Chapter 6

Validation of Circuit Quality

As discussed earlier, heuristic algorithms such as GEN are best compared on the basis of their actual results. The primary applications of the benchmark circuits produced by GEN are FPGA architectural exploration and software tools for computer-aided design. Thus, our method of validation will use well accepted metrics of routability to compare "real" benchmark circuits with clone circuits produced by GEN. Because the GEN algorithm contains a number of random and probabilistic techniques, it is also interesting to compare the GEN-circuits against standard random graphs of the same size.

In the case of combinational circuits, we use the MCNC benchmarks as our real circuits. For sequential circuits, the author was able to use industrial circuits provided by the Altera Corporation while employed there on an internship.

Our validation process is outlined in Figure 6.1. We take a real circuit, its clone circuit from GEN, and a random graph of the same size. These are individually placed and routed, and comparisons are made based on reconvergence number (from CIRC), track-count and wirelength (from VPR), and the "wiring resources" used on an Altera 10K20RC240 commercial FPGA (from MAX+PLUS2).

In Section 6.1 we show how to create reasonable random graphs for this comparison. Section 6.2 then gives a number of examples to visually indicate the differences between random graphs, GEN-circuits and real benchmarks, itself a form of validation. In Section 6.3 we discuss the empirical results for the combinational MCNC circuits, and in Section 6.4 we discuss empirical results for sequential industrial circuits.

Because GEN creates circuits using only a small parameter list, the goal is to show how

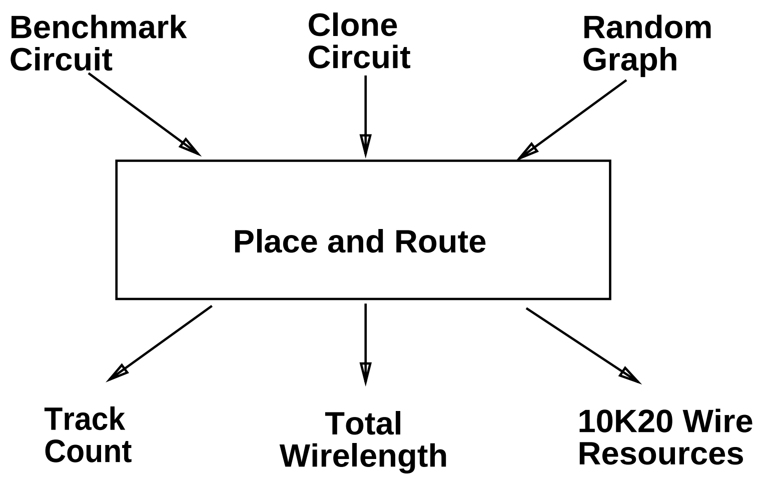


Figure 6.1: The validation process.

close they are to existing circuits. For a method such as proposed by Iwama *et. al.* [43] (See Section 2.2.3), where new circuits are generated by repeated small transformations or mutations, it would be equally important to show that the result was significantly *different*.

It is important to point out that the default parameterizations and the benchmark circuits produced by GEN are not at all restricted to the existence of an initial circuit to clone, other than for this validation process. We are able to generate benchmark circuits of up to 200,000 LUTs, well beyond the level of current FPGAs or ASIC circuits, but we can only *validate* the process up to the largest circuits in the MCNC and industrial collections, currently about 4500 LUTs.

6.1 Generating Comparison Random Graphs.

As mentioned earlier in Section 2.2.3, there are several natural models under which it is relatively easy to generate uniform random graphs. The most common model used is $G(n, p)$: a graph on n nodes where each edge exists independently with probability p. However, these graphs have either too many edges, or are disconnected (depending on p—see Section 2.2.3), so they are too unrealistic even to form a basis for comparison. The closest form of random graph that we can generate as a fair comparison is a random t-regular undirected graph, which we then force to be directed.

6.1.1 Random Directed Acyclic Graphs.

To generate a random graph the same size as a combinational circuit with n nodes and m edges, we calculate the largest t such that $t \cdot n < 2m$, generate a t-regular graph, then add the required number of leftover edges by random sampling. We direct the edges by taking a random ordering of the nodes and directing each edge from the lower to the higher numbered vertex.

A random t-regular graph can be generated as follows[1]:

1. Create a random permutation σ of size $2 \cdot t \cdot n$, to represent $2 \cdot t \cdot n$ nodes of a new graph (with no edges).

