Compact, Multilayer Layout for Butterfly Fat-Tree

André DeHon
California Institute of Technology
Department of Computer Science, 256-80
Pasadena, CA 91125
andre@acm.org

ABSTRACT

Modern VLSI processing supports a two-dimensional surface for active devices along with multiple stacked layers of interconnect. With the advent of planarization, the number of layers can be large (6 or 7 in modern designs) and more layers are feasible if the cost is justified. Using a multilayer-wiring VLSI area model, we show how a butterfly fat-tree (or fat-pyramid) with N processors can be laid out in $\Theta(N)$ active device area using $\Theta(\log(N))$ wiring layers. This result may have practical value in laying out efficient, single-chip multiprocessors and FPGAs. It may also provide a theoretical basis for the rate of layer scaling empirically seen in VLSI designs.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Types and Design Styles—*VLSI*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network Topology*; C.1.4 [**Processor Architectures**]: Parallel Architectures

General Terms

VLSI Layout Theory, Fat-Tree, Fat-Pyramid, Scaling, Universal Network, Multiprocessor, FPGA

1. INTRODUCTION

Traditional VLSI area models (*e.g.* [10]) assume two, or a small fixed number of, wiring layers, which was very appropriate for early VLSI process capabilities. With this model it was possible to identify many interesting cases where wiring limitations determined the size required by chips. Modern VLSI processes, perhaps in response to the empirical recognition of these wiring limitations, now offer many layers of wiring. It is, consequently, interesting to review VLSI wiring restrictions exploiting the new multilayer wiring model.

This paper looks specifically at fat-tree style wiring. The fat-tree was constructed specifically to be efficient for VLSI layouts, and the canonical 2D fat-tree is an example of a structure whose area is wiring limited. Further, the fat-tree can be used as a universal interconnect or wiring substrate. We show that the wiring structure in the fat-tree is sufficiently regular to permit a layout in $\Theta(N)$ area (the area dictated by the nodes and switches) using $O(\log(N))$ wiring layers. This should be compared to an area of $O(N\log^2(N))$ using a conventional, 2D, bounded wiring layers, layout for a fat-tree.

The paper starts with an abstract of modern, multilayer VLSI layout (Section 2) and a review of the butterfly fat-tree and fat-pyramid (Section 3). In Section 4, we demonstrate the major result that a butterfly fat-tree can be placed and routed efficiently using multiple wiring layers. In Section 5, we look at how this result may relate to VLSI wiring growth. We identify a few, interesting, open questions which this raises in Section 6.

2. MODERN, MULTILAYER VLSI LAYOUT

Contemporary VLSI processes easily offer 6 layers of metalization for wiring. With the advent of Chemical Mechanical Planarization (CMP) [9], it is feasible for process technology to continue stacking additional metal layers as long as the cost of the extra mask steps and processing are justified by the area benefits. This produces an interesting twist on the traditional VLSI models. With current technology, active devices (transistors, gates, buffers) are still largely limited to two-dimensional layout on the silicon substrate. However, wire layers can feasibly be stacked on top of each other creating a three-dimensional structure for interconnect and wiring.

This gives us a model where:

1. Devices which actually compute upon, store, or switch data must be laid out in two dimensions.
2. Wires which interconnect these devices have finite width and spacing.
3. Wires on any two wiring layers can be interconnect with vias and will take up finite space on all intervening layers.

If the active devices for some structure take up total area A, then it is interesting to ask if the active devices can be laid out compactly to fit in $O(A)$ two-dimensional surface area and be supported by the multilayer wiring. Further, we should ask how many wiring layers are required to support this compact active area layout.

3. BUTTERFLY FAT-PYRAMID AND FAT-TREE

The particular structure we are interested in here is Leiserson's Fat-Tree [7] and, by extension, Greenberg's Fat-Pyramid [3]. Results from Leiserson and Greenberg show that an N-node fat-tree (and fat-pyramid) can be laid out in $O(N\log^2(N))$ area [4] using the

SPAA 2000 Bar Harbor, Maine
Copyright ACM 2000 1-58113-185-2/00/07...$5.00

fold-and-squash technique of Leighton and Bhatt [1]. Figure 1 shows the Butterfly version of Leiserson's Fat-Tree [5] (the fat-pyramid is similar, adding a constant number of additional wires between physical adjacent switch nodes at the same tree level) along with its compact, fold-and-squash layout.

For this layout it is important to note that each 4-ary tree layer, corresponding to multiplying the number of nodes in the tree by 4, adds:

- a constant number of wire tracks (6 as shown) per "cubie"[1]
- a constant number of switches (1) to *some* cubies

Hence we get the logarithmic growth in the side width of each cubie due to wiring. Since wire width alone in the 2D VLSI model dictates a side growth of $O(\log^2(N))$, it does not (theoretically) matter than some cubies have a switch count which is growing as $O(\log(N))$. The overall result is that the area of the N-node fat-tree grows as $O(N\log^2(N))$.

