Congestion Estimation and Localization in FPGAs: A Visual Tool for Interconnect Prediction

David Yeager Darius Chiu Guy Lemieux
Dept of Electrical and Computer Engineering
The University of British Columbia
Vancouver, BC, Canada
{ dyeager | dariusc | lemieux } @ ece.ubc.ca

ABSTRACT
In this paper, we are concerned with *locating* the most congested regions in FPGA designs before routing is completed. As well, we are interested in the amount of congestion in these locations relative to surrounding areas. If this estimation is done accurately and early enough, *e.g.* prior to routing or even prior to placement, the data can be used during clustering, placement and perhaps during routing to avoid or spread out congestion before it becomes a problem. We implemented several estimation methods in the VPR tool set and visually compare estimation results to an actual routing congestion map. We find that standard image processing techniques such as blending and peak saturation considerably improve the quality of estimation for all metrics.

Categories and Subject Descriptors
B.7.2 [**Integrated Circuits**]: Design Aids – *graphics, placement and routing, layout.*

General Terms
Algorithms, Measurement, Performance, Design.

Keywords
Congestion estimation, congestion localization, FPGA routing, FPGA interconnect, interconnect prediction.

1. INTRODUCTION
Interconnect congestion prediction involves the estimation of the quantity of routing resources required prior to routing. Accurately predicting this *early* in the CAD flow allows CAD tools to plan ahead and take evasive action to reduce congested regions by spreading required routing resources more evenly. Reducing routing resources in an FPGA is extremely important, probably more than in ASICs, because logic designers cannot add more resources to an FPGA. Also, it can reduce costs for *all* FPGAs: reducing local peak congestion in an FPGA may allow an FPGA architect to design an FPGA layout tile with fewer routing resources, *i.e.*, a narrower channel width, to save costs for everyone. A CAD flow that reduces the peak routing resource requirements across a suite of benchmarks therefore allows FPGAs to have fewer routing tracks, yet maintain the same percentage of routable circuits. Because up to 80% of an FPGA's area is devoted to interconnect, this global savings can also significantly improve logic density throughout the entire chip. In contrast, spreading out congestion can also improve ASIC routability on a case-by-case basis, but it does not produce the same global savings that occurs in an FPGA. This makes good interconnect prediction potentially more valuable for FPGAs.

Interconnect prediction for FPGAs is also *different* in nature than for ASICs. Previous FPGA congestion estimation heuristics focus on the more ASIC-centric *accuracy* of predicting the *absolute peak* amount of routing resources. The quality of these heuristics is often measured in terms of global quantities such as peak routing requirements or the ability to correctly guess whether or not a circuit is routable.

Unfortunately, previous interconnect estimation techniques seldom account for features in FPGAs that make estimation different: fixed wire lengths and fixed channel widths. Fixed long wires result in routing overlap between two nets that should otherwise appear to be non-interacting based on terminal

ESTIMATE 1 ESTIMATE 2 FROM ROUTER

Figure 1: Congestion estimates along one row (left, middle) and the corresponding real congestion values obtained from routing (right). If properly rescaled, estimate 2 has greater fidelity and is more valuable.

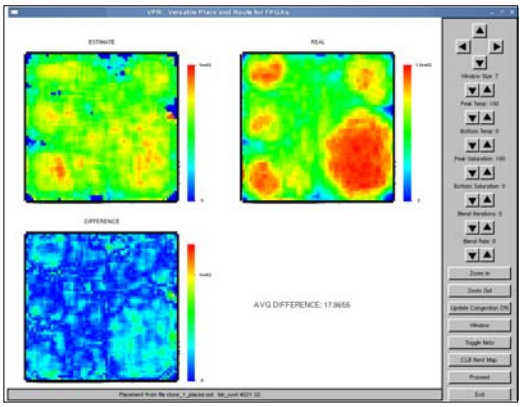

Figure 2: Congestion map display using modified VPR 4.30. The images are the congestion estimate map *E* (top left), the actual congestion map *R* (top right), and the difference abs(*E*–*R*) (bottom).

* Code available for download at http://www.ece.ubc.ca/~lemieux/downloads

SLIP'07, March 17–18, 2007, Austin, Texas, USA.

placement. Also, fixed channel width capacities force routers to use less-direct paths around congested regions. Again, estimation techniques do not adequately compensate for this – although they estimate the *peak capacity* required, they usually cannot estimate the *size of the peak region*. However, the amount and size of congestion is important in locating regions of significance. In this paper, we are concerned with how to estimate the *entire congestion map* of an FPGA. In particular, we want the hills and valleys to closely match the actual congestion profile of a circuit. We say that an estimate that closely corresponds to an actual routed congestion map across the entire device has good *fidelity*.

Figure 1 illustrates the difference between absolute accuracy and relative fidelity with a hypothetical example. Two different congestion predictions along the same single row of tiles are shown beside a third "actual" congestion profile determined by the router. Here, the first estimate is more accurate since absolute congestion values are closer to the real results. In fact, it correctly estimates peak congestion. However, the shape of the second estimate more closely matches that of the router. Even though the actual peak value itself is not correct, the *location* of the peak congestion is determined correctly. Previous congestion quality metrics would report estimate 1 as better [1]. Even though the first estimate would more accurately predict the peak channel width, the second estimate would be better at guiding a CAD tool to eliminate congested regions. Accurate peak information tells us whether congestion is a problem or not, but the *location* and *size* of the congested areas are important for properly mitigating the congestion before it reaches the router.