2. Join the nodes σ_{2i} and σ_{2i+1} with a new edge, $i = 0..(t \cdot n) - 1$. This creates a graph on $2 \cdot t \cdot n$ nodes with $t \cdot n$ edges, where each node is connected to exactly one other, i.e. a random matching.

3. Collapse (i.e. "identify") all nodes labeled $\sigma_{ti}..\sigma_{(t+1)i-1}$ into a single node x_i, for $i = 0..n-1$.

The result of this process is an n node undirected graph where the degree of each node is exactly t. The algorithm does not, however, guarantee that the graph is *simple* (contains no double-edges or self-loops). The expected number of loops (edges from vertex v to itself) is given by

$$\begin{aligned}
\mu_1 &= \text{Pr(edge is loop)} \cdot \text{edges in } G \\
&= \frac{\text{\#pairs which produce a loop}}{\text{\# pairs}} \cdot \text{edges in } G \\
&= \frac{\binom{t}{2} \cdot n}{\binom{nt}{2}} \cdot \frac{nt}{2} \\
&= \frac{t(t-1)n}{(nt)(nt-1)} \cdot \frac{nt}{2} \\
&= \frac{t-1}{2} \qquad (n >> 1)
\end{aligned}$$

[1]Thanks to Mike Molloy [53] for showing me this construction and the analysis of it. The configuration model was introduced in this form by Bollobás[9] and motivated in part by the work of Bender and Canfield[7]. This model arose in a somewhat different form in the work of Bekessy, Bekessy and Komlós[6] and Wormald[71, 72].

and the expected number of double connections (multiple uv edges) is given by

$$\begin{aligned}
\mu_2 &= \frac{\text{possible double edges}}{\text{possible edge-pairs}} \cdot \text{edges in } G \\
&= \frac{\binom{n}{2}\binom{t}{2}\binom{t}{2} \cdot 2}{\binom{nt}{2}\binom{nt-2}{2}} \cdot \left(\frac{nt}{2} \cdot \frac{nt-2}{2}\right) \\
&= \frac{n(n-1)t^2(t-1)^2}{nt(nt-1)(nt-2)(nt-3)} \cdot \frac{nt(nt-2)}{4} \\
&= \frac{(t-1)^2}{4} \qquad (n >> 1).
\end{aligned}$$

It is quite interesting that the expected number of loops and double edges is a function purely of t, independent of n. The distribution of the events is Poisson, so the probability that a given G generated by the construction is loop-free, double-edge-free is $e^{-(\mu_1+\mu_2)}$. For $t = 5$ the probability that G is simple is 0.006. Thus we can expect to find a simple t-regular graph within a couple of hundred iterations. In practice, however, we can (and do) just delete the loops and multi-edges and choose new edges when adding the $m - tn$ other edges. Constructions due to Frieze [28] and McKay and Wormwald [51] allow this to be done without sacrificing perfect uniformity, but this is not necessary for our purposes.

For the direction of edges, we just use the (natural) ordering which comes from the random permutation σ. To add the extra edges, we uniformly choose a node with low fanin, uniformly choose a node from those with lesser numbers, and add an edge. We repeat this process until the number of edges in the graph is m.

One problem with these random graphs is that they have an overly high number of I/Os. For any random ordering of the nodes used to choose the edge directions, the probability that the i'th node x has all its edges directed forward (i.e. is a PI) is approximately $(\frac{i}{n})^t$, so the expected number of PIs is

$$\begin{aligned}
E[n_{PI}] &= \sum_{i=1}^{n} \left(\frac{i}{n}\right)^t \\
&= \frac{O(n^{t+1})}{n^t} \\
&= O(n).
\end{aligned}$$

Empirically, we calculate that for $t = 5$, about 8% of nodes are primary inputs, and by symmetry 8% are primary outputs.

6.1.2 Random Directed Graphs with Cycles.

For sequential circuits, we also want to have a given number of flip-flops and back edges. The introduction of back-edges also offers the opportunity to "repair" the I/O bias in the acyclic circuits.

We generate a random directed graph on n nodes and m edges with n_{PI} primary inputs, n_{PO} primary outputs, with n_{DFF} available flip-flops (for breaking combinational cycles, as we want only synchronous designs) and k-bounded fanin. The algorithm is as follows.

