Programmable interconnect is the dominant contributor to die area and cycle time in configurable devices. To support their large, active functional density, the computational units must be richly interconnected and support highly parallel data routing. FPGAs, more than other general-purpose devices, place most of their area into interconnect.

We review interconnect issues in the context of on-chip networks for reconfigurable architectures. We establish typical size and delay contributions by analyzing conventional FPGA implementations, then we look at how resource requirements grow with increasing array size. Understanding conventional sizes and growth factors help us characterize the design space. It also serves as background context for the architectural developments described in Part IV.

In this chapter, we:

1. Decompose FPGA area into three component parts and establish the relative areas of each: fixed logic, configuration memory, interconnect resources
2. Review issues in configurable network design
3. Establish growth rates for interconnect and description requirements as a function of network size
4. Establish relationships between network size and richness of network interconnect
5. Examine the efficiency of device utilization when viewed in relation to network resource utilization rather than programmable gate utilization
6. Examine the effects of multibit granularity on interconnect resource requirements

7.1 Dominant Area and Delay

7.1.1 Fixed Area

Reviewing LUT-based FPGA implementations from Table 4.13, and calculating the area per 4-LUT (Table 7.1), we see that each 4-LUT is roughly 600Kλ^2. The flip-flop and 16:1 LUT multiplexor make up very little of this area, easily less than 20Kλ^2. [BFRV92] estimates the area of the 4-LUT multiplexor with flip-flop as 13Kλ^2. In our own DPGA implementation these items occupied 15Kλ^2 (See Chapter 10). The majority of the area associated with each 4-LUT (97%), goes into programmable interconnect and configuration memory

This breakdown, alone, shows us one reason why a full 4-input lookup table is often used as the programmable logic element, rather than a more restricted gate. The area required for the full LUT, including its configuration memory, is less than 10% of the area of the 4-LUT cell, such that there is little advantage to reducing the cell's functional size.

Year	**Design**	λ^2 **area** / 4-LUT
1986	Xilinx 2K [CDF+86]	500K
1988	Xilinx 3K [Xil89, HDJ+88]	650K
1991	UTFPGA [CSA+91]	670K
1992	Xilinx 4K [Xil94b]	630K
1994	LEGO [Seo94]	1020K
1995	DPGA [TEC+95]	660K
1995	Xilinx 5K [Xil91]	560K
1995	Altera 8K [Alt95]	930K
1995	Orca 2C [ATT95]	1060K

Table 7.1: FPGA 4-LUT Size

Part	Approximate Bits/4-LUT
Xilinx xc2k	160
Xilinx xc3k	100
Xilinx xc4k	200
Xilinx xc5k	120
UTFPGA	48
LEGO	120
DPGA	4×40
Altera 8k	190
Orca 2C	120

Table 7.2: Bits per 4-LUT

7.1.2 Interconnect and Configuration Area

The number of programming bits per 4-LUT for these devices is summarized in Table 7.2. Using a rather large memory cell ($\approx$4.5Kλ^2/bit), the memory accounted for 35% of the area on UTFPGA. With 4-contexts and 600λ^2 3T-DRAM memory cells, memory only occupied 33% of the area on the DPGA. If we assume 1000λ^2 static memory cells, for the Xilinx parts, memory accounts for about 15-30% of that area ($\frac{160\text{K}\lambda^2}{500\text{K}\lambda^2}$ (32%), $\frac{100\text{K}\lambda^2}{650\text{K}\lambda^2}$, (15%), $\frac{200\text{K}\lambda^2}{630\text{K}\lambda^2}$, (32%), $\frac{120\text{K}\lambda^2}{560\text{K}\lambda^2}$, (21%)). Making similar assumptions, memory accounts for 21% of an Altera 8K part ($\frac{190\text{K}\lambda^2}{930\text{K}\lambda^2}$) and 11% ($\frac{120\text{K}\lambda^2}{1060\text{K}\lambda^2}$) of an Orca 2C part. Interconnect and routing occupies the balance of the area (70-90%).

7.1.3 Delay

Most vendors lump interconnect timing in with lookup table evaluation, making it difficult to distinguish the components of delay. Table 7.3 summaries interconnect and LUT logic delay for Altera's 8K series [Alt95] and our own experience with the DPGA (Chapter 10). From here, we

Design	Path	Total Delay	LUT delay	Interconnect
Altera 8K	LUT-local-LUT	2.5 ns	2 ns	20%
[Alt95]	LUT-row-local-LUT	7.5 ns	2 ns	73%
	LUT-row-column-local-LUT	10.5 ns	2 ns	81%
MIT DPGA	LUT-LUT (in subarray)	3.5 ns	1.5 ns	60%
[TEC+95]	LUT-xbar-LUT	7 ns	1.5 ns	80%

Table 7.3: FPGA Delay Breakdown

see that interconnect typically accounts for 80% of the path delay.

7.2 Problems with "Simple" Networks

FPGA networks, which already need to interconnect thousands of independent processing elements, do not, typically, look like conventional multiprocessor networks. In particular, a number of conceptually "simple" network structures commonly used as the basis for multiprocessor networks do not scale properly for use in FPGAs. In this section, we review three typical organizations and highlight their shortcomings on the scale required for FPGA networks.

1. crossbars
2. multistage networks
3. mesh networks

This review helps identify and motivate important design issues for reconfigurable interconnect which we will address in the following section.

7.2.1 Crossbars

To guarantee arbitrary, full, connectivity among elements, we could build a a full crossbar for the interconnection network. In such a scheme we would not have to worry about whether or not a given network could be mapped onto the programmable interconnect nor would we have to worry about where logic elements were placed. Unfortunately, the cost for this full interconnect is prohibitively high.

For an n element array where each element is a k-input function (*e.g.* k-LUT), the crossbar would be an $n \times k \cdot n$ crossbar. Arranged in a roughly square array, each input and output must travel $O(\sqrt{n})$ distance, before we account for saturated wire density. Since interconnect delay is proportional to interconnect distance, this implies the interconnect delay grows at least as $O(\sqrt{n})$. However, the bisection bandwidth for any full crossbar is $O(n)$. For sufficiently large n, this bisection bandwidth requires that the side of an array be $O(n)$ to accommodate the wires across the bisection. In turn, the $O(n)$ bisection bandwidth dictates an area $O(n^2)$. This also dictates input and output wires of length $O(n)$. For large crossbars, wire size dominates the areas. These growth rates are not acceptable even at the level of thousands of LUTs. If we were to build devices using a single monolithic crossbar for interconnect:

- area growth would be as the square of the number of LUTs supported
- cycle time would slow down linearly with the number of LUTs in the network

Consider, for the sake of illustration, the size of a crossbar required to interconnect a 2,500 4-LUT device. We will assume the minimum wire pitch is 8λ and the crossbar is implemented with two layers of dense metal routed at this minimum wire-pitch. The area of such an array, as dictated simply by the wiring would be:

$$(8\lambda \times 4 \times 2500) \times (8\lambda \times 2500) = 1.6\text{G}\lambda^2$$

Making for an area of $\frac{1.6\mathrm{G}\lambda^2}{2500} = 640\mathrm{K}\lambda^2$ per 4-LUT just to handle the requisite wiring. Conventional FPGAs use a single SRAM cell to configure each of the crosspoints in the crossbar. If this were done, the area would be memory bit dominated rather than wire dominated and take up:

$$1000\lambda^2 \times (4 \times 2500) \times 2500 = 25\mathrm{G}\lambda^2$$

Which results in $10\mathrm{M}\lambda^2$ per 4-LUT just to hold the configuration memory. The area per LUT, of course, continues to grow linearly in the number of LUTs for larger networks.

7.2.2 Multistage Networks

Multistage interconnection networks (*e.g.* butterfly, omega, CLOS, Benes) can reduce the total number of switches required from $O(n^2)$ to $O(n \log(n))$, but have the same bisection bandwidth problem. Between any two pair of stages in a butterfly network, the total bisection bandwidth is $O(n)$, such that the wiring requirements dictate that area grows at $O(n^2)$.