In this work, we visually compare congestion estimates, E, of several heuristics to actual congestion maps produced by the VPR router, R, to assess their quality. In addition, we measure *average absolute normalized error*, or *a.a.n.e.*, across the congestion map

$$a.a.n.e. = \frac{1}{M^2} \sum_{x=1}^{M} \sum_{y=1}^{M} \frac{\left| E_{x,y} - R_{x,y} \right|}{R_{\max}} \qquad (1)$$

where x and y represent all possible locations in the $M \times M$ FPGA array, $E_{x,y}$ is the estimated channel width at a precise x,y location, $R_{x,y}$ is the routed channel width obtained from VPR 4.30 [3] equal to the maximum number of wires carrying a signal across the 4 adjacent channel segments to the CLB located at x,y. Also, $R_{\max}$ is a normalizing factor equal to $\max(R_{x,y})$. Note that the range of $E_{x,y}$ is first linearly rescaled so that it matches the min and max values in $R_{x,y}$. By counting wires carrying signals, $R_{x,y}$ includes the architectural effect of long wires extending further than required.

To produce visual congestion map comparisons, we have modified VPR to present congestion estimation maps as shown in Figure 2. They are presented as a two-dimensional colour image of $E_{x,y}$ or $R_{x,y}$, using a temperature scale (blue=low, red=high congestion). The estimate, E, appears to the left of the real map, R, and above an error map, $\text{abs}(E-R)$. The user has real-time control of several fitting parameters used in the heuristics through a button interface. The parameters can be adjusted to explore how the heuristic performs with instant visual feedback.

The FPGA CLB architecture assumed in this paper contains 16 LUTs, 51 inputs and 16 outputs. The routing architecture uses all length-4 wires with bidirectional buffered routing switches. A single large benchmark circuit is used to develop heuristics in Section 3, while Section 4 uses the well-known MCNC circuits.

2. PREVIOUS WORK

Congestion estimation for FPGAs can be found in several forms. One indirect form is found in congestion-driven cost functions used in technology mapping, clustering, and placement algorithms. For example, Rmap [4] performs technology mapping to improve a circuit's routability by reducing and balancing the pin counts of LUTs. DART [5] reduces and balances pin count during both technology mapping and clustering. iRAC [6] also does this during clustering. In this respect, pin count distribution has been shown by several authors to be an indicator of routability and congestion. The iRAP [7] placement tool modifies VPR's placement cost function with a congestion estimate by approximating the local Rent exponent for each CLB. Used together, iRAC and iRAP reduce routed peak channel width requirements [6]. VPR's placement cost function is also modified in [8] with a congestion factor which has a large influence in the early stages of placement and a lesser influence during the later stages. The congestion factor counts the number of bounding boxes of different nets that overlap that region. We use a similar heuristic in Section 3.4.

Another form of congestion estimation can be found in routability models which predict the likeliness of routing individual nets as well as circuits. An example is [9] which describes a tool that automatically formulates a Boolean satisfiability problem to determine if a circuit is routable. It therefore provides a discrete yes or no answer, but it does not indicate the degree of routability or congestion of a circuit. Another example is Brown's stochastic routability model for symmetric FPGAs [10] which estimates the probability of making individual two-point connections based on conditional events on the switches in each connection's path. The routability of the circuit is then calculated as the sum of all the individual probabilities divided by the number of connections. In [11], Brown's model is adapted to FPGAs with a routing hierarchy. A stochastic model based on Rent's rule [12] is used to predict the quantity of different types of routing resources consumed by a circuit before placement for hierarchical FPGAs. An estimate of peak channel width is computed in [17] for determining routability. None of the above techniques is aimed at predicting the location of congestion within a circuit.

A third form of congestion estimation is found in the prediction of local routing resource demand. This is explored in [1], which compares 4 congestion estimation heuristics for FPGAs. This is most comparable to our work since we also present several heuristics for predicting local congestion. In [1], it first presents fGREP and fGREP2 [2] as FPGA-specific heuristics that assign routing resource demand values to elements in a routing resource graph. It is based on the distance from the net terminals and the number of alternatives at that distance. Second, [1] describes an enhancement to an ASIC technique where a chip is partitioned into rectangular regions, and the number of shortest paths to connect net terminals that pass through each region is summed to produce the demand for the routing resources in these regions. Third, it describes an FPGA adaptation of RISA [13]. Fourth, [1] calculates a circuit's Rent exponent to estimate peak routing demand of the entire circuit. This last technique does not produce local congestion estimates. Although the first three techniques do produce local congestion values, their quality is measured by peak channel widths (per-channel or globally) for short-wire

architectures. It does not present the data as a visual map and does not measure the *overall* local accuracy of the estimates.

Our work is unique in that it measures and compares the local fidelity of congestion estimates for FPGAs with long wires. Several of the post-placement heuristics explored are novel, and we show how post-processing techniques of blending and saturation can further improve all these heuristics. As well, in Section 3.5, we develop a pre-placement congestion estimation heuristic using blending to propagate congestion information. CLBs that might initially appear to be in low-congestion regions may end up being placed in a region of high-congestion – our propagation allows these CLBs to improve their own local estimate and reduce this source of prediction error.