1. Generate a t-regular graph as in the combinational case.
2. Randomly label n_{PI} fanin-zero nodes as PI (similarly n_{PO} fanout-zero nodes as PO).
3. Randomly connect unlabeled fanout-zero and fanin-zero nodes by new edges until they are exhausted. When it is necessary to connect a node to a node of a lower number, separate the two by a flip-flop if one remains to allocate, otherwise ignore this choice and restart the search for an alternate connection that does not involve a back-edge. This reduces the number of unwanted I/Os in the circuit, while also adding back edges and flip-flops.
4. Continue randomly connecting random nodes to random nodes with fanin less than k until the graph contains exactly m edges.

The graphs generated by this process could be seen as a "first pass" version of GEN which takes fewer parameters into account. In fact, this algorithm alone would be an improvement over most naive approaches to generating random graphs for benchmarks, and thus represents an extremely fair comparison of GEN circuits to "random graphs." Comparing real circuits to clones and these random graphs is essentially measuring how far along the scale from "random" to "real" the current GEN approach has traveled.

6.2 Visual Validation: Examples.

For smaller circuits, we can observe the output of GEN pictorially. One command-line option of CIRC causes a DOT script to be output. The DOT program [47] takes this description of the graph and generates a drawing in postscript. We show a number of these drawings here. Further examples are shown in Appendix C.

6.2.1 Gen Circuits from Defaults.

Figure 6.2 shows four different combinational circuits produced by GEN using the default parameter distributions. We note that these circuits appear to be "normal" circuits, and include many features such as areas of high fanout. The visual "quality" of the circuits is most striking when one observes the similarity to MCNC circuits, shown in Figure 6.3, and the contrast between MCNC circuits and the random graphs shown in Figure 6.4.

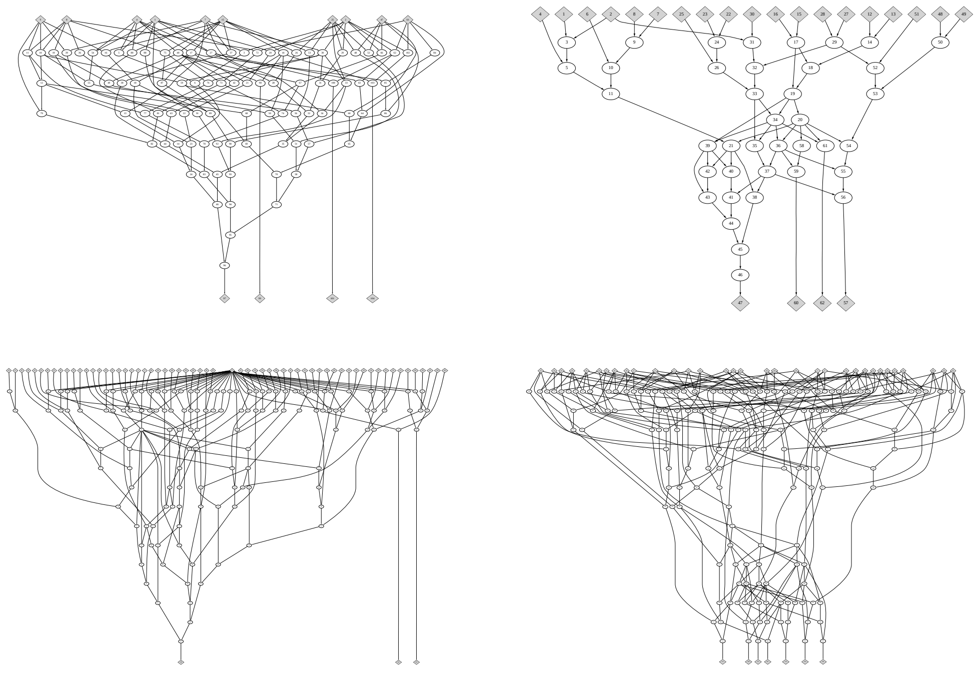


Figure 6.2: Varied circuits produced by GEN, using the default profile.

