Chapter 4

Characterization of Sequential Circuits

Combinational circuits have limited application, and any general CAD tool or FPGA must be able to deal with sequential circuits. In this chapter, we expand our characterization of combinational circuits towards this goal.

Before we can proceed it is necessary to have a more detailed model of what we mean by a sequential circuit. In Section 4.1 we describe such a model, defining sequential circuits in terms of combinational building blocks. Section 4.2 describes the basic statistical characterizations arising from the model and our empirical analysis with the MCNC benchmark circuits, in particular the issue of "ghost" inputs and outputs arising in the decomposition of a sequential circuit. Section 4.3 extends the combinational characterization of reconvergence from the previous chapter to sequential circuits.

4.1 The Sequential Model.

We model a sequential circuit as a hierarchy of two or more combinational circuits connected with flip-flops and "back-edges." A single level sequential circuit is simply a combinational circuit.

For this work we consider only synchronous sequential circuits with a single global clock. This ensures that there is a well-defined notion of time in the logical operation of the circuit, and we can define "sequential levels" on the basis of time increments.

In addition to a single clock restriction, we ignore reset/preset/clear lines, assume uni-

directional I/O pins, and do not allow internal tristate buffers. None of these are major restrictions in a theoretical sense: our model can be generalized to allow for circuits to be analyzed or generated hierarchically, so multiple clocks could be hidden within sub-circuits generated separately without a great deal of difficulty. (The generalized model has not yet been implemented in CIRC and GEN.) Similarly, bidirectional pins and tristates can be simulated in standard logic.

In a practical sense, however, we point out that bidirectional pins and the correct physical layout of busses in a design are important, and a commercial system would certainly deal with them explicitly. This is particularly true for FPGA architectural experiments, as tristates or other buffers could be consumable resources and bidirectional pins may (depending on the architecture) introduce greater stress on the routing network than do separate input and output pins. Also, though clocks are often special resources, FPGAs have a limited number of them, and the software may have to deal with some clocks or reset lines as ordinary logic signals. We leave implementation of these detailed features for future work.

For simplicity we assume that the only registers allowed are D-type flip-flops (as is common for most commercial FPGAs). Thus all nodes are of type PI (primary input), LOG, (logic) or DFF (flip-flop). Recall from the previous chapter that PO (primary output) is a property of a logic node, not a separate node type.

Our abstract model of a sequential circuit is shown in Figure 4.1. The figure shows a 3-level sequential circuit. The definitions of primary input, primary output, and all measures of fanout remain as described in Chapter 3. The *sequential level*, $level(x)$ of node x is defined as 0 if x is a primary input, $1 + level(y)$ for a flip-flop x with input y, and MIN(level(y_i)) over all inputs y_i to x otherwise. Notice that all primary inputs must thus occur in sequential level 0. Define an edge (x, y) to be a *forward-edge* if $level(x) = level(y)$ and a *back-edge* if $level(x) > level(y)$. By definition, any other edge is necessarily from a node at sequential level i to a DFF at level $i + 1$, and we call it a *FF-edge*.

It is important to point out that, though this model could appear to apply only to certain types of circuits which have a pipelined appearance, it does not actually preclude other views of sequential connections. Rather we just *define* sequential levels in this way.

With the introduction of sequential levels, we have to modify the definition of combinational delay: for node x, delay(x) = 0 if x is a PI or a DFF and one greater than the maximum delay over its fanins, otherwise. The definition of edge-length is as before, even if

