steiner trees and the Steiner minimal tree (SMT), a Steiner tree with minimum total cost. Our methods can use any SMT algorithm for two-terminal decomposition.

Each net is given an initial route and then a rip-up and reroute phase is applied to further minimize the total overflow. This technique (or variants of it) appears in most global routers in order to deal with the net ordering problem [10]. During rip-up and reroute, the bin edges are sequentially searched. If an edge is overflown, then all of the nets that pass through that edge are ripped and rerouted. This process continues until the total cost converges to a local minimum. That is, if the total cost does not decrease (the goal is to minimize the total cost) after λ iterations, the rip-up and reroute process has completed. We found that a λ of 200 gave good results for the designs that we tested. Larger designs may need an increased λ which decreases the chance of getting stuck in a local minimum. In general, smaller designs can afford to decrease λ which would decrease the runtime.

3 Predictable Routing

3.1 Pattern Routing

Pattern routing is the notion of using predefined patterns to route two-terminal nets. Usually these are simple patterns such as a L-shaped (sometimes called 1-bend) or a Z-shaped pattern (2-bends, route restricted within bounding box) as shown in figure 2.

Patterns can speed up the global routing process. Instead of maze routing every net, we pattern route a portion of the nets. In general, maze routing will consider many bins that the final route will not actually use. When using pattern routing, only a constant number of edges are searched for overflow information. For example, L-shaped pattern routing will only search the edges on the bounding box of the two-terminal nets. Then, depending on cost of these edges, it will choose the upper-L or lower-L and place the route there. Similarly, Z-shaped pattern routing needs to search the edges on the perimeter and inside the two-terminal bounding box. On the other hand, maze routing will search every edge (on the worst case). Therefore, pattern routing has a better upper bound

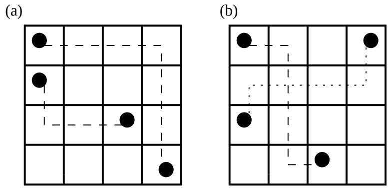

Figure 2: **(a)** L-shaped routing of 2 two-terminal nets. **(b)** Z-shaped routing of 2 nets.

on runtime complexity. We found that on average, the pattern routing approach searches fewer edges than the maze router. We formally summarize the complexities:

1. Given a net $n = \{(x_1, y_1), (x_2, y_2)\}$ and a grid graph G(V,E).
2. Let A be the edges on and within the bounding box of n. A $\subseteq$ E. $|A| = 2 \cdot |x_1 - x_2| \cdot |y_1 - y_2| + |x_1 - x_2| + |y_1 - y_2|$
3. Let P be the edges on the bounding box of n. P $\subseteq$ A. $|P| = 2 \cdot (|x_1 - x_2| + |y_1 - y_2|)$
4. Maze routing - $O(|E|)$
5. L-shaped pattern routing - $O(|P|)$
6. Z-shaped pattern routing - $O(|A|)$

Theorem 3.1: $|P| \leq |A| \leq |E|$.
Proof: The proof is trivial since, by definition, $P \subseteq A \subseteq E$. □

The maze router ensures that the least cost route (according to the cost function) is found. Pattern routing does not give you this luxury. In fact, an L-shaped pattern routing could produce the second worst possible route. This occurs if both the upper-L route and the lower-L route are the two worst paths. Pattern routing will choose the better of these two solutions, giving you a bad routing. In general this is not the case, as our results show.

Another benefit of pattern routing lies in the predictability of a pattern-routed net. If you know that a net will be pattern routed, you can quickly and accurately estimate its route at a higher level. For example, you know that an L-shaped pattern route will take one of two routes. This allows higher level CAD tools, such as the placement or synthesis engines, to estimate routings which will lead to better congestion and area estimates. In order to exploit predictability, the tools need placement information. Many industrial logic synthesis tools are moving towards layout-driven synthesis. Additionally, an academic behavioral level synthesis tool has recently incorporated placement information [12].

With emergence of deep submicron (DSM) fabrication technology, interconnect has an increasingly dominant role. Now the circuit delays are determined by the gate resistance and capacitance as well as the interconnect resistance and capacitance [2]. When optimizing for delay in a circuit, logic synthesis tools look at the critical path and often ignore the interconnect. If we could predictably route the gates on the critical path, then we more accurately estimate the interconnect resistance and capacitance.

The number of vias on a pattern-routed net is fixed. Since vias further increase the capacitance and resistance, it is beneficial to keep them at a minimum. Also, vias negatively affect the routability of the circuit [1].

Pattern routing can be used to minimize coupling in global and detailed routing [8].