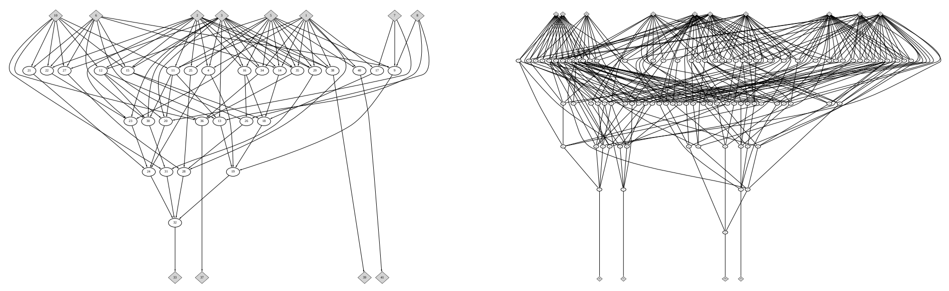


Figure 6.3: MCNC combinational circuits **sqrt8** and **sa02**.

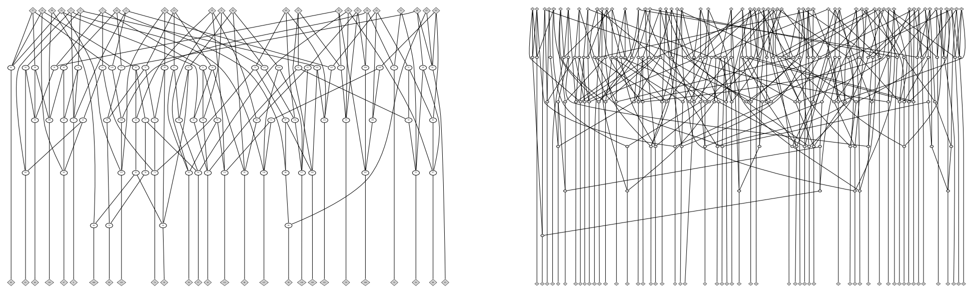


Figure 6.4: Random 4-regular digraphs

6.2.2 Gen Clone-Circuits.

Figures 6.5 and 6.6 show two MCNC circuits, each original circuit pictured with two different clone circuits generated from its characterization by CIRC. Notice that the clones have a similar structure in terms of the parameters given to GEN, but are different in the implementation of that structure, just as they are different from the original.

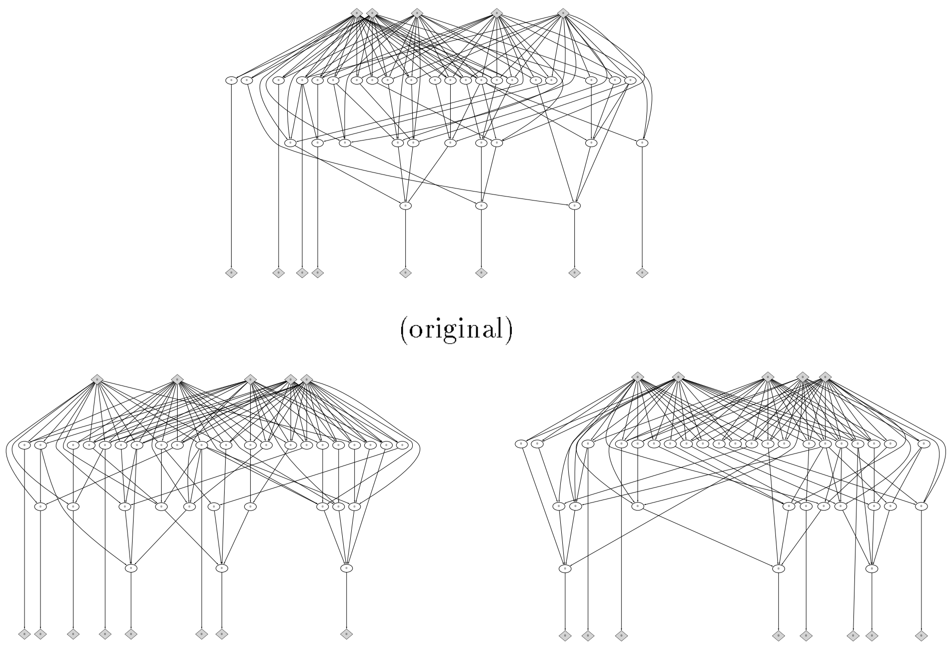
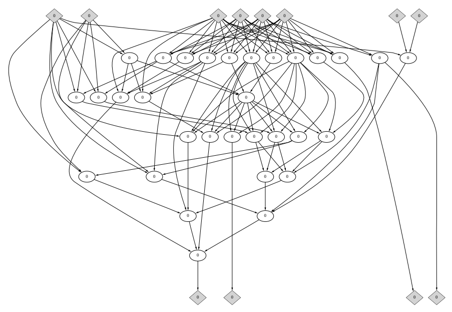


Figure 6.5: MCNC combinational circuit **squar5** and two clone circuits from GEN.

