Chapter 2

Background and Previous Work

2.1 Terms and Definitions.

A graph $G = (V, E)$ has n nodes (vertices) and m edges, unless otherwise specified.

A *boolean network* G is a directed graph whose nodes, also called *gates*, are labeled as primitive boolean functions: typically $\wedge$ (and), $\vee$ (or) and $\neg$ (not). Edges are also referred to as *wires*. A boolean network is *combinational* if it is acyclic. A *sequential* network is traditionally defined as a circuit with memory. We will assume that memory is implemented by atomic *flip-flop* nodes in the representation of the circuit as a graph, rather than built from gates. All sequential circuits discussed in this thesis will be single-clock synchronous networks, unless stated otherwise, which means that all directed cycles must be "broken" by one or more flip-flops. We will ignore the issue of pipelining, whereby flip-flops are added for timing reasons but logically function as buffers, so we assume that all sequential circuits have back edges[1]. When referring to the graphical representation of a practical boolean network, we will use the term *circuit graph* or *circuit*. The term *graph* will refer to an arbitrary graph which may or may not arise from a boolean network. A *random graph* is one drawn from some natural distribution by a stochastic process. For example, a random graph $G(n, p)$ is a graph on n nodes such that each potential edge exists with independent probability p.

In a circuit graph G, nodes with no incoming edges are called *primary inputs* and nodes with no outgoing edges *primary outputs*. The *fanin* (*fanout*) of a node is the number of

[1] A back edge is a "feedback" edge which goes from one sequential level to a previous level. Sequential levels are formally defined later.

incoming (outgoing) edges. The *depth* of a circuit is the longest input to output directed path. In a combinational circuit, this distance is the *unit combinational delay*, delay(G), of the circuit. The length of a shortest directed path from an input to a particular node x defines the unit combinational delay for x, delay(x). In a sequential circuit, the combinational delay of a node is the length of the shortest directed path from either an input *or* a flip-flop, and the combinational delay of the circuit is the maximum combinational delay over all nodes.

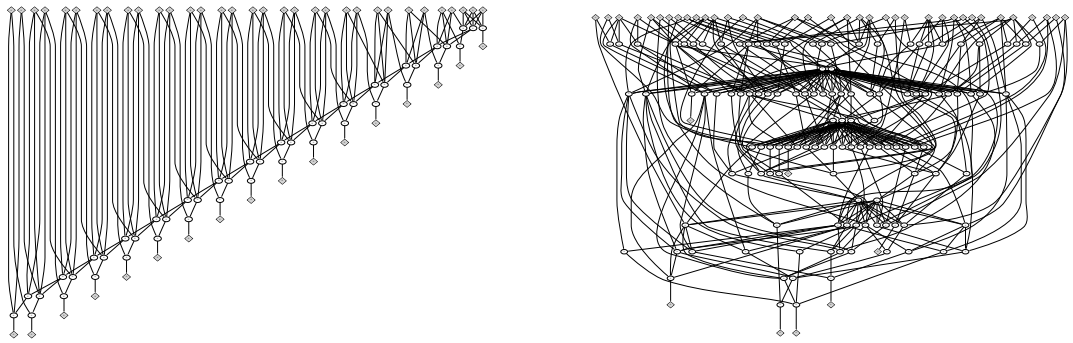
A circuit is often modeled as a *hypergraph*, $H = (V_H, E_H)$, particularly for the partitioning problem. $V_H = V_G$ and each node and its set of fanouts collectively form a hyperedge in E_H, usually called a *net*. Electrically, this is the more correct model of a circuit, but most problems are more easily defined in terms of graphs.

The *recursive fan-in* (*fan-out*) of a node, also called a *cone*, is the set of all preceding (following) nodes in the partial order underlying G (undefined for sequential networks). When two disjoint directed uv paths exist in G, we say that G is a *reconvergent* network and that G is "reconvergent at v." In a non-reconvergent combinational network every fanout-cone is a tree. The increasing presence of reconvergence is known to introduce difficulty into many CAD problems, as the input graphs become less and less "tree-like."

Circuits are often classified into two distinct types. *Datapath* circuits are repetitive, simple sequences where each node is often connected only to immediate physical neighbours. Arithmetic functions such as an adder or multiplier are typical datapath circuits. *Random logic* or *control* circuits are loosely defined as everything else. They typically lack the regularity of datapath circuits. Since the structure of a datapath circuit is usually well-known to the designer, and the type of functions computed are typically more generic (rather then application specific), they are often treated as special cases for layout. It is relatively easy to synthesize a datapath using commercial CAD tools, so we will be primarily interested in circuits in the random-logic category. Figure 2.1 shows examples of datapath and random logic, taken from the MCNC benchmark collection.