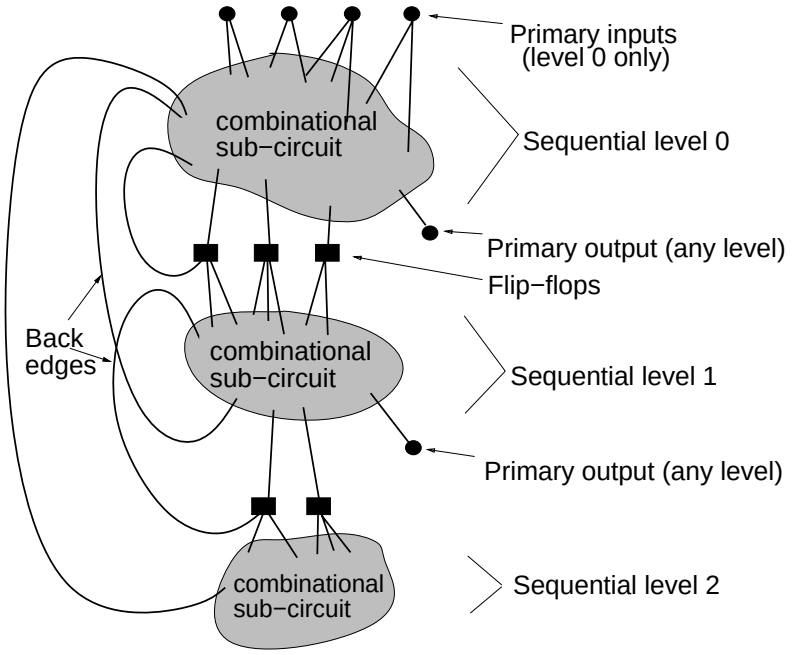

Figure 4.1: Abstract model of a 3-level sequential circuit

the nodes are at different sequential levels, except that edges to a DFF are always of length one. The *size* of the circuit is $n = n_{LOG} + n_{PI} + n_{DFF}$.

4.2 Characteristics of Sequential Circuits.

There are a number of new sequential characteristics arising directly from the model, and we describe them here. Note that all empirical results are based on the MCNC circuits, as mentioned previously in Section 3.1.

4.2.1 Basic Characteristics

The division of a circuit into its combinational sub-circuits introduces the concepts of *sequential shape*, the number of nodes in each successive sequential level, and the number of sequential *levels*. We also have counts of the numbers of flip-flops and back-edges. Table 4.1 shows this information for a sample of sequential MCNC circuits.

The number of I/Os is greatly decreased for sequential circuits, and we find that the Rent-like parameterization that we used before is no longer an adequate reflection of the I/O consumption for the circuit. In fact, we find that there is no real statistical correlation between the size of the circuit and the number of I/Os. In the default profile for sequential

Name	Nodes	IOs	nDFF	Edges	nBack	Levels	Seq. Shape
s838	167	37	32	556	256	2	169 65
s953	214	39	29	739	184	3	191 65 3
styr	238	19	5	814	219	2	207 45
planet	266	26	6	910	300	2	169 110
sbc	372	96	27	1273	300	2	388 51
mm30a	467	63	90	1697	235	2	500 90
dsip	1362	425	224	5440	896	2	1590 224
s298	1930	9	8	6944	2218	2	1636 305
bigkey	1699	425	224	6108	1344	2	1591 560
clma	8361	127	31	30114	5596	3	5810 2640 3

Table 4.1: Sequential circuit characteristics for selected MCNC circuits.

circuits, we use one quarter of the combinational I/O calculation as an upper bound on the number of I/Os, then choose the number of PI and PO for the circuit uniformly between 2 and the upper bound. In practice this yields reasonable values.

We find that the number of sequential levels is a small constant. Recall that a circuit with one sequential level is a combinational circuit. Of 78 sequential MCNC circuits, 69 have two sequential levels, 6 have three levels, and there is one circuit each of 4, 7, and 8 sequential levels. In all cases we saw, the majority of the combinational logic lies in the zeroth sequential level. We typically see successive sequential levels of logic having less than half the logic of the preceding level.