3. ESTIMATION HEURISTICS

This section presents the visual exploration of several estimation heuristics using a circuit *clone* from [14]. *Clone* is large, synthetic circuit generated with GNL [18]. It emulates a system-on-chip design by specifying 20 synthetic sub-circuits. Each sub-circuit emulates the properties of the 20 largest MCNC benchmarks. *Clone* has over 50,000 6-input LUTs packed into 3618 CLBs. After placement, CLBs from each "subcircuit" tend to clump together, providing well-defined regions of high and low congestion. This makes it easy to visually assess quality as well.

3.1 Local Rent Exponent

In the 1960's, E.F. Rent at IBM plotted the log-log graph of the number of terminals at boundaries of integrated circuits vs. the number of internal components such as gates within those boundaries. It was found that this plot could be fit to a straight line which can be written in the form $\log(T) = P \bullet \log(G) + \log(a)$ which is equivalent to $T = aG^P$, where P is the key fitting parameter known as the Rent exponent, T is the number of terminals for a region containing G gates or primitives, and a is the average number of terminals for one gate.

In this work, we are estimating local interconnect demand. Rent exponent values are typically used to represent overall interconnect demands for the entire circuit. To extract this information over local regions after placement, we extract equal-size regions from the FPGA and use a placement partitioning approach to calculate the Rent exponent for each region [15]. The resulting Rent exponent for this region is assigned to the elementary component, or CLB, in the center of this region. Figure 3 shows the use of this technique to calculate the Rent exponent to assign to a single CLB using a region size of five. This 5x5 region is only a subset of the entire circuit.

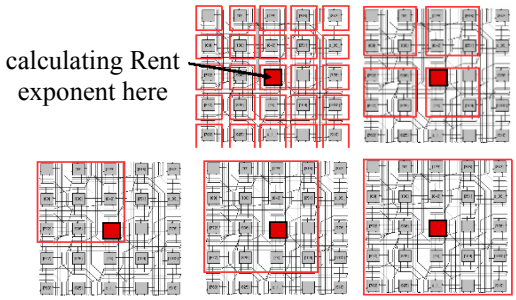

Figure 3: Calculating the Rent exponent for each CLB using a placement approach is demonstrated with a 5x5 window.

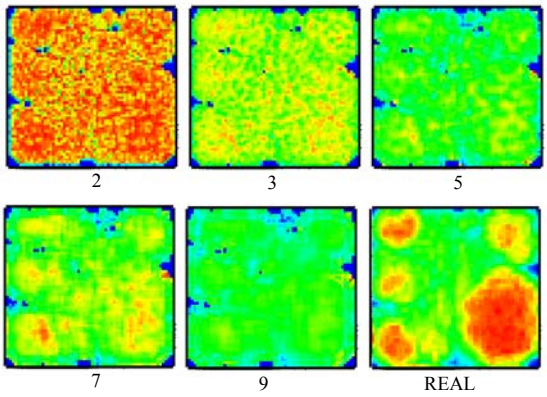

Figure 4: Local Rent exponent congestion maps, window size 2 to 9.

Cuts are counted along the red partition lines and plotted for each region size. Because there are more samples at smaller partition sizes which bias the line of best fit towards smaller samples, the average cuts per partition size are plotted to obtain the line of best fit. The Rent exponent for each CLB is then stored in a grid that becomes the congestion map, E. These calculations are repeated for each CLB. Figure 4 illustrates the estimates for different region / window sizes and the real congestion map R.

The best result from the above exploration is a minimum error or *a.a.n.e.* of 13% using a window size of 7. While the estimation map looks roughly similar to the actual results to a limited extent, there are some definite regions that are incorrect, such as red peaks near the center or the low green areas dispersed throughout the bottom right of the estimation map, which is completely red in the real version. One of the reasons why the Rent exponent may not be the best heuristic for local regions is because the Rent exponent is simply the slope of the graph of the log cuts vs. the log of the number of CLBs in a region. A higher slope therefore indicates that we have a greater number of cuts over a larger region relative to a smaller region. This roughly indicates wire length since a large percentage of longer wires would result in more cuts over a larger region compared to a smaller region. While this can help predict interconnect demands to a certain extent, it is an indirect indicator. Also, anomalies can result from sparse regions with low interconnect demands producing high slopes in the Rent plot, thus a large Rent exponent, since they may produce a high rate of change in the number of cuts when making the transition from counting the cuts over small regions to larger regions. To offset this, we set the entries in the congestion map which correspond to empty CLBs to equal the minimum non-empty congestion value in the map. Due to the recursive nature of calculating Rent exponents, this method is computationally expensive. It may be possible to accelerate the computation using dynamic-programming techniques where some partial calculations are retained and re-used for other CLBs/windows.

3.2 Net Cuts per Region

In the previous heuristic, it was shown that the local Rent exponent can be used to predict interconnect congestion to a certain extent. However, it is an indirect indicator since it is really the rate of change of the number of cuts as we increase the number of components in a region. A more direct indicator would be the absolute number of cuts in a physical region. Figure 5 illustrates this.

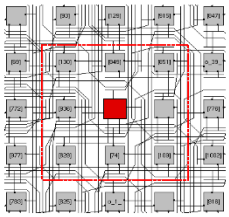

Figure 5: The number of net cuts crossing the red boundary is the congestion for the red CLB using a 3x3 window.