7.2.3 Mesh Interconnect

At the opposite interconnect extreme, we can use only local connections within the array between adjacent, or close, array elements. By limiting all the connections to fixed distances, the link delay does not grow as the array grows. Further, the bisection bandwidth in a mesh configuration is $O(\sqrt{n})$ and hence, never dominates the logical array element size. However, communicating a piece of data between two points in the array requires switching delay proportional to the distance between the source and the destination. Since switching delay through programmable interconnect is generally much larger than fanout or wire propagation delay along a fixed wire, this makes distant communication slow and expensive. For example, in a topology where direct connections are only made between an array element and its north, east, south, and west neighbors (typically called a NEWS network), a signal must traverse a number of programmable switching elements proportional to the Manhattan distance between the source and the destination ($O(\sqrt{n})$). For the interconnect network topologies typically encountered in logic circuits, this can make interconnect delay quite high – easily dominating the delay through the array element logic.

7.3 Issues in Reconfigurable Network Design

With this background, we can begin to formulate the design requirements for programmable interconnect:

1. **Provide adequate flexible** – The network must be capable of implementing the interconnection topology required by the programmed logic design with acceptable interconnect delays.

2. **Use configuration memory efficiently** – Space required for configuration memory can account for a reasonable fraction of the array real-estate, as we saw in Section 7.2.1. However, as we will see in Section 7.8, configuration encodings can be tight and do not have to take up substantial area relative to that required for wires and switches.

3. **Balance bisection bandwidth** – As discussed above, interconnect wiring takes space and can, in some topologies, dominate the array size. The wiring topology should be chosen to balance interconnect bandwidth with array size and expected design interconnect requirements.

4. **Minimize delays** – The delay through the routing network can easily be the dominant delay in a programmable technology (See Section 7.1.3). Care is required to minimize interconnect delays. Two significant factors of delay are:

   (a) **Propagation and fanout delay** – Interconnect delay on a wire is proportional to distance and capacitive loading (fanout). This makes interconnect delay roughly proportional to distance run, especially when there are regular taps into the signal run. Consequently, small/short signal runs are faster than long signal runs.

   (b) **Switched element delay** – Each programmable switching element in a path (*e.g.* crossbar, multiplexor) adds delay. This delay is generally much larger than the propagation or fanout delay associated with covering the same physical distance. Consequently, one generally wants to minimize the number of switch elements in a path, even if this means using some longer signal runs.

   Switching can be used to reduce fanout on a line by segmenting tracks, and large fanout can be used to reduce switching by making a signal always available in several places. Minimizing the interconnect delay, therefore, always requires technology dependent tradeoffs between the amount of switching and the length of wire runs.

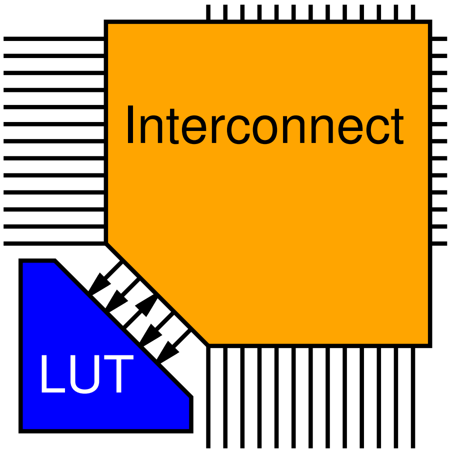


Figure 7.1: Conventional FPGA Interconnect Topology

7.4 Conventional Interconnect

Conventional FPGA interconnect takes a hybrid approach with a mix of short, neighbor connections and longer connections. Figure 7.1 shows a canonical FPGA LUT tile. Full connectivity is not supported even within the interconnect of a single tile. Typically, the interconnect block includes:

- A hierarchy of line lengths – some interconnect lines span a single cell, some a small number of cells, and some an entire row or column
- Limited, but not complete, opportunity for corner turns
- Limited opportunity to link together shorter segments for longer routes
- Options for the value generated by the LUT to connect to some lines of each hierarchical length in each direction – perhaps including some local interconnect lines dedicated to the local LUT output
- Opportunity to select the k-LUT inputs from most of the lines converging in the interconnect block

The amount of interconnect in each of the two dimensions is not necessarily the same. Figure 7.2 shows these features in a caricature of conventional FPGA interconnect.

The University of Toronto has performed a number of empirical interconnect studies aimed at establishing basic FPGA interconnect characteristics, including:

- How densely to populate the interconnect with switches and the number of routing tracks required to route representative circuits [RB91]
- The merits of hierarchical interconnect [AL94]

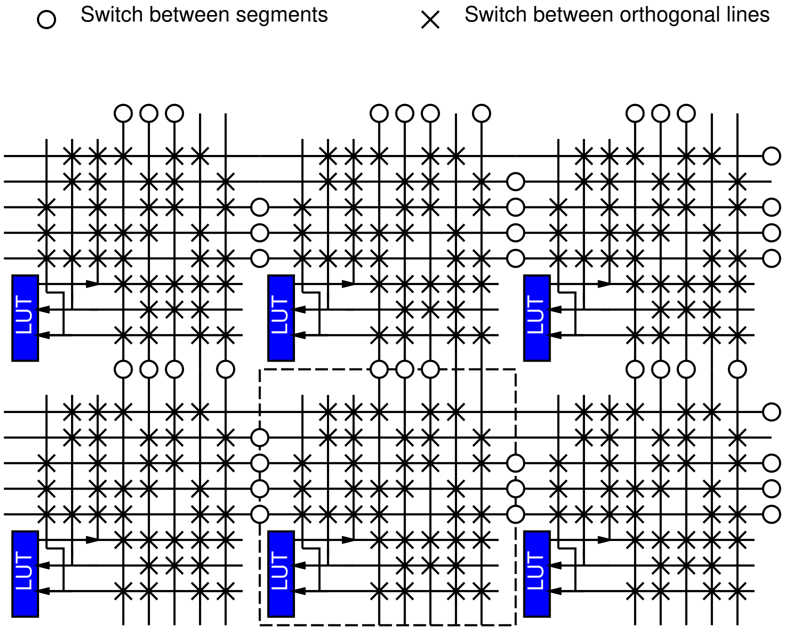


Figure 7.2: FPGA Interconnect Caricature

- The distribution of line lengths [Seo94]

One of the key differences between FPGAs and traditional "multiprocessor" networks is that FPGA interconnect paths are locked down serving a single function. The FPGA must be able to simultaneously route all source-sink connections using unique resources to realize the connectivity required by the FPGA. Another key difference is that the interconnection pattern is known *a prior* to execution, so offline partitioning and placement can be used to exploit locality and thereby reduce the interconnect requirements.

7.5 Switch Requirements for FPGAs with 100-1000 LUTs

Before we examine how network requirements scale with connectivity and network size, in this section, we briefly review the number of switches conventionally employed by networks supporting 100 to 1000 4-LUTs. Brown and Rose [RB91, BFRV92] suggest each 4-LUT in a moderate sized FPGA with 100's of 4-LUTs will require 200-400 switches. Agarwal and Lewis suggest approximately 100 switches per LUT for hierarchical FPGAs [AL94] with some reduction in logic utilization. Conventional, commercial FPGAs do little or no encoding on their interconnect bit streams – that is, each interconnect switch is controlled by a single configuration bit. From the configuration bit summary in Table 7.2, we see that commercial devices also exhibit on the order of 200 switches per 4-LUT. The fact that conventional FPGAs can, with difficulty, route most all designs using less than 80-90% of the device LUTs, suggests that they chose a number of switches which provides reasonably "adequate" interconnect for the current device sizes – hundreds to a couple of thousand 4-LUTs.

7.6 Channel and Wire Growth

In Sections 7.1 and 7.5, we have empirically established the size of conventional interconnect. However, as we glimpsed in Section 7.2, the area which these resources occupy is not necessarily independent of the number of LUTs interconnected. In this section we look at how interconnect requirements will grow with the number of LUTs supported.

The best characterization to date which empirically meters interconnect requirements is Rent's Rule [LR71, Vil82]:

$$N_{io} = C N_{gates}^{p} \tag{7.1}$$

N_{io} is the number number of interconnection in/out of a region containing N_{gates}. C and p are empirical constants. For logic functions $0.5 < p < 0.7$, typically.