The number of flip-flops in a circuit also has little correlation to the amount of logic in the circuit. This can occur for many reasons. For example, the designer of a state-machine has the choice of encoding the state directly or in logarithmic size with extra decoding logic. Thus the number of flip-flops in the defaults file is also calculated with a wide degree of variation. We use a Gaussian distribution around a constant-deflated square root of the number of nodes as an approximation. See Appendix A. Note that this roughly models the number of flip-flops as the number of I/Os in a combinational circuit, not an unreasonable thing looking at the model. In FPGAs, the number of *available* flip-flops is usually proportional to the number of LUTs (often 1:1), but this is more to do with the design cost of adding the flip-flops and with logic block homogeneity than with the raw numbers of flip-flops required by circuits.

The number of back-edges varies between one and two times the number of nodes at the first sequential level, and we model it as such.

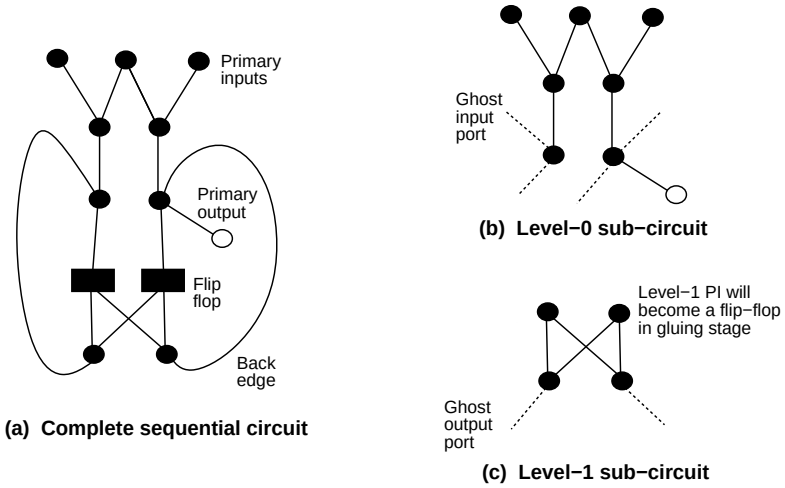

Figure 4.2: Example decomposition of a 2-level sequential circuit.

4.2.2 Decomposing Sequential Circuits.

Our model defines a sequential circuit based on its combinational sub-circuits, back-edges, and flip-flops. As part of the characterization process, we want to decompose a sequential circuit into its component parts. To describe the sequential interface within combinational sub-circuits we introduce the concepts of a *ghost input port* and *ghost output port*. Intuitively, these are points that connect different sequential levels (combinational sub-circuits).

These are best understood with a small example. Figure 4.2(a) shows a sequential circuit with three primary inputs, one primary output, two flip-flops (hence two flip-flop edges) and two back-edges. The decomposition of this circuit is shown to the right: Figure 4.2(b) shows the level-0 sub-circuit with 3 primary inputs and one primary output. We have two *ghost input ports* (GI) which record the existence of back-edges from a succeeding level, and two *ghost output ports* (GO) which record the location of back-edges connected to a preceding level or, as in this case, edges to flip-flops at a succeeding level. Similarly, Figure 4.2(c) shows the level-1 sub-circuit with two primary inputs (which used to be flip-flops) and two ghost outputs. Note that GI and GO ports correspond more closely to edges than nodes, since a single node can have up to $k-1$ ghost inputs, and *max_out* ghost outputs. We note that in any sub-circuit, a zero-fanout node must have at least one GO or PO attached to it.

In the parameterization of the combinational sub-circuits, it is not sufficient simply to record the number of ghost inputs and outputs, as this ignores a great deal of information about the interface between sub-circuits. In particular, if we are to use this model as the

basis for a generation algorithm, it is important to ensure that sub-circuits are *compatible*. For a single GO and GI to be compatible, the combinational delay of the node with the GO must be less than that of the node with the GI (i.e. it is legal/sensible to connect the GI to the GO in the context of combinational delay). For two sub-circuits to be compatible, there must exist a matching of GI and GO between them, all of which are compatible.