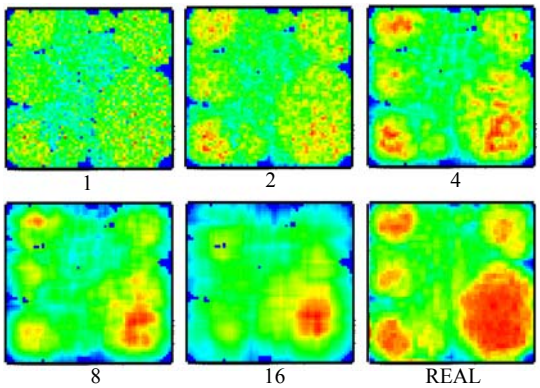

Figure 6: Net cuts per region with a window size from 1 to 16.

This heuristic is explored in Figure 6 by tuning the window size. A window size of 4 offered the best match to the actual congestion map, with an *a.a.n.e.* of 11.2%. Unlike the local Rent exponent, calculating the net cuts per region is not recursive, and therefore much faster.

Although this last heuristic is an improvement over the local Rent exponent, the above images show several regions of very low congestion dispersed throughout regions of medium-to-high congestion. In contrast, the real congestion map produces regions that are more continuous. Although not reflected in *a.a.n.e.*, the visual helps us identify this problem and propose solutions more easily. For example, if one blurs their eyes while staring at any of the estimated congestion maps, they can more easily match the regions of high congestion to those in the real image. This suggests that their may be some value in image blending, and this exact exercise leads to the next technique.

3.3 Image Blending

The previous heuristics incorrectly predict small regions of low congestion within regions of high congestion. This produces unrealistic results if one considers the nature of the routing problem. If the channel for which an ideal path a net should take is fully utilized, the router will take the next best option, which is likely in the vicinity of the congested path. Congestion should therefore be a more gradual phenomenon. If a particular spot within that region does not indicate high congestion according to a particular heuristic, it should still be assigned high congestion if it surrounded by regions of high congestion. One way of doing this "low pass filter" is through an iterative nearest-neighbor blend which is described below.

```
for number of blend iterations N {                                    (2)
  for x from 1 to M {
    for y from 1 to M {
      E'x,y = (1-α)Ex,y + α(Ex-1,y + Ex,y+1 + Ex+1,y + Ex,y-1)/4  } }
  E = E'
}
```

Here, α is the blend rate, and *E* is the congestion map. Each blend iteration spreads congestion from one point to its neighbours. Distant values across the congestion map combine after multiple iterations. Increasing either α or N has similar effects on results.

The final result brings the error of the estimated congestion down to 7.4% using α = 6 and N = 50 iterations. As blending is done, the peaks lower and the troughs increase to the point that the image becomes uniform. To re-emphasize the variation, estimation data is linearly rescaled after blending to the original min and max values. This maintains image contrast.

3.4 Bounding Box Overlap, Wirelength per Area

The next approach is to account for the overlap of nets, which is essentially what creates regions of high congestion. This is accounted for in [8] by counting the number of nets whose bounding box overlap in a region, and assigning that as the congestion value of that region. This heuristic [8] is reported by the name "Bounding Box Overlap" in Section 4 results. However, this approach neglects the fact that large bounding box areas reduce the probability that the net is actually routed to a specific point in that region. We improve upon that technique with little computational overhead by estimating amount of wire that a net will produce, and distributing that amount over its bounding box. The amount produced by each bounding box that overlaps a region is then summed for that region to produce the congestion map. A simple way of quantifying this is to calculate the expected amount of wire that will exist at any given point for a given net. One approach is through the equation (3).

$$W = L / A \qquad (3)$$

where *W* is the expected wire quantity per unit area, *L* is the total wire length of the net, and *A* is the possible area the wire may occupy. We estimate *A* using the bounding box area, and we estimate *L* using equation (4), which includes a tuning parameter β and fanout correction factor *q*.

$$L = \tfrac{1}{2} BB + \beta \cdot q \qquad (4)$$

$$q = \min(BB\ Width, BB\ Height) \cdot \max(0, num\_pins - 3) \qquad (5)$$

Here, BB is the bounding box perimeter. For nets with a fanout of three or less, the half perimeter bounding box is a good estimate of wire length. However, for higher fanouts, the ideal routing

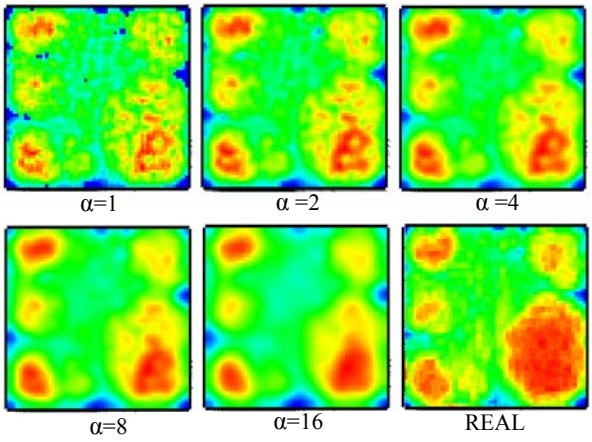

Figure 7: Net cut congestion maps with blending applied, using a net cut window size = 4, N = 50 blend iterations.

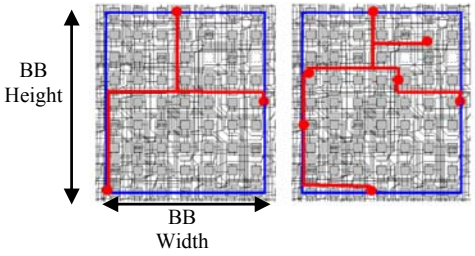

Figure 8: Bounding boxes for a 3-terminal and 7-terminal net.