3.2 1-density Routing

The *1-density (1-d) routing* problem tries to find a 1-bend routing of two-terminal nets so that no two routings overlap (crossing of wires is allowed). Let us define the 1-density routing problem formally:

1. Given a set of two-terminal nets N and a grid graph G(V,E).
2. Does there exist a 1-bend routing for every net n $\in$ N such that $d_e \leq 1$ for every edge e $\in$ E?

Theoretical aspects as well as detailed description of algorithms for this problem are discussed in [9].

4 Evaluation

In this section, we show the effect of pattern routing on the quality of the routing solution. We show that you can pattern route up to 80% of the nets with smallest bounding boxes while incurring loss of quality. Then, we show how a set of nets that satisfies the 1-density routing problem can be pattern routed without sacrificing the routing quality. This gives us the ability to predictably route a subset of all the nets, even if the nets have a large bounding box.

To perform our experiments, we used five MCNC standard-cell benchmark circuits [3] (see Table 1. The circuits were placed into global bins using the Dragon placement engine [6]. Some of the benchmarks (e.g. prim1 and prim1.2) are repeated. Repeated benchmarks differ in the number of global bins; they consist of the same number of nets, cells and pins but may have a completely different placement.

Data file	Num Cells	Num Nets	Num Pins	Global Bins
prim1	833	1156	3303	8 X 16
prim1.2	833	1156	3303	16 X 16
prim2	3014	3671	12014	8 X 16
prim2.2	3014	3671	12014	32 X 32
avqs	21584	30038	84081	30 X 80
avqs.2	21584	30038	84081	80 X 80
biomed	6417	7052	22253	20 X 40
biomed.2	6417	7052	22253	40 X 40
struct	1888	1920	5407	20 X 16

Table 1: Benchmark circuit information

For our experimental results, we choose to use L-shaped pattern routing over Z-shaped for a several reasons. First, for two-terminal nets there are only two possible L-shaped routes to consider. The number of Z-shaped routes grows linearly with the bounding box size. Since we are aiming towards predictable routes, L-shaped patterns reduce the choices of routings. Secondly, we want the predictable routes to be chosen quickly. Once again, the time to find the congestion of the routes is $O(|P|)$ whereas the Z-shaped routes is $O(|A|)$. Theorem 3.1 states the $|P| \leq |A|$. Also, our experiments show that the congestion costs using Z-shaped routing gives a congestion that is close to that of L-shaped routings. Figure 3 shows the congestion when we lock a percentage of the nets. We route the locked nets considering every possible route (maze), L-shaped and Z-shaped routes. An experiment was done that locked x% of the smallest (in terms of bounding box) nets. A locked net can only be routed once; it cannot be considered for rip-up and reroute. The locked nets are routed by L-shaped, Z-shaped or maze routings. These nets will choose the route with minimum congestion. After the locked nets are routed, the remaining unlocked nets are maze routed and are considered in the rip-up and reroute stage. The y-axis is scaled to the congestion when there are no locked nets. Therefore, point at (5, 1.10) means that if 5% of the nets are locked, then the congestion is 10% greater than that where every net is maze routed and considered in the rip-up and reroute stage.

We comment on a few observations. Even though pure maze routing has the greatest freedom in terms of finding the least congested solution, the overall algorithm is a heuristic therefore it is not guaranteed to find an optimal solution. Therefore, pattern routing nets may lead the heuristic to better solutions. This is seen in Figure 3. When we lock some of the nets, we get a better overall congestion. The tradeoff between fast routing time, reduced number of routings (better predictability) and quality of solution favors L-shaped routing. Therefore, we will exclusively use L-shaped routing for all of our pattern routing experiments.

Our experiments focused on determining which nets to pattern route while incurring little to no congestion penalty. Our first heuristic (referred to as the Largest First Pattern Route or LFPR heuristic) split the multi-terminal nets into two-terminal nets and sorted them from largest bounding box to smallest bounding box. Then, we pattern routed the x% largest nets while maze routing the rest of the nets. The pattern routed nets were not rerouted during the rip and reroute phase. As shown in Table 2, pattern routing large nets gives unfavorable overflow results. If you pattern route only the largest 5% of the nets, your overflow increases

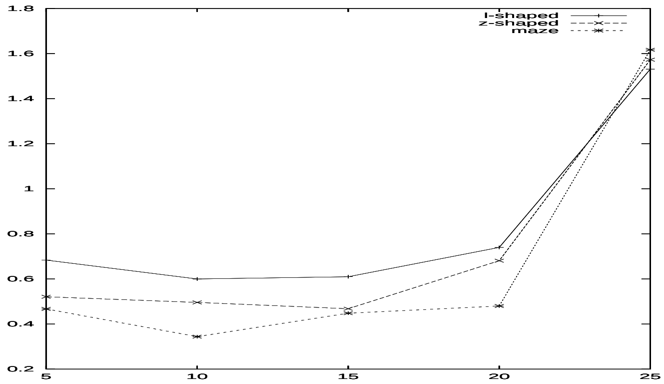

Figure 3: Comparison of congestion using maze, l-shaped or z-shaped routing. The x-axis is % of fixed nets, the y-axis is overflow compared to maze routing every net.