El Gamal used a stochastic model to estimate the interconnection requirements for channeled gate arrays [Gam81]. He found that each routing channel requires $O(\overline{R})$ tracks if the average wire length, $\overline{R}$, grows faster than $O\left(\log(n)\right)$. n here is the total number of circuits in the array, generally arranged in an $\sqrt{n} \times \sqrt{n}$ array. Brown used El Gamal's routing model for FPGAs and found good correspondence between it and FPGA interconnect requirements [Bro92]. For large numbers of gates (N_{gates}) and $p > 0.5$, Donath finds that $\overline{R} \propto N_{gates}^{p-0.5}$ [Don79]. Together this means the channel width grows as $O(N_{gates}^{p-0.5})$. From which we can derive the interconnect requirements growth:

$$\begin{aligned}
N_{channels} &\propto \sqrt{N_{gates}} \\
N_{interconnect_width} &= N_{channels} \times N_{channel_width} \\
N_{interconnect_area} &= N_{interconnect_width}^{2} \\
N_{interconnect_area} &\propto \left(\sqrt{N_{gates}} \times N_{gates}^{p-0.5}\right)^{2} \\
N_{interconnect_area} &\propto N_{gates}^{2p}
\end{aligned} \tag{7.2}$$

$p = \frac{2}{3}$ is often considered a good, conservative, value for p to handle most interconnect requirements.

For $p < 0.5$, Donath finds that $\overline{R}$ grows as $O(\log(N_{gates}))$ or smaller. For $\overline{R} \leq O(\ln(N_{gates}))$, El Gamal's model suggests the the track width grows as $O(\ln(N_{gates}))$. In this case, total interconnect requirements grow as $O(N_{gates} \log^{2}(N_{gates}))$.

7.6.1 Rent's Rule Based Hierarchical Interconnect Model

To make this size estimate more concrete, let us consider a specific structure built according to Rent's Rule. We build a fully hierarchical interconnect with inter-level signaling bandwidth growing according to Rent's Rule. To simplify analysis, we consider only unidirectional signal wires.

The gates are recursively partitioned into n equally sized sets at each level of the hierarchy. The principal interconnect occurs at each node of convergence in the hierarchy (See Figure 7.3). At a level l in the hierarchy, each node has a fan-in from below of $n \times n_{out_{l-1}}$ signals and a fan-in from above of n_{in_l}. Similarly, it has a fan-out of $n \times n_{in_{l-1}}$ toward the leaves and n_{out_l} towards

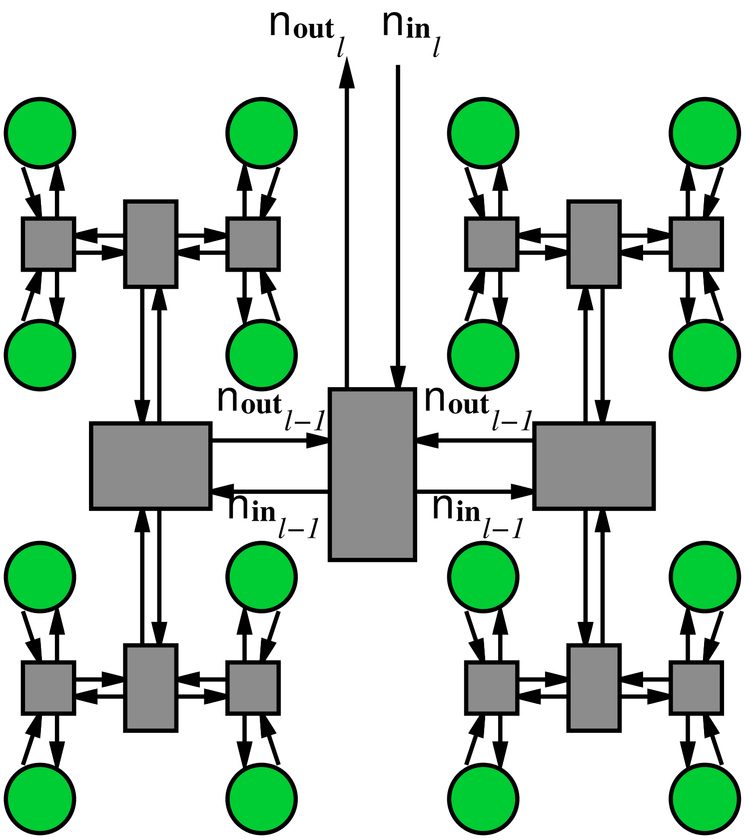


Figure 7.3: Logical Structure of Hierchical Interconnect

the root. At each level l, we have n_{LUT_l} LUTs, n_{in_l} external inputs, and n_{out_l} external outputs. According to the hierarchical combining and Rent's Rule growth, we have:

$$\begin{aligned} N_{LUT_l} &= n^l \\ n_{in_l} &= C\left((n)^l\right)^p \\ n_{out_l} &= C\left((n)^l\right)^p \end{aligned} \tag{7.3}$$

We take $C = K$, the number of LUT inputs. When $\left(C\left((n)^l\right)^p\right) > n^l$, which will be true for small l, we take $n_{out_l} = n^l$ – that is, all outputs are passed out of the region when this Rent bandwidth permits.

Logically, we have $n + 1$ distinct output directions from each node of convergence in the interconnect – n for the n leaves, plus one for the root. Allowing full connectivity within each tree node, each of the n leaves picks its n_{in} inputs from the $(n-1) \times n_{out}$ outputs from its siblings and from the n_{in} inputs from the parent node. The n_{out} outputs of this node are selected from

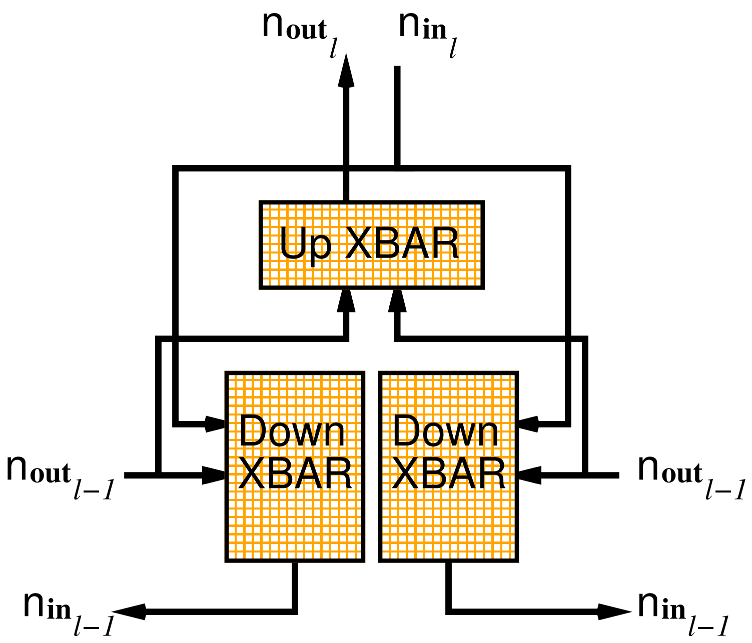


Figure 7.4: Switching node in 2-ary Hierarchical Interconnect

the $n \times n_{out}$ outputs from all n subtrees converging at this point. Figure 7.4 shows this basic arrangement for $n = 2$.

7.6.2 Wire Growth in Rent Hierarchy Model

First, let us consider how wiring resources grow in this structure. At each stage of the hierarchy, there are $n_{in_l} = n_{out_l} = C\left((n)^l\right)^p$ wires coming and leaving each subarray. This makes the bisection width of $O\left(\left(n^l\right)^p\right) = O\left((N_{LUT_l})^p\right)$. For a two-dimensional network layout, this bisection width must cross out of the subarray through the perimeter. Thus the perimeter of each subarray is $O\left((N_{LUT_l})^p\right)$. The area of the subarrays will be proportional to the square of its perimeter, making:

$$A_{subarray} \propto \left(N_{LUT}^p\right)^2$$

The area required for each LUT based on wiring constraints, then, goes as:

$$A_{LUT} = \frac{A_{subarray}}{N_{LUT}} \propto \frac{\left(N_{LUT}^p\right)^2}{N_{LUT}}$$

$$A_{LUT} \propto \left(N_{LUT}^{(2p-1)}\right) \qquad (7.4)$$

Not unsurprisingly, this matches the interconnect growth we derived in Equation 7.2. Of course, if $p < 0.5$, wiring is not the dominant resource constraining LUT area. A_{LUT} may be $O(1)$ for $p \leq 0.5$ as far as strict wiring requirements are concerned.

