Rent's Rule Based Switching Requirements

[Extended Abstract]

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

ABSTRACT

A Rent's Rule characterization of recursive bisection captures a measure of the locality in a netlist or graph. From this characterization, we know we can establish a lower bound on layout area implied by wiring. When applying this lower bound to the design of programmable or switched networks, we are ultimately concerned for both the switching requirements and the wiring requirements. Switch requirements are particularly important in conventional VLSI where (a) a switchpoint consumes considerably more area than a wire crossing and (b) switchpoints must reside on the active surface, whereas wiring may take place on any of several wire layers. The most straight-forward, limited-bisection switching networks have switching requirements which grow as $O(N^{2p})$, similar to wiring requirements. In practice, however, this leaves switching dominating wiring. We show that there are limited-bisection networks with $O(N)$ switching growth and highlight some of the tradeoffs between wire utilization and switching, routing complexity, routing guarantees, and switch latency within this design space.

Categories and Subject Descriptors

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

Keywords

Rent's Rule, Switching Requirements, Hierarchical Networks

1. INTRODUCTION AND BACKGROUND

1.1 Conventional, Flat Switching Networks

We have a well developed theory and design space for switching networks which can connect a number of sources (N) to a number of sinks (M). In many common cases, the sources and sinks are the same ($N = M$); we will use that simplification for now, but the results have certainly been generalized for the asymmetric case. This switching theory assumes our only limitations are the number of inputs and outputs, making no assumptions or restrictions about the locality of communication.

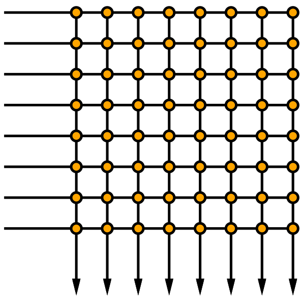


Figure 1: 8×8 Crossbar

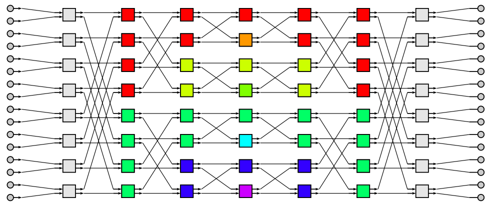


Figure 2: 16×16 Beneš Network

From this switching theory, for example, we know:

- A **crossbar** (Figure 1) can connect any N inputs to N outputs with N^2 switches. The path from any input to any output goes through a single series switch, but the lines connecting input and output are each $O(N)$ long and have $2N$ switches connected to them which add some amount of capacitive loading. There is exactly one switch associated with each possible input to output connection, so routing is trivial and guaranteed as long as the input to output mapping is a permutation (or a subset of a permutation).

- a **Beneš** network (Figure 2) can also route any permutation and can do so with only $O(N \log(N))$ switches [1] [2]. It achieves this reduction in switches at the cost of a multistage arrangement. Any path from input to output goes through $2\log_r(N) - 1$ switches (where r is the radix of each switching stage; each switching stage

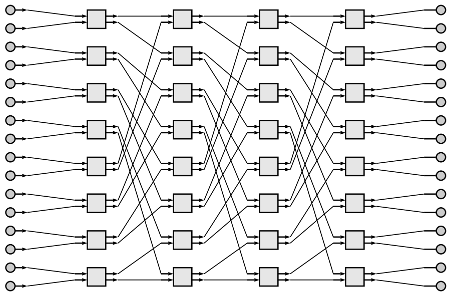


Figure 3: 16×16 Omega Network

may be an $r \times r$ crossbar) on the path from input to output and sees a constant amount of capacitive loading (for constant r) per stage, making total capacitive loading $O(\log(N))$. Total wire length for any route remains $O(N)$. Assuming the permutation is known prior to routing (offline routing), computing the configuration of switches to realize any permutation on a Beneš network is guaranteed and deterministic and can be done in $O(N \log(N))$ time.

- a **banyan** (*e.g.* butterfly, baseline, omega (Figure 3)) network retains the ability to route any single input to output connection, but is not guaranteed to route any permutation. It requires half as many switches as the Beneš network [asymptotically, still $O(N \log(N))$] and half as many switches in any path between input and output ($\log_r(N)$). Total wire length remains $O(N)$ and capacitive loading $O(\log(N))$.

