CMU-CS-80-111

Statistics on VLSI Designs[1]

Jon Louis Bentley[2]
Dorothea Haken
Robert W. Hon

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

17 April 1980

Abstract -- This paper presents a statistical study of the components on VLSI chips. We examine the size and shape of components, and their placement over the chip area. The data is useful for building efficient VLSI design tools: the form of the distribution shows that some simple strategies can lead to efficient algorithms, and the parameters of the distribution aid in choosing program parameters. To illustrate the application of the statistics to VLSI design tasks, we present an algorithm for solving the "rectangle intersection" problem that arises in design rule checkers.

[1]This research was supported in part by the Defense Advanced Research Projects Agency under Contract F33615-78-C-1551 (monitored by the Air Force Office of Scientific Research) and in part by the Office of Naval Research under Contract N00014-76-C-0370.

[2]Also with the Department of Mathematics.

Table of Contents

1. Introduction

The "VLSI explosion" seems to be making the world of computers better and better: with every technological advance we can build machines that are smaller, faster, and cheaper than their predecessors. There are, however, several concomitant difficulties associated with these advances, one of the more noticeable being the problems inherent in designing a chip that contains hundreds of thousands of transistors. For this reason a great deal of recent research has focussed on building a VLSI design system that can automate certain design tasks. These tasks include such problems as *laying out* components, *routing* wires between components, and *design rule checking* to ensure that the design satisfies a set of constraints.

The primary purpose of this paper is to present a set of statistics on VLSI chips that can be used when constructing a VLSI design system. The statistics describe the geometry of chips: that is, the shape of their components and how those components are placed on the chip. To illustrate the application of the statistics, we will describe an algorithm useful in design rule checking that was developed by using the statistics. (A design rule checker that uses this algorithm is currently under development at Carnegie-Mellon.) The primary audience for this paper, therefore, is the community interested in constructing VLSI design systems.

A secondary purpose of this paper is to provide a study in "applied algorithm design". A great deal of work has been done recently on the probabilistic analysis of algorithms, and the design of algorithms with fast expected running time. One weak part of that work, however, is that it has merely assumed that the input data is drawn from a particular underlying distribution, and has not questioned how practitioners might justify that assumption. This paper provides an example of the justification of probabilistic assumptions, and the development of both an abstract algorithm and a concrete program based on the assumptions.

Chapter 2 contains the statistical study of a set of VLSI designs. An algorithm for computing rectangle intersections that exploits those statistics to achieve fast expected running time is presented in Chapter 3, and conclusions are offered in Chapter 4.

2. Statistics on Designs

In this chapter we will study the distribution of components on VLSI chips. There are two primary knowledge sources that we will bring to bear in this study. The first is an *a priori* knowledge of the design style of (at least one school of) chip designers. The second source is an empirical study of a set of actual chip designs.

This chapter is organized as follows. In Section 2.1 we briefly describe both the basic design philosophy underlying the chips we studied and the chips themselves. Section 2.2 contains statistics on the *shape* of the components placed on chips, and Section 2.3 contains statistics on the *placement* of the components on the chips. A probabilistic model summarizing these studies is presented in Section 2.4. (All the measurements in this chapter are used in the construction of the rectangle intersection algorithm described in Chapter 3. Since issues of area are not important for the algorithm of Chapter 3, we defer discussion of the data on the area occupied by components and wires to Appendix VI.)

2.1 Description of the Designs

Before we describe the statistics in detail, it is important to give a few words of background on the chip designs studied. All were the result of VLSI design courses taught at research institutions, including Caltech, Carnegie-Mellon, MIT, Stanford, and Xerox PARC. The designers were researchers schooled in the Mead and Conway [1980] style of hierarchical IC design; the target process was silicon-gate nMOS, using Mead and Conway's design rules and a value of λ from 2.0 to 3.0 microns. (The constant λ is the size of the smallest features resolvable by the implementation process; typical minimum-sized transistors have gate widths of 2λ.) All of the designs were expressed in a geometric specification language called *Caltech Intermediate Format* (or CIF -- see Hon and Sequin [1980]), regardless of the design system used to lay out the chip.