Active Devices

It is, however, worthwhile to note that the number of active devices in the butterfly fat-tree (and fat-pyramid) converges to a constant independent of the number of tree levels. It should be trivially clear that the number of endpoints nodes is N. It is also true that the number of switching nodes is $\Theta(N)$. For example, if we assume a 4-ary tree with switches with 4 down links and 2 up links, as shown, then the total number of switches is at most $\frac{N}{2}$. To see this, note that each group of 4 leaf nodes needs one switch at the lowest level (labeled 4 in Figure 1). At the next level, we need half as many switches (every 4 switches on the lower level needs 2 switches at the next level). This relationship continues with each succeeding level requiring half as many switches as the level before. Consequently, the number of switches needed per endpoint can be calculated as a classical geometric series:

$$N_{switch} = \frac{N}{4} + \frac{1}{2}\left(\frac{N}{4}\right) + \frac{1}{4}\left(\frac{N}{4}\right) + \frac{1}{8}\left(\frac{N}{4}\right) + \cdots \leq \frac{N}{2} \tag{1}$$

Since switches and endpoints make up the entire set of active devices, this demonstrates the active device area for a butterfly fat-tree is $\Theta(N)$.

4. LAYOUT

Having established that the active device requirement for a butterfly fat-tree is $\Theta(N)$, the question remains as to whether or not the device can be conveniently laid out in this area and the wiring can all be performed in a reasonable number of wiring layers. We also note from our observations in the previous section that the number of wiring channels per cubie is $O(\log(N))$. Since it is necessary to build a cubie in space $O(1)$ if we are to layout the entire tree in active, two-dimensional area $O(N)$, then that sets a trivial lower bound of $\Omega(\log(N))$ on the number of wire layers required to wire the fat-tree. In fact, it is possible to organize the fat-tree so that it can be laid out in $O(N)$ active area and $O(\log(N))$ wiring layers. Two show this is possible, we demonstrate two things:

1. The switches can be arranged to be placed into cubies so there are at most a constant number (2) of switches in any cubie.
2. When we account for **both** the wiring per layer and the through vias required between layers, we do not saturate any of the wiring layers.

4.1 Switch Placement

Figure 2 shows the rearrangement of the basic fat-tree and its fold-and-squash layout. The rearranged fat-tree is topologically equivalent to the original fat-tree (Figure 1). However, when this rearrangement is folded up, at most 2 switches end up in the cubie along with 4 processing nodes (Figure 8, provided at the end of the paper, builds the tree one level higher to better show this effect).

In the original fat-tree arrangement, all the switches lie along the same diagonal. In the new arrangement, the diagonals are complementary so that, when folded together, the next level diagonal is always left open. Figure 3 shows the actual folding sequence to display the basic invariant maintained by this arrangement. Each final cubie will contain the 4 leaf processing element, the switch associated with those four processing elements, and, at most, one additional switch. For clarity, the processors and first switch (labeled 4) are not shown in Figure 3 once folding begins.

Notice, at each stage, that, after folding, the lower level(s) manages to leave **both** main diagonals free. One main diagonal is then consumed by the new switches added at the level onto which the lower levels are being folded. This, in turn, leaves one diagonal free in the folded box. As a consequence when this new level is now folded with its peers to create the next tree level, it will also create a structure with both main diagonals free so that the next level of switches can be added and the folding can continue in this manner *ad infinitum*.

4.2 Wires

The basic strategy for wiring is to give each tree layer its own pair of wire layers—one for horizontal wiring and one for vertical wiring. In all likelihood the constants will work out such that more than one tree layer can share the same wiring layer, but for the sake of clear exposition, we will use this generous assumption. As shown, the wiring per tree layer is, at most, 6 wires wide,[2] so we immediately see we have a constant number of wires running through each cubie side on each of the $2 \cdot \log(N)$ wiring layers.

Now, we must also show that we can accommodate all of the through vias in constant area. Since there are at most two switches per cubie, there must be at most $6 \times 2 = 12$ through interconnect vias from the substrate to some routing layer in each cubie.[3] We can allocate a via track for each wire channel in each wiring layer in order to make the connection down to the substrate. Further, the vias in this channel will need to be spaced one wire channel apart to avoid blocking the wires running the orthogonal direction (see Figure 4b). As shown in Figure 4a-b, each of the channels stacked on top of each other on different routing layer can route out to the single via channel and down to the substrate when it needs to connect without creating interference with the other channels in its stack. Note also that we assume the horizontal and vertical layers for a given tree layer are adjacent so that via connections between them can be made without disturbing wiring on any other wiring layers. This composite construction shows that we can wire each cubie in

[1] Cubies shown here contain 4 processing nodes, but are otherwise similar to the cubies shown in [4].

[2] Again, this could almost certainly be done with less wires per channel, but that would only complicate the description.

[3] Actually, since 4 of those connections are to the endpoint nodes, we only have 8 to worry about for the tree wiring layers.