We will occasionally refer to qualitative size of circuits (small, medium, large). Current generation FPGAs have one to four thousand 4-input lookup tables (or LUTs)[2], and next

[2]A k-input lookup table is a logic element which can be programmed to implement *any* single-valued boolean function on k inputs. Though an FPGA could have a more restrictive type of logic block, or have different logic functions available throughout the architecture, the industry standard is to use the k-input LUT uniformly across the chip.



Datapath logic
MCNC circuit **my_adder**

Random logic
MCNC circuit **C432**

Figure 2.1: Datapath vs. random logic.

generation devices will have double that. So we will use "small" to refer to circuits with 500 LUTs or fewer, "medium" for 500 to 5,000 LUTs, "large" for up to 10,000 LUTs, and very large for beyond 10,000 LUTs. Gate-array technology typically quantifies logic in terms of standard 2-input gates. Industry typically translates one LUT/flip-flop pair as comprising about 12 such gates. State of the art gate arrays are currently in the 1,000,000 gate range, or about ten times the capacity of current FPGAs.

2.1.1 Computer-Aided Design for Digital Circuits.

It is important to have a common view of what is implied by a particular computer-aided design software problem. We give enough detail here to be self-contained, and refer the interested reader to Lengauer's comprehensive book [50] for more detail.

Technology-independent optimization refers to the manipulation of a network to achieve some common basic requirements for all technologies (such as constraining fan-in/out [32, 38]) or to effect a result deemed to be of value for any destination technology; e.g. isolating and merging common boolean expressions to reduce the size of the network.

Partitioning refers to separating the nodes of a graph into two or more disjoint sets or *modules* to minimize some graph-theoretic measure, usually the number of edges or hyperedges which cross the partition boundaries, subject to such constraints as the minimum or maximum module size. An equivalent notion to the number of inter-module edges is the number of vertices in each module that have external connections, often referred to as the number of logical *pins* in the module. Standard formulations of the problem are NP-hard. Various heuristic algorithms exist. One popular approach is the Kernighan-Lin-Fiduccia-

Mattheyses (KLFM) [46, 26] algorithm, which performs incremental improvements (swaps) from an initial solution until some predetermined tolerance is reached. In practice, such algorithms can perform reasonably well, despite theoretical proofs of pathological non-optimality.

Though primitive boolean functions ($\wedge$, $\vee$, $\neg$) are the basic blocks of the abstract description of a circuit, hardware implementation typically draws from a larger library of available basic functions, often determined by physical design concerns. *Technology mapping* is the process of converting from a circuit whose nodes are basic blocks of one (e.g. the generic) type into one whose basic blocks are of another (the technology specific) type. For field-programmable gate arrays the basic block is usually a k-input lookup table (LUT), in which case the problem of finding a *size* or *depth* optimal mapping is somewhat different from the subgraph matching problem of typical library based mapping. Existing software to compute such a mapping includes flowmap [13], chortle [27], rmap [60], xnfmap [73] and mis/sis-pga [54]. For a general reference, see the textbook by Brown *et. al.* [11].

Placement is the embedding of a graph G into the physical, geometric, world. This is often abstracted as a mapping of the nodes of G into the nodes of the $N \times N$ grid-graph $G_{N,N}$ (the "host") to minimize some approximation of channel width (see below) or a total wirelength metric (e.g. sum of Manhattan distances between adjacent nodes in G). In this thesis we will use both this model, which closely resembles a number of Xilinx FPGAs, and hierarchical variations such as occur in the Altera 10K programmable device. Once placement has taken place, we can define the *length* of a particular edge and the total *wire-length* R , *average wire-length* $\bar{R}$, and distribution of wire lengths $\mathcal{R} = \{R_l\}$, with respect to the placement. *Wireability*, which refers to the types and distributions of wire-lengths which can be supported by a given host (e.g. $G_{N,N}$) independent of any particular circuit, will be discussed in more detail in Section 2.2.2[3]. By the term *routability* we refer to the "ease" of successfully placing and routing a specific network G into a host, using these and other related metrics.

