A Method for Generating Random Circuits and Its Application to Routability Measurement

Joel Darnauer and Wayne Wei-ming Dai
Computer Engineering, UC Santa Cruz
Santa Cruz, CA 95064
(joel,dai)@cse.ucsc.edu

December 1, 1995

Abstract

FPLD architectures are often designed based on the results of experiments with "typical" benchmark circuits. For very large FPLDs, it may be difficult to obtain enough benchmark circuits to accurately evaluate an architecture. In this paper, we present a method for generating large random circuits with a fixed number of inputs, outputs, blocks, pins per cell, and approximate rent exponent. The circuits generated are used to evaluate several routability measures. We find that routability is best predicted by estimating the total wirelength in the circuit, not the mean wirelength times pins per cell.

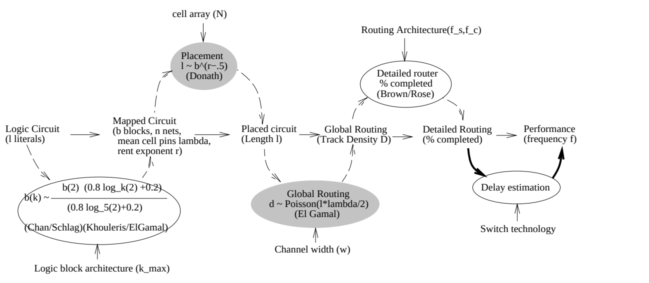

Figure 1: Predictions available in the FPLD mapping process

1 The Problem of Routability Prediction

0

Routability prediction is uesful in several settings: 1) as a cost function for automatic FPLD partitioning, 2) part selection for single FPLDs, 3) FPLD architecture design, 4) FPLD comparison and characterization.

FPLD synthesis usually travels through logic minimization, technology mapping, placement, routing, and delay optimization. Each of these steps represents the progressive refinement of the design and a commitment to certain decisions. At each step we can generate statistics that indicate the performance of the algorithms, the number of resources used, and the likelihood of subsequent steps to succeed. An overview of some estimates compiled from the literature is shown in figure 1 [SCK93, SKC94, Don79, RFLC90, KG92, Feu82].

Although estimation at early stages is useful, early estimates of fittability rest on a foundation of understanding about the subsequent placement and routing process that is less strong that it could be. Experimental evidence shows that Freuer and Donath's techniques for estimating the mean wirelengths are typically off by 50%, which would lead to a 50% difference in capacity of an FPLD and unacceptably low utilizations, and and equal increase in cost per gate.

As a result, we will study routability prediction for mapped circuits. In particular, we would like a general technique for characterizing the routability of a given FPLD.

Previous efforts have studied a number of routability estimation techniques, but these techniques have only been applied to a relatively small set of benchmark circuits. Because the circuits tested have relatively similar characteristics, they do not provide a statistical basis on which to evaluate a measure of routability. Clearly we need a better definition of routability and a way of measuring it.

0

2 Characterizing designs

If we assume that the design has been mapped to k-input, 1 output LUTs,[1] possibly followed by a latch, designs can be characterized by the following parameters:

- g: the number of LUTs
- f: the mean LUT fanin
- r: the Rent parameter of the design.
- i: the number of inputs
- o: the number of outputs

Note that these parameters are not independent. For example, the number of nets is exactly $n = g + i$, total LUT inputs is $p = fg$, and the mean net degree is $d = (g + p + i + o)/n$. If we assume Rent's rule applies at the chip level, then $i + o$ will be approximately $(f + 1)g^r$. More subtle relationships exist. For example, the Rent parameter must be 0 if $p = g$ and $i = o = 1$, because a linear chain of gates can be partitioned very effectively. All of these parameters (with the possible exception of the Rent exponent) are likely to be known before placement and routing.

Given a deterministic way of measuring r, it is easy to see if a circuit corresponds to a particular set of values (g, p, r, i, o). This 5-tuple defines a space of circuits which we can use as the basis for architecture characterization.