Figure 6.7 shows the MCNC sequential circuit **dk15** and two clone circuits produced by GEN. Unfortunately, DOT is only designed to display directed acyclic graphs, so we are unable to automatically display graphs according to our sequential model. To generate



(original)

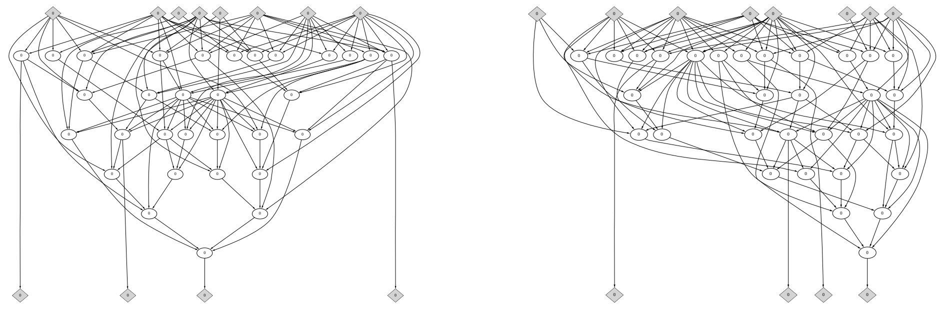


Figure 6.6: MCNC combinational circuit **sqrt8ml** and two clone circuits from GEN.

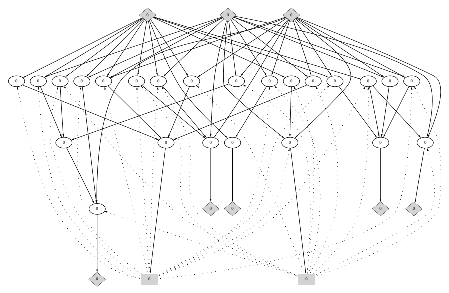
acceptable input for DOT, CIRC reverses all back-edges and gives instructions for DOT to display them as dotted in the drawing.

6.3 Combinational MCNC Circuits.

In this section we deal with the validation question for combinational circuits. We judge the quality of the generated circuits with respect to parameters not specified in generation: reconvergence, and post-placement and routing wirelength and track count. We note that a validation process for other characteristics such as node activity in simulation or timing analysis could also be performed; we leave this for future work.

We constructed the clone scripts (See Section 5.4.3) for 42 combinational MCNC circuits[2] with CIRC (i.e. n, n_{PI}, n_{PO}, d, shape, fanout and edge length distributions), and generated corresponding circuits meeting those profiles with GEN. Our method of validation is to compare unspecified characteristics of the MCNC circuits against those of the corresponding

[2]There are actually 109 combinational circuits in the LGSynth93 benchmark suite, but the majority are too small to be useful. We have restricted the experiments to circuits with 100 LUTs or more.



(original)

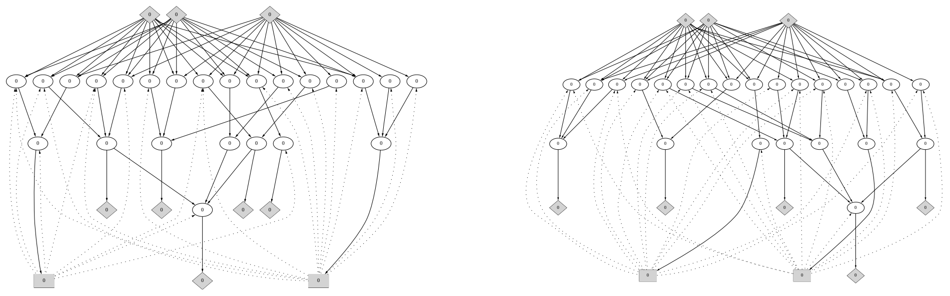


Figure 6.7: MCNC sequential circuit **dk15** and two clone circuits by GEN.

generated circuits and against random graphs of the same size (as discussed in the previous section).

Validating Reconvergence.