While these networks allow us to reduce switching complexity from N^2 to $O(N \log(N))$, they all still require wiring area that goes as $\Omega(N^2)$ in any two-dimensional layout. This is easy to see using, for example, Thompson's argument about bisection width [12]. The minimum bisection width of any of these networks is $\Omega(N)$, meaning in any layout, there will need to be $\Omega(N)$ wires crossing between the two halves of the layout. When we are limited to 2D-VLSI, this means $\Omega(N)$ wires must cross the 1D-line that bisects the chip. That places a lower bound of $\Omega(N)$ on the width of the chip, if we assume a fixed number of wire layers. Since this property holds recursively, we can make a similar argument for the next cut of the chip and establish that both the width and height of the layout must be $\Omega(N)$, making the entire chip area $\Omega(N^2)$.

This result looks unfortunate, because it says that we can reduce the switches, but, asymptotically, we are stuck with a $\Omega(N^2)$ wiring area. Alternately, it might say that we would need $\Omega(N/\log(N))$ wiring layers to even approach the $O(N \log(N))$ switching area limit.

1.2 Rent's Rule Locality

Fortunately, we have empirical evidence, in the form of Rent's Rule [9], for example, which suggests there is additional structure to typical computations which we can exploit to avoid this limit. That is, the evidence from Rent's Rule suggests that the bisection bandwidth for a design does not need to be $O(N)$, but rather, can be $O(N^p)$ with $p < 1$ for typical designs. This gives us a model for capturing the locality structure of a design—that is, we can look at the bisection growth for a class of designs and characterize their locality by p; this amounts to modeling non-local IO requirements for a well-clustered sub-region as having geometric growth (*e.g.* [3]).

This restricted bandwidth gives us leverage to design a class of networks with lower wire growth and will allow us to reduce the switching requirements. Using an argument like Thompson's in the non-switched case, we see we now need only $\Omega(N^p)$ wires in the bisection, this leads to a smaller 2D-VLSI area lower bound of $\Omega(N^{2p})$.

1.3 Implication on Switching Networks

Once we make this bisection restriction, we can ask what the switching requirements are for these networks. This paper highlights major points in this network design space; owing to space limitations, we cannot comprehensively cover the design space, but we can show a few major alternatives that demonstrate the key tradeoffs involved. We show that the most straightforward layouts also require $O(N^{2p})$ switches. Since switches are larger than wires in conventional VLSI, we explore network organizations which allow us to reduce the asymptotic switching requirements, sometimes at the cost of increased wiring, increased switch latency, or reduced routing guarantees; these tradeoffs are roughly analogous to the tradeoffs for flat networks reviewed above.

This paper focuses on the case where we are trying to understand how to build a universal network which will allow us to route any design with a characteristic (c,p) [from Rent's Rule IO=cN^p]—or, strictly speaking, any design whose recursive bisection is dominated by a geometric growth wiring schedule given by the Rent Relationship and set of (c,p) parameters. A separate problem arises when we want to map a design whose recursive bisection wire capacities exceed the network's limited bisection capacities. In such a case, the design must be placed sparsely on the physical node sites to meet the limited wiring capacity of the network. In earlier work [5], I show one approach to accommodating such mismatches, and use this to understand what Rent growth parameters should be used in designing a network which may see a variety of design characteristics.

2. SWITCHES

The large area of switches relative to wires is one of the key reasons that we care about the number of switches required by a network. If the wire pitch is 5 to 8 λ, the area of a wire crossing is 25–64λ^2. Contrast this with roughly 1200λ^2 for the static memory cell one would use to configure a switch. Once you add a decent sized pass transistor to make up a passive switch, 2500λ^2 is a more typical switchpoint size. This alone makes for a factor of 40 difference in area. This area would show up directly if we were simply building a passive crossbar. In many case, we'll want more than just a pass gate for the switch. We may want to actively rebuffer the switch or even register it. Such switches can easily be 5-10Kλ^2. So, while asymptotic switch counts may match asymptotic wire area, the large constant factor area differences mean that we definitely need to watch the switch count details. (See, for example, Figure 12.)