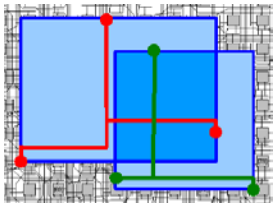

Figure 9: W for the overlapped region of two nets (dark area) is the sum of the W (wire length contribution) of each net (light areas).

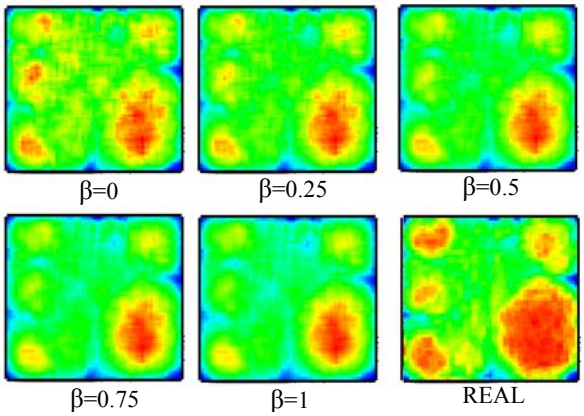

Figure 10: Wire length per area congestion maps.

solution may contain backtracking. Here, it is assumed that the backtracking is performed in an intelligent manner and therefore in proportion to the minimum dimension of the bounding box. This is illustrated in Figure 8.

While this overestimates wire length for very high fan-out nets, it gives us an easily tunable equation through the tuning factor β. For our purposes, we found β to be more effective than VPR's fan-out factor [3]. To illustrate overlap, Figure 9 shows that the W contribution from each net is summed only within the bounding box area of each net. Figure 10 shows the resulting congestion estimation with β representing the extra pin weight.

The resulting algorithm is as follows:

```
for all x,y initialize E_x,y = 0
for each net n {
  q_n = min(BB_n width, BB_n height)*max(0, num_pins_n – 3)    (5)
  L_n= ½ BB_n +  β · q_n                                       (4)
  W_n = L _n / A_n                                             (3)
  for each clb located at clb.x,clb.y in the bounding box of n {
    E_clb.x,clb.y += W_n
  }
}
```

Starting at β = 0, this heuristic has an *a.a.n.e.* of 9%, which continuously increases to 13.9% when β = 1. Although these results are not as low as the 7.3% obtained earlier using net cuts with blending, this heuristic visually produces quite realistic shapes and locations in the congestion map. Another observation from Figure 10 is that although β = 1 produces the most realistic shapes, it under-predicts the size of the congested regions. This is addressed in the next section through saturation.

3.5 Channel Width Constraints

Unlike routing solutions in an FPGA with a channel width constraint, optimal routing solutions for an FPGA with an arbitrarily large channel width do not include paths that route around congested regions. This congestion avoidance, if accounted for, would convert these peaks into wider, lower plateaus. So far none of the heuristics presented consider channel width constraints, and so the peak congestion regions produced tend to be sharper than in the real congestion map. Such images hint toward the use of congestion saturation.

If one considers the effects of a channel width constraint on the routing solution, a limit on the peak saturation is produced as illustrated in Figure 11. This results in clipping, producing the broad plateau desired. However, to maintain image definition, the estimate map must also be rescaled so that clipped peaks are brought back up to their previously-maximum values. The result is displayed in Figure 12 using β = 1.

A peak saturation of 75% produces the best results with an *a.a.n.e.* of 5.6%. Visually, the congestion regions appear to be more accurately located and sized compared to before.

3.6 Single-Pass Route

Another heuristic to consider is the result of a single pass route from the PathFinder algorithm employed by VPR. Here, the congestion estimate is taken from maximum number of tracks used on all four sides of a CLB. In the first routing pass, PathFinder largely ignores congestion as every net takes its favourite route. Although this calculation is not nearly as fast as any of the heuristics explored, a single routing iteration of PathFinder is still significantly faster than a full route. Figure 13 illustrates the result using a saturation of 75% compared to the actual congestion map, producing *a.a.n.e.* of 4.7%.

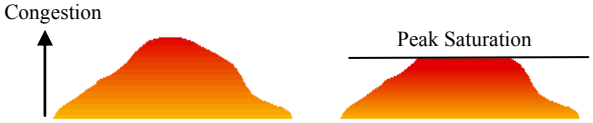

Figure 11: Emulating a channel width constraint by saturating the congestion peaks.

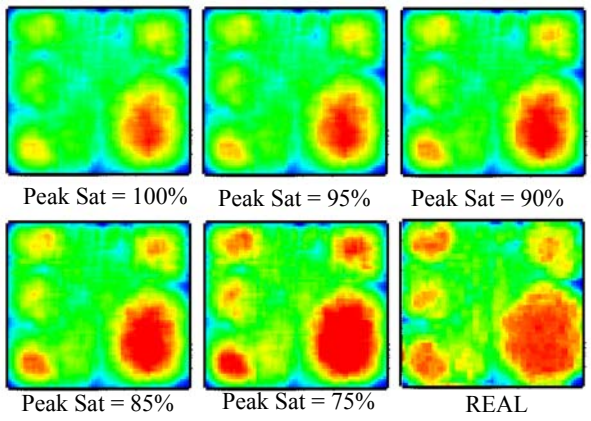

Figure 12: Wire length per area congestion maps with saturation.