To deal with compatibility issues between sub-circuits, we introduce the GI and GO *shape* within sub-circuits. Define the vector *GIshape[i]* as the number of ghost inputs at combinational delay i, i=0..d, and $GOshape[i]$ similarly for ghost outputs. These will introduce a topological constraint on the connections between different sub-circuits in addition to simply the number of connections. In practice, we find that these vectors are important, because they often uncover "quirky" aspects of different circuits. Note that the GIshape for one level and the GOshape for the other level in a 2-level circuit will roughly correspond, but would only correspond exactly if all edges in the circuit were unit-edges, which is not usually the case.

For the circuit in Figure 4.2(b) we have GIshape = (0,0,2) and GOshape = (0,2); Figure 4.2(c) has GIshape = (0,0) and GOshape = (0,2). We note that flip-flops are not included in the GIshape of a level, because they are already recorded in n_{DFF} (a purely semantic detail).

As an example, the circuit **clma** has 3 sequential levels:

```
Level 0:
    GIshape = (  526 1245 664 354 451 429 860 502 295 48 37 25 22 2 4 0 0 )
    GOshape = (    0    0   0   8   4   7   1   2   0  2  0  0  0 1 1 2 1 )
Level 1:
    GIshape = (   74   45   3   8   2   0   0   0   0  0 )
    GOshape = ( 1289 1282 412 671 372 364 555 360 151  4 )
Level 2:
    GIshape = (    0    0 )
    GOshape = (  136    2 )
```

We find that the GI and GO shapes of MCNC sequential circuits do not statistically show any common shape beyond GIshape[i] being roughly proportional to shape[i] within sub-circuits. We have a heuristic process for generating reasonable GI and GO shapes which are compatible, and the interested reader is referred to the GEN source-code for details.

Note that the shape distributions for the combinational sub-circuits of sequential circuits differ from those of purely combinational circuits. This is because the second sequential level often has many more flip-flops (inputs) than is typical for a combinational circuit of

the same size.

4.2.3 Extensions to the Sequential Model.

With ghost input and output ports now defined, it is worth pointing out that the sequential model can be generalized to describe arbitrary levels of hierarchy, rather than just the interface between multiple levels in a simple sequential circuit.

For example, we can define a purely combinational circuit as a hierarchy of combinational sub-circuits simply by combinational specifications and a compatible GI and GO interface (without requiring that the circuit have flip-flop or back-edges). In combination with a partitioner this would allow us to form a partition tree model of an input circuit.

It would also be interesting to use this mechanism to describe an interface to other forms of circuits (e.g. memory), or to deal with circuits at the block diagram level.

The ability to generalize the use of ghost inputs in generation and outputs would open the door to a hierarchical generation process.

In this dissertation, however, we will restrict ourselves to simple sequential circuits.

4.3 Generalizing Reconvergence.

In Section 3.4 we defined the reconvergence number of a node r in a combinational circuit as the proportion of reconvergent nodes to total nodes in the out-cone of r. We also pointed out the combinatorial significance of the numerator as the $\log_2 t$, where t is the number of spanning out-trees rooted at r.

Recall a *spanning out-tree* $T(G)$ of a directed graph G with respect to a designated root node r is a spanning tree of G such that, for all x in G there is a (necessarily unique) directed r-x path in T.

Recall that the combinational *out-cone* of r in G (which we now denote G_r^c) is defined recursively as follows: r is in G_r^c and if x is in G_r^c and xy is a *forward* edge of G then the node y and the edge xy are also in G_r^c. Define the *sequential out-cone*, G_r^s of r to be identical, but *without* the restriction of xy being a forward edge. Then G_r^c is always a subgraph of G_r^s.