In standard VLSI, we can use additional metal layers to increase the effective density of wire crossings. For example, with 4 metal layers, and a 7λ wire pitch, we could have two wire crossing in the same 49λ^2 of area. However, switches, in current technology, requires space on the substrate, so

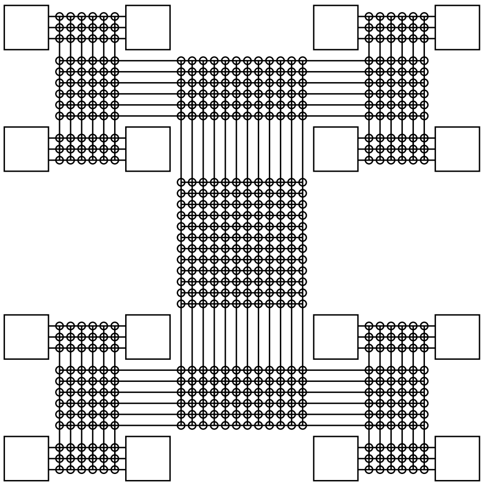


Figure 4: 16 Node Tree of Meshes

we cannot stack two switches in a $2500\lambda^2$, regardless of the number of metal layers. So, while we may increase the wiring layers to increase the wiring density, we get no such increase in switch density. This further suggests the need to keep switch requirements significantly, and perhaps asymptotically, below wire crossings.

3. LIMITED-BISECTION SWITCHING

3.1 Leighton's Tree of Meshes

One realization of these limited-bisection bandwidth networks is Leighton's Tree of Meshes (Figure 4) [10]. Each level of the tree is connected with an $N \times M$ mesh, where the M and N can be chosen to provide the desired growth rate in the tree. For example, if we are trying to achieve a $p = 0.5$ growth rate and the lower channel has M=10, then the upper channel would have $10 \times 2^{0.5} \approx 10 \times 1.4 \approx 14$ wire channels. By keeping the $1 : 2^{0.5}$ ratio on each mesh, we realize a network whose bandwidth grows geometrically according to Rent's Rule with a $p = 0.5$. The mesh allows us to route connections from children subtrees up the network, and vice versa, and allows connections to cross between a pair of sibling subtrees. We will call the mesh a *switchbox* and will be looking at alternate scheme to realize the switching function throughout this section. We denote the number of wires into the base of the switchbox as W (*e.g.* $W = 10$ in this example).

Strictly speaking, this network may need channel capacities which are a constant factor larger than the design it is trying to route. That is, if we construct the network to a (c',p), where $c' = k \times c$, any design dominated by a (c,p) Rent Relationship will be routable. For Leighton's construction $k = 4$ [3].

For $p > 0.5$, a Tree of Meshes network with N nodes will require $O(N^{2p})$ switches. This is easy to see since the top-level mesh will be $N^p \times \left(\frac{N}{2}\right)^p$. If we recursively sum all the switches in the network, we will see that the total number of switches converges to this asymptote. For $p = 0.5$, the number of switches is $O(N \log(N))$, and the number of switches converges to $O(N)$ for $p < 0.5$. By a similar argument, the number of switches in the worst-case path across the network will be $O(N^p)$ for $p > 0.5$. Since this class of networks supports locality, routes only have to go through the root switchbox for the least common ancestor which contains both the source and the sink, making many paths much shorter than the worst case.

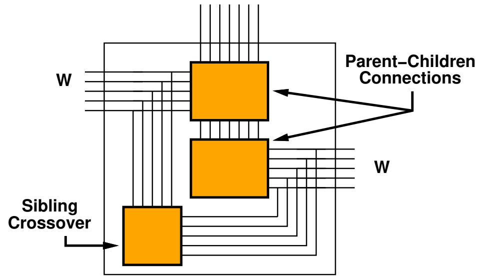


Figure 5: Switchbox Topology for Crossbar, N-choose-M, and Beneš-based Networks: this replaces the mesh switchbox used by the Tree of Meshes.