Given a specific architecture and a placement and routing algorithm, we can find out if a design will fit by running the algorithms. Our goal is compute the probability of success based on the parameters above -– i.e. to produce a statement that a random design drawn from the set of all possible designs with the same parameters has a certain probability of routing. Let us give this mystery function a name:

$$P_{r_{arch}} : (g, p, r, i, o) \rightarrow [0, 1] \tag{1}$$

Clearly, before we can estimate this function, we need to fully define how circuits are drawn from the class (g, p, r, i, o).

3 A new procedure for generating random circuits

We have developed a program called `rmc` - random mapped circuit. The program generates an XC4000 netlist from several input parameters:

[1]For simplicity, we assume that tristate buses have been mapped to LUTs.

- g: the number of F and G LUTs
- p: the total number of input pins on the all LUT
- i: the number of inputs
- o: the number of outputs
- r: the target Rent parameter of the design

If i and o are unspecified, the program selects appropriate values based on the Rent exponent and a user-specified maximum (to allow the circuit to fit in a particular package). The circuit generated will consist of i input IOBs, o output IOBs and a total of g LUTs followed by a flip-flop to force each LUT to be mapped to the F and G LUTs in the XC4000 CLB, allowing us to exactly control the CLB utilization. (A clock net is connected to each CLB using the dedicated routing). The total number of inputs pins on the LUTs is exactly p, which produces a mean LUT fanin of p/g. p is mean to control the pins per CLB, but because merging in CLBs will depend on the Rent parameters, pins per CLB is only partially determined by p. As will be discussed, r is a target value that the program aims for, but cannot guarantee.

3.1 Generating a feasible split

The basic procedure that is employed to generate the circuit is as follows. The g gates are created and a net is created for each gate and each input for a total of $n = i + g$ nets. We make sure that the number of net drivers is no more than the number of net receivers before continuing: $n <= p + o$. The circuit is treated as a "cluster" with G gates, P pins, I inputs and O outputs, and the cluster is then split into two "cells": A and B. (Figure 2).

Each cell has its own set of input and output nets: i_a, i_b, o_a, o_b, and its own set of gates and pins: g_a, g_b, p_a, p_b. The structure (rent parameter) of circuit is given by the average number of pins leaving each cell: $t_a = i_a + a_a, t_b = i_b + o_b$. (Because we are ignoring tristate connections, no net can be both a cluster input and output). Given the split, there are several different types of net that can cross the cell boundaries:

i_a, i_b	Cluster inputs routed to A or B
i_{ab}	Cluster inputs routed to both A and B
o_a, o_b	Cluster outputs routed from A or B
c_{ab}, c_{ba}	Outputs of A(B) used by B(A)
o_{ab}, o_{ba}	Outputs of A(B) used by B(A) and the cluster outputs

It follows that:

$$I_a = \quad i_a + i_{ab} + c_{ba} + o_{ba} \tag{2}$$
$$I_b = \quad i_b + i_{ba} + c_{ab} + o_{ab} \tag{3}$$

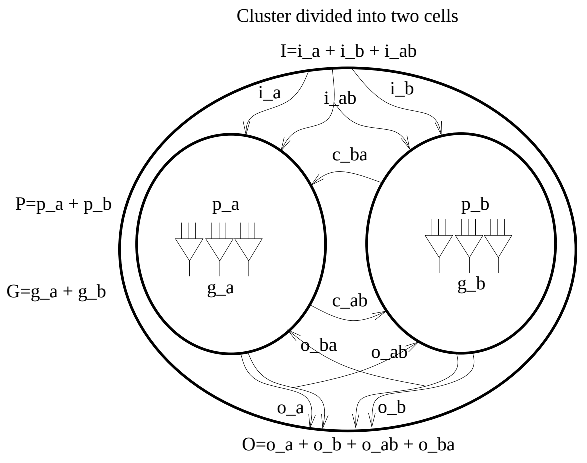

Figure 2: Types of nets that can occur between two split cells in a cluster

$$O_a = \quad o_a + o_{ab} + c_{ab} \tag{4}$$
$$O_b = \quad o_b + o_{ba} + c_{ba} \tag{5}$$
$$\tag{6}$$

We do not allow inputs to route to outputs, since that can always be accommodated in some other splitting operation.