Given the placement, a *global routing* is an assignment of the edges of G to paths in $G_{N,N}$. Then we have the notion of *channel width*, W, defined as the maximum over all

[3]Note that the term "wireability" refers mostly to the process of determining statistical relationships on the connectivity and distribution of wires once circuits are already placed on a grid-like architecture. It is not usually used in the sense of a quality judgement on a network. In general, we don't use the terms wireable and unwireable in that sense, rather we use routable and unroutable.

edges in $G_{N,N}$ of the number of paths using that grid edge. The *optimal channel width* over all placements is denoted W^*. A *detailed routing* assigns the paths of the global routing to realizable electrical connections with respect to the technology. In the case of mask-programmable technology this means physical wires which can interact (cross) other wires in only specific ways. Field-programmable gate arrays have preexisting wires laid out in *tracks* in each channel, each track is broken up into *segments* (actual wires) connected by *programmable connections* with which to select a given path within a track or between tracks. A detailed routing is then a refinement of the global routing which specifies the settings of the programmable switches to code a physical path in the segments of the coarse routing (channels only). It is possible to consider global and detailed routing together as a single problem. For some FPGA architectures the concept of detailed routing makes less sense, and this approach is taken.

Since known deterministic algorithms for NP-hard problems are considered infeasible, existing practical algorithmic solutions often have no provable performance (correctness or quality). For evaluation of competing techniques the community uses various "standard" benchmark suites. By running a new algorithm on the benchmark circuits a quantitative measure (run-time, channel width, percent of routable connections, speed of the circuit) can be obtained for comparison with existing algorithms. The currently accepted standard in academia is to use the the MCNC benchmarks [74]. We will occasionally refer to and take examples from this collection of circuits; two such examples have already been shown in Figure 2.1. Industry would typically use proprietary benchmark sets, and would not announce results of their experiments.

Some terms with respect to implementation technologies: *full-custom* VLSI refers to the layout of a design (transistors and wires) on a totally empty and unconstrained space. *Standard-cell* refers to a technology where the basic blocks come from a library of predetermined logic elements which can be placed in rows at the specification of the designer. Detailed routing then reduces to channel routing (with feed-through cells) in the horizontal channels between the rows. A *gate array* technology constrains the logic elements to lie on a rectangular grid with both horizontal and vertical routing channels. *Mask programmable* gate array (MPGA) technology then allows the wires to be freely placed on a separate fabrication layer at manufacturing time.

2.1.2 Field-Programmable Gate Arrays.

The design of an application specific integrated circuit (ASIC) using either gate array or standard cell technology requires that the wiring is added as one step in the fabrication process. A recent technological alternative to this type of ASIC is the field-programmable gate array, which has both programmable logic elements and a programmable routing network to connect the logic. FPGAs can be programmed using just a personal computer and simple hardware interface, giving them flexibility and time-to-market advantages over traditional ASICs, which must have all wiring completed in a fabrication plant. However, programmability typically incurs a factor of ten in decreased chip density and a factor of three in decreased speed for the resulting hardware. This tradeoff is increasingly more acceptable to designers, and the FPGA industry has grown from an insignificant portion of the ASIC business in 1984 to a 1.4 billion US dollar industry today.

The advent of FPGAs spawns a host of new problems for CAD designers. Because FPGAs have a fixed routing network instead of "open real estate" the layout problem becomes more graph theoretic than geometric in nature. For rapid prototyping, it is common to implement a single design on multiple FPGAs or even boards of FPGAs, creating new variations on the partitioning problem which do not arise in higher capacity, more finely grained, ASICs. While the routing problem for gate arrays is one of minimizing channel width, CAD software for FPGAs deals with a binary fit/no-fit problem. Because of the programming logic, FPGAs also produce new challenges for timing estimation.

In addition to these new software problems, there is the issue of the FPGA *architecture*. Numerous choices exist in the design of an FPGA: Do I organize the logic and routing architecture hierarchically, or in a flat grid? How big should logic elements be? How many tracks should be placed in each row/column and how should they be connected together? Should the programming be permanent, or stored in a way which is reconfigurable? All of these issues must be addressed in the context of device cost, routability, timing, power consumption, noise, and the ability to write efficient CAD software. The architectural design process is inherently approximate, so many of these questions can only be answered empirically with benchmarks.