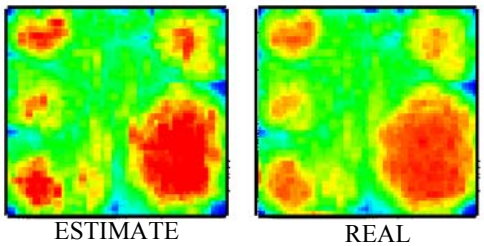

Figure 13: First pass route with saturation.

3.7 Pre-placement: Blending Pin Count

All the previous heuristics that were explored require the Cartesian locations of the CLBs after placement. In this section, we try to estimate congestion before placement. This is a challenge if we do not know the positions of the CLBs. One hint as to how this could be overcome is to carefully look at the images produced by the previous heuristics. An astute observer may notice that in Figure 6 the image of the net cuts per region heuristic using a window size of 1 still hints towards regions of high and low congestion that correspond to the actual congestion map. This has been repeated in Figure 14 for convenience.

Notice the red and yellow points scattered where the continuous high congestion regions exist in the real map. If one repeats the exercise of blurring their eyes, the different regions in the estimation map take shape. The reason this particular image hints to a pre-placement heuristic is that at a window size of 1, the net cut heuristic is simply coloring each CLB based on its pin count, which is information that is know before placement. It also corresponds with El Gamal's stochastic routability model which states that the number of channels used on a masked programmable gate converges to a Poisson distribution with parameter $W = \gamma L_{avg}/2$, where L_{avg} is the average wire length and γ the average pins per cell [16]. In this relation, only pins per cell is known before placement.

A simple solution might then be to repeat the image blending performed previously, however this is not possible since a CLB's neighbors are not known before placement. Instead, we substitute a CLB's Cartesian neighbours used in the previous blending technique with all the CLBs that a CLB connects to, *i.e.*, shares a common net with. This difference is illustrated in Figure 15.

The following blending algorithm is a modification of (2). Here each point in the congestion map corresponds to a CLB that has not yet been assigned a physical location:

```
initialize each point in E[ ] to CLB pin count
for the number of blend iterations N {
  for each CLB a {
    neighbour_congestion = 0
    neighbour_count = 0
    for each pin p of CLB a {
      for each CLB b connected to pin p that is not CLB a {
        neighbour_congestion += E[b]
        neighbour_count++  } }
    neighbour_congestion /= neighbour_count
    E'[a] = (1-α)E[a] + α(neighbour_congestion)   }
  E = E' }
```

Here, α is also the blend rate and N the blend iteration count. Increasing either α or N will increase the amount of blending. Figure 16 illustrates different amounts of blending and compares this image to the real congestion map. To produce visual estimation maps for comparison with the real congestion map, the estimates were correlated with CLB placement data.

Notice that the regions of high and low congestion are clearly highlighted and correspond fairly well with congestion regions in the real congestion map. This technique allowed us to achieve an error of 13% for this circuit with α = 1. Figure 16 shows that with further increases in α, the isolated CLBs of low congestion within regions of high congestion eventually blend to the values of their neighbors. Through this blending, global information is distributed throughout the netlist to allow the CLBs to form a consensus as to whether or not they will be located in a region of high congestion.

4. INDIVIDUAL MCNC RESULTS

Each heuristic was applied to compute the total of the *a.a.n.e.* across the 20 largest MCNC circuits. For each heuristic, the same parameter settings are used across all benchmarks; these were determined in advance to minimize the total error. Peak saturation and blending were also used as a post processing step to improve the quality of the heuristics. The results are plotted in Figure 17.

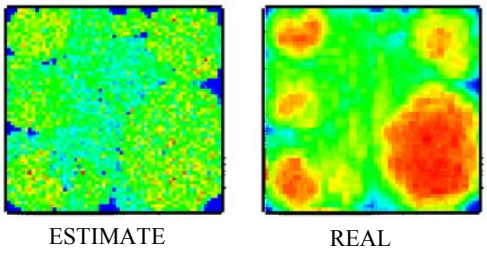

Figure 14: Net cuts per region with a window size of 1 CLB.

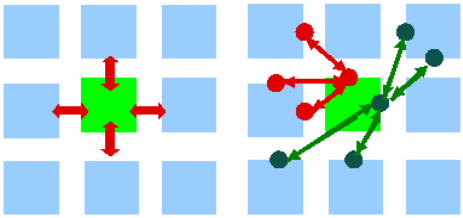

Figure 15: Determining the center CLB's neighbors: Cartesian nearest neighbor blending (left) requires placement information, where as blending CLBs with a common net (right) can be performed in the absence of placement information.

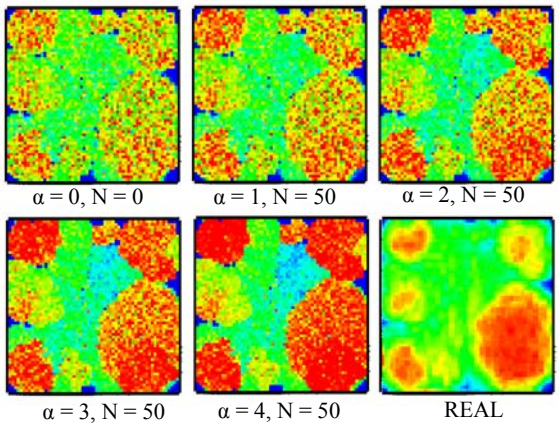

Figure 16: Blending of CLBs that share common nets, using a peak saturation of 75%. This estimation values are calculated before placement, but we use placement data to visualize the resulting maps.