By limiting the inputs and outputs at each level we guarantee that a partitioning with a given Rent exponent exists (although the partitioner may not find it). We want to make sure that there are not better partitions. Since random graphs have very high Rent exponents, it seems reasonable that we can achieve this by scrambling the interconnections at each interface, so they all appear identical. Thus the splitting procedure reduces to finding not a set, but a number of nets for the following 13 non-negative variables: $g_a, g_b, p_a, p_b, i_a, i_b, i_{ab}, o_a, o_b, o_{ab}, o_{ba}, c_{ab}, c_{ba}$ under the following constraints:

$$p_a \geq g_a \tag{7}$$
$$p_b \geq g_b \tag{8}$$
$$i_a + i_{ab} + c_{ba} + o_{ba} > 0 \tag{9}$$
$$i_b + i_{ab} + c_{ab} + o_{ab} > 0 \tag{10}$$
$$o_a + o_{ab} + c_{ab} > 0 \tag{11}$$
$$o_b + o_{ba} + c_{ba} > 0 \tag{12}$$
$$g_a + g_b = G \tag{13}$$
$$p_a + p_b = P \tag{14}$$
$$i_a + i_b + i_{ab} = I \tag{15}$$
$$o_a + o_b + o_{ab} + o_{ba} = O \tag{16}$$

Which guarantees that each cell has at least one LUT, one input, and one output, that each LUT has at least one input, and that all cluster inputs, outputs, gates, and LUT inputs get assigned. We also need to satisfy the following:

$$I_a + G_a \leq O_a + P_a \tag{17}$$
$$I_b + G_b \leq O_b + P_b \tag{18}$$
$$O_a + P_a \leq G_a(G_a - 1) + G_a O_a \tag{19}$$
$$O_b + P_b \leq G_b(G_b - 1) + G_b O_b \tag{20}$$

Which guarantee that each net in each cell has at least one source and one destination, as well as making sure that there are not too many destinations in each cell (each net can be used at most once by a LUT). We have 13 variables, 4 equalities, and 10 inequalities, leaving 9 degrees of freedom in which to find a solution.

We will approach the solution in the following way.

First we choose g_a randomly from $[1, G-1]$, also fixing g_b. Then we divide P into p_a and p_b evenly according to $p_a = \lfloor P\frac{g_a}{g_a+g_b} \rfloor$, fixing p_b as well. Since the choice of g_a is random, we could fix it to be G/2, but we felt that exact bisection might produce circuits whose clusters were too regular. In the same spirit, it is possible to choose different values for p_a and p_b, but doing it this way guarantees that we can satisfy the last pair of inequalities, and the constraints on p_a and p_b are difficult to formulate without knowing the cell inputs and outputs.

Now we try make sure that each cell has one input and one output. The basic method is to assign one input from I to i_a, use a c_{ab} to run a connection from A to B, and then assign one cluster output in O to o_b. (There are some cases where this does not work, consult the code for details.) After each cell has one input and output, we assign the remaining inputs and outputs to cells, occasionally adding crossovers to make sure that each pin in the cluster has a potential driver. At this point we have a feasible circuit - i.e. one that satisfies the constraints above.

The major shortcoming is that the procedure generates very narrow pins-per-block distributions, so we cannot make the claim that the random procedure generates all circuits of a class, only a large random subset of them.

3.2 Achieving the target Rent exponent

At this point, we try to augment the communication between A and B to increase pins on each cell to a target value. The target values r_a, r_b, are set to the value predicted by Rent's rule plus or minus a small variation (about 25%). Generally this involves taking random unused connections in A and routing them to B, or moving a net from i_a to i_{ab} or moving a net from o_a to o_{ab}. Care must be take not to violate the constraints we originally worked so hard to satisfy. This occasionally causes the procedure to stop short of the target values - usually in the case where the requested variation is not deliverable.