It is by no means clear which architectural choices are correct, or even if there are correct choices. The Actel Corporation manufactures FPGAs using a standard-cell like architecture,

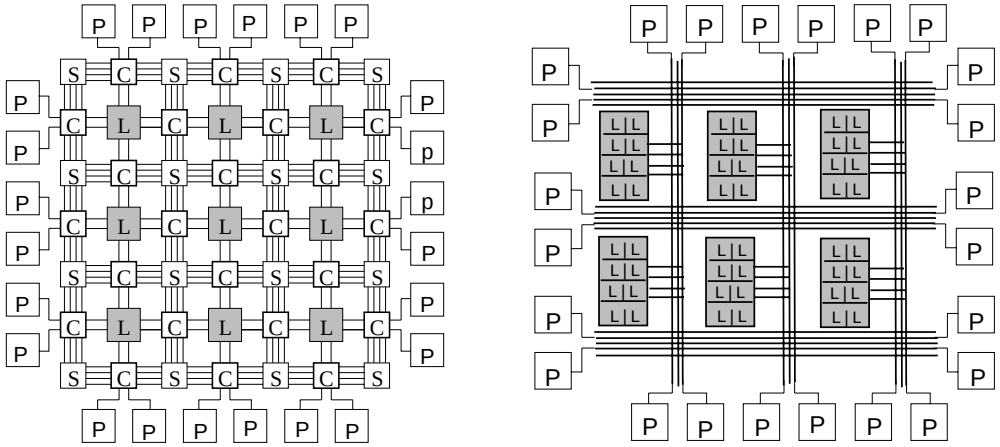


Figure 2.2: Different FPGA architectures.

and uses anti-fuse technology for permanent programmability (the only major vendor to do so). Altera's 10K series of devices organize logic elements into a shallow hierarchy: cliques of fully connected logic and a more sparse interconnection structure between cliques. Xilinx uses a "flat" architecture reminiscent of a gate array, with a routing architecture consisting of multi-track channels with "switch" (S) block modules at the intersection of channels, and "connection" (C) block modules where logic-block pins enter the routing network (P and L stand for pin and logic block, respectively). Abstract representations of Xilinx and Altera architectures are shown in Figure 2.2. Both Altera and Xilinx use SRAM bits to program the parts, which means the logic can be re-programmed repeatedly, in some cases during the computation itself though this is not commonly done.

The research described in this thesis applies both to the ASIC and the FPGA world, but it is of particular interest for FPGAs. As mentioned previously, hardware and software architects of a "new" 1,000 LUT FPGA have to deal with the discrete fit/no-fit issue rather than more fine-grained optimization problems. Typically this means a large number of circuits in the 900-950 LUT range would be required to exercise the device, while neither a 400 LUT circuit nor a 1,200 LUT circuit would be an interesting test case. The circuits must also be representative enough to deal with the vastly different types of circuits that a user might wish to implement. Thus FPGA vendors consider their benchmark suites to be closely guarded proprietary information, and universally feel that there are "never enough benchmarks."

2.1.3 Graph Classifications.

A *class* of graphs is a set of graphs which are related in some way. A class can be defined by some specific graph-theoretic property, for example "A graph is *regular* if all vertices have the same degree." The members of the class can sometimes be defined by recursive construction: "A single vertex is a *tree*, a tree T with a new vertex x and an edge from x to some vertex v of T is also a tree." The class can be determined by virtue of what it does not contain, for example a forbidden configuration or subgraph: "A *tree* is a connected graph with no cycles."; "A *planar* graph is a graph which contains no subgraph "homeomorphic" to the graphs K_5 or $K_{3,3}$." Any other well defined mathematical definition would also be appropriate. Note that often a graph class can be defined in multiple equivalent ways. For example a planar graph is commonly defined geometrically as "a graph for which there exists a embedding which maps vertices to points in the plane and edges to Jordan curves connecting their respective endpoints that do not intersect except at those endpoints."

If $G = (V_G, E_G)$ is a graph then any graph $H = (V_H, E_H)$ where $V_H \subseteq V_G$ and $E_H \subseteq E_G$ is a *subgraph* of G. If xy is an edge of G, and xy is in H whenever both x and y are in H, then we say that H is an *induced* subgraph of G, otherwise it is a *partial* subgraph. It is often interesting when the definition of a class is closed under the taking of subgraphs; that is, the definition of the class is *hereditary*. Planarity is hereditary, because any subgraph of a planar graph is clearly planar.