In this section we will present statistics on sixteen VLSI designs. All designs are taken from recent multiproject chips managed by the LSI group at Xerox PARC. Communication and file transfer were facilitated by the ARPANET. The sixteen chips we study are all those available to us that contain over ten thousand geometric primitives. The designs were for a variety of tasks, ranging from a digital clock to a machine that interprets Lisp code; a short description of the sixteen chips is available in Appendix I. In addition to the tables in this chapter summarizing the sixteen chips, the six largest chips are examined in more detail in Appendices II through V.

The results that we will present in this chapter are drawn from a relatively small number of designs, yet we feel that they are applicable in a much broader context. We will now enumerate several issues regarding the chips chosen for our study, and discuss how those issues affect the scope of our conclusions.

- *The designs are fairly small.* Although all designs consisted of at least 10,000 rectangles, the largest had only 109,862 rectangles.[3] Our results are quite consistent across this order-of-magnitude difference, and we see no reason to believe that they will change significantly for larger chips.

- *All designs were in the Mead-Conway style.* This implies that the designers placed emphasis on regular, modular designs with clean communication schemes, rather than optimizing the circuit for the densest (in terms of active devices per λ^2 area) layout. This is a fundamental departure from common industrial practice.

- *Are the results biased towards a particular design system?* The CIF descriptions we processed were produced by a variety of design systems, ranging from an interactive drawing system to a simple version of a "silicon compiler." For this reason, we expect the set of designs to be free of heavy bias from one particular design system. It should be noted that many cells used in the designs came from a common library (for example, input/output pads and PLA cells).

- *Our studies ignored the hierarchy inherent in CIF.* Our study of the chips dealt only with the instantiated symbols on the chip, ignoring the hierarchical structure of their CIF descriptions.

- *Logos on the chips were not excluded from our study.* Several of the chips we studied contain pictures, ranging in complexity from the designer's initials to a map of the Boston subway system. The components in these logos were included in our studies as though they were active components in the design. Because there were relatively few logos, and all contain relatively few rectangles, the logos had little impact on our statistics.

2.2 Statistics on Shape

The first set of statistics we gathered has to do with the *shape* of the components on chips. All of the chips we examined were designed using primarily rectangles with edges parallel to the coordinate axes. The few components that were not rectangles were surrounded by their "bounding rectangles" before being passed to our analysis programs. We may therefore refer to the chip components as rectangles, and the task of this section is to describe the shapes of the rectangles on the chips. In Section 2.2.1 we introduce a trichotomy that places each rectangle in one of three distinct classes, and the rectangles within these classes are then studied in Sections 2.2.2, 2.2.3 and 2.2.4, respectively.

[3] To relate the rectangle count to the more typical measure of active device count, we note that chip 15 had approximately 96,000 rectangles and 13,500 active devices, while chip 16 had approximately 110,000 rectangles and 11,000 active devices.

2.2.1 A Trichotomy of Rectangles

Any attempt to describe homogeneously all rectangles on a VLSI chip is doomed to failure. No matter what measurements are chosen (such as average edge length, aspect ratio, or area), there will be a tremendous variance in the results. The reason for this variance is that designers use several fundamentally different classes of rectangles, and although the rectangles within these classes are quite similar, rectangles across the classes are remarkably different. As a first approximation, it is safe to say that most designers have the following classes of rectangles in mind as they design a VLSI circuit: *components* -- the small rectangles used to synthesize logic or storage devices; *wires* -- the skinny rectangles used to connect distant components; and *other* rectangles (for example, large pads).

To formalize our intuitive notions, we gave these fuzzy concepts the following precise definitions.

- A *component* is any rectangle with neither side greater than 10λ.
- A *wire* is a rectangle with one side greater than 10λ and the other side not greater than 6λ.
- An *other* rectangle is any rectangle that is neither a component nor a wire.

It is important to emphasize that a rectangle classified as a wire, for instance, in the above trichotomy is not necessarily viewed as such by the designer[4]; nonetheless, this trichotomy will serve well in explaining the various shapes on chips. The robustness of this trichotomy is substantiated both by the consistency of the data soon to be presented, and by the histograms of Appendices II and IV. (Discussions of the robustness can be found in those appendices.)