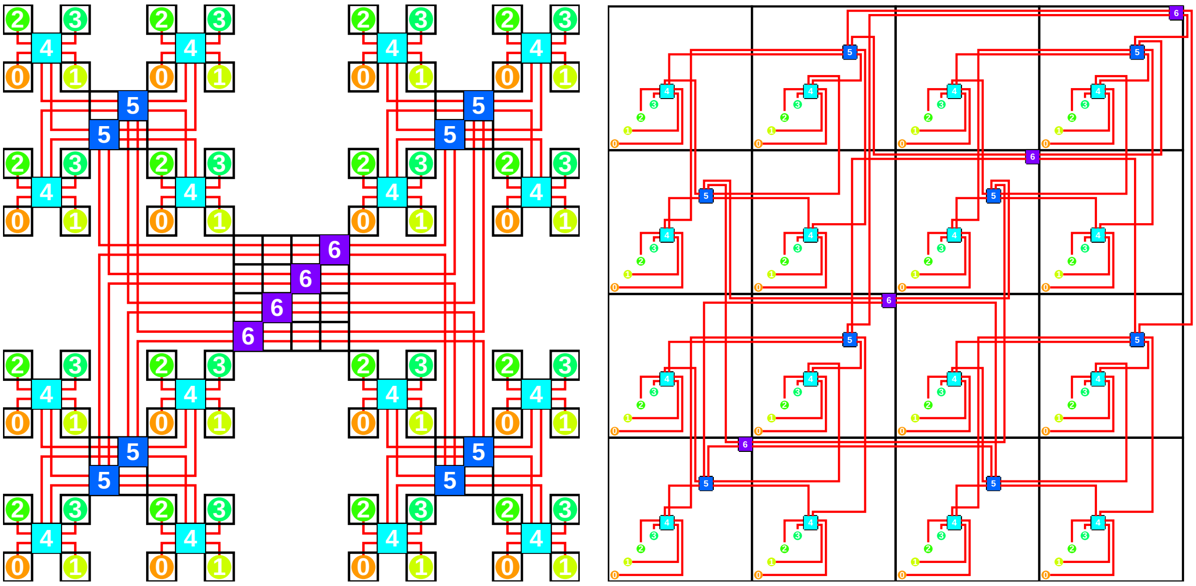

Figure 1: Butterfly Fat-Tree and Compact Layout

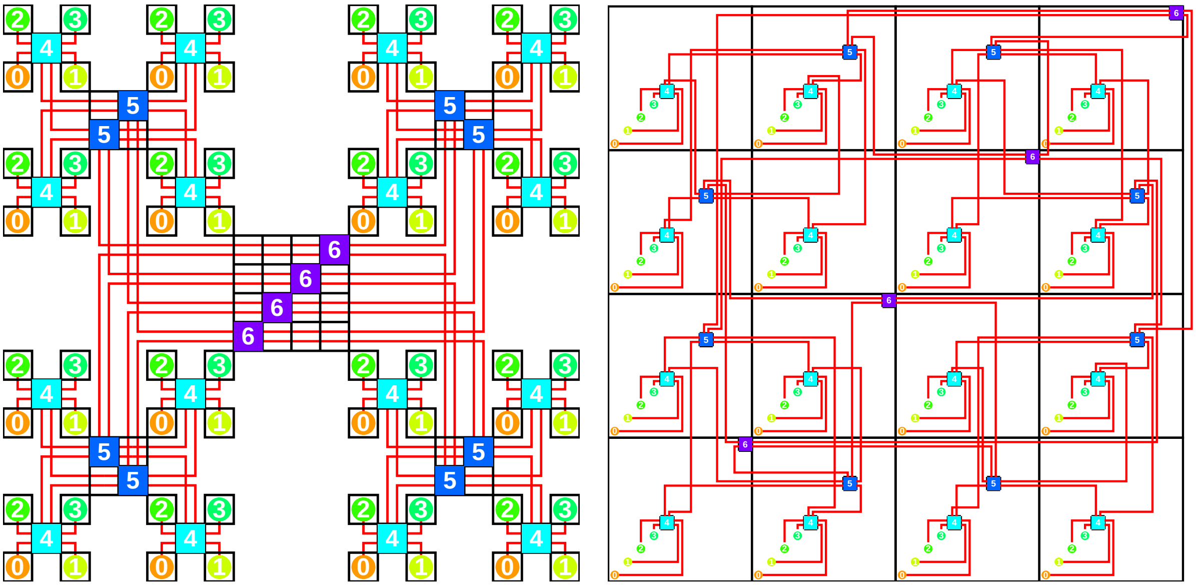

Figure 2: Rearranged Butterfly Fat-Tree and Compact Layout

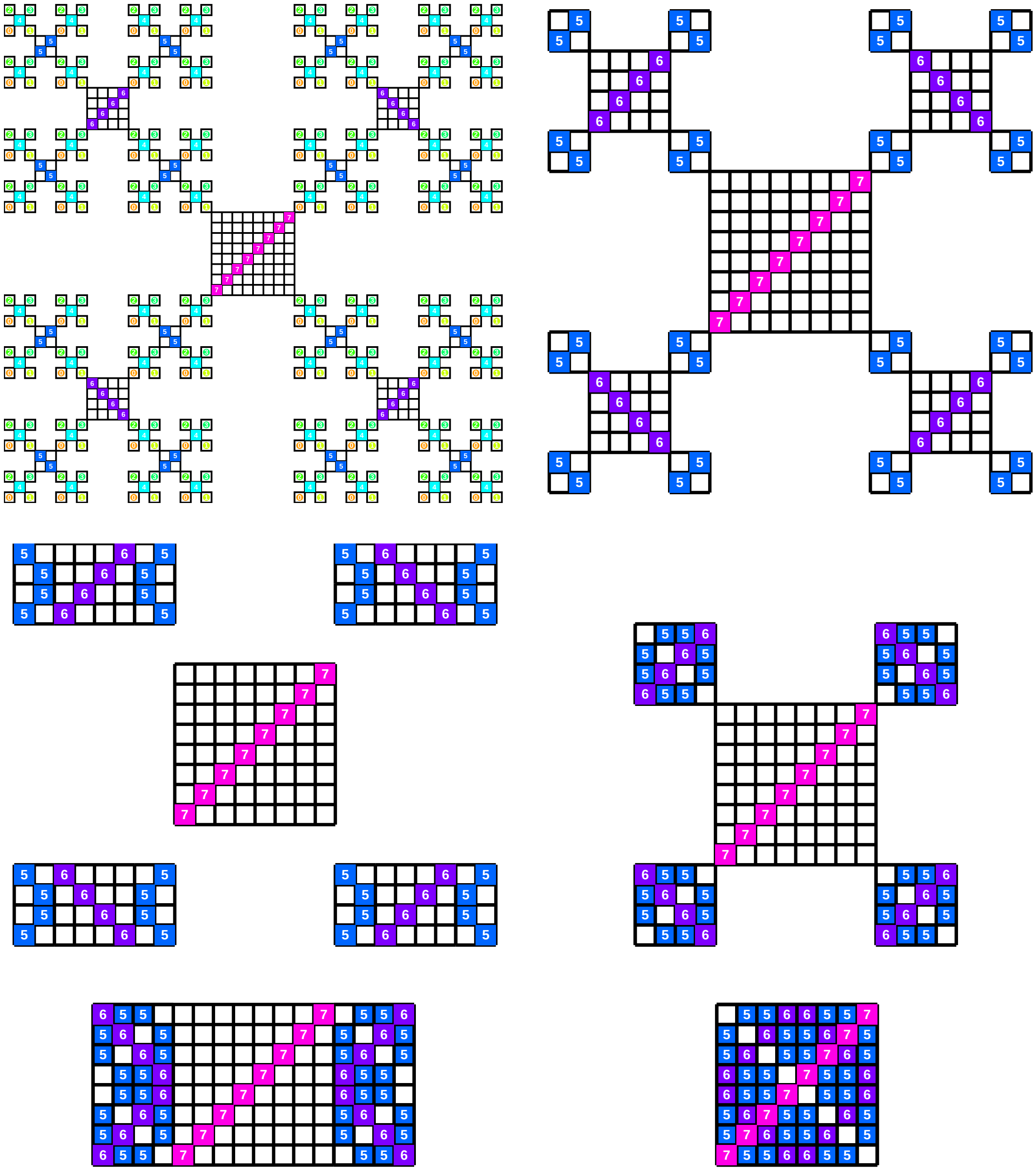

Figure 3: Fold Sequence for Rearranged Butterfly Fat-Tree

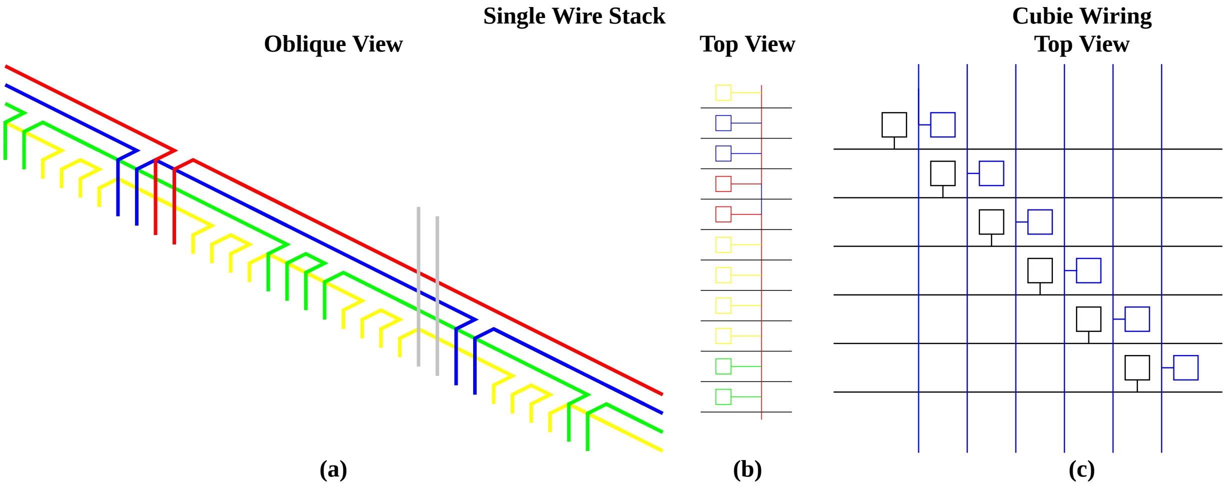

Figure 4: Wiring Pattern

constant, two-dimensional surface area if given $O(\log(N))$ wire layers.

In practice, we would not want to run a pair of wires directly in parallel for long runs due to potential coupling and hence crosstalk effects. Using standard techniques for twisting wires among the channels we can reduce the crosstalk coupling while maintaining our asymptotic bounds. Strictly speaking, adding a shielding layers between wiring runs would also provide such protection without changing the asymptotic bounds, but that should not be necessary.

Together with earlier observations about switch placement, this demonstrates our original claim that the entire butterfly fat-tree can be laid out compactly in $\Theta(N)$ active area and $\Theta(\log(N))$ wiring layers.

5. VLSI IMPLICATIONS

The immediate implication of this result is that we can use the butterfly fat-tree routing topology to compactly layout single-chip multiprocessors and FPGAs (*e.g.* [11]). This says that, given enough metal layers, we can layout an $\alpha = \frac{1}{2}$ bifurcating fat-tree [1], which Leiserson identifies as area universal [8], in area linear in the number of graph nodes. As noted above, this is better than the $O(N \log^2(N))$ area required if we limit the number of metal layers to a constant independent of N.