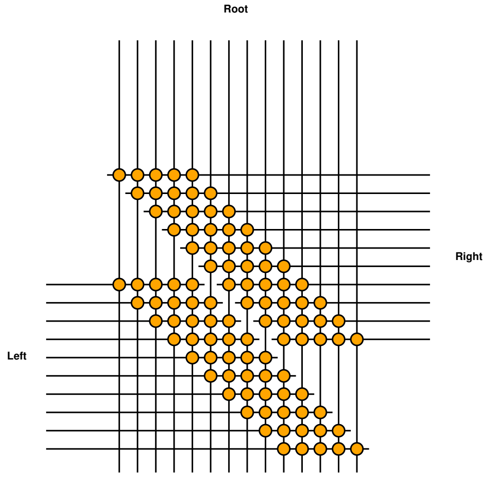


Figure 6: Parent-to-child Connections using 14-choose-10 Depopulated Crossbars: this arrangement allows each child to independently select any subset of 10 signals from the 14 parent signals.

3.2 N-choose-M variation

As an alternative to the Tree of Meshes, we can use depopulated, crossbar-like switchboxes for each of the tree stages in place of Leighton's mesh switchbox (Figure 5). If we fully populate the crossbars so that we have a $W \times (2^p)\,W$ crossbar for each child channel connecting to the parent and a $W \times W$ connection between the two child channels, we get guaranteed routability with no constant overhead—that is, any design whose recursive bisection IO capacities do not dominate the network's channel capacities can be routed. It is also easy to see we have the same asymptotic growth in total switches as the Tree of Meshes. Now, however, we only have $2\log_2(N) - 1$ switches in the worst case path between any source and sink.

A quick look at the $W \times (2^p)\,W$ crossbar switch for each parent to child connection suggests that we have more switching freedom than we strictly need. In particular, the crossbar allows us to connect any of the W lower channel signals to any of the $(2^p)\,W$ upper channel signals, whereas we only

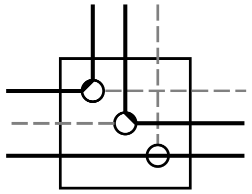


Figure 7: Single Crossover in $p = 0$ Case: the routed (highlighted) connections show an example where the physical network needs to have 1.5× more wires in each channel than the design in order to be fully routable.

really need to make sure that each lower channel signal is connected to some signal in the upper channel. That is, we do not really need a full crossbar; we simply need to select a subset $M = W$ of signals from a set $N = (2^p)\,W$. This can be done with an N-choose-M depopulated crossbar as shown in Figure 6 (for details, see [4]; this result was separately developed by Fujiyoshi et al. [7]). This arrangement needs only $M\times(N-M+1)$ switches. In our case, that means $W \times ((2^p)\,W - W + 1)$. When $p > 0$ and W is large, the number of switches is roughly $(2^p - 1)\,W^2$. This reduces the constant factor required for these switchboxes, which is important in practice, but retains the same asymptotic growth we derived above.

By making a similar observation, the $W\times W$ child crossover can be reduced to $W^2/2$ switches. Here we again note that the exact wire is irrelevant; it is only important that we connect one channel from each child for each crossover connection. Wu and colleagues show that each channel need only fanout to half of the sibling channels in order to guarantee full connections [13]. For large p, if we exploit the crossbar connections between the root and the children when routing siblings (meaning some connections will go through two switches in making a sibling-to-sibling cross connection), we can reduce the number of crossover switches further.

An option to even further reduce crossover switches is to use a single switch for each sibling connection. That is, we number each channel coming in from the two siblings from top to bottom and provide a switch only between the channels numbered in the same manner. This switching scheme, however, can prevent us from being able to use all the wires in the channel. Used in conjunction with the N-choose-M switches above, it means some wires may be used on only one sibling side of the switchbox. Since sibling channels are only connected to a single other channel, it may leave orphaned connections on both sides of the channel. The practical effect is that channels will need to be a constant size larger than the wiring they need to support. This constant is 1.5 for $p = 0$ [14], and decreases with increasing p.

Note that the 3-way switchbox and the optimal 3-way greedy routing switchbox from [13] are degenerate cases of the N-choose-M switching scheme described here for the case where $p = 0$. Figure 7 shows the single crossover 3-way switchbox and demonstrates how this switch reduction may underutilize the physical wires by a constant factor of 1.5.