Table 1 contains the first set of data we gathered; we will explain the various columns of the table by describing in detail its first row. That row describes chip number 1, which contains 10803 rectangles. Of those rectangles, 6723 (or 62.2%) are classified as components. Additionally, 1665 rectangles are vertical wires and 2002 are horizontal wires; therefore a total of 33.9% of all rectangles are wires (either vertical or horizontal). The remaining 413 rectangles were classified as "others", for a total of 3.8% of all rectangles. The last three columns of the table report data on all 21606 edges of the rectangles (note the number of edges on a chip is twice the number of rectangles). The mean edge length is 8.8λ, with a standard deviation of 18.6λ, and the longest edge on the chip is 660λ.

---

[4] Likewise, what is viewed as a wire by the designer may be classified by this trichotomy as many components. This is because long data paths may be built up out of many pieces in hierarchical designs. For example, if a memory array is constructed by a sequence of instantiations of a symbol defining a single memory cell, then each cell would contain a piece of the data line running through the entire array. The measurements under discussion are based on the rectangles that result from the fully instantiated symbol structure. No effort was made to coalesce separate pieces of the same wire. Although an analysis of the "coalesced" structure is also interesting, we present our results because many programs that work with chip designs are given input in this form.

Chip #	# Rects	Components #	Components %	Wires # Vert	Wires # Hor	Wires %	Others #	Others %	Edge Length Mean	Edge Length S. D.	Edge Length Max
1	10803	6723	62.2	1665	2002	33.9	413	3.8	8.8	18.6	660
2	11265	8409	74.6	1356	1235	23.0	265	2.4	7.9	39.3	1711
3	11566	8491	73.4	1164	1737	25.1	174	1.5	7.9	18.7	800
4	11853	7733	65.2	1826	1914	31.5	380	3.2	8.5	22.5	1088
5	11915	9027	75.8	1277	1356	22.1	255	2.1	7.5	23.0	954
6	12063	7405	61.4	2324	2149	37.1	185	1.5	8.0	22.6	909
7	12800	9176	71.7	1605	1744	26.1	275	2.1	7.6	14.4	406
8	14186	10936	77.1	920	2076	21.1	254	1.8	8.6	36.8	1155
9	14423	11324	78.5	1189	1611	19.4	299	2.1	6.3	10.1	106
10	15097	10786	71.4	2012	2015	26.6	284	1.9	6.9	11.6	356
11	16194	11128	68.7	2565	2103	28.8	398	2.5	8.9	22.6	968
12	17565	10219	58.2	3598	3124	38.3	624	3.6	6.7	9.5	460
13	18056	13731	76.0	2381	1751	22.9	193	1.1	7.0	16.2	1068
14	33387	25431	76.2	3078	4628	23.1	250	0.7	6.8	18.1	1675
15	95901	76116	79.4	6773	12108	19.7	904	0.9	6.7	18.7	2902
16	109682	73939	67.4	17838	16874	31.7	1031	0.9	7.0	21.9	1523

Table 1. General data on rectangles.

The percentages in Table 1 show that our trichotomy of rectangles is consistent with chip designs -- that is, the percentages of components, wires and others remain approximately constant across various chips. This can be seen roughly by scanning the fourth, seventh, and ninth columns of Table 1. To make the comparison more obvious, we offer the following summary of those columns of Table 1.

	Median	Mean	S.D.
% Components	72.5	71.1	6.34
% Wires	25.6	26.9	5.84
% Others	2.0	2.0	.91

That is, of the sixteen component percentages, the median is 72.5%, the mean is 71.1% and the standard deviation is 6.3%. This summary shows that all three percentages remain consistent across the sixteen chips: the median and the mean percentages are very close, and the standard deviations are fairly small.