Figure 18 illustrates how saturation and blending are effective post-processing steps that help congestion estimation quality. Error is reported as the average across the 20 circuits. Blending uses $\alpha = 1$, and the wire length per area heuristic uses $\beta = 0.75$. These techniques significantly reduce error for all estimates. Figure 19 also shows the standard deviation in the error.

The average amount of time it takes to compute the entire congestion map (from scratch) of the 20 largest MCNC benchmarks vs. the error of the estimate is plotted in Figure 20. In some cases, it may be possible to quickly compute incremental updates to an estimate (*e.g.*, after a placement swap), but we did not consider that here. Our wire length per area heuristic was the fastest and produced good quality results, while the single pass route produced the best quality results but was also the slowest.

5. FUTURE WORK

The next step is to use these heuristics within an FPGA CAD flow. We are presently modifying VPR to be more congestion-aware. This should reduce the number of place-and-route iterations in Un/DoPack [14], perhaps fully eliminating iteration. Also, we would like to consider using root-mean-square error instead of *a.a.n.e.* so that larger errors are emphasized more than smaller ones.

6. CONCLUSIONS

In this work we described a visual aid for congestion estimation. Using this tool, we qualitatively compared the effectiveness of several straight-forward congestion estimation techniques and used a numerical metric, the average absolute normalized error or *a.a.n.e.*, to further improve the visual match to the real congestion map. Based on visual feedback, we developed two post-processing techniques, peak congestion saturation and blending, which help to emulate the spreading of congestion and routing with a channel width constraint as well as the interaction of nets caused by long wires in the architecture. These techniques improve the quality of all heuristics. Finally, we showed how blending the pin count of each CLB with its netlist neighbors can propagate global congestion information across a circuit to improve pre-placement estimation heuristics. Overall, the highest quality heuristic is found to be a single pass route combined with blending and saturation to produce an average error of 10.9% over the MCNC 20. Our wire length per area model is shown to be the best heuristic when speed is necessary, incurring an average error of only 12.7% for the same benchmarks while executing three orders of magnitude faster.

7. ACKNOWLEDGMENTS

We thank David Leong and Julien Lamoureux for providing some tools and making helpful suggestions throughout this work.

8. REFERENCES

[1] P. Kannan, S. Balachandran, and D. Bhatia, "On Metrics for Comparing Interconnect Estimation Methods for FPGAs," *IEEE Trans. on VLSI*, pp. 381—385, April 2004.

[2] P. Kannan and D. Bhatia, "Interconnect Estimation for FPGAs," *IEEE Trans. on Computer-aided Design*, pp. 1523—1534, August 2006.

[3] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-submicron FPGAs*, Kluwer, 1999.

[4] M. Schlag, J. Kong, and P.K. Chan, "Routability-driven Technology Mapping for Lookup Table-based FPGA's," *IEEE Trans. on CAD*, pp. 13—26, January 1994.

[5] A. Lu, E. Dagless, and J. Saul, "DART: Delay and Routability Driven Technology Mapping for LUT based FPGAs," *ICCD*, pp. 409–414, October 1995.

[6] A. Singh and M. Marek-Sadowska, "Efficient Circuit Clustering for Area and Power Reduction in FPGAs," *FPGA*, pp. 59-66, Feb. 2002.

[7] G. Parthasarathy, M. Marerk-Sadowska, A. Mukherjee, and A. Singh, "Interconnect Complexity-aware FPGA Placement Using Rent's Rule," *SLIP*, pp. 115—121, March 2001.

[8] Y. Zhuo, H. Li, and S. Mohanty, "A Congestion Driven Placement Algorithm for FPGA Synthesis," *FPL*, pp. 683—686, August 2006.

[9] G.-J. Nam, K.A. Sakallah, and R.A. Rutenbar, "Satisfiability-based Detailed FPGA Routing," *Int'l Conf. on VLSI Design*, pp. 574–577, January 1999.

[10] S. Brown, J. Rose, and Z.G. Vranesic, "A Stochastic Model to Predict the Routability of Field-Programmable Gate Arrays," *IEEE Trans. on CAD*, pp. 1827—1838, Dec. 1993.

[11] Z. Dai and D.K. Banerji, "Routability Prediction for Field-Programmable Gate Arrays with a Routing Hierarchy," *Int'l Conf. on VLSI Design*, pp. 85—90, January 2003.

[12] W. Li and D.K. Banerji, "Routability Prediction for Hierarchical FPGAs," *Great Lakes Symp. on VLSI*, pp. 256—259, March 1999.

[13] C.-L.E. Cheng, "RISA: Accurate and Efficient Placement Routability Modelling," *ICCAD*, pp. 690—695, Nov. 1994.

[14] M. Tom, D. Leong, and G. Lemieux, "Un/DoPack: Re-Clustering of Large System-on-Chip Designs with Interconnect Variation for Low-Cost FPGAs," *ICCAD*, November 2006.

[15] J. Pistorius and M. Hutton, "Placement Rent Exponent Calculation Methods, Temporal Behavior and FPGA Architecture Evaluation," *SLIP*, pp. 31—38, March 2003.

[16] A. El Gamal, "Two-dimensional Stochastic Model for Interconnections in Master Slice Integrated Circuits," *IEEE Trans. on Circuits and Systems*, pp. 127—138, February 1981.

[17] J. Swartz, V. Betz and J. Rose, "A Fast Routability-driven Router for FPGAs," *FPGA*, pp. 140—149, February 1998.