3.3 Beneš Switching

In the aforementioned networks we pay $O(W^2)$ switches for each of our intermediate switchboxes. We can, however, use our knowledge about tradeoffs in flat networks to reduce the switching cost.

First, we can replace each of the "crossbars" in the switchbox formulation of Figure 5 with Beneš networks. This reduces the total number of switches in the switchboxes to $O(W \log(W))$. Note that the switching networks between child and parent will be asymmetric, but we can easily imagine building a $((2^p)\,W)$-endpoint Beneš network and pruning the unused nodes on the child side of the network. This has an important effect. The total number of switches now converges to only $O(N)$ as shown in Figure 8. Note, however, that the switchbox area will remain $\Omega(W^2)$ in 2D-VLSI due to wiring since we have previously established that all of the flat networks have $O(N)$ bisection bandwidth. Also note that the $\Omega(W^2)$ switchbox wiring does not increase the total wiring lower bound from the $\Omega\left(N^{2p}\right)$ bound which is true for all of these limited-bisection networks.

With the Beneš switching, the problem remains fully routable and can be done so deterministically in $O(N)$ time. Note that we simply need to route each of Beneš switches in the network, which we can do in $O(W \log(W))$ time, so the routing time analysis is the same summation as we just performed to count total switches (*i.e.* Figure 8). The major drawback of this scheme is that routes will now have to cross $O(\log^2(N))$ switches in the worst-case path between a source and a sink.

We can, in theory, use superconcentrators to achieve a linear number of switches in the child to parent path [11]. This leaves us with Beneš switches for the crossover, and the superconcentrators still have $O(\log(W))$ delay in the child to parent connection in every switchbox. This variation does not change the asymptotic bounds we have already achieved.

3.4 Linear Population

We can further reduce the switching requirement by only connecting each channel wire, from siblings or from parents, to a small, constant number of channels in the sibling or parent channel of the switchbox. That is, rather than providing full concentration between the various switchbox directions, we only connect to a small number of channels and use the fact that we have flexibility over multiple switchboxes to get rich routing.

One example of this is Leiserson's Butterfly Fat-Tree [8] (Figure 9); a similar version is our "linear" population tree (Figure 10). We call this "linear" because the number of switches in each switchbox is linear in the channel width, W. Using these topologies, we can program the growth rate similar to Leighton's Tree of Meshes; however, it is most convenient to grow tree stages in quanta. In the scheme used for HSRA, each stage would either be a 1:1 stage or a 2:1 stage. By choosing which levels do (1:1) or do not (2:1) reduce the total wires out of the top of the switchbox, we can target a particular p value over multiple stages of growth (Figure 11).

The number of switches in any of these linearly populated networks converges to a constant independent of the number of tree levels. For example, if we assume a 4-ary butterfly fat tree with switches with 4 down links and 2 up links, as shown in Figure 9, 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

Here we show that the total number of switches in a limited-bisection network using Beneš switchboxes is $O(N)$. There are $cW\log(W)$ switches in each switchbox. Summing over all switchboxes in the tree, we get:

$$\begin{aligned} N_{switch} &= c\Bigg(1\cdot(N^p)\log(N^p) + 2\cdot\left(\frac{N}{2}\right)^p \log\left(\left(\frac{N}{2}\right)^p\right) + 4\cdot\left(\frac{N}{4}\right)^p \log\left(\left(\frac{N}{4}\right)^p\right) + \cdots \\ &\quad + \frac{N}{2}\cdot\left(\frac{N}{N/2}\right)^p \log\left(\left(\frac{N}{N/2}\right)^p\right) + N\cdot\left(\frac{N}{N}\right)^p \log\left(\left(\frac{N}{N}\right)^p\right)\Bigg) \end{aligned}$$

This is better re-expressed using $n = \log(N)$ and extracting the common term p.

$$\begin{aligned} N_{switch} &= cp\Bigg(N\left(\frac{2^p}{2}\right)^n \cdot n + N\left(\frac{2^p}{2}\right)^{n-1} \cdot (n-1) + N\left(\frac{2^p}{2}\right)^{n-2} \cdot (n-2)\cdots \\ &\quad + N\left(\frac{2^p}{2}\right)\cdot 1 + N\cdot 0\Bigg) \end{aligned}$$