2.2 Previous Work.

2.2.1 Rent's Rule.

The commonly accepted relationship called "Rent's rule" dates back [49] to E. F. Rent of IBM, who made an empirical observation regarding the partitioning problem:

Rent's rule: Let G be a circuit with n blocks (nodes) and m wires (edges). Consider a "reasonable" partition of the blocks of G into modules $M_1, M_2, \cdots, M_l$ where the modules each satisfy a pin constraint: the number of external vertices in any M_i is constrained to some value P^*, and the number of modules is no less than five. Then the empirical relationship

$$P = kB^r \tag{2.1}$$

is found to hold in general, where

k = the average number of edges incident on a block,

P = the average number of pins (external vertices) in a module,

B = the average number of blocks in a module, and

r = the "Rent exponent", empirically $0.5 \leq r \leq 0.8$.

Satisfyingly enough, for the trivial "total" partition of G into modules of one block each Rent's rule with $B = 1$ correctly gives the average number of pins as the average degree over the blocks in the network. This does not hold empirically for partitions into only a few modules, as stated in the definition and discussed later.

The algorithm for separating the circuit into modules is undefined in the standard formulation of Rent's rule. Later refinements by Feuer [25] specify that placement provides such a "good" partition into modules in terms of geometric proximity in the sense that from any circle (closed set of grid points of Manhattan distance r from a fixed centre point) the number of external connections will follow Rent's rule on average. So we can think of Rent's rule as both a law that holds for a given partition, on average, and at the same time as the *expected* relationship for a specific module in terms of its terminal and non-terminal vertices.

It is crucial to note how closely the notion of Rent's rule is tied to that of a good empirical modularization. For example, it is possible to self-embed $G_{N,N}$ (equivalently, give a partition) badly so that every wire is of length $\frac{N}{2}$, yielding a channel width of $O(N)$ and Rent exponent $r = 1$, even though the trivial embedding has $W = 1$ and $r = 0.5$. Thus any discussion of Rent's rule holding in an abstract sense must capture somehow the *existence* of some modularization, either in a non-constructive sense or by exhibiting the modularization directly. Hagen *et. al.* [35] investigated this in detail, and defined the *intrinsic Rent parameter* of a circuit as the minimum possible Rent parameter over the set of all partitioning algorithms. They gave empirical evidence to show that different algorithms do yield different values for the Rent parameter. We also stress that Rent's rule applies to modules *on average* and does not address maximum or minimum behaviour for a particular module.

Empirical calculations.

Landman and Russo [49] discuss the historical origins of Rent's rule. They calculate $P = 4.17B^{0.65}$ for Rent's initial data and cite various other independent confirmations: A study by Meade and Geller [52] yields $P = 4B^{0.7}$. Notz *et al.* [55] find $P = kB^{\frac{2}{3}}$ where k is one plus the average fan-in of the network. Radke [57] also mentions the "common knowledge" of the rule (attributing it to Rent), noting observations of p varying from 0.5 to 0.7 with values of k between 3 and 5.

The typical method for calculating r is to perform an empirical partition, sample modules for values (P_i, B_i), and perform a linear regression on $\log P_i = \log k' + r \log B_i$ $(P_i = c^{k'} B_i^r)$, usually constraining $k = c^{k'}$ as a constant. Landman and Russo specifically point out that Rent's Rule is unstable when the number of modules is less than 5. One reason that this would be true is that the number of pins on a chip is usually a hard constraint in practice, and the engineer must build the design within the given number of I/Os. Hierarchy inside the chip does not suffer from these hard constraints, and should exhibit more consistent behaviour.

Russo [58] notes that more parallel "high performance" circuits tend to exhibit larger r, because they tend to have a higher pins-per-gate ratio, hence Rent exponent.

Theoretical Issues.

In an attempt to understand the determinants underlying Rent's rule, and also to investigate the tradeoff between logic (control) and memory, Donath [17] developed a model of the process of designing computer hardware. He models the modular decomposition process of hardware design, and argues that Rent's Rule is a natural consequence of a structured design methodology. Donath also investigated the "information content" involved in trading memory bits for logic (i.e. implementing logic functions as lookup-tables in ROM), and derived a rough rule of thumb which states that one basic logic element (gate) is equivalent to 8.5 bits of memory[4].

Landman and Russo cite an old unpublished manuscript of Donath [20] in which he proves that a random graph G, defined as "a graph with edges distributed randomly among

[4]This suggests that a 2K ROM, used as a lookup table, would be expected to implement a boolean function comprising about 1900 2-input NAND gates (on average). Similarly, a 2K truth table could be expected (on average) to optimize into about 1900 gates of combinational logic.

its vertices," exhibits a linear relationship between P and B (i.e. $r = 1$). The statement is difficult to interpret without a concrete random graph model, but the basic property will also be visible in the theoretical wirelength studies of the next section.