7.6.3 Switch Growth in Rent Hierarchy Model

We can also look at the number of switches required if each of the logical switching units is a fully-populated crossbar. At each level, l, the total number of switches is:

$$
\begin{aligned}
n_{SW-total_l} &= \left(n \times \underbrace{\left(\overbrace{\left((n-1)\cdot n_{out_{l-1}} + n_{in_l}\right)}^{\text{inputs to down xbar}} \times n_{in_{l-1}} \right)}_{\text{each down xbar}} \right) + \underbrace{\left(n_{out_l} \times \left(n \cdot n_{out_{l-1}}\right)\right)}_{\text{up xbar}} \\
&= \left(n \times \left(\left((n-1)\cdot K n^{p(l-1)} + K n^{pl} \right) \times K n^{p(l-1)} \right) \right) + K n^{pl} \times \left(n \cdot K n^{p(l-1)} \right) \\
&= \left(n \times \left(\left((n-1+n^p)\cdot K n^{p(l-1)} \right) \times K n^{p(l-1)} \right) \right) + K n^{pl} \times \left(n \cdot K n^{p(l-1)} \right) \\
&= \left(n \cdot (n-1+n^p) \cdot K^2 n^{2p(l-1)} \right) + \left(n^p \cdot n \cdot K^2 n^{2p(l-1)} \right) \\
&= \left(n \cdot K^2 n^{2p(l-1)} \right) \cdot \left((n-1+n^p) + n^p \right) \\
&= \left(n \cdot K^2 n^{2p(l-1)} \right) \cdot \left(n-1+2n^p \right) \qquad (7.5)
\end{aligned}
$$

Amortizing across the number of LUTs supported at level l, we can count the number of switches per LUT at each level:

$$
\begin{aligned}
n_{SW_l} &= \frac{\left(n \cdot K^2 n^{2p(l-1)} \right) \cdot (n-1+2n^p)}{n^l} \\
&= \left(K^2 n^{(2pl-2p-l+1)} \right) \cdot (n-1+2n^p) \\
&= \left(K^2 n^{(2p-1)(l-1)} \right) \cdot (n-1+2n^p) \qquad (7.6)
\end{aligned}
$$

Summing across all levels, we can thus calculate the number of switches per LUT as a function of the size of the network.

$$
n_{SW} = \sum_{l=1}^{\log_n(N_{LUT})} \left(K^2 n^{(2p-1)(l-1)} \right) \cdot (n-1+2n^p)
$$

Substituting N_{LUT} for n^l, and expanding sum:

$$
n_{SW} = \left(K^2 N_{LUT}^{(2p-1)} \right) \cdot \left(\frac{1}{n^{(2p-1)}} + \left(\frac{1}{n^{(2p-1)}} \right)^2 + \cdots + \left(\frac{1}{n^{(2p-1)}} \right)^l \right) \cdot (n-1+2n^p) \qquad (7.7)
$$

For $p > 0.5$, this gives us:

$$
n_{SW} = \left(K^2 N_{LUT}^{(2p-1)} \right) \cdot \left(\frac{\frac{1}{n^{(2p-1)}} - \left(\frac{1}{n^{(2p-1)}} \right)^{\left(\log_n(N_{LUT})+1\right)}}{\left(1 - \frac{1}{n^{(2p-1)}}\right)} \right) \cdot (n-1+2n^p)
$$

$$
\begin{aligned}
&< \left(K^2 N_{LUT}^{(2p-1)}\right)\cdot\left(\frac{\frac{1}{n^{(2p-1)}}}{(1-\frac{1}{n^{(2p-1)}})}\right)\cdot(n-1+2n^p)\\
&< \left(N_{LUT}^{(2p-1)}\right)\cdot K^2\cdot\left(\frac{1}{n^{(2p-1)}-1}\right)\cdot(n-1+2n^p) \qquad (7.8)
\end{aligned}
$$

For $p = 0.5$, each sum term in Equation 7.7 goes to one:

$$
\begin{aligned}
n_{SW} &= \left(N_{LUT}^{0}\log_n(N_{LUT})\right)\cdot K^2\cdot(n-1+2n^p)\\
n_{SW} &= \log_n(N_{LUT})\cdot K^2\cdot(n-1+2n^p) \qquad (7.9)
\end{aligned}
$$

For $p < 0.5$,

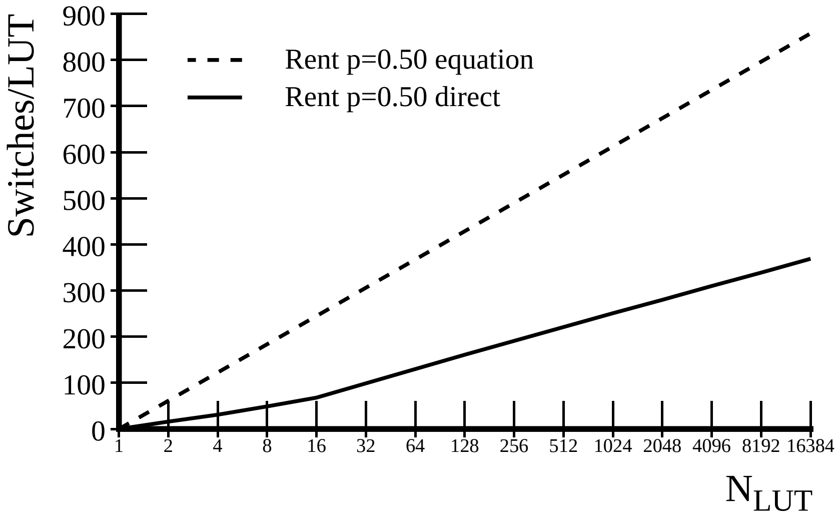
$$
\begin{aligned}
n_{SW} &= \left(K^2 N_{LUT}^{(2p-1)}\right)\cdot(n-1+2n^p)\left(\left(n^{(1-2p)}\right)^{\log_n(N_{LUT})}+\cdots+\left(n^{(1-2p)}\right)^2+\left(n^{(1-2p)}\right)\right)\\
n_{SW} &= \left(K^2 N_{LUT}^{(2p-1)}\right)\cdot(n-1+2n^p)\cdot\left(\left(\left(n^{(1-2p)}\right)^{\log_n(N_{LUT})}\right)\cdot\left(\frac{1-\left(\frac{1}{n^{(1-2p)}}\right)^{\log_n(N_{LUT})}}{1-\frac{1}{n^{(1-2p)}}}\right)\right)\\
n_{SW} &< \left(K^2 N_{LUT}^{(2p-1)}\right)\cdot(n-1+2n^p)\cdot\left(\left(n^{(1-2p)}\right)^{\log_n(N_{LUT})}\right)\cdot\left(\frac{1}{1-\frac{1}{n^{(1-2p)}}}\right)\\
n_{SW} &< \left(K^2 N_{LUT}^{(2p-1)}\right)\cdot(n-1+2n^p)\cdot\left(n^{\left((1-2p)\log_n(N_{LUT})\right)}\right)\cdot\left(\frac{n^{(1-2p)}}{n^{(1-2p)}-1}\right)\\
n_{SW} &< \left(K^2 N_{LUT}^{(2p-1)}\right)\cdot(n-1+2n^p)\cdot\left(n^{\log_n\left(N_{LUT}^{(1-2p)}\right)}\right)\cdot\left(\frac{n^{(1-2p)}}{n^{(1-2p)}-1}\right)\\
n_{SW} &< \left(K^2 N_{LUT}^{(2p-1)}\right)\cdot(n-1+2n^p)\cdot\left(N_{LUT}^{(1-2p)}\right)\cdot\left(\frac{n^{(1-2p)}}{n^{(1-2p)}-1}\right)\\
n_{SW} &< \left(N_{LUT}^{(2p-1)}\right)\cdot\left(N_{LUT}^{(1-2p)}\right)\cdot K^2\cdot(n-1+2n^p)\cdot\left(\frac{n^{(1-2p)}}{n^{(1-2p)}-1}\right)\\
n_{SW} &< K^2\cdot(n-1+2n^p)\cdot\left(\frac{n^{(1-2p)}}{n^{(1-2p)}-1}\right) \qquad (7.10)
\end{aligned}
$$