The second interesting aspect of Table 1 is contained in the last three columns, which summarize the edge lengths observed on the chip. Note that the mean edge length observed is very consistent: the mean of the sixteen values is 7.57λ, while the median is 7.55λ. Unfortunately, the standard deviation of edge lengths is also consistent -- consistently very high. The median standard deviation is 18.7λ and the mean is 20.3λ. The high standard deviations are due to the fact that there is a tremendous discrepancy among different kinds of rectangles; to reduce this deviation we must study rectangles in the various classes individually.

2.2.2 Data on Components

The data in Table 2 describes the rectangles that were classified as components (that is, those rectangles with neither edge greater than 10λ). The first four columns repeat information from Table 1; the fifth column reports the mean edge length of all components on the chip (in λ), and the sixth reports the standard deviation of the edge lengths. The edge lengths are remarkably consistent: the median of the sixteen averages is 3.9λ, and their mean is also 3.9λ (with a standard deviation of only .16λ). Note that the standard deviations are all very close to 2λ.

Chip	#	Components		Edge Length	
#	Rects	#	%	Mean	S. D.
1	10803	6723	62.2	3.9	2.0
2	11265	8409	74.6	3.8	1.9
3	11566	8491	73.4	4.2	2.3
4	11853	7733	65.2	4.0	2.1
5	11915	9027	75.8	3.6	1.9
6	12063	7405	61.4	3.9	2.0
7	12800	9176	71.7	3.9	2.0
8	14186	10936	77.1	3.9	1.9
9	14423	11324	78.5	3.8	1.6
10	15097	10786	71.4	3.9	2.0
11	16194	11128	68.7	4.0	2.0
12	17565	10219	58.2	3.5	1.6
13	18056	13731	76.0	3.9	1.8
14	33387	25431	76.2	3.7	2.0
15	95901	76116	79.4	4.0	2.1
16	109682	73939	67.4	3.8	1.7

Table 2. Data on components.

The above data is only a brief sketch of the distribution of components. More detailed studies are contained in Appendix II: we give two-dimensional histograms of component sizes for the six largest chips we studied. Those histograms show that the overwhelming majority of components on a chip are 2λ-by-2λ, 2λ-by-4λ, 4λ-by-4λ, or 4λ-by-6λ in size, which correspond to minimum-size contact cuts, polysilicon, diffusion, or metal flashes, and the contact cuts used in "butting contacts" (see Mead and Conway [1980]).

2.2.3 Data on Wires

Data on the rectangles that were classified as wires is contained in Table 3. The first five columns of that table are duplicated from Table 1. The next two columns describe the lengths of the short edges of the wires (which are, by definition, not greater than 6λ). The final three columns describe the lengths of the long edge of the wires (which are greater than 10λ).

Chip #	# Rects	Wires # Vert	# Hor	%	Short Edge Mean	S. D.	Long Edge Mean	S. D.	Max
1	10803	1665	2002	33.9	3.1	1.3	25.6	28.3	461
2	11265	1356	1235	23.0	2.9	1.3	33.8	106.8	1711
3	11566	1164	1737	25.1	3.3	1.6	28.3	28.8	340
4	11853	1826	1914	31.5	3.1	1.1	24.8	34.6	488
5	11915	1277	1356	22.1	2.9	1.2	30.9	41.1	612
6	12063	2324	2149	37.1	3.2	1.0	23.4	39.7	630
7	12800	1605	1744	26.1	2.9	1.1	27.9	25.8	406
8	14186	920	2076	21.1	3.1	1.5	39.4	92.0	831
9	14423	1189	1611	19.4	3.2	1.5	23.5	17.0	96
10	15097	2012	2015	26.6	3.2	1.3	22.6	21.2	356
11	16194	2565	2103	28.8	3.0	1.2	32.1	40.5	791
12	17565	3598	3124	38.3	3.2	1.1	17.0	10.4	322
13	18056	2381	1751	22.9	3.2	1.4	26.8	23.5	303
14	33387	3078	4628	23.1	3.2	1.5	28.9	40.5	1284
15	95901	6773	12108	19.7	3.1	1.3	28.3	40.2	1771
16	109682	17838	16874	31.7	2.9	1.1	22.2	45.6	1523

Table 3. Data on wires.