Discussion.

Rent's Rule works well as a predictor of I/O to logic ratios for internal connections to a chip. Most researchers would use a safe overestimation of r to predict the number of pins required for a chip, or to generate a theoretical "envelope" on the number of tracks or wirelength but, in practice, would combine this with empirical analysis.

For our purposes, Rent's Rule is not a good "characterization" of circuits, because of its reliance on an existing parameterization and on its reference to the average-case behaviour of the partition hierarchy. However, Rent's Rule is a well accepted guideline in the community, and important to keep in mind as a general rule of thumb about circuits.

2.2.2 Stochastic Wireability Models.

Routability refers to estimating the wirelength or fittability of a circuit on a given host graph or architecture. Early research on gate-arrays gave us a number of statistical properties and distributions which can be used to predict routability for circuits.

Wire length distributions.

Using random placements [16, 36, 59], or assumptions about stochastic properties of placement [24, 61] and Rent's rule [18, 19, 25] various theoretical models of wire-length have been proposed.

Donath [16] studied the statistical properties of randomly placing a random graph on a grid. He developed a lower bound on the average wire-length $\bar{R}$, over *all* placements, for an embedding of a given G into $G_{N,N}$. He showed that this lower bound is dependent only on n and m (the number of edges), and is independent of the structure of the graph:

$$\bar{R} = \frac{m n^{\frac{1}{2}-\frac{n}{2m}}}{e^{1-\frac{n}{2m}}} \tag{2.2}$$

$$= \frac{m n^{\frac{1}{2}-\frac{1}{2k}}}{e^{1-\frac{1}{2k}}} \qquad \text{(average vertex degree } k\text{)}. \tag{2.3}$$

The bound provides some information, since we expect that any reasonable algorithm does better than the expected random placement. However, the bound implies an $\bar{R}$ of approximately $\frac{N}{3}$ and $W = O(N)$ [18], so the bounds are too loose to have any practical utility: studies have shown both $\bar{R}$ and W^* to be roughly proportional to $\log N$ in practice. Note that this result also implies the unit Rent exponent for the random graphs in Donath's construction.

Donath [18] later developed a formula for the upper bound on expected average wire length $\bar{R}$ based on a "pseudo-random" placement. The placement is partly stochastic, but attempts to "reflect both the characteristics of logic complexes as they are designed by engineers and the effect of the placement procedure." By assuming that Rent's rule holds recursively, he developed a new upper bound for the expected average wire-length.

$$\begin{aligned} \bar{R} &\sim B^{r-\frac{1}{2}}, && r > \tfrac{1}{2} \\ \bar{R} &\sim \log B, && r = \tfrac{1}{2} \\ \bar{R} &\sim f(r), && r < \tfrac{1}{2} \qquad \text{(independent of } B.) \end{aligned} \tag{2.4}$$

An important note is that the estimator under the Rent assumption differs from that of a purely random placement, which yields $\bar{R} \sim \sqrt{B}/3$ as mentioned earlier. Donath compares his upper bound to experiments on five real circuits and finds that the estimate is about double the average wire length found in practice. The dependence on both B and p is supported by the experiments.

Feuer [25] does a similar analysis to develop wire length estimators from Rent's rule, and also calculates the distribution of wire lengths. He derives, from Rent's rule and several simple geometric assumptions about the placement, an expression

$$R_l = c(r, B) l^{2r-4} \tag{2.5}$$

for the expected number of connections between any two grid points of Manhattan distance l apart in a placed circuit. The parameters r and B are from Rent's rule, and c is a constant function of these parameters only, hence constant for a given graph.

This distribution leads to expressions for the average wire length of connections *internal* and *external* to a region of radius d:

$$\bar{R}^i = \sqrt{2}\frac{\alpha(5-\alpha)}{(3-\alpha)(4-\alpha)}\frac{B^{r-0.5}}{(1-B^{r-1})}, \tag{2.6}$$

and

$$\bar{R}^e = \sqrt{2}\frac{(1-\alpha)(5-\alpha)}{(3-\alpha)(4-\alpha)}B^{r-0.5} \tag{2.7}$$

where $\alpha = 2 - 2r$.