Empirically, one can observe that the number of metal layers *has been* steadily increasing with the active device capacity of our chips. Bohr observes that the number of metal layers has been increasing at the rate of 0.75 layers per IC generation [2]. Each generation represents a feature size reduction to 0.7× the previous generation. Assuming die sizes stay roughly constant,[4] this means each generation roughly doubles the area for active computing devices. Adding a constant number of metal layers per capacity doubling represents a logarithmic growth, or the same asymptotic bound which we demonstrated above for the fat-tree.

Two-dimensional VLSI layout theory would have predicated that, if our circuits have interconnect as rich as $p \geq 0.5$ (Rent's Rule [6]) or natural bifurcators with $\alpha \geq \frac{1}{2}$, then the number of active device we can usefully place on a VLSI component will scale sublinearly as devices are pushed out to accommodate the necessary interconnect wiring. This result and Bohr's suggest that processes have evolved to avoid this effect by correspondingly adding metal layers at a logarithmic rate to accommodate the richer interconnect requirements of our designs. Our results demonstrate that wiring layer growth is, in fact, sufficient to allow us to wire up universal routing structures efficiently; that is, with the logarithmic growth in wiring layers, we can place a number of active devices on the die which is linear in the total component area.

6. OPEN QUESTIONS

Can we layout an $\alpha > \frac{1}{2}$ tree in $O(N)$ area with any number of wire layers? with $O(N^{2p-1})$? how? A more general butterfly fat-tree can have a different growth rate in aggregate channel capacity [8] than the area-universal one where the main channel doubles when the subtree quadruples (matching he $\sqrt{A}$ perimeter I/O to area ratio of a two-dimensional layout). For any larger geometric channel growth rate (less than a complete doubling at every stage – *i.e.* $\alpha < 1$), the number of switches in the butterfly fat-tree will still be only $O(N)$. So, the question here, is: is there a similarly clever way to arrange the switches in this more general case? And, can the wiring and through via connections also be arranged to work out?

7. SUMMARY

We have noted that the assumption of a fixed number of wiring layers independent of device capacity does not match technology advances in modern VLSI. Using a multilayer model, we showed that the fat-tree can be arranged and laid out in $\Theta(N)$ area using $\Theta(\log(N))$ wiring layers. Finally, we noted that the growth rate derived here matches empirical observation of the growth rate of wiring layers in VLSI processes, suggesting that general designs have encountered similar wire limitations, encouraging processes to scale wire layers to meet wiring demands.

The primary contributions of this paper are:

- Show how to arrange the switches for folding so there is conveniently a constant number of switches along with each processing node tile (cubie).

[4] Die sizes are not entirely constant, but this is a reasonable approximation for our purposes.

- Show that the wiring can be arranged so as not to saturate intervening layers with through via connections.
- Assemble these two results to demonstrate the aforementioned claim for compact fat-tree layout.

8. ACKNOWLEDGMENTS

This research was part of the Berkeley Reconfigurable Architectures Software and Systems effort supported by the Defense Advanced Research Projects Agence under contract number DABT63-C-0048.

9. REFERENCES

[1] S. Bhatt and F. T. Leighton. A framework for solving vlsi graph layout problems. *Journal of Computer System Sciences*, 28:300–343, 1984.

[2] M. Bohr. Interconnect scaling – the real limiter to high performance ulsi. In *International Electron Devices Meeting 1995 Technical Digest*, pages 241–244. Electron Devices Society of IEEE, December 1995.

[3] R. Greenberg. The fat-pyramid and universal parallel computation independent of wire delay. *IEEE Transactions on Computers*, 43(12):1358–1365, December 1994.

[4] R. I. Greenberg and C. E. Leiserson. A compact layout for the three-dimensional tree of meshes. *Applied Math Letters*, 1(2):171–176, 1988.

[5] R. I. Greenberg and C. E. Leiserson. *Randomness in Computation*, volume 5 of *Advanes in Computing Research*, chapter Randomized Routing on Fat-Trees. JAI Press, 1988. Earlier version MIT/LCS/TM-307.

[6] B. S. Landman and R. L. Russo. On pin versus block relationship for partitions of logic circuits. *IEEE Transactions on Computers*, 20:1469–1479, 1971.

[7] C. E. Leiserson. Fat-trees: Universal networks for hardware efficient supercomputing. *IEEE Transactions on Computers*, C-34(10):892–901, Oct. 1985.

[8] C. E. Leiserson. Vlsi theory and parallel supercomputing. MIT/LCS/TM 402, MIT, 545 Technology Sq., Cambridge, MA 02139, May 1989. Also appears as an invited presentation at the 1989 Caltech Decennial VLSI Conference.

[9] W. T. Siegle. Interconnection technology for modern logic devices; an exercise in system engineering to assure manufacturability. In *Proceedings of the 1994 Materials Research Society Symposium*, volume 337, pages 3–11. Material Research Society, 1994.

[10] C. Thompson. Area-time complexity for vlsi. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, pages 81–88, May 1979.

[11] W. Tsu, K. Macy, A. Joshi, R. Huang, N. Walker, T. Tung, O. Rowhani, V. George, J. Wawrzynek, and A. DeHon. Hsra: High-speed, hierarchical synchronous reconfigurable array. In *Proceedings of the International Symposium on Field Programmable Gate Arrays*, pages 125–134, February 1999.