$\frac{1665}{7787} = 21\%$ over maze routing every net. A similar trend occurs as you increase the pattern route percentage. Pattern routing only 20% of the nets results in an overflow over 36% larger than the 0% overflow. (Note, the 0% pattern route is exactly equivalent to maze routing every net; the rip and reroute stage will consider every net.)

Datafile	0%	5%	10%	15%	20%
prim1	70	34	35	46	41
prim1.2	383	26	35	53	57
prim2	117	79	130	133	138
prim2.2	675	199	272	330	328
avqs	115	149	198	290	346
avqs.2	3148	722	1002	1220	1238
biomed	3024	283	285	327	339
biomed.2	22	126	214	248	295
struct	233	47	98	131	153
total	**7787**	**1665**	**2269**	**2778**	**2935**

Table 2: Congestion Data for Largest-First Pattern Route heuristic. 0% is the base case congestion. The remaining results take the congestion and subtract the base case congestion. So 34 means a total congestion of 70+34=104

The Smallest First Pattern Route (SFPR) heuristic gave encouraging results. This heuristic is similar to LFPR except here, we sort the two-terminal nets from smallest to largest. Thus, an LFPR of 5% will pattern route the smallest 5% of the nets. Referring to Table 3, we can see that we can pattern route up to 80% of the nets with only a small increase in overflow. In fact, pattern routing the small nets actually leads to better overflow results! These results further confirm our previous statement that pattern routing can lead the maze router to better overflow solution.

This SFPR heuristic results may seem surprising. Looking at Table 4, you can see the percentage of the total route length that the smallest x% of the nets comprises. Even when you pattern route the smallest 90% of the nets, the route length of these small nets is, on average, only 58.32% of the total route length. This means that the remaining 10% of the nets that are maze routed are much longer than the short nets. This allows the maze router enough freedom to find a good routing, even when 90% of the nets are fixed. This gives some insight as to why the LFPR heuristic does not work. If you fix the long nets to a pattern, you greatly reduce the routing freedom that the maze router needs to produce a good route. Since the small nets are close in physical proximity, there are limited number of routes that these nets could take. Therefore, the maze router may find a less congested solution, but due to the small number of feasible routes, the pattern route solution will not significantly vary from the best (i.e. maze-routed) solution. Additionally, small nets are often entirely located within a congested region. In this case, any shortest length path will be essentially equivalent in terms of overflow minimization. Since there is no quality improvement using maze routing, the pattern route is preferable due to its faster run

Datafile	0%	50%	60%	70%	80%	90%
prim1	622	-8	-13	-1	-7	-2
prim1.2	379	-3	-3	-3	1	-1
prim2	1370	-2	-10	-11	-2	18
prim2.2	665	0	-1	21	21	47
avqs	3149	-109	-25	22	-53	146
avqs.2	401	-9	3	-38	61	6
biomed	2994	-9	14	-89	-29	-43
biomed.2	15	0	0	-2	0	1
struct	769	-7	-17	-13	14	89
total	**10364**	**-147**	**-52**	**-114**	**6**	**261**

Table 3: Congestion Data for Smallest-First Pattern Route heuristic. 0% is the base case congestion. The remaining results take the congestion and subtract the base case congestion. A negative result means that the current congestion is better than the base congestion.

time and predictability.

Datafile	10%	20%	30%	50%	80%	90%
prim1	5.75	11.54	17.34	28.53	50.69	64.09
prim1.2	5.60	11.23	16.88	28.11	54.67	69.44
prim2	6.31	12.64	18.97	31.62	52.19	65.19
prim2.2	3.71	7.41	11.12	18.54	40.96	54.65
avqs	2.77	5.54	8.31	13.91	32.83	49.11
avqs.2	3.61	7.23	10.86	18.11	36.56	50.50
biomed	3.64	7.28	10.91	18.08	40.80	54.83
biomed.2	2.94	5.97	8.96	14.92	36.26	49.99
struct	3.27	6.57	9.86	21.92	52.26	67.09
avg	**4.18**	**8.38**	**12.58**	**21.53**	**44.14**	**58.32**

Table 4: Percentage of route length used by SFPR nets. For example, when you pattern route the 10% smallest nets in prim1, the route length of those nets is only 5.75% of the total route length.

We have shown that you can predictably route up to 80% of the nets with small bounding boxes (Table 3). Unfortunately, you can not do predictable routing on nets with large bounding boxes using the LFPR heuristic without suffering a huge loss in the quality of solution. Now, we will show that any set of 1-density nets can be predictably routed without degrading the solution quality. This allows us to predictably route the nets with large bounding boxes.