Now, if we pull out N and set $r = \frac{2}{2^p}$, we get:

$$N_{switch} = cpN\sum_{n=1}^{\log(N)}\left(\frac{n}{r^n}\right)$$

This is certainly less than the infinite sum where we do not bound n. We further note that the ratio of consecutive terms in this series is:

$$\frac{n+1/r^{n+1}}{n/r^n} = \left(\frac{1}{r}\right)\left(\frac{n+1}{n}\right) = \left(\frac{2^p}{2}\right)\left(\frac{n+1}{n}\right)$$

For $p < 1$ and $n > \frac{2}{2-2^p}$, this ratio is strictly less than one. Hence, this series is bounded by a geometric series and, consequently, the summation is asymptotically bounded by a constant. Since the sum is a constant, we have a total number of switches which is simply a constant times p and N, or $O(N)$ switches. □

Figure 8: Total Switch Accounting when using Beneš Networks in Switchboxes

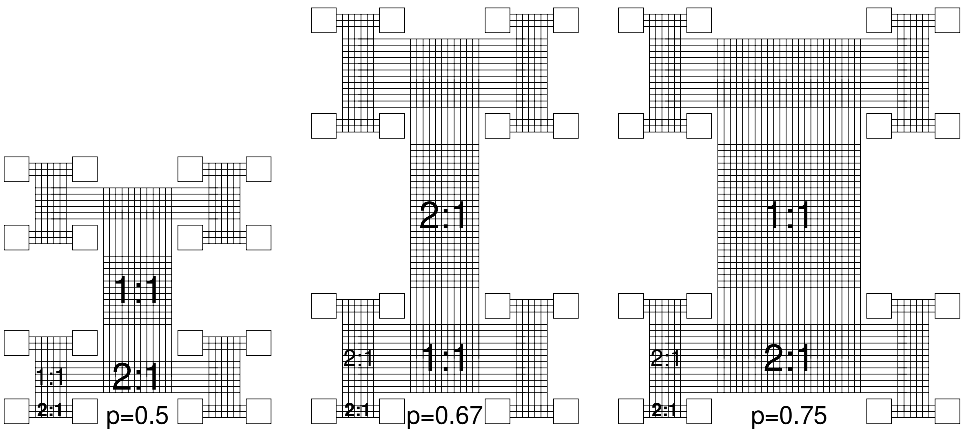


Figure 11: Programming Growth for Tree of Meshes: note that the number of base channels (c) is 3 in all these examples.

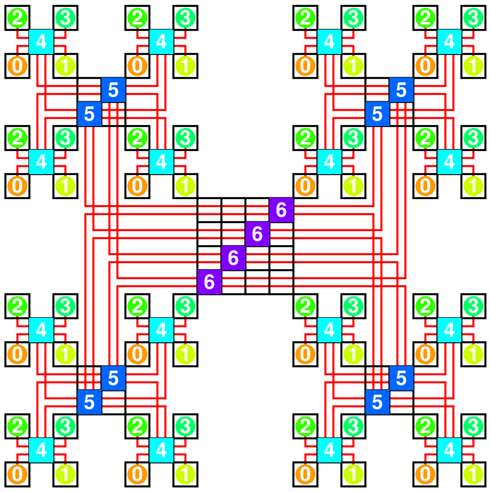


Figure 9: 4-ary Butterfly Fat-Tree

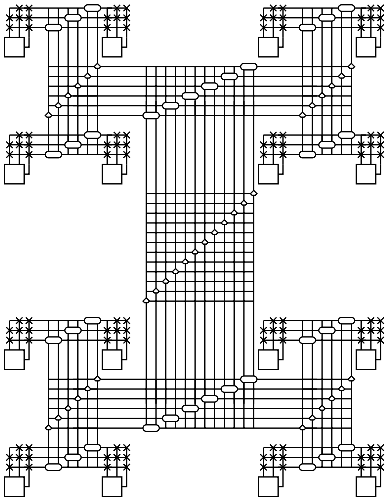


Figure 10: Hierarchical Network with Linear Switch Population

switch at the lowest level (labeled 4 in Figure 9). 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 can be calculated as a classical geometric series:

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

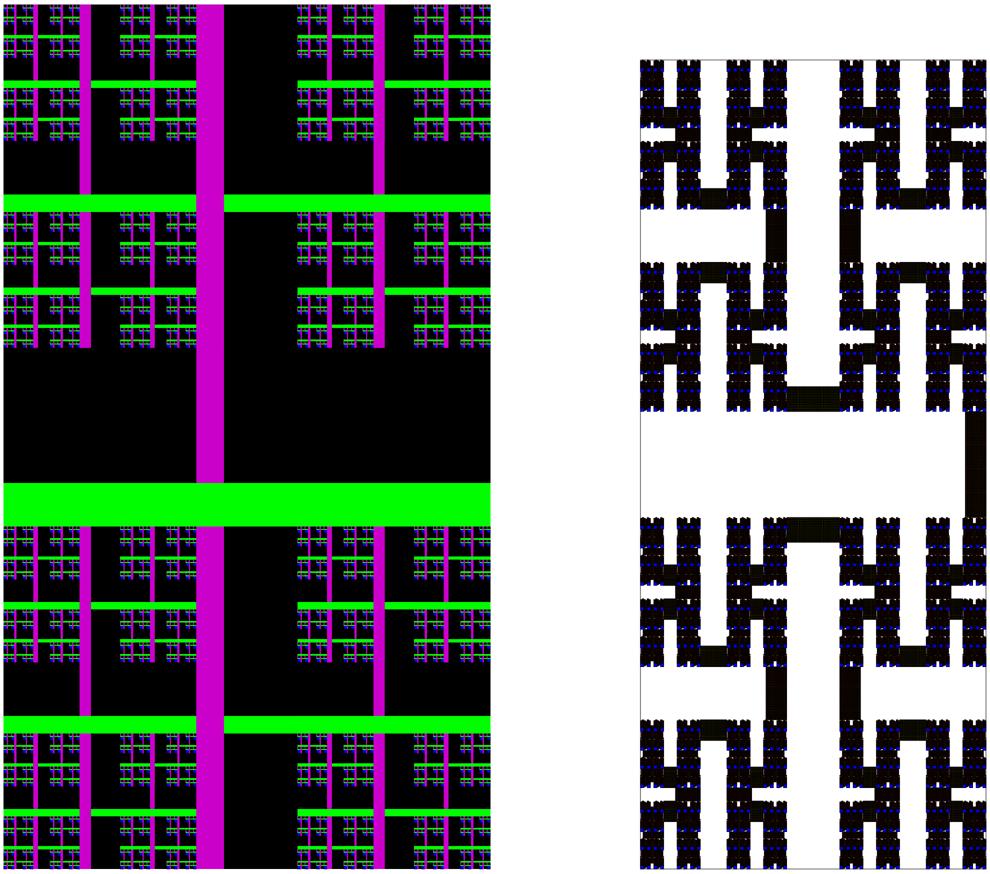
These linear organizations also allow us to have only a single switch at each tree level, such that the total switches along the worst-case path is again only $2\log_r(N) - 1$.

This linear population may come at the cost of routability and wire utilization. It is clear from the 3-way single-switch sibling-crossover case (Figure 7) that we will not be able to use every wire in the network perfectly. However, even if we had to pay a small constant factor in wire utilization (*e.g.* we had to guarantee to have $2\times$ as many available wires in each channel as wires in the netlist), this tradeoff between wires and switches would still be a net win, as shown in Figure 12. We have empirical evidence that a small constant overhead is adequate to route typical benchmark circuits. However, we have not yet proved the existence or non-existence of a constant mapping ratio. Our working hypothesis is that there may not be a constant mapping ratio, but that typical circuits do have additional structure which makes this work. These represent important open questions in our understanding of switching requirements for these networks.

These networks with a linear number of switches are particularly attractive when we consider multi-level wiring. Since the number of switches per node converges to a constant independent of N, it may be possible to layout the networks so that they take up a fixed amount of active substrate area and support the super-linear interconnect requirements with additional metal layers. For the linear populated network, like the Butterfly Fat Tree ($p = 0.5$), I have shown [6] that it is possible to layout the network in $\Theta(N)$ active area using $\Theta(\log(N))$ wiring layers.