As pointed out above, the Rent parameter can depend on the block fanin and net degree: a chain of inverters with a single input and output can only have a Rent exponent of 0. However, it is not easy to formulate this constraint. The target value for each cluster was obtained in almost all cases, and it usually is off only for very high (> 0.9) Rent parameters.

Regardless how well the cluster splitting procedure performed, the overall Rent exponent of a circuit could vary because of accumulated deviation due to random choices made in forming the smaller clusters, the impact of IO limits, or the existence of better partitions. As it turned out, the clustering procedure was able to eliminate any discrepancy between the number of IO at the top level and the Rent prediction within a few levels of hierarchy. A chart of the pins vs. LUTs for the clustering generated by `rmc` is shown in figure 3. The Rent parameter that is the best-fit to the clustering that `rmc` generates is usually within +/-0.05 of the target value.

Of course, the clustering produced by `rmc` is only one of many possible clusterings. We need to consider the possibility that a better partitioning could be found. To examine the real Rent parameters of the circuits, we used the ratio cut partitioner [WC91] to decompose the circuit into clusters. The clustering produced by ratio cut is shown in figure 4.

Figure 5 shows the rent exponents as measured by Rcut against the target rent exponent given to `rmc`. We observe that value of the measured Rent paramter varies by +/-0.1 on either side of the target value. Some of this error is produced because the clustering generated by `rmc` is not best possible clustering, but some of the error is also due to the measurement process (either Rcut did not find the best partition, or the regression fit is too loose). We have yet to characterize the deviation in this procedure.

3.3 Other circuit parameters

We should expect that circuits have normal distributions for net and cell degrees. Figure 6 shows the cell and block degrees produced by rmc for the circuit shown above. We see that the net degrees seem to follow a sort of geometric distribution, while the cell degrees are confined to the two values adjacent to the average.

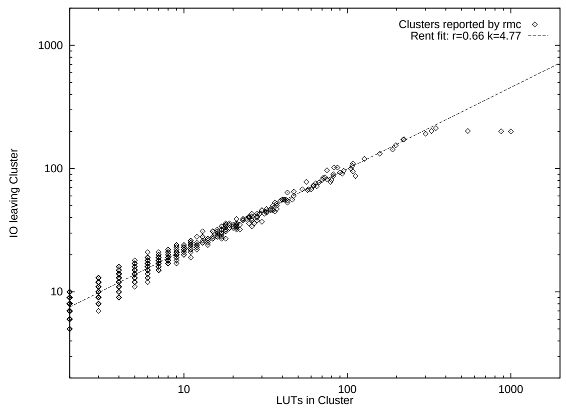

Figure 3: Pins vs Blocks relationship for clustering of 1000 LUT circuit generated by `rmc` and least-squares Rent parameter fit

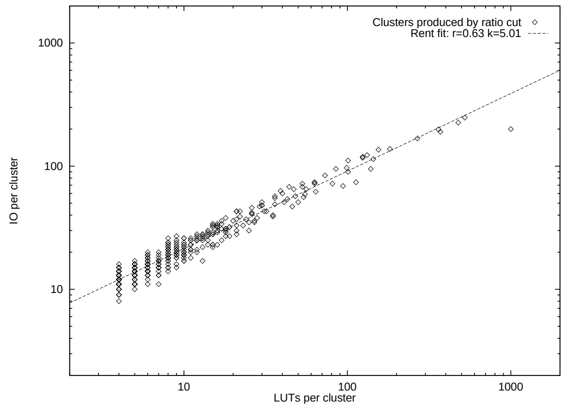

Figure 4: Pins vs Blocks relationship for clustering of the same 1000 LUT circuit, this time partitioned with the ratio cut partitioner.