Putting these cases, together:

$$
n_{SW} \le \begin{cases} K^2\cdot(n-1+2n^p)\cdot\left(\frac{n^{(1-2p)}}{n^{(1-2p)}-1}\right) & p < 0.5\\ \log_n(N_{LUT})\cdot K^2\cdot(n-1+2\sqrt{n}) & p = 0.5\\ \left(N_{LUT}^{(2p-1)}\right)\cdot K^2\cdot\left(\frac{1}{n^{(2p-1)}-1}\right)\cdot(n-1+2n^p) & p > 0.5 \end{cases} \qquad (7.11)
$$

Here, we see switching area per LUT grows as $O(1)$, for $p < 0.5$, and $O\left(N_{LUT}^{(2p-1)}\right)$ for $p > 0.5$. Again, this matches our wiring growth expectations (Equation 7.2).

While Equation 7.11 gives the correct growth rates it overestimates the required number of switches on two accounts:



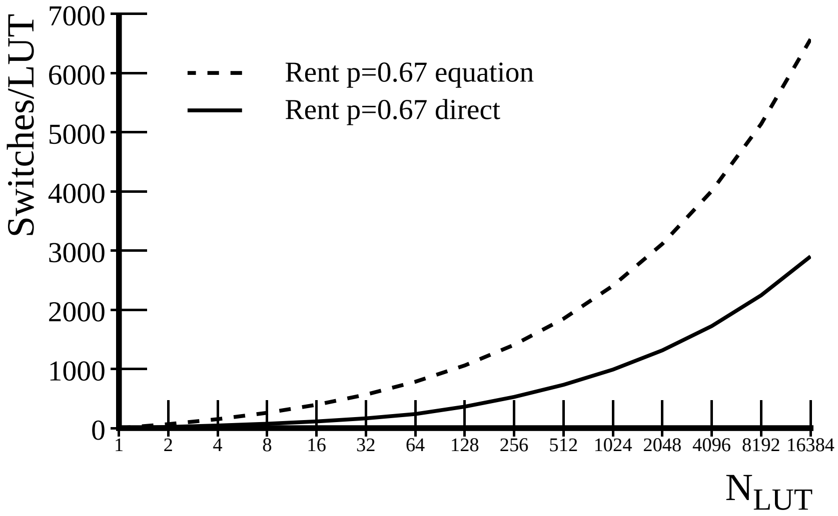
$k = 4$, $n = 2$, $p = 0.5$

Figure 7.5: Switches per LUT – Equation versus Direct Calculation

1. It does not take into account the limited number of distinct outputs at the lowest stages of the network – *i.e.* when there are less outputs than the Rent i/o suggests.

2. It approximates each crossbar as requiring $n \times m$ switches. However, since each crossbar is performing an n choose m operation, only $n \times (m - 1)$ crosspoints are actually required to provide full connectivity at each tree interconnect node.

Figures 7.5 and 7.6 show the difference between Equation 7.11 and a direct calculations which includes the above two effects. Asymptotically, the difference is in the constant factor. Note that for $p = 0.5$, the number of switches per LUT computed by the direct calculation in the 256–1024 LUT range is 190-250, which is on par with contemporary interconnects (Section 7.5).



$k = 4$, $n = 2$, $p = 0.67$

Figure 7.6: Switches per LUT – Equation versus Direct Calculation

7.7 Network Utilization Efficiency

In the previous section, we saw that the amount of interconnect we need to provide depends upon the connectivity of the network. This makes it difficult to design a single network which will *efficiently* accommodate arbitrary designs. If the design has limited connectivity, but the network provides a large amount of connectivity, the network is over designed relative to the design and provides less functional density than achievable. If the design has considerable connectivity, but the network provides less, the design must be routed sparsely on the interconnect, leaving many of the device LUTs unusable.

Using the switching models derived in the previous section, we can examine the relative inefficiencies of using a design with Rent exponent p_{design} on a network with Rent exponent p_{net}. We do this by looking at the ratio of the area occupied by a design with N_{LUT} LUTs and on top of a network built using Rent exponent p_{net}. If $p_{net} > p_{design}$, then the ratio is simply the ratio of the area per LUT of a p_{net} interconnect of N_{LUT} LUTs to the area per LUT of a p_{design} interconnect of N_{LUT} LUTs. However, if $p_{design} > p_{net}$, we cannot simply map the design netlist on top of the device LUTs. Here, we have to figure out how much larger the network must be than the number of LUTs in the design in order to accommodate the highly connected design. Let us call this scaling factor C. In order for the network to accommodate the design, it must have enough i/o bandwidth into each subregion. Starting at the top level in the design, this means:

$$n_{io_{design}} \leq n_{io_{net}}$$

The only way to accommodate this requirement with a fixed p_{net} is to scale up the network used. Applying Rent's Rule (Equation 7.1), this means:

$$KN_{LUT}^{p_{design}} \leq K(C \cdot N_{LUT})^{p_{net}}$$

Solving this relation for equality:

$$\begin{aligned} N_{LUT}^{p_{design}} &= (C \cdot N_{LUT})^{p_{net}} \\ N_{LUT}^{p_{design}/p_{net}} &= C \cdot N_{LUT} \\ C &= N_{LUT}^{\left(p_{design}/p_{net}-1\right)} \end{aligned} \quad (7.12)$$

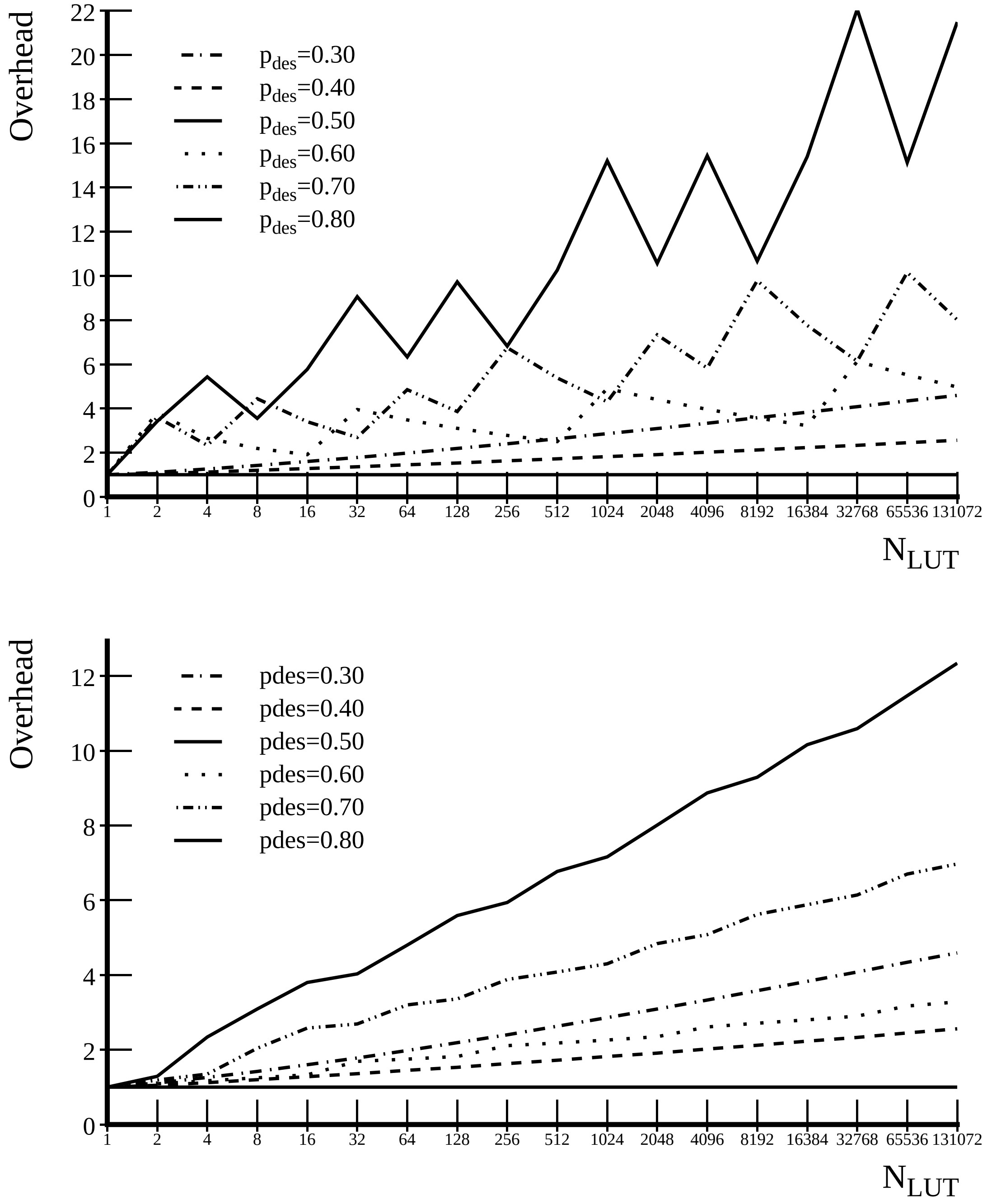
Note that once we accommodate the top level of the design, all other levels are also accommodated as well. That is, once we have chosen C as above, at the top level:

$$Kn^{lp_{design}} \leq K(Cn)^{lp_{net}} \quad (7.13)$$

Since $p_{net} < p_{design}$, at level $l-1$, the connectivity required for the design will shrink faster than the network connectivity, so lower levels are satisfied by the same scale up factor which satisfies the top level in the design. The overhead ratio for the $p_{design} > p_{net}$ case, then, is the ratio of the size of a $C \cdot N_{LUT}$ interconnect with Rent exponent p_{net} compared to the size of an N_{LUT} interconnect with Rent exponent p_{design}.

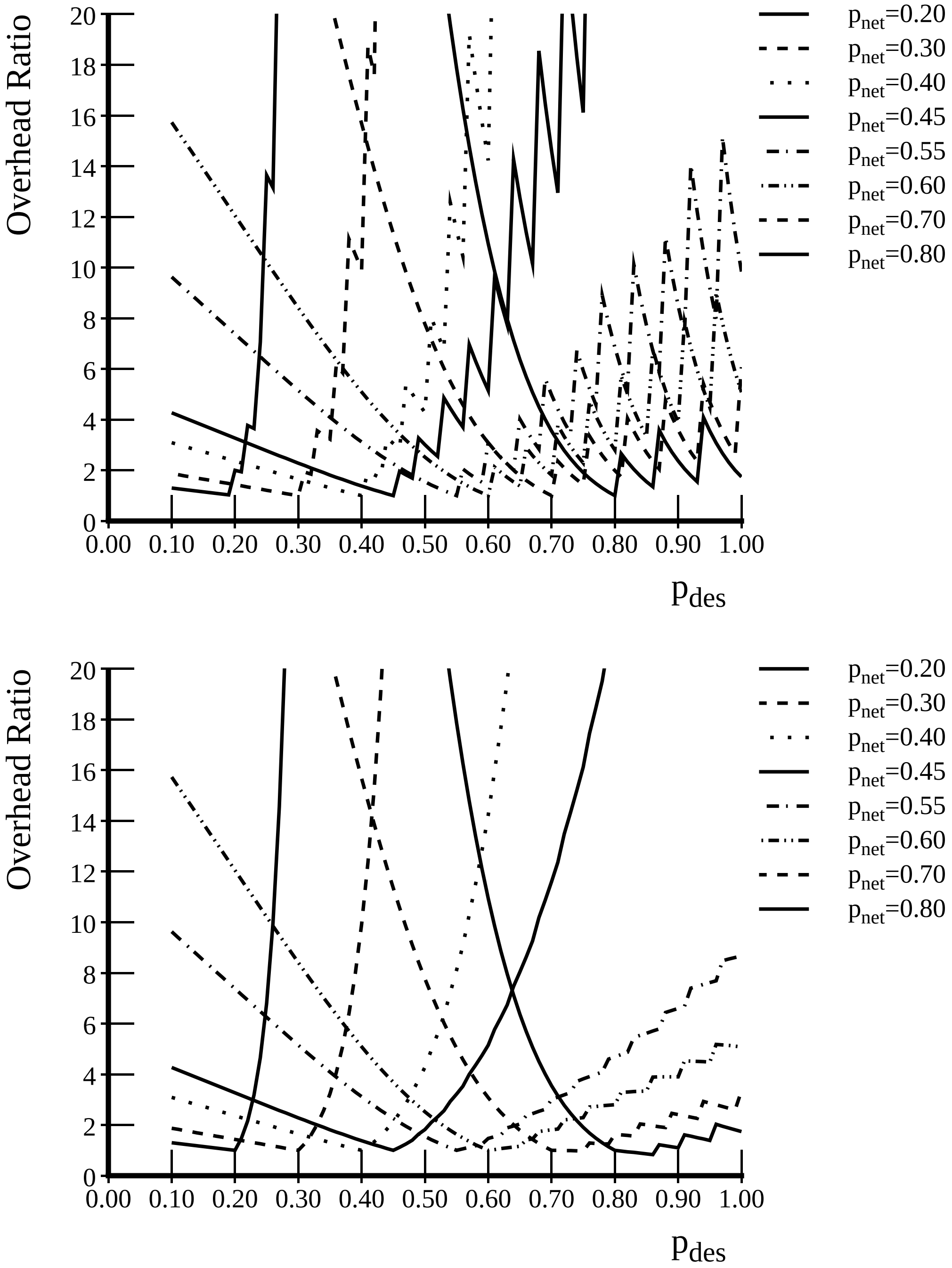
In making this area comparison, we assume that switching area dominates non-switching area, and we approximate LUT area as proportional to the number of switches. From Section 7.1, we saw that this is true of conventional devices. In the previous section, we saw that switching requirements grow at least as fast as wires, and generally faster than non-switching resources. This suggests that switching area will continue to dominate non-switching area as device capacities grow.

If we solve for C strictly according to Equation 7.13, the ratios are continuous and do not take into account the discretization affects associated with network size and levels. The continuous approximation gives us a smooth way to compare general overhead growth trends. Figure 7.7 shows both the discrete and continuous comparisons for various $p_{designs}$ implemented on a network with $p_{net} = 0.5$ as a function of N_{LUT}. Figure 7.8 similarly shows the relative overheads for implementing p_{des} designs with $N_{LUT} = 4096$ on p_{net} designs. Figure 7.9 plots the same data as the continuous case from Figure 7.8 on three axes. Figure 7.10 plots the continuous efficiency, the inverse of overhead, and Figure 7.11 plots the continuous efficiency on a logarithmic scale.