In Table 5, we show that predictable routing on a set of 1-d routable nets does not affect the overall routing solution quality. Since we are trying to show that nets with large bounding boxes can be predictably routed, we used a heuristic that focused on finding such nets. Like the LFPR heuristic, we sort the nets from largest to smallest bounding box. Then, we assign an upper or lower routing to the nets so that they can be 1-d routed. Therefore, some of the largest nets are always in the set of 1-d nets. Table 5 also shows the overflow results when we pattern route a set of 1-d, 2-d, 3-d, 4-d and 5-d nets. A 2-d (3-d,4-d,5-d) routing is similar to 1-d but allows a maximum edge capacity of 2 (3,4,5). Notice that some circuits allow up to 4-d routing without loss of quality. This highly depends on the number of nets and number of bins in the benchmark. For example, *avqs* is a large benchmark and the nets in the 3-d routing only account for 17.7% of the total routing. Compare this to *prim1.2* where the nets of the 3-d routing are 35.5% of the total routing. Notice that 1-d routing doesn't hurt the solution quality for all but one benchmark (here *avqs.2* seems to be an anomaly since the 2-d, 3-d and 4-d routings show little degradation of the overall routing quality).

5 Conclusion

In this paper, we argued that predictable routing is beneficial as it allows wire planning, buffer insertion and wire sizing to occur earlier in the design flow. In addition, we showed that predictable routing can help even at the global routing stage by leading the router find a better solution.

Datafile	base	1-d	2-d	3-d	4-d	5-d
prim1	622	0	-10	-4	-5	7
prim1.2	379	-3	4	9	22	39
prim2	1321	0	4	6	3	5
prim2.2	665	-3	32	30	11	42
avqs	3149	-13	-20	-121	6	20
avqs.2	401	22	3	-23	7	82
biomed	4837	-5	-41	-52	18	-7
biomed.2	47	-6	2	-4	11	6
struct	769	-4	4	24	38	56
total	**12190**	**-12**	**-22**	**-135**	**121**	**250**

Table 5: Overflow information for pattern routing a set of x-d nets. x is varied from 1 to 5. The base case is the total overflow with pure maze routing. The next columns are current overflow - base case. A lower value means better overflow hence a better solution.

We looked for nets that can be predictably routed with little degradation in the quality of the routing solution. Even with this limitation, we show that we can pattern route up to 80% of the nets. Also, we show that pattern routing works with large nets if they are 1-d routable.

Our future research will focus on better heuristics for finding a set of 1-d nets. Furthermore, we plan to integrate our ideas into high level CAD tools.

References

[1] A. Kahng, S. Mantik and D. Stroobandt. "Requirements for Models of Achievable Routing". In *Proc. International Symposium on Physical Design*, April 2000.

[2] D. Sylvester and K. Keutzer. "A Global Wiring Paradigm for Deep Submicron Design". In *IEEE Transactions on Computer Aided Design*, February 2000.

[3] K. Kozminski. "Benchmarks for Layout Synthesis - Evolution and Current Status". In *Proc. ACM/IEEE Design Automation Conference*, June 1991.

[4] L.P.P.P van Ginneken. "Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay". In *Proc. International Symposium on Circuits and Systems*, 1990.

[5] M. Sarrafzadeh and C.K. Wong. *An Introduction to VLSI Physical Design.* McGraw-Hill, New York, NY, 1996.

[6] M. Wang, X. Yang and M. Sarrafzadeh. "DRAGON: Fast Standard-Cell Placement for Large Circuits". In *Proc. IEEE International Conference on Computer Aided Design*, November 2000.

[7] N. Sherwani. *"Algorithms For VLSI Physical Design Automation"*. Kluwer Academic Publishers, Boston, MA, 1993.

[8] R. Kastner, E. Bozorgzadeh and M. Sarrafzadeh. "Coupling Aware Routing". In *IEEE International ASIC/SOC Conference*, September 2000.

[9] R. Kastner, E. Bozorgzadeh and M. Sarrafzadeh. *Predictable Routing.* Technical Report, Northwestern University Department of Electrical and Computer Engineering, 2000.

[10] R. Nair. "A Simple Yet Effective Technique for Global Wiring". In *IEEE Transactions on Computer Aided Design*, March 1987.

[11] T. Lengauer. *"Combinatorial Algorithms for Integrated Circuit Layout"*. John Wiley and Sons, New York, 1990.

[12] W. Dougherty and D. Thomas. "Unifying Behavioral Synthesis and Physical Design". In *Proc. ACM/IEEE Design Automation Conference*, June 2000.

[13] W.J. Sun and C. Sechen. "Efficient and Effective Placement for Very Large Circuits". In *Proc. International Conference on Computer Aided Design*, November 1993.