Reconvergence (from Section 3.4), R, is not a parameter to GEN. Reconvergence captures numerous properties of a circuit, including high fanout, and the interaction between shape, edge length and fanout distribution, all of which affect the ability to place and route the circuit. We calculated R for the generated circuits and compared them to those of the original circuits from which the generation profiles were extracted and to those of random graphs of the same size. The results for the MCNC circuits and their corresponding GEN-clones and random graphs are shown in Table 6.1. Recall that $0 \leq R \leq 2$ for 4-LUT mapped circuits.

We found that, for over half of generated circuits, R was within 0.1 of the value for the corresponding MCNC circuit. On average R differed by 22% in absolute value (if cancellation is allowed the difference is only 9%). This indicates that the correlation for an important descriptive parameter, R, did carry through the generation process.

		Reconvergence			Tracks			Wirelength		
	size	MCNC	GEN	RND	MCNC	GEN	RND	MCNC	GEN	RND
sao2	100	0.48	0.57	0.45	4	4	6	616	602	879
cht	102	0.10	0.17	0.10	3	3	5	353	445	572
9symml	106	0.41	0.57	0.44	4	4	7	606	582	867
C1355	115	0.80	0.56	0.21	5	4	6	677	655	825
C499	115	0.80	0.56	0.22	5	4	6	668	655	831
bw	137	0.67	0.66	0.67	4	4	9	842	794	1342
clip	149	0.59	0.63	0.79	4	4	9	978	896	1579
9sym	153	0.45	0.51	0.44	4	4	8	950	858	1424
C432	160	0.96	0.95	0.15	4	4	7	855	895	1347
rd84	165	0.53	0.78	0.60	5	4	9	1171	999	1927
o64	176	0.00	0.00	0.05	3	3	5	395	375	1204
C1908	178	0.84	0.95	0.28	5	6	8	1196	1249	1777
i3	178	0.00	0.00	0.05	3	3	6	332	344	1209
alu2	207	0.88	0.97	0.64	5	5	10	1425	1425	2591
i5	221	0.00	0.16	0.06	3	3	5	655	1180	1620
exmpl2	223	0.36	0.30	0.05	4	4	6	1053	1289	1523
toolrg	225	0.31	0.46	0.37	5	5	9	1520	1417	2494
t481	230	0.62	0.76	0.62	6	6	10	1763	1728	3071
C880	234	0.57	0.64	0.16	5	6	7	1419	1655	2233
duke2	273	0.56	0.56	0.36	6	5	10	2169	2008	3277
i2	275	0.02	0.06	0.02	3	3	6	727	716	2203
i4	290	0.00	0.01	0.03	3	3	6	592	639	2393
vda	305	0.72	0.77	0.55	7	5	12	2787	2557	4613
i6	320	0.24	0.21	0.05	3	3	7	1181	1262	2501
i7	402	0.20	0.20	0.03	3	3	6	1352	1403	4114
i9	464	1.07	0.72	0.22	5	5	12	2770	3072	6913
C3540	481	0.86	0.84	0.38	6	8	15	3726	4887	8321
cordic	489	0.80	0.89	0.39	7	7	15	4279	4859	8891
table3	494	0.73	0.87	0.49	8	6	15	5442	4847	8840
table5	500	0.78	0.86	0.39	8	7	15	5612	5018	9159
x3	512	0.26	0.24	0.08	4	5	10	3454	4289	7029
ex4p	514	0.41	0.25	0.23	4	5	12	3425	3914	8604
apex6	528	0.25	0.21	0.08	4	6	10	3217	4331	7115
C6288	559	0.90	1.16	0.45	4	8	16	2900	6207	10287
k2	559	0.60	0.60	0.18	7	7	14	5190	5191	9139
misex3c	563	0.53	0.63	0.37	6	5	15	4841	4493	10989
dalu	575	0.46	0.48	0.19	5	6	13	3827	4871	9547
i8	614	0.77	0.43	0.18	5	7	15	5729	6391	10181
apex1	740	0.67	0.56	0.36	8	7	19	8124	7725	15326
apex3	921	0.66	0.59	0.30	8	7	19	10658	9831	34423
C7552	945	0.53	0.45	0.05	5	6	13	5751	10384	15918
ex5p	1072	1.12	1.20	0.27	10	8	21	14343	12615	27904
i10	1252	0.72	0.55	0.09	6	8	19	15085	23915	28738
apex4	1270	0.90	0.69	0.23	9	8	23	16312	14279	34423
misex3	1411	0.55	0.77	0.24	8	7	24	16139	14799	40152
alu4	1536	0.50	0.62	0.22	7	6	26	15818	13561	45177
seq	1791	0.48	0.67	0.21	8	7	27	21348	19796	57040
des	1847	0.50	0.39	0.07	6	9	23	17898	33925	50294
apex2	1916	0.47	0.64	0.20	8	8	29	23203	22742	63418
spla	3706	0.97	1.07	0.13	10	9	19	49724	52583	167832
pdc	4591	1.01	1.27	0.10	11	10	19	74553	66131	225679
signed difference			9%	-45%		3%	123%		10%	119%
absolute difference			22%	48%		14%	123%		17%	119%