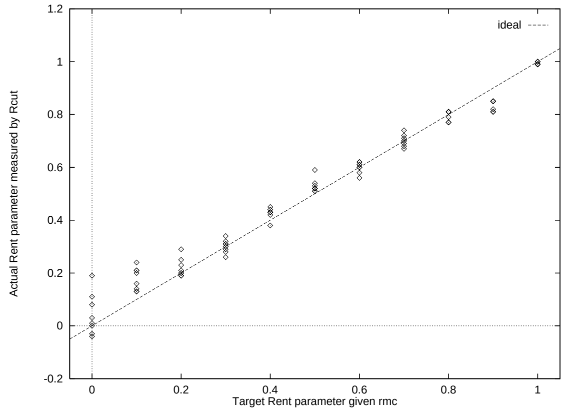

Figure 5: Variation of Rent parameter measured with Rcut against rmc's target rent parameter.

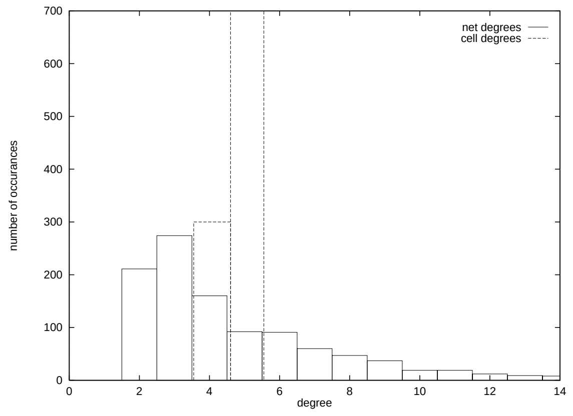

Figure 6: Cell and net degrees for a sample circuit

4 Comparing Routability Measures

In order to put the technique to some use, we decided to measure the routability of random circuits in the XC4002. First we will see how the X4002 fares against a collection of random circuits, and then we will try to draw conclusions some routability heuristics.

4.1 Using the RMC to generate a data set

We can use `rmc` to benchmark a device by throwin random circuits at it and asking which ones can be routed with it. By performing a suitable number of experiments, we can begin to estimate its routability function:

$$P_{r_{arch}} : (g, p, r, i, o) \rightarrow [0, 1] \tag{21}$$

We used RMC and PPR to geneate a data set for the X4002. The small part was chosen because each data point could be collected in a short amount of time. Four random designs were placed and routed in the XC4002 for each combination of the following parameters:

- g: 77 80 83 106 115 128
- f: 3 3.5 5
- r: 0.5 0.6 0.7 0.8

i and o were allowed to follow the Rent parameters, but were limited to the maximum IO of 63. The parameters were chosen to highlight the region of circuits which correspond to real designs. There were 72 classes of circuits chosen and we generated four circuits for each class. Of the 288 circuits generated, 15 failed to route in the 4002.

For each placement, the following data were recorded:

- Internal and exteral wirelengths from placement based on the half perimeter approximation.
- Percent utilization of LUTs
- Measured rent parameter using Rcut
- Number of internal and external nets
- Degree of internal and external nets
- Pins per CLB
- Number of unrouted connections
- Delay as estimated by ppr

We can now use this data to compare some routability heuristics.

4.2 Previous work on routability

El Gamal developed a theoretical model for the distribution of channel densities on masterslice ICs given a mean pins per cell and a wirelength distribution[Gam81]. He assumed that the terminals were scattered randomly in an infinite lattice and that we wished to estimate the distribution of channel densities for a finite subsection. Each terminal generates a single wire which travels to its destination on a random shortest path. He was able to establish that the channel densities were Poisson distributed with mean $\lambda\bar{l}/2$. This average channel density was the figure used to gauge routability in [CSZ93] and was the basis for the statistical derviation in [BFRV92].

El Gamal does not stop here, however. El Gamal also derives an expression for the number of tracks whose density exceedes the maximum width w, based on the expected number of tracks with density d. He also derives bounds for the probability that a circuit routes within a given channel density requirement by computing the expected number of overflowing channels. This chance of failing routing is *proportional to the number of cells,* in addition to being a strong function of the pins per cell and mean wirelength.