Using the sequential out-cone, the numerator in our reconvergence calculation no longer corresponds exactly to the number of spanning out-trees. Consider the sequential circuit

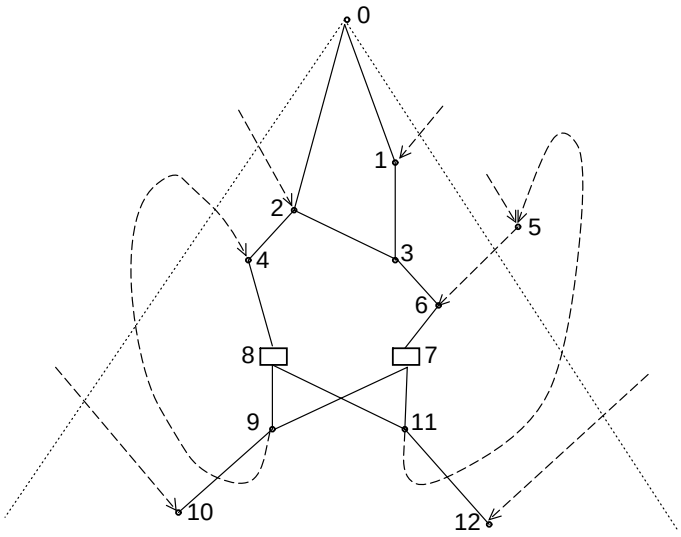

Figure 4.3: Reconvergence in a circuit.

represented in Figure 4.3. The combinational out-cone of node 0 is shown within dotted lines from 0. The number of reconvergent nodes in the combinational out-cone of node 0 is 3 (nodes 3, 9 and 11), and there are 2^3, or 8, spanning out-trees. However, the sequential out-cone of node 0 additionally includes vertex 5, and edges (11,5), (5,6) and (9,4). The number of reconvergent nodes in the sequential out-cone of node 0 is five, (nodes 3, 4, 6, 9, and 11), but the number of spanning out-trees is 15, not 32. The reason for this is that the choice of edges is no longer independent: no spanning out-tree can contain both (5,6) and (7,11).

Define the n by n matrix K with respect to a digraph G as follows:

$$K_{ij} = \begin{cases} \text{in-degree}(i) & i = j \\ -1 & i \neq j, (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

We note that K_{ii} is 0 if and only if i is a source in G, and that the sum of the entries in any column i is 0. Furthermore, if the vertices are in topological order[1], K is upper-triangular if and only if G is acyclic.

Now consider the graph G_r^s (with n' nodes) for digraph G with root r. Let K_r be the minor with respect to r of the Kirchoff matrix of G_r^s (i.e. the matrix formed by removing

[1]A *topological order* on the vertices of a directed acyclic graph G is any order σ such that the existence of edge xy implies that $\sigma(x) < sigma(y)$. Such an order always exists for an acyclic digraph.

row and column for r, resulting in a square matrix of dimension $n' - 1$). They we can apply the following to count the number of spanning out-trees from r in G_r^s.

Theorem (Kirchoff, c.f. [31]) *The number of spanning out-trees rooted at r in a finite digraph G is equal to the determinant of K_r.*

The basic idea of the proof is that as the determinant of this matrix is broken into terms by a standard linear algebra decomposition, the number of leaf corresponds to the number of trees from a given root vertex. The combinatorial justification for this process is explained fully in the book by Gibbons [31][pages 49-54], and the interested reader is referred there for the details.

For our purposes here, it is sufficient to explain the process with our example (Figure 4.3).

The intuition is clearer for acyclic graphs. Ignoring all back edges, the out-cone of node 0 consists of the 12 nodes and solid edges shown inside the cone. The Kirchoff matrix of the out-cone of 0 is then

$$K = \begin{pmatrix} 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Combinatorially, the number of spanning out-trees from G can be calculated as the product of the in-degrees of the vertices of G (not including the root)—if the in-degree of vertex x is 1, then that edge must be present in any spanning out-tree. If x has two or more inputs then any one can be chosen independently of other choices of edges in $T(G)$. Since K_r is upper triangular, its determinant is the product of the diagonal elements. (Note, because we chose the out-cone, the value is always at least 1.) Thus, the number of spanning out-trees in the out-cone of 0, ignoring back edges, is 2^3 or 8.