Table 6.1: Empirical validation using combinational MCNC circuits.

In contrast, the reconvergence numbers of the random graphs did not match the MCNC circuits well at all. We observe that these random graphs also exhibit diminishing R as n increases. This is partly due to the two factors mentioned earlier: the absence of high-fanout nodes and the large number of I/Os. Thus any generator which does not take these factors into account will fail to emulate crucial behaviour of real circuits.

Validating Routability.

To test the "routability" of our output circuits, we used a locally available tool, VPR [8], to place and global route the sets of MCNC circuits, generated circuits, and random graphs described above. The circuits are compared on two different metrics: the maximum number of tracks per channel required to successfully route, and the total wirelength of the global routing. VPR is a high-quality tool, currently the best academic global router available, so it provides a good quality solution for our comparisons.

VPR [8] chooses a minimal square grid to support the size of the circuit, and minimizes both maximum track-count per channel and total wirelength (by re-routing with successively fewer tracks per channel until failure occurs).

Table 6.1 also shows the routing statistics for the MCNC circuits, clones and random graphs with summary statistics (percentage pairwise differences) on the last line. We see that the track count for the generated circuits differed by 14%, on average, from the corresponding MCNC circuit, whereas the random graphs differed by 123%. Wirelength differed by 17% for the generated circuits and 119% for random graphs.

For both track-count and wirelength, we note that the variation for GEN clones lies in both directions whereas random graphs were universally harder to place and route. Thus, the *signed* differences for the GEN clones were only 3% in track-count and 10% in wirelength, meaning that the difference applies as much to the variance of GEN circuits as to an inherent specification bias. The random graphs, on the other hand, showed an obvious and consistent bias.

Though not shown in the table, we note that there is a corresponding increase in the cpu time required place and route the GEN circuits and random graphs, which is roughly proportional to the increase in wirelength (i.e. small for GEN circuits, and double or more for random graphs).

These results clearly show the circuits produced by GEN are very similar to the MCNC originals and significantly more realistic than random graphs as benchmark circuits.

Locality Revisited.

The above empirical results are all for the original method of producing locality—using the locality parameter L described in Chapter 5.

As mentioned in the description of the algorithm there, our hope is to eventually use a form of locality generation which is based on the locality characterization in Section 3.5. We are pursuing ongoing work to that end, and have made changes to GEN which use *spread* and *span* to parameterize edge connections in Step 5 of the generation algorithm.

Unfortunately, these efforts have not yet shown any numerical improvements over simply using the locality parameter L. There are several possible explanations for this. One is that, although the span of a node models the *distance* between nodes in the delay based layout, it does not model the interaction between edges, i.e. crossings. It is possible that edges are at the correct distance, but exhibit a balanced (rather than clustered) distribution of crossing numbers across horizontal slices of the layout, making the place and route problem more difficult.

Though the empirical results show that we already have an excellent method of producing circuits, it is theoretically displeasing to not tie the issue of locality characterization into the generation algorithm. For this reason, we feel that further characterizations of locality, especially the ability to parameterize locality generation in ways such as described in Section 3.5, are an important direction for ongoing and future work.

6.4 Sequential MCNC Circuits.

We validate the sequential GEN-circuits by generating clones of 22 industrial benchmark circuits (provided by the Altera Corporation), and comparing the post-placement and routing statistics from VPR and Altera's *max+plus2* for the original circuit with that of the clone circuit and a equivalently sized (in terms of nodes, edges, flip-flops and I/O) random graph.

The benchmark circuits[3] were output as BLIF after synthesis and fitting with Altera's commercial place-and-route tool MAX+PLUS2 into an Altera 10K20RC240 FPGA, and all analysis by CIRC, including the extraction of clone scripts, takes place from that point. Given industrial criticisms of the MCNC circuits, it is extremely useful to be able to compare our results with real industrial circuits.