In contrast, more recent work on routability prediction uses a metric which is only indirectly employs the number of cells in the design. [CSZ93] Reviews previous work and uses Feuer's model to estimate wire internal and external lengths based on a extracted Rent exponent from min-cut based initial placement. A star expansion is used to derive a modified pins-per cell, γ', which takes into account the extra routing imposed by high-fanout nets. They propose the following routability criterion:

$$\gamma'\overline{l_{ext}}/2 \leq w_g - 0.5 \tag{22}$$

$$\gamma'\overline{l_{int}}/2 \leq w_g + w_d - 0.5 \tag{23}$$

Where w_g and w_d are the number of general purpose and direct interconnect, and $\overline{l_{ext}}$ and $\overline{l_{int}}$ are the mean internal and external wirelengths. Circuits which exceede either of these conditions by more than one are deemed "unroutable". Those in between are deemed "marginally routable". On the benchmark circuits studied, this predictor generates no false positives or negatives, but the "marginal" region is large (half of the benchmarks) and provides few clues to the ultimate routability of the circuits (6 of the 14 marginal cells failed). If we classify the marginal circuits as routable, then the predictor has a success rate of 76% on the MCNC benchmarks.

Chan, Schlag and Zien's estimator directly takes into account only the number of pins per cell and the mean wirelength. Although the wirelength is derived from the Freuer model based on the number of cells and the estimated Rent exponent, it seems that this does not give enough weight to the number of cells.

4.3 Empirical Evaluation of Predictors

In order to simplify the collection of data, we measure the wirelength after placement. Since all of the predictors we will be using will be based on wirelength, this removes some of the error that could cloud our appraisal. The following predictors are examined:

H1 El Gamal's Channel Width Estimator ($\lambda\bar{l}/2$)

H2 El Gamal modifed by star expansion from [CSZ93].

H3 Total placed wirelength in circuit (perimeter approximation)

Figures corresponding to each hypothesis are shown below. In each case, the horizontal axis is the value of the metric and the vertical axis is the value of the PPR-reported delay in nanoseconds. A delay of 100ns represents an unroutable design. Some of the data points in the unroutable category overlap, so it is hard to see the distribution.

In order to measure the metrics, we assume that we will use them to evalaute partitions in a multi-FPLD partitioning system where we cannot tolerate unroutable partitions. We will also assume that we can choose the threshold point very close to the boundary of the unroutable circuits. The L value in the following figures is the number of the 263 routable circuits that can be sucessfully classified under each metric without any false positives. Higher numbers indicate that the metric is somewhat better at separating routable and unroutable designs. This is not a perfect technique, just one way of comparing routability metrics.

Based on this data, we can tenatively conclude that the total wirelength may be a superior metric compared

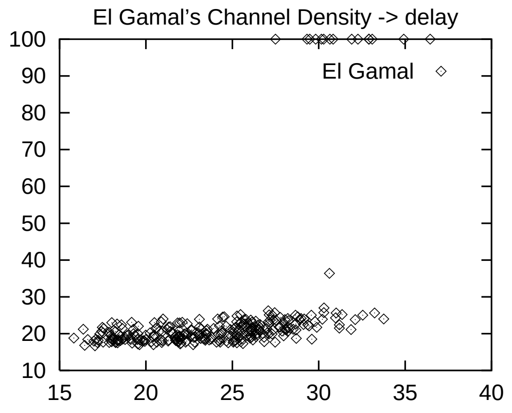

Figure 7: El Gamal's channel width estimator. $L = 228$. El Gamal's measure. In order to reject all the unroutable designs, we need to reject about 15% of the good ones.