The situation is more complicated when we allow cycles. Adding the vertex 5 and edges (11,5), (5, 6) and (9,4) increases the dimension of K by 1, and makes K_r no longer

upper triangular; correspondingly, the choice of edges is no longer independent; no spanning subtree can contain both (5,6) and (7,11). Thus we utilize the thorem of Kirchoff, with

$$K = \begin{pmatrix}
0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 2 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 2 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

and $|K_0| = 15$. There are 15 spanning out trees from 0, not 32—more than the 8 in the combinational out-cone, but significantly less than the 32 obtained from counting reconvergent nodes in the sequential out-cone as if they were independent.

It should be clear that the number of spanning out trees can be seen as a true measure of the reconvergence of r, more so than the counting method. With this in mind, we define the sequential reconvergence number, R^s of a vertex v in G as

$$R^s(v) = \log_k \det(K_r(v))/|G_r^s|,$$

where K is calculated on G_r^s, and k is the maximum in-degree (LUT-size) of the circuit G.

So the reconvergence of any node v is the logarithm of the number of spanning out-trees normalized by the size of the out-cone G_v^s. The purpose of taking the logarithm, as before, is to scale the number to within a comprehensible range for large graphs; this, with the normalization by the size of the out-cone, generates $0 \leq R^s(v) < 1$ for G mapped into 2-LUTs and $0 \leq R^s(v) < k$ for G mapped into k-LUTs.

Note that $R^s = 0$ if and only if the out-cone is already a tree.

To calculate the sequential reconvergence number of a graph we take, as in the combinational case, the weighted average of the reconvergence numbers of its primary inputs.

Note that the combinational reconvergence number of a sequential circuit is still well-defined. It is equivalent to performing the calculation on the circuit G with all back-edges removed or ignored. It is often, but not always, true that $R^c < R^s$; it depends on the

relative growth of the out-cone compared to the additional reconvergence in it.

Implementation Details.

Calculating the determinant of an n by n matrix uses $O(n^3)$ time. In CIRC we use a sparse-matrix implementation, which greatly decreases the required computation time. However, it is still not practical to calculate R^s for circuits with more than about 5,000 LUTs.

We take care to deal with the numerical stability of the determinant calculation with row pivoting, but above 2,000 LUTs, we sometimes encounter ill-conditioned matrices. CIRC will warn the user in these cases.

Empirical Calculations of R^c and R^s.

We calculated the combinational and sequential reconvergence numbers for all MCNC circuits. A sample of these is shown in Table 4.2. For comparison, we give R^c for both 2-LUT and 4-LUT mapped circuits, and R^s for 4-LUT mapped circuits.

We note, as in the combinational case, that any results here are biased by the contents of the MCNC benchmark set, which has limited documentation and could be missing large classes of logic. Thus our comments can only be based on the data that is available.

Observe that, as in the combinational case, there is a reasonable amount of grouping among the different types of logic. The arithmetic logic falls in the lower part of the spectrum, and finite state machines in the higher end. Within these bands, we notice, for example, the closeness of the reconvergence numbers for different implementations of multipliers, and for multipliers of different sizes. This data indicates that the reconvergence number is useful information, and captures some part of the fundamental nature of circuits. It would be very interesting to do these comparisons with greater information about the circuit functionality than we have, but the MCNC circuits have no documentation beyond these brief one-line descriptions.

There are several reasons that the finite state machines tend to have large reconvergence numbers: they often have very few I/Os, and their first sequential level often has a very exaggerated conical shape. Because of the small number of I/Os we often see that the sizes of the combinational and sequential out-cones are close, also explaining why they tend to have large R^c.