The overall average wire length predicted by the model is

$$\bar{R} = \sqrt{2}\frac{(2-\alpha)(5-\alpha)}{(3-\alpha)(4-\alpha)}\frac{B^{r-0.5}}{(1+B^{r-1})}. \tag{2.8}$$

Since $(1+B^{r-1})$ vanishes for large B, the latter is proportional to $B^{r-\frac{1}{2}}$; exactly as derived by Donath for $r > \frac{1}{2}$ (Equation 2.4).

Feuer's analysis yields justification that geometric proximity after placement is itself a "good modularization" for application of Rent's rule, as mentioned earlier, because the derivation from the proximity assumptions generates Rent's rule, which is then itself assumed for the derivation of the wire-length estimators.

El Gamal and Syed [24] define a purely stochastic model in which wire lengths are distributed Poisson(λ) and wire trajectories are parameterized by $\gamma, \alpha, \beta, p, u$. They develop a formula for average wire length in terms of these parameters, and estimate the parameters using empirical data. As an application of their model they vary the parameter u, the percentage of utilized gates, holding other parameters fixed and find that "it is better to use an array of size $\frac{n}{.8}$ with more tracks (channel width) than a larger array of size $\frac{n}{.5}$ with fewer tracks." It is stated that 100% utilization ($u = 1$) is unrealistic, and implied that the model bears this out as well.

Sastry and Parker [59] show that "any placement which satisfies Rent's rule, or any similar pin-to-block relationship," will have a wire-length distribution which is Weibull:

$$R_l = \alpha\beta l^{\beta-1}e^{-\alpha l^\beta} \tag{2.9}$$

with mean

$$\bar{R} = \frac{1}{\beta}\left(\frac{1}{\alpha}\right)^{\frac{1}{\beta}}\Gamma\left(\frac{1}{\beta}\right). \tag{2.10}$$

The parameters α and β are calculated from the empirical data by log-linear regression on empirical data. Though there is definitely a relationship between r and these parameters, the authors do not develop a closed formula, and instead rely on the regression to give empirical values.

Channel Width.

Wire length alone does not capture the effect of difficult areas or "hot-spots" with high channel width. What we often would like is to have a prediction of the greatest channel width in the array. This, of course, would have to be less than the available channel width if routing is to take place.

El Gamal [23] gives such a model. He calculates W in terms of $\bar{R}$ assuming that the distribution of lengths is geometric and adding the additional assumptions of trajectory along a minimum (Manhattan) distance path in the array: each lattice point emits X_i wires of length L_{ij}—given an initial trajectory (up-right, up-left, down-right, down-left) flip L_{ij} coins and move up or down on heads and left or right on tails, as appropriate.

The conclusions to be drawn vary with $\bar{R}$. If $\bar{R}$ is finite, then the distribution of channel densities is Poisson:

$$W_t = \mathcal{P}\left(\frac{\lambda \bar{R}}{2}; t\right) \tag{2.11}$$

where W_t is the number of channel segments with width t. The expected maximum channel density converges to $O(\ln N)$ (almost always) when $\bar{R} \leq O(\ln N)$ and $O(\bar{R})$ (almost always) otherwise. Since the former seems the most reasonable occurrence the primary conclusion is that channel densities are distributed Poisson with a mean channel density of $\frac{\lambda \bar{R}}{2}$. Brown *et. al.* [10, 11] find the accord between this prediction and several actual circuits to be very good. They note, however, that the model becomes less accurate if the FPGA model is expanded to give segments of more than unit length.

Applying Routability.

Chan, Schlag and Zien [12] recently combined several of the results just discussed to predict routability for a Xilinx 3000 series FPGA. Circuits are classified as "unroutable", "marginally routable" and "easily routable" based on the Feuer's expectation of channel width for the circuit vs. the available channel width, an estimator for the Rent parameter

r using mincut partitioning and El Gamal's estimator for W.

Another (earlier) model for routability was given by Brown [10]. Here, routing is a stochastic process with parameters specifying the network for the FPGA (e.g. the number of connections in the switch and connection blocks, the channel width), several model-parameters (event probabilities) and basic properties of the circuit to be routed (size, connections and expected wire length $\bar{R}$ (from El Gamal)). An expression for the expected percentage of unrouted connections is generated. The model has been used both as an indicator of routability and as a vehicle for determining good settings for the parameters which specify the FPGA; e.g. to determine how much flexibility (how many switches) to put in a Xilinx C-block or S-block.

Discussion.