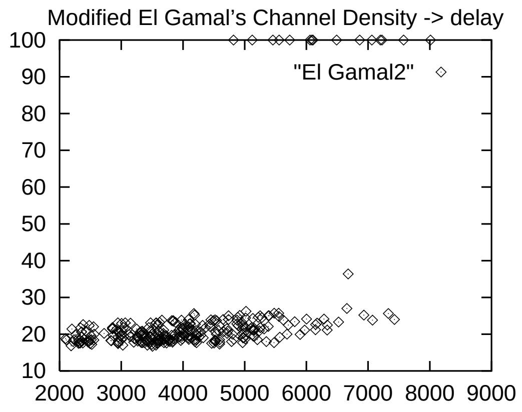

Figure 8: El Gamal's width estimator modified by star expansion. (Before placement) $L = 208$. The star expansion seems to hurt this estimation.

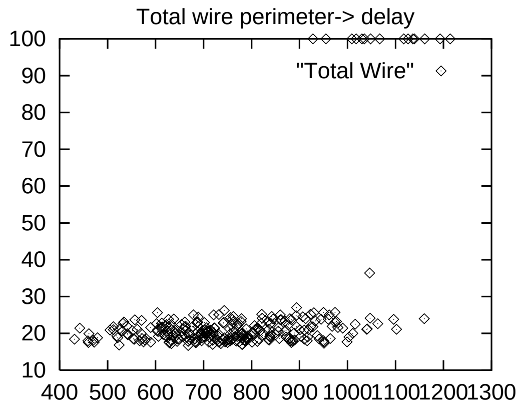

Figure 9: Total placed wirelength. $L = 242$.

to the average channel density which has be previously proposed. The reader should be warned that these results are very tenative. It remains to repeat the experiments with a larger set of circuits and better data analysis. We do hope however, that this example illustrates a *technique* for routability analysis that overcomes some of the limitations of relying on benchmarks.

5 Summary

Chief results of this work as as follows:

1. A method for generating random circuits with a given number of LUTs, LUT fanin, inputs, and outputs, and a target Rent parameter. The technique has an error in the Rent parameter of +/-0.1.
2. A proposed technique for measuring routability using many random circuits.
3. The observation that, after placement, the total length of wire is the best predictor of routability. El Gamal's unmodified channel density measure is also good, but not as good as total wire length. This suggests we should try to estimate total wire length before placement as well.

Acknowledgment

The authors wish to thank Martine Schlag and Pak Chan for their very helpful comments and instruction.

References

[BFRV92] S.D. Brown, R.J. Francis, J. Rose, and Z.G. Vranesic. *Field-programmable gate arrays*. Kluwer Academic Publishers, 1992.

[CSZ93] Pak Chan, Martine Schlag, and Jason Zien. On routability prediction for field programmable gate arrays. In *Proceedings of the 1993 Design Automation Conference.*, pages 326–330, 1993.

[Don79] Wilm E. Donath. Placement and average interconnection lengths of computer logic. *IEEE Transactions on Circuits and Systems*, CAS-26(4), April 1979.

[Feu82] Michael Feuer. Connectivity of random logic. *IEEE Transactions on Computers*, C-31(1):29–33, January 1982.

[Gam81] A. El Gamal. Two-dimensional stochastic model for interconnections of master slice integrated circuits. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, CAS-28(2):127–138, February 1981.

[KG92] J.L. Kouloheris and A. El Gamal. Fpga area versus cell granularity - lookup tables and pla cells. In *FPGA '92*, pages 9–14, 1992.

[RFLC90] Johnathan Rose, Robert J. Francis, David Lewis, and Paul Chow. Architecture of field-programmable gate arrays: the effect of logic block functionality on area efficiency. *IEEE Journal of Solid State Circuits*, 25(5):1217–1225, October 1990.

[SCK93] Martine Schlag, Pak Chan, and Jackson Kong. Empirical evaluation of multilevel logic minimization tools for a lookup-table-based field-programmable gate array technology. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(5):713–22, may 1993.

[SKC94] Martine Schalg, Jackson Kong, and Pak Chan. Routability-driven technology mapping for lookup table-based fpgas. *IEEE Transactions on CAD*, 13(1):13–26, January 1994.

[WC91] Y.C. Wei and C.K. Cheng. Ratio cut partitioning for hierarchical designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(7), July 1991.