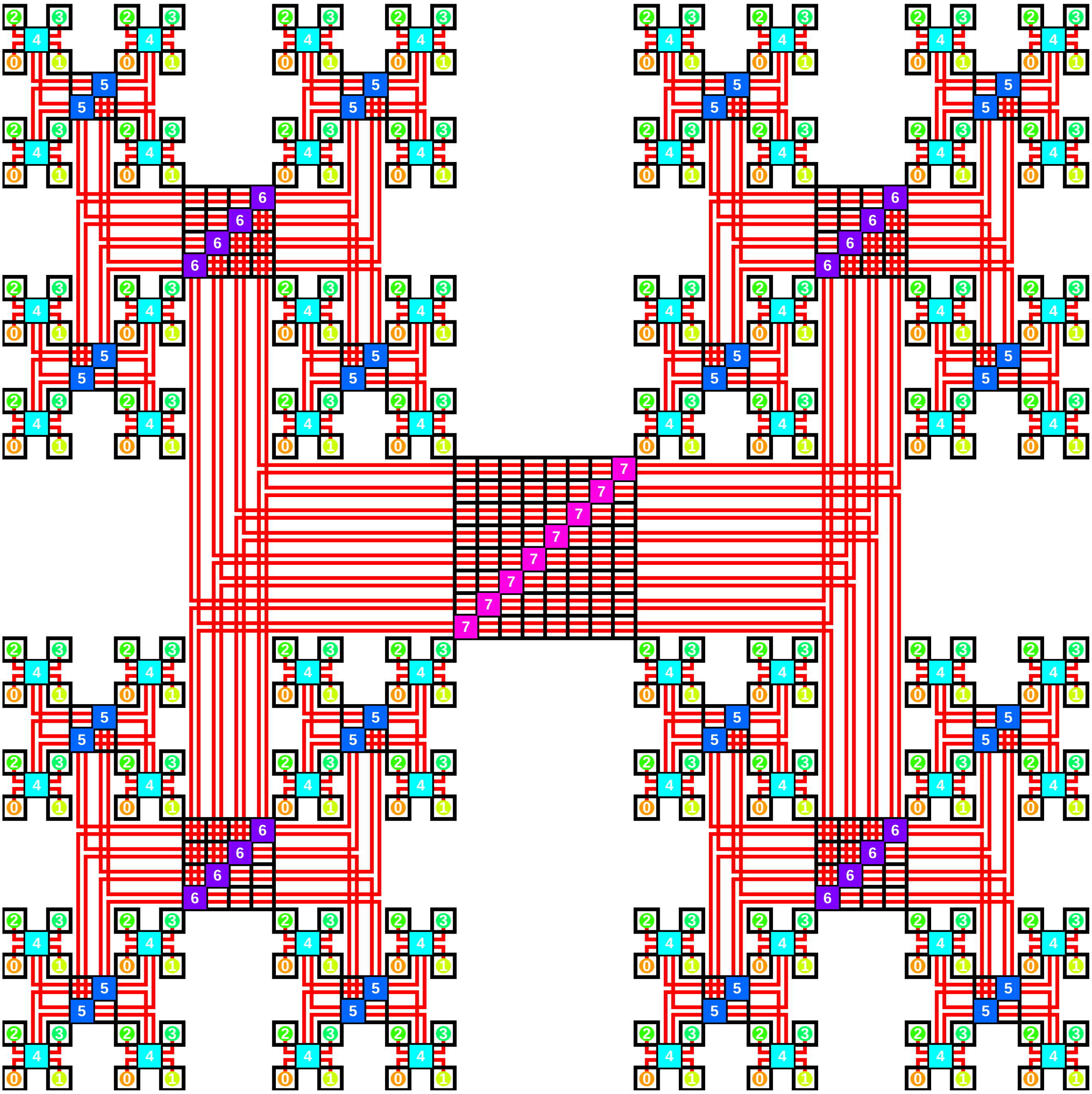

Figure 5: 256-node Butterfly Fat-Tree

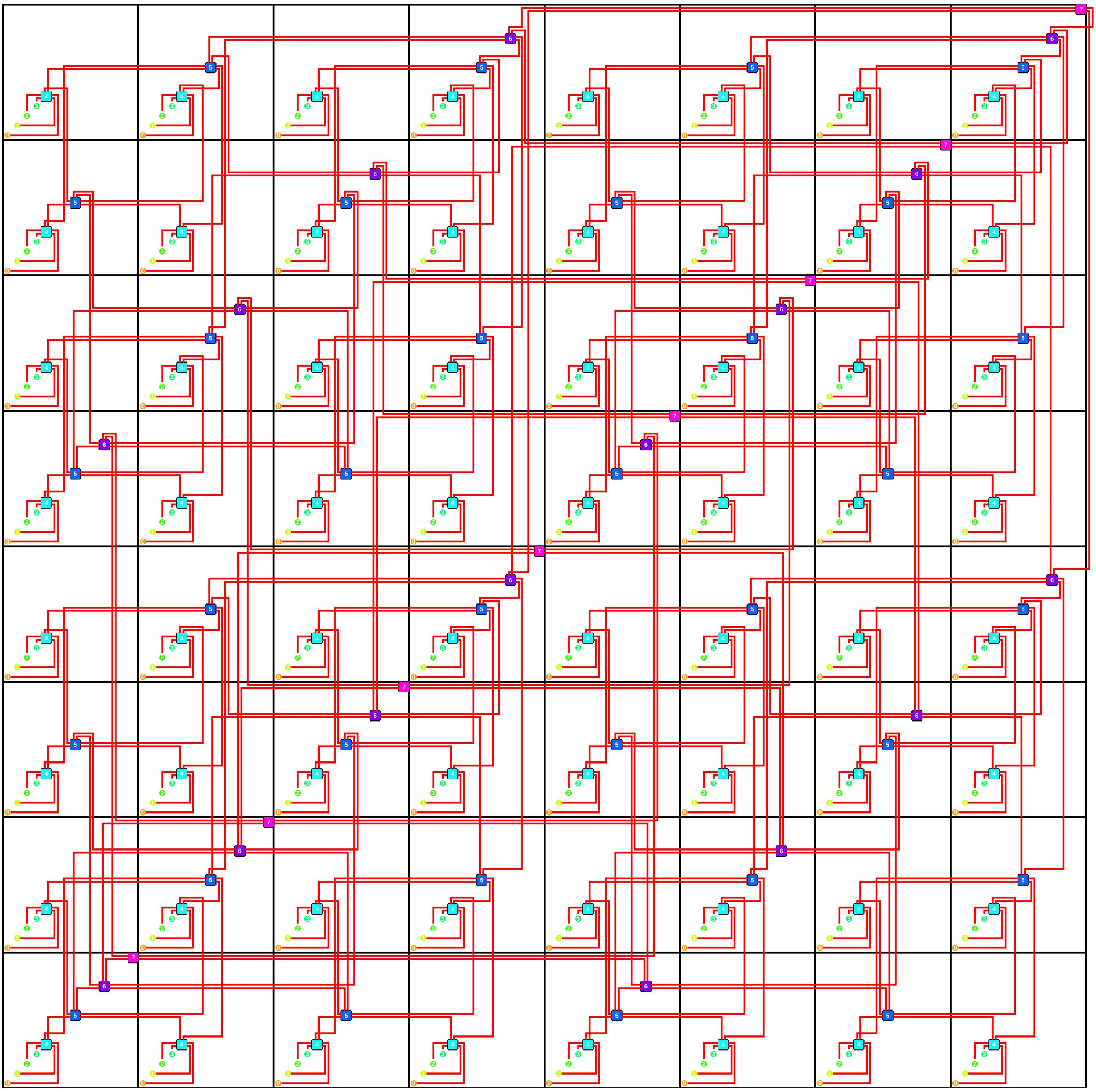