The stochastic results cited in this section are the traditional approaches to characterizing circuits and determining theoretical bounds for architectural parameters. The goals of this thesis are quite different, in that we want to determine graph-theoretic characteristics taken from analyzing the circuit *graph* itself. The purpose of including this previous work is more to provide context for the current research, and because the terms introduced here are used elsewhere in the thesis.

2.2.3 Other Generation Efforts.

Random Graphs.

In this thesis, the term a *random graph* will refer to graphs generated by stochastic methods which do not take into account the properties of digital circuits. Such random graphs are drawn uniformly from the set of all graphs, or uniformly from a partially restricted set of all graphs, such as "all regular graphs." The most common such model is the random undirected graph $G(n, p)$, defined as a graph on G nodes in which each potential edge exists with uniform independent probability p. These graphs can be easily generated, but are not realistic as circuits: for $p < n \log n$ they are disconnected and otherwise they contain $O(n \log n)$ edges and have most nodes with degree $\log n$, *almost always*. The former makes the graph uninteresting, and the latter makes it electrically infeasible as a circuit.

There are also well-known methods for generating random degree-constrained graphs

uniformly. We use one such method (with modifications) for comparison to our circuits in Chapter 6. However, there are no known methods to uniformly generate directed I/O constrained graphs, or to generate directed graphs with restricted path-lengths, or which properly satisfy the electrical constraints of a synchronous sequential circuit.

There is a long history of using random undirected graphs as benchmarks. Kernighan and Lin [46], Johnson *et. al.* [45], Krishnamurthy [48] and Hagen and Kahng [34] (for others see [50]) used random graphs to compare and evaluate partitioning algorithms. Vargese *et. al.* [68] and Hauk *et. al.* [37] also used random graphs to study architectural parameters and algorithmic issues for logic emulation systems with FPGAs, which require very large circuits. Random graphs are currently unavoidable for experimentation beyond the size of existing circuits.

Generating Circuits by Transformation.

Iwama *et. al.* [43], in independent work, discuss how to apply transformation rules to a initial seed circuit and create a different structural circuit with the same logic function. This work applies only to combinational circuits, and is limited to generating variations on the initial circuit. In a paper to appear later this year [44], they will discuss an improvement on the work which generates seed circuits from random truth-tables, rather than requiring an input circuit.

This work is primarily aimed at benchmarking for logic synthesis (logic independent optimization) algorithms. The authors do not describe any applications of the approach to dealing with physical design algorithms or architectural issues.

Generating Circuits with Rent's Rule.

In independent work, Darnauer and Dai [15] have recently given an algorithm for generating random undirected graphs to meet a given Rent parameter. The basic idea is to generate a random partition hierarchy, and recursively generate a graph from it. The approach has an obvious attraction for partitioning, which was its primary application. Darnauer and Dai showed the empirical validity of their algorithm for relatively small combinational circuits on partitioning problems.

The primary drawbacks of the method are that the tool loses control over combinational delay and does not have the ability to generate sequential circuits with the properties which

we concentrate on in this thesis and which are important for FPGA architectures and all other physical-design CAD problems. We believe it would be possible to incorporate the most important aspects of the Rent-based method into the high-level hierarchy of our sequential generation; one area for further work would be to investigate combining our approach for generating medium-size circuits with a high-level partition hierarchy. We will remark further on this in Chapter 7.

2.3 "Obvious" Properties of Circuit Graphs.

Based on common knowledge of digital circuits, we can make a number of preliminary observations about their combinatorial structure.

One obvious property is that the class of circuit-graphs is hereditary: if G is a circuit then any induced subgraph of G will also be a circuit. Because electrical fanin and fanout are constrained in all but special circumstances, we observe that the number of edges should be linear in the number of nodes. For convenience, we will assume that any complete circuit is connected, since a CAD algorithm could easily check connectivity and significantly decrease the problem complexity on disconnected graphs.

From Rent's rule, we can expect that circuits exhibit some type of "hierarchical" structure. However, this is an abstract notion only, since Rent's rule and the wireability studies mentioned earlier do not give any applicable graph-theoretic restrictions which we can use directly.

Also from empirical studies of Rent's rule, we note that the number of inputs and outputs in a circuit is sub-linear in the number of nodes (unless $r = 1$ for the circuit, which is not seen empirically). For a chip with a reasonable aspect ratio and packaging constraints this follows independently of Rent's rule, since the number of I/Os can only be a small constant multiple of the perimeter.