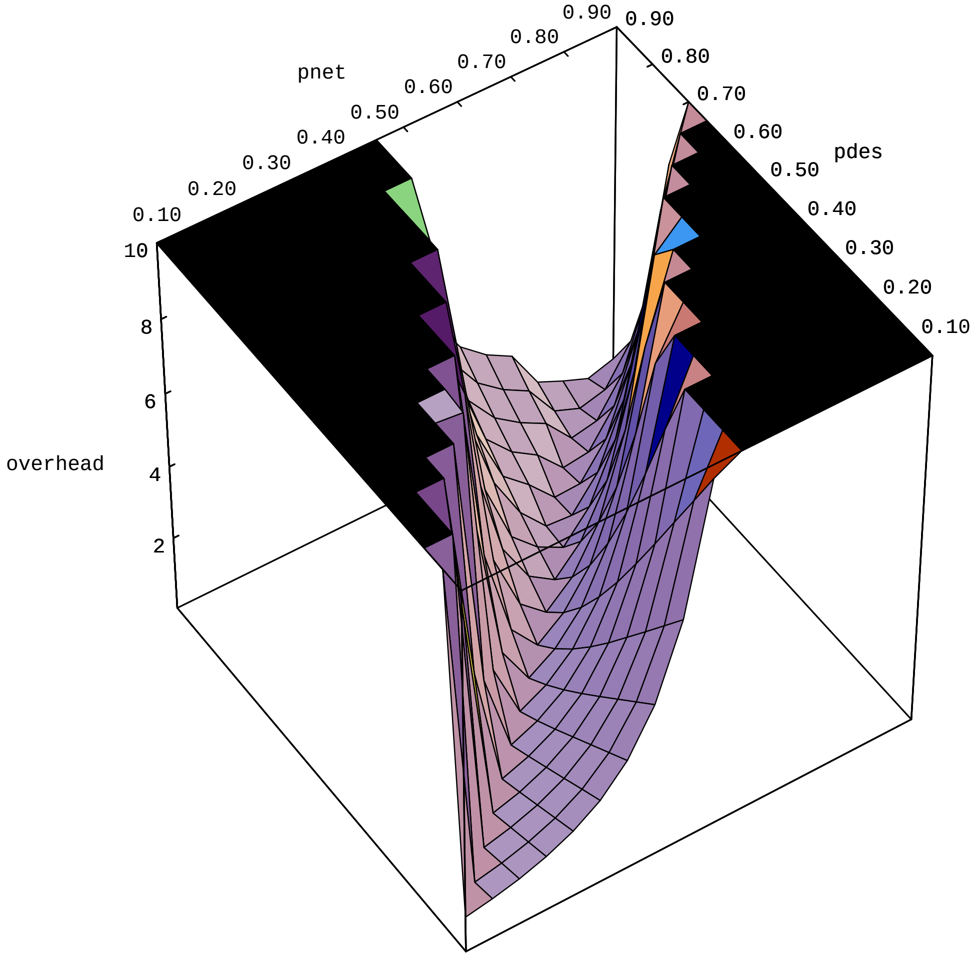
$k = 4$, $n = 2$, $p_{net} = 0.5$
Top - discretized ratios; Bottom - continuous ratios

Figure 7.7: Overhead Growth versus N_{LUT} for various p_{net}



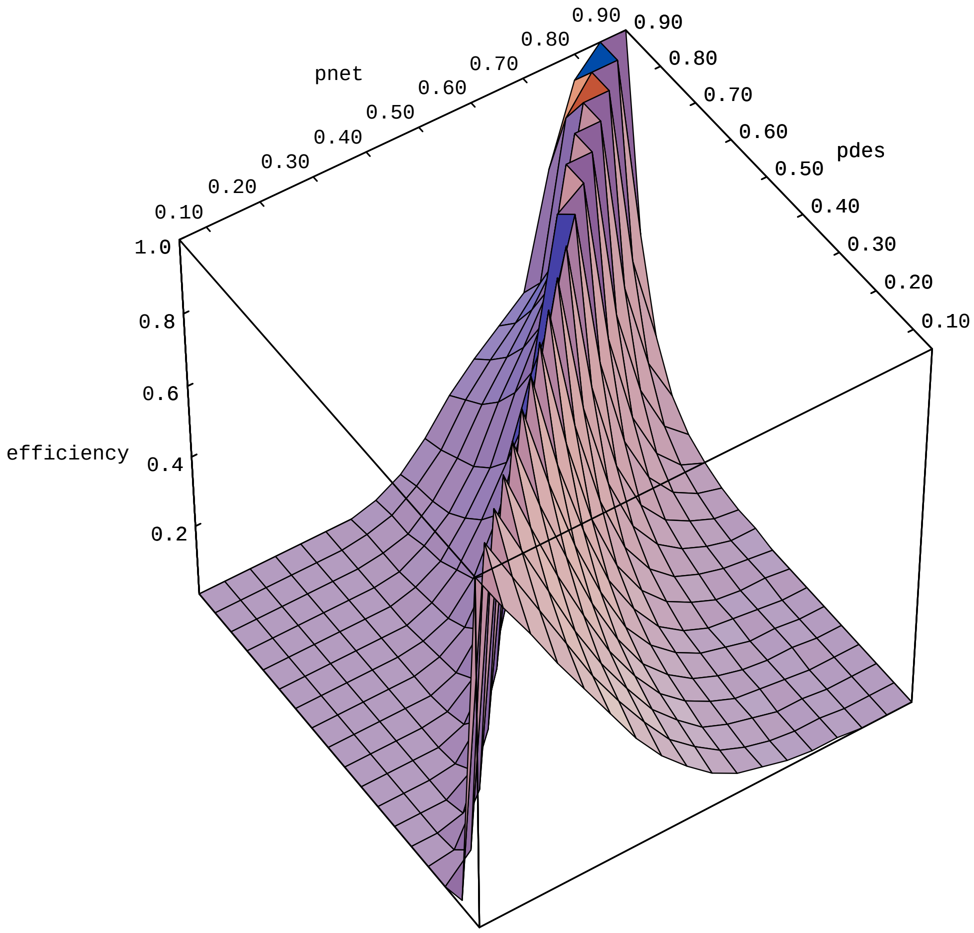
$k = 4$, $n = 2$, $N_{LUT} = 4096$
Top - discretized ratios; Bottom - continuous ratios