Figure 6: Fold-and-Squash Layout for 256-node Butterfly Fat-Tree

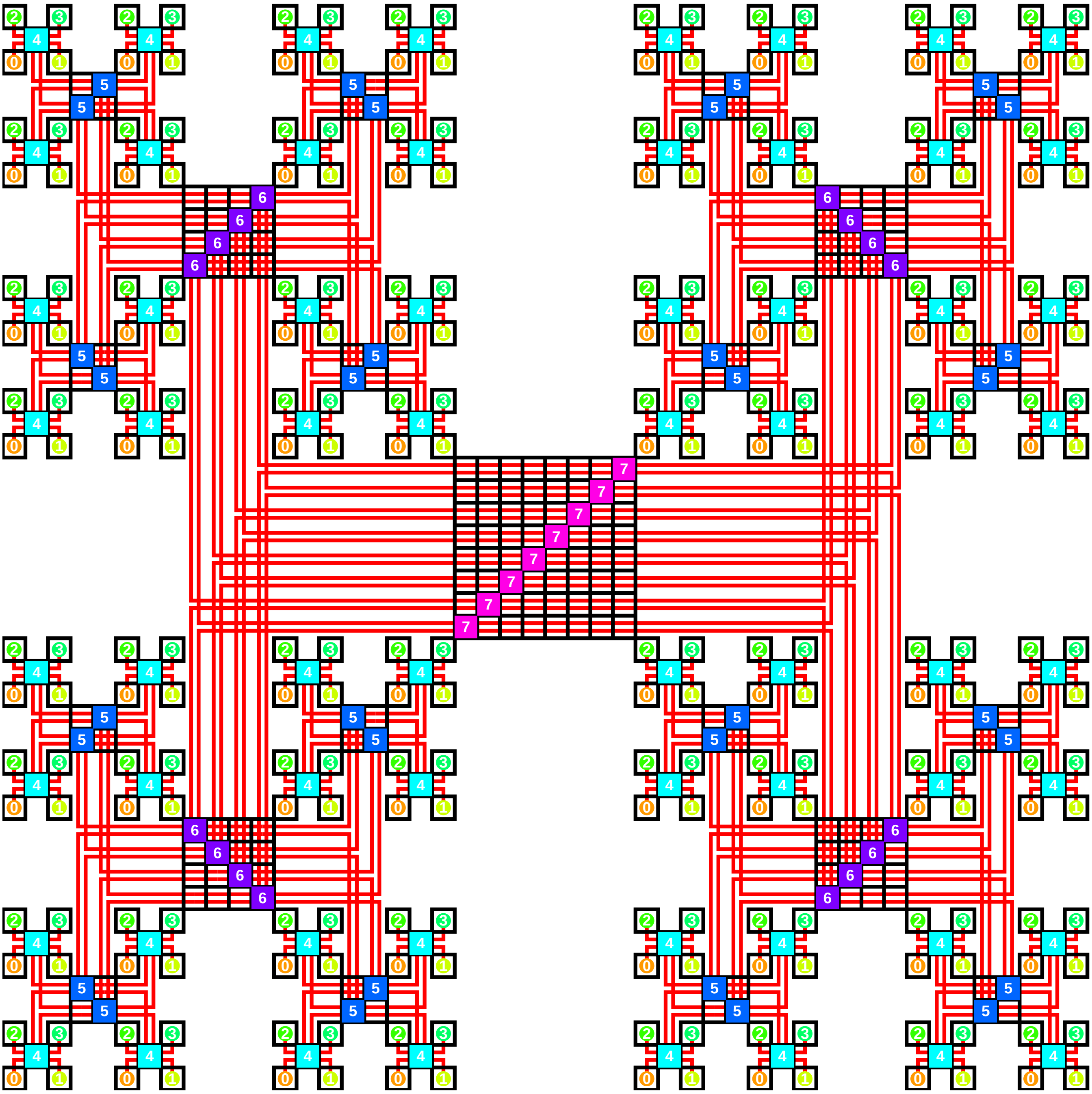

Figure 7: 256-Node Rearranged Butterfly Fat-Tree