Table 6.2 shows the comparison between the original, GEN and random circuits after placement and global routing by VPR and implementation on an Altera 10K20-RC240 FPGA [4] by MAX+PLUS2. The benchmarks used are all of the appropriate size (between 60 and

[3]Use of Altera circuits was made while the author was a summer intern there and had access to proprietary data and software.

	VPR wire			VPR tracks			10K20 tracks	
		clone	rand		clone	rand	clone	rand
Circuit	orig	%diff	%diff	orig	%diff	%diff	%diff	%diff
A	5102	21	144	6	16	83	14	132
B	7719	64	215	5	80	160	71	.
C	6344	27	160	6	16	116	30	.
D	6818	20	147	6	16	133	32	.
E	6609	53	266	5	60	160	35	.
F	4293	57	188	5	40	140	41	197
G	4147	2	158	5	0	140	16	208
H	5107	21	137	5	40	120	0	123
I	4692	19	155	5	40	160	23	132
J	6087	34	153	5	60	120	51	165
K	9313	42	202	6	33	133	38	.
L	6546	36	222	6	33	100	55	.
M	7748	86	248	5	100	220	85	.
N	10794	-43	52	10	-40	30	-41	.
O	8070	17	140	7	14	100	25	.
P	5562	88	268	5	80	180	90	.
Q	6460	71	167	5	80	160	.	.
S	6417	29	166	5	40	140	24	.
T	4662	28	170	6	0	83	16	108
U	8828	2	156	6	16	150	53	.
V	4876	81	201	4	75	175	63	174
W	4837	28	143	4	50	150	34	117
mean	6358	35%	175%	5.5	38%	134%	36%	151%

Table 6.2: Empirical validation using sequential circuits from industry.

100% logic utilization, with most in the higher end of the range) for exercising this 10K20 part, which has 1152 LCELLS (logic blocks or LUT+FF combinations) and 240 user I/O pins.

The first column identifies the circuit. The second column gives the total wirelength after global routing. Then we give the percentage of extra wiring (beyond that required for the original) required by the corresponding clone circuit and random graph. Similarly, we then have the track-count (channel width) followed by the percentage increase in track-count for the corresponding clone circuit and random graph. The last two columns show the percentage increase in "routing resources" used by the clone circuit and the random circuit when implemented on the 10K20 FPGA. To respect information about the benchmark circuits which is proprietary to Altera the actual resource usage in the device is not displayed—for this study it is only the percentage difference that is of interest.

For our metric of FPGA resource usage, we count the total number of full-horizontal, half-horizontal and vertical lines used by the design in a 10K20, as reported by MAX+PLUS2. Because we are using an actual device, it is possible that a design does not "fit" (see Section 2.1.2). Though all original circuits do fit in the 10K20, one of the clone circuits and thirteen of the random graphs did not, and these are indicated by a '.' in the table.

The last row of the table indicates the averages for each column. For the last two columns, the missing data is *not* included in the average, which means that the numbers for random circuits are deceptively low.

We find that the clone circuits are, in general, harder to place and route than are the original circuits we took the specifications from, though a given clone is always closer to the original than the corresponding random graph. On average, the clone circuits used 35% more wirelength and 38% more tracks than the original circuit, whereas the random graphs used 175% more wirelength and 134% more tracks. This is further reflected in the implementation of the clone and random circuits on the commercial FPGA where (when they did fit) the clone circuits used an average of 36% more routing resources and the random graphs used 151% more routing resources. We also find that about half of the random graphs do not fit at all in the part, whereas only one clone failed to fit. In Section 4.3 we gave the definition of a measure quantifying generalized reconvergence for sequential circuits. By this measure, GEN circuits differ by about 0.19 on average, while random graphs differ by 0.28 on average. The difference in the average wirelength and track count between the original and clone circuits likely results from as yet unknown parameters. We hope to address the issue with future work on local structure in circuits.

These empirical results show that the GEN circuits are significantly more realistic than even carefully generated random graphs. Though not perfectly close, the GEN is able to generate circuits which are quite similar to the original benchmark circuits. We remark that, due to the proprietary nature of the circuits, we are not able to update the empirical results to take into account the new locality characterizations discussed in Sections 3.5, 5.2.2 and 6.3.