We point out the growth in the amount of reconvergence in a circuit as k increases, which

Name	R^c (k=2)	R^c (k=4)	R^s (k=4)	Circuit Description
elliptic	0.47	0.85	0.26	elliptic eqn solver
dsip	0.27	0.81	0.28	encryption
mult16b	0.36	0.56	0.30	16-bit multiplier
mult16a	0.54	0.62	0.36	16-bit multiplier
mult32a	0.54	0.61	0.36	32-bit multiplier
s208.1	0.38	0.39	0.45	digital fractional multiplier
s344	0.38	0.59	0.57	4-bit multiplier
ecc	0.66	0.96	0.58	error-correcting
lion	0.52	0.79	0.63	fsm
bbtas	0.76	0.84	0.73	fsm
traffic	0.50	0.73	0.78	fsm, traffic light
bigkey	0.53	0.89	0.83	key encryption
sbc	0.47	0.53	0.83	snooping bus controller
dk27	0.77	1.00	0.88	fsm
dk15	0.65	0.88	0.92	fsm
s382	0.63	1.04	0.92	fsm, traffic light
bbara	0.70	0.94	0.93	fsm
mm30a	0.69	1.12	0.95	min-max
mark1	0.58	0.76	1.03	fsm
s526n	0.63	1.04	1.03	fsm, traffic light
mm4a	0.66	1.12	1.04	min-max
tseng	0.49	0.78	1.04	bus-controller
keyb	0.72	0.99	1.14	fsm
opus	0.66	0.85	1.14	fsm
dk14	0.65	1.17	1.15	fsm
ph-dcd	0.61	1.02	1.19	phase decoder
diffeq	0.57	0.97	1.20	differential eqn solver
gcd	0.37	0.66	1.22	compute gcd
s832	0.65	0.90	1.22	fsm
bbsse	0.69	1.04	1.23	fsm
ex6	0.66	1.11	1.23	fsm
mm9b	0.68	1.15	1.23	min-max
sse	0.69	1.04	1.23	fsm
s820	0.67	0.96	1.24	fsm from a PLD
dk17	0.71	1.09	1.25	fsm
sand	0.77	1.06	1.25	fsm
styr	0.77	1.09	1.25	fsm
s510	0.77	0.75	1.33	fsm controller
dk512	0.80	1.39	1.34	fsm
s1	0.76	1.29	1.44	fsm
s1488	0.83	1.32	1.45	fsm controller
pma	0.80	1.32	1.46	fsm
s1494	0.82	1.37	1.47	fsm controller
planet	0.84	1.36	1.50	fsm
s298	0.84	1.60	1.60	fsm from a PLD
bbrtas	0.93	1.65	1.65	fsm

Table 4.2: Reconvergence for selected MCNC circuits.

is as one would expect: the number of nodes in an out-cone decreases, but the number of reconvergent paths remains unchanged (except when entirely "consumed" by a larger LUT).

There is a reasonably strong correlation between R^c and R^s, however not enough that one can predict the other. There are a number of cases where the two are drastically different. We reiterate our belief that R^s has a more theoretically pleasing value because of its combinatorial interpretation. However, as noted earlier, we can only effectively calculate it up to about 5,000 LUTs. Beyond that point R^c becomes the only available value.

Reconvergence and Routability

It is interesting to compare the routability of circuits with their reconvergence numbers. However, routability is (obviously) sensitive to both the number of nodes and the number of edges in the circuit so we need a large number of circuits which are very close in size. Such a subset does not exist in the MCNC circuits.

Some such experiments were possible with the Altera benchmark circuits, where we do have large numbers of similarly sized circuits. We find that R can be used in combination with other parameters to form a model of routability, but that any predictions are still dominated by other parameters which prevent us from isolating reconvergence. Further details of this particular study constitute proprietary information, but we leave the direction of research for future work.