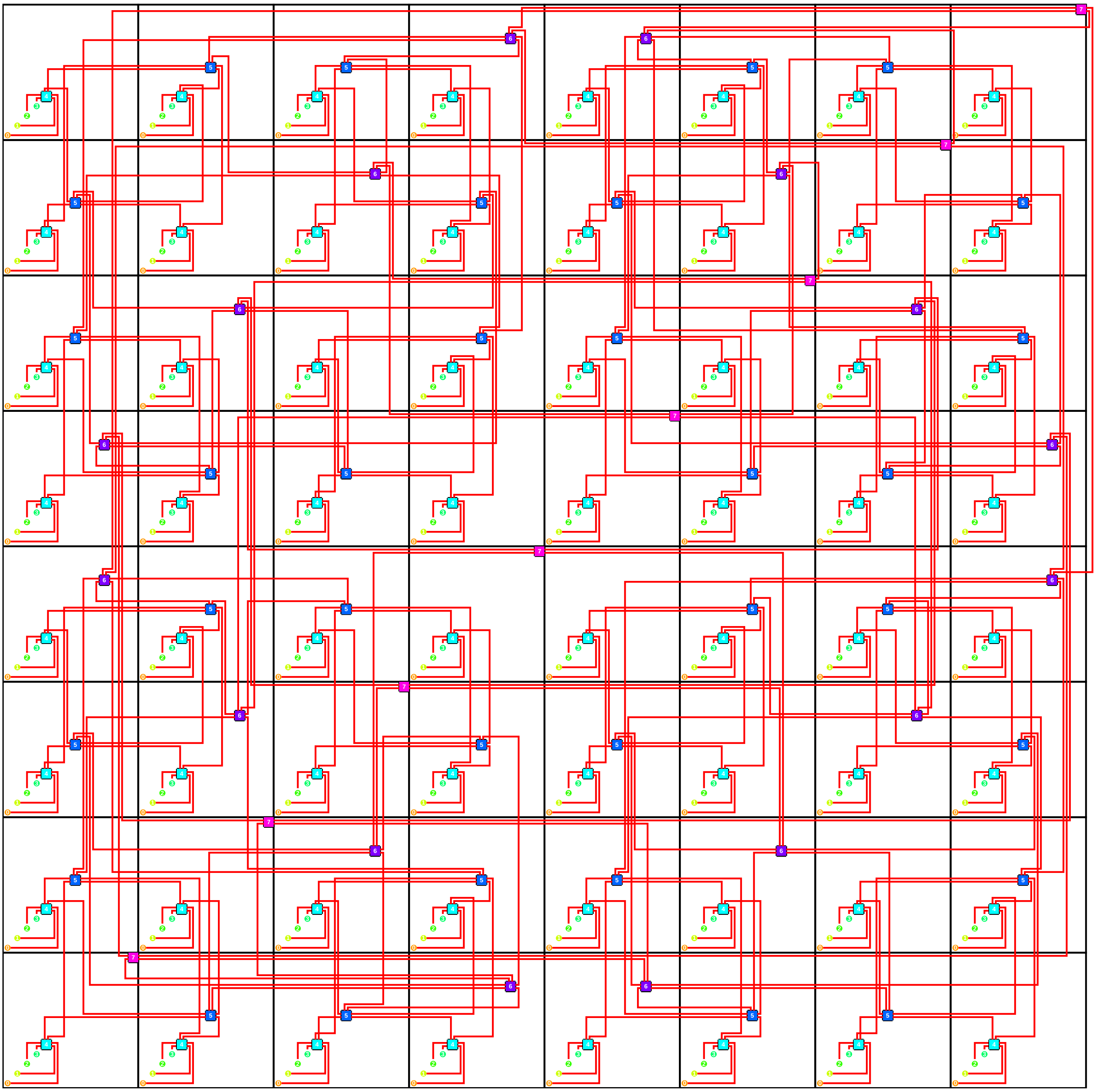

Figure 8: Fold-and-Squash Layout for 256-Node Rearranged Butterfly Fat-Tree