[18] D. Stroobandt, P. Verplaetse, and J. van Campenhout, "Generating Synthetic Benchmark Circuits for Evaluating CAD Tools," *IEEE Trans. on Computer-aided Design*, pp. 1011–1022, September 2000.

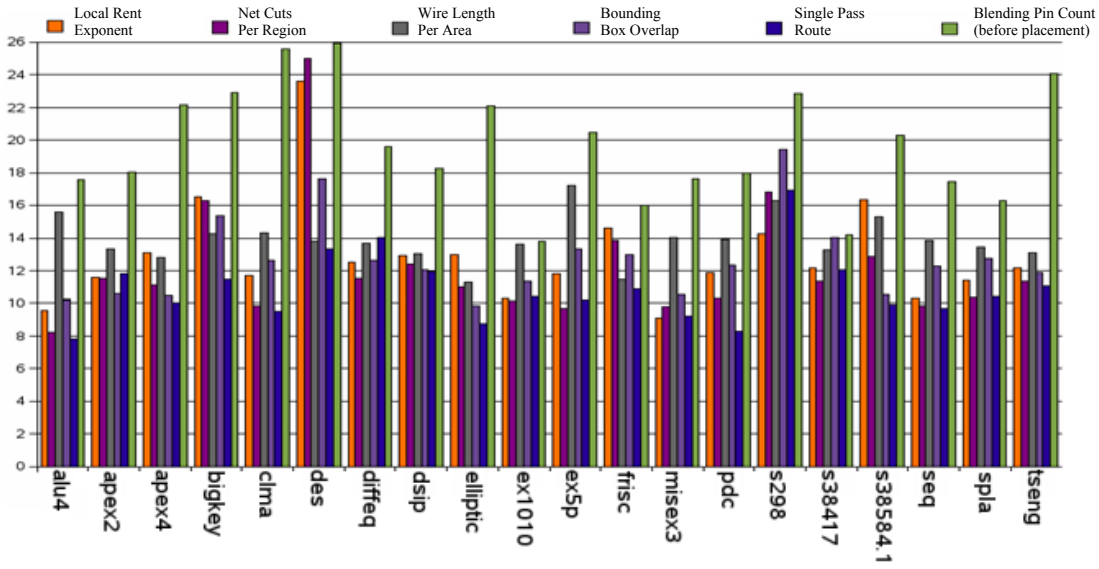

Figure 17: Error (*a.a.n.e*) produced by each heuristic for the 20 largest MCNC circuits (after saturation and blending).

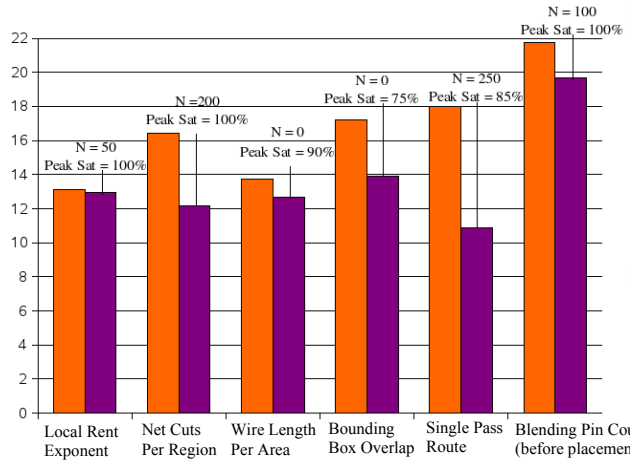

Figure 18: Error (*a.a.n.e.*) before and after saturation and blending.

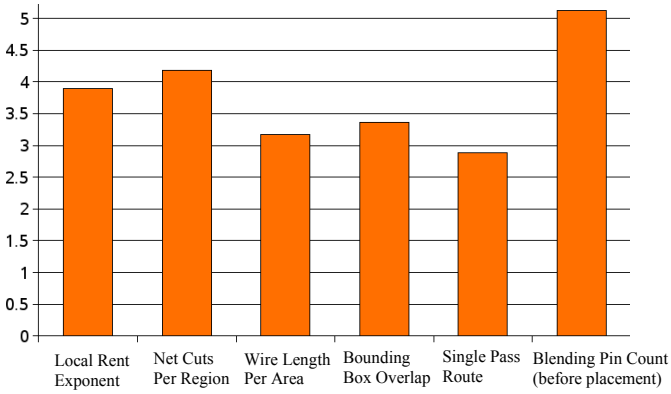

Figure 19: Standard deviation of error (average across 20 MCNC).

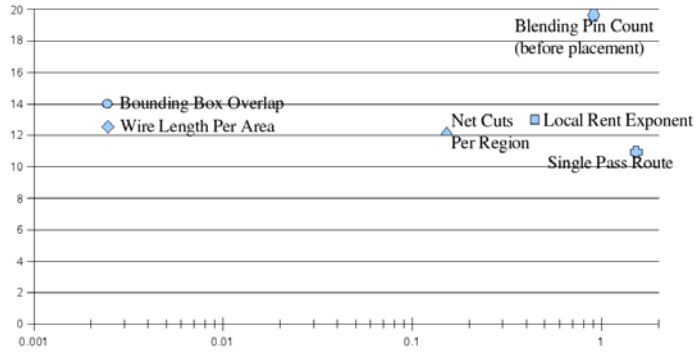

Figure 20: The quality/runtime plot for the heuristics explored in this paper. Note the time axis is logarithmic.