4. NETWORK ARITY

So far, I have described and shown most of these limited-bisection networks as two-ary trees—that is, each switchbox has only two children. Another topological option is to vary the arity of the switching nodes. In general, increasing the arity will decrease the number of switchboxes, and hence switches, on the path between the worst-case source and sink pair. For example, going from a 2-ary network to a 4-ary network will cut the number of series switches in half for the linear and N-choose-M populated networks. If we are trying to maintain the same wiring guarantees for a given set of (c, p) parameters, the flattened network will have more wires in some intermediate channels than the non-flattened network since bandwidth reduction only occurs at the switchboxes which are now less frequent. Flattening from 2-ary to slightly larger arity may be able to reduce the total number of switches in the network in some cases, but, asymptotically, increased arity will increase switch count. Flattening by any constant amount will only change the constants.



N-choose-M
$c = 5$
(wires shown)

both
$p = 0.67$

Linear
$c = 10$
(wires not shown)

Figure 12: Area Comparison for a 1024 Node Network: this shows that even if we cannot use all the wires in a linearly populated network, typical constants work out that the total area is still less than that of a network which provides switching so that all wires can be used perfectly. This comparison assumes two metal layers dedicated to routing. The non-black area in the N-choose-M case shows the horizontal and vertical wire layers. It should be clear from these diagrams, that additional metal layers would not substantially reduce the area of the N-choose-M layout since it is already switch dominated; the linear layout is clearly wire-channel dominated and could be reduced if additional metal layers were available for routing.

5. SUMMARY

Limited-bisection networks designed with growth according to Rent's Rule, allow us to reduce switch requirements as well as wire requirements. Since switches are large relative to wires and can only exist on the substrate in VLSI, switch area can easily dominate the wiring area in these networks. We showed a number of design points in this space, with switch areas which range from $O(N^{2p})$, matching wire area asymptotes when we have fixed wire layers, down to $O(N)$ switch area, which may permit $O(N)$ layout area with sufficient metalization.

6. REFERENCES

[1] V. Beneš. Permutation groups, complexes, and rearrangeable multistage connecting networks. *Bell System Technical Journal*, 43:1619–1640, July 1964.

[2] V. Beneš. *Mathematical Theory of Connecting Networks and Telephone Traffic*. Academic Press, New York, NY, 1965.

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

[4] A. DeHon. Entropy, counting, and programmable interconnect. In *Proceedings of the 1996 International Symposium on Field Programmable Gate Arrays*. ACM/SIGDA, February 1996. See extended version <`http://www.cs.caltech.edu/~andre/abstracts/entropy_fpga96.html`>.

[5] A. DeHon. Balancing interconnect and computation in a reconfigurable computing array (or, why you don't really want 100% lut utilization). In *Proceedings of the International Symposium on Field Programmable Gate Arrays*, pages 69–78, February 1999.

[6] A. DeHon. Compact, multilayer layout for butterfly fat-tree. In *Proceedings of the Twelfth ACM Symposium on Parallel Algorithms and Architectures (SPAA'2000)*, pages 206–215. ACM, July 2000.

[7] K. Fujiyoshi, Y. Kajitani, and H. Niitsu. Design of minimum and uniform bipartites for optimum connection blocks of fpga. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 16(11):1377–1383, November 1997.

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

[9] 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.

[10] F. T. Leighton. New lower bound techniques for vlsi. In *Twenty-Second Annual Symposium on the Foundations of Computer Science*. IEEE, 1981.

[11] N. Pippenger. Superconcentrators. *SIAM Journal of Computing*, 6(2):298–304, 1977.

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

[13] Y.-L. Wu, D. Chang, M. Marek-Sadowska, and S. Tsukiyama. Not necessarily more switches more routability. In *Proceedings of the 1996 Asia Pacific Design Automation Conference*, 1996.

[14] Y.-L. Wu, S. Tsukiyama, and M. Marek-Sadowska. Graph based analysis of 2-d fpga routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(1):33–44, January 1996.