Figure 7.8: Overhead for p_{des} versus p_{net}



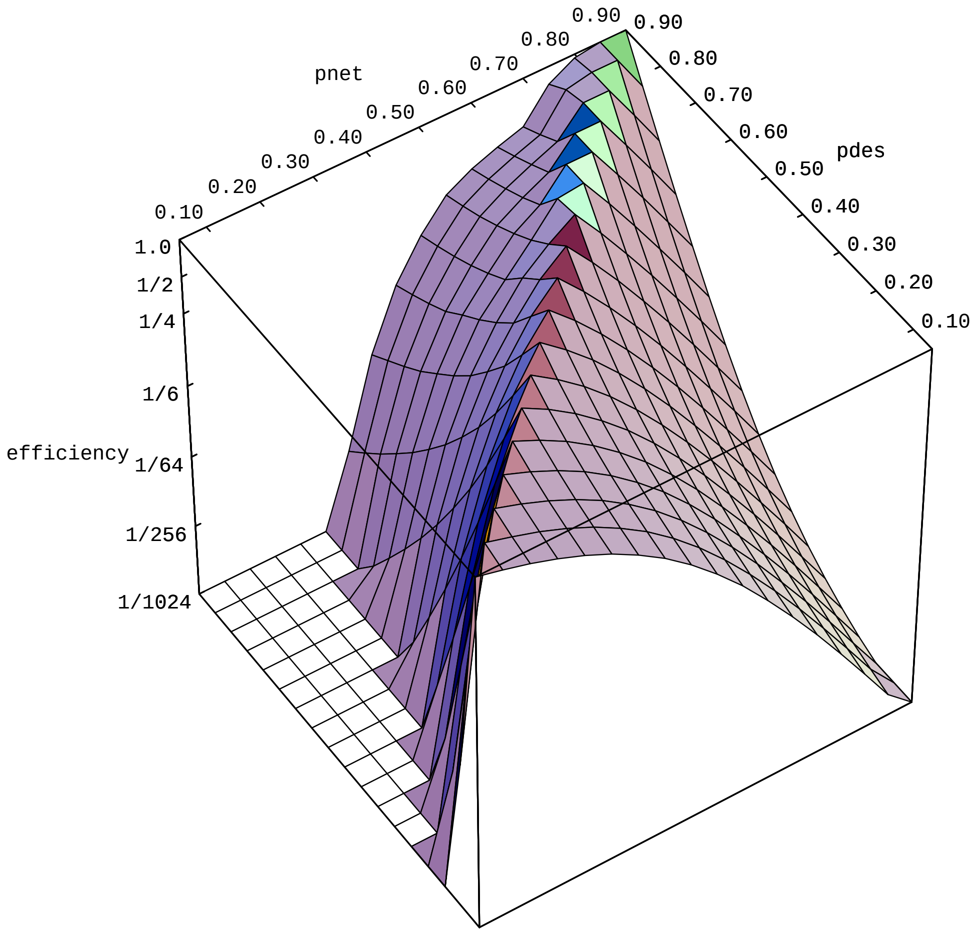
$k = 4$, $n = 2$, $N_{LUT} = 4096$

Figure 7.9: Continuous Overhead for p_{des} versus p_{net}



$k = 4$, $n = 2$, $N_{LUT} = 4096$

Figure 7.10: Continuous Efficiency for p_{des} versus p_{net}



$k = 4$, $n = 2$, $N_{LUT} = 4096$

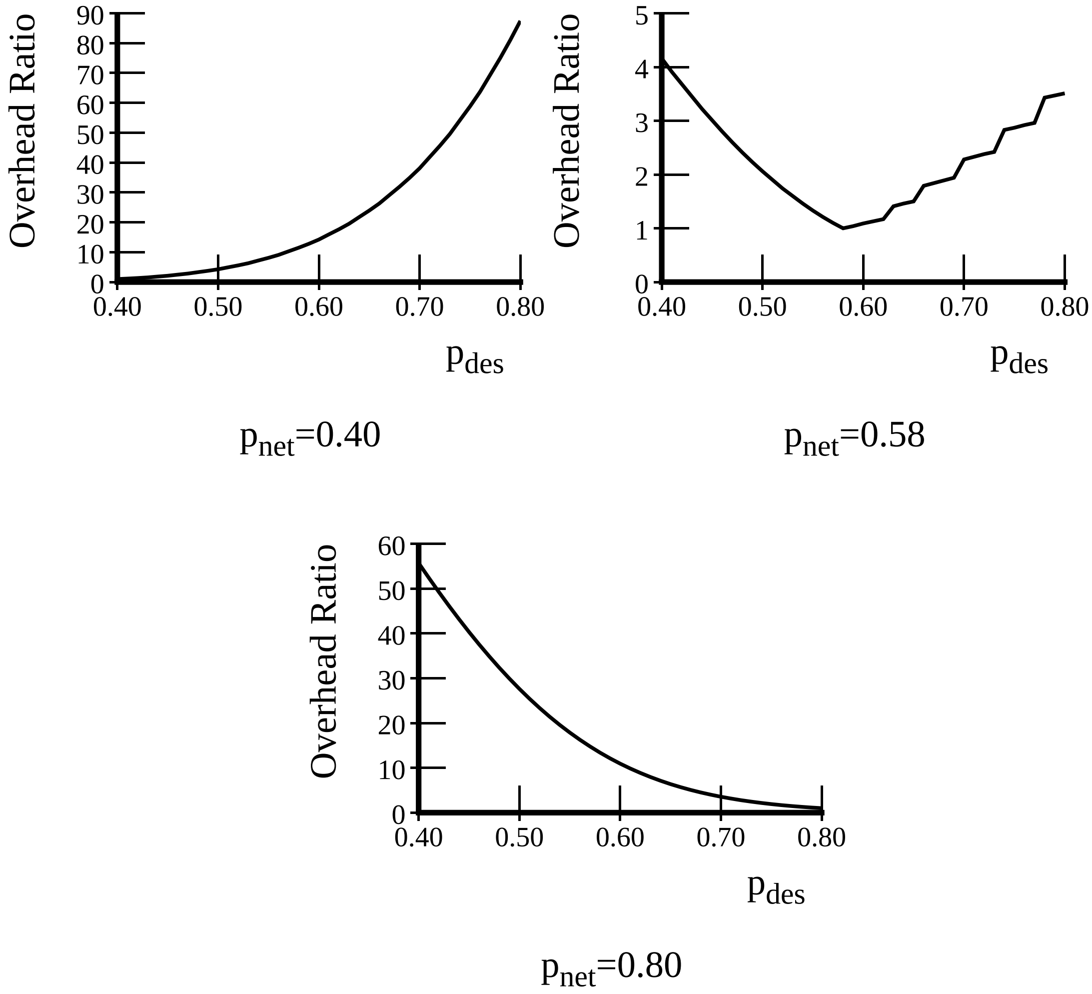
Figure 7.11: Continuous Efficiency for p_{des} versus p_{net} (Log Scale)

Ideally, we would like to match the programmable network connectivity to the design connectivity. Unfortunately, we do not generally get that choice. Figures 7.7 through 7.11 show us that it is just as inefficient to provide too much interconnect for a design as it is to provide too little. *This is important to notice, since there is a tendency to demand rich interconnect that provides high gate utilization across all designs. However, since the non-interconnect area is trivial compared to network area in FPGA devices, optimizing for gate utilization is often short sighted.*

As a final, illustrative example, let us consider the task of picking the network connectivity, p_{net}, assuming that we know typical designs will have a p_{design} between 0.4 and 0.8. Figure 7.12 shows the overheads for p_{net} values of 0.4, 0.58, and 0.8 as a function of p_{design}, respectively. If we further assume that the design Rent exponents are evenly distributed in this range, we can calculate an expected overhead:

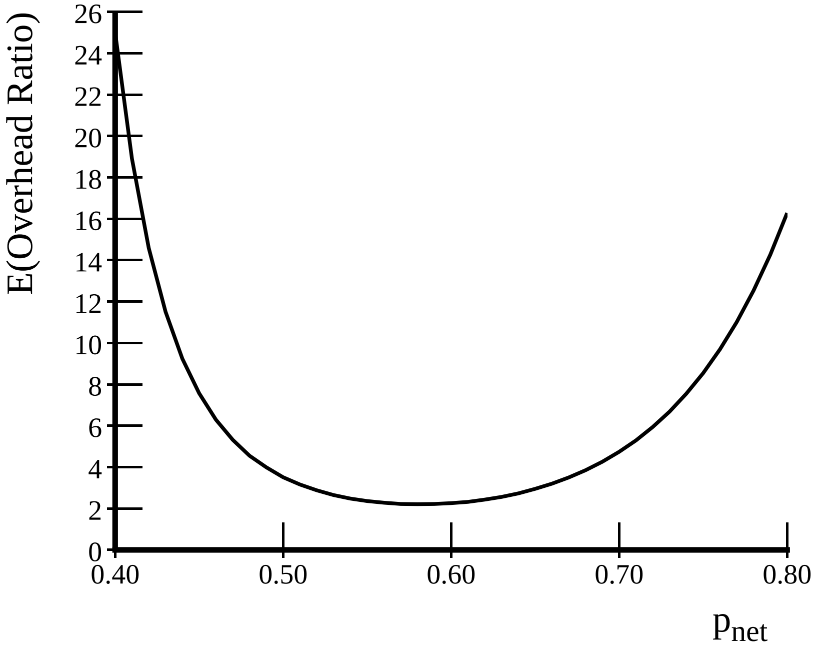
$$E\left(\text{overhead}(p_{net})\right) = \int_{0.4}^{0.8} \text{overhead}\left(p_{design}, p_{net}\right) dp_{design}$$

Figure 7.13 plots this expected overhead for the identified range. We see that the expected overhead is quite flat between p_{net}=0.5 and 0.6 with an expected overhead of just over 2$\times$. At the ends of the spectrum, the expected overhead is 8$\times$ worse. Note, in particular, if we chose to build $p_{net} = 0.8$ in order to guarantee full utilization of every LUT, we would pay a 16$\times$ overhead on average, and a 56$\times$ overhead in the worst case. In contrast, choosing $p_{net} = 0.58$ has a worst-case overhead of 4.2$\times$ and an average overhead of 2.2$\times$.



$k = 4$, $n = 2$, $N_{LUT} = 4096$

Figure 7.12: Sample p_{des} versus p_{net} Overheads



$k = 4$, $n = 2$, $N_{LUT} = 4096$

Figure 7.13: E(overhead) versus p_{net} for Uniform p_{des} Distribution