A number of facts are apparent from Table 3. The first is that most designs have a number of horizontal wires approximately equal to the number of vertical wires, as one would expect. There are several designs, however, that are obviously not "balanced" in this sense: chips 3, 8, 14 and 15 all have at least 50% more horizontal wires than vertical.

The data concerning the short (not greater than 6λ) side of the wires is quite consistent. The mean of the sixteen mean values is 3.1λ (with standard deviation $.13\lambda$) and the median is 3.1λ. The standard deviation of the short edge length is consistently very close to 1.3λ.

The lengths of the longer edge of the wires are more difficult to describe. The median value of the sixteen mean edge lengths is 27.4λ, and their mean is 27.2λ (with standard deviation of 5.1λ). The typical standard deviation is approximately 40λ. Histograms of the distribution of the long edges can be found in Appendix III.

2.2.4 Data on Other Rectangles

Table 4 presents data on the "other" rectangles that were classified as neither components nor wires. The first four columns are duplicated from Table 1, and the last three columns describe the edge lengths of the components in λ.

The median of the sixteen average lengths is 45λ; their mean and standard deviation are 43.1λ and 8.5λ, respectively. The standard deviations are centered around 80λ, but vary widely. Two-

Chip #	# Rects	Others #	Others %	Edge Length Mean	Edge Length S. D.	Edge Length Max
1	10803	413	3.8	39.0	52.3	660
2	11265	265	2.4	35.7	71.8	1118
3	11566	174	1.5	53.7	98.6	800
4	11853	380	3.2	45.9	80.9	1088
5	11915	255	2.1	48.5	103.1	954
6	12063	185	1.5	45.7	95.0	909
7	12800	275	2.1	36.5	38.1	353
8	14186	254	1.8	57.6	128.2	1155
9	14423	299	2.1	37.2	29.2	106
10	15097	284	1.9	33.9	28.6	106
11	16194	398	2.5	44.3	75.8	968
12	17565	624	3.6	24.4	28.0	460
13	18056	193	1.1	56.1	104.9	1068
14	33387	250	0.7	46.1	88.5	1675
15	95901	904	0.9	38.2	114.5	2902
16	109682	1031	0.9	46.6	95.1	1416

Table 4. Data on "other" rectangles.

dimensional histograms showing the distributions of other components are available in Appendix IV.

2.3 Data on Placement

The data we have mentioned so far has discussed only the *shape* of the components; in this section we will describe how those shapes are distributed. There are two questions that we must answer in this study: in what sort of region are the rectangles placed, and how are they distributed over that region?

The first issue to be addressed is the region over which the rectangles are distributed. All the chips that we studied were designed to be enclosed within a bounding rectangle, referred to as its "bounding box". Table 5 contains data on the bounding boxes of the chips we studied. The first two columns are duplicated from Table 1. The next two columns give the width and the height of the chips' bounding boxes in λ. The fifth column of the table contains the aspect ratios of each chip, which is the ratio of the chip's longer dimension to its shorter dimension. The mean aspect ratio is 1.65, while the median is 1.31; the mean is larger than the median primarily because of the one "outlying" design with the aspect ratio of 4.57. This data allows us to conclude that most designs are approximately square.

Having investigated the aspect ratio of the bounding box, we must now study its size (as a function of the number of rectangles on the chip). To do this, the last column of Table 5 shows the ratio of the area of the bounding box (measured in λ^2) to the number of the rectangles on the chip. Note that this

Chip #	# Rects	Chip Bounding Box DX	DY	Ratio	Area (in λ^2)/ # Rects
1	10803	848	1073	1.27	84.23
2	11265	2072	453	4.57	83.32
3	11566	840	800	1.05	58.10
4	11853	896	1088	1.21	82.24
5	11915	954	856	1.11	68.54
6	12063	856	910	1.06	64.54
7	12800	1155	630	1.83	56.85
8	14186	900	1258	1.40	79.80
9	14423	1272	657	1.94	57.94
10	15097	850	1110	1.31	62.50
11	16194	1190	1106	1.08	81.20
12	17565	605	1272	2.10	43.81
13	18056	1268	800	1.58	56.18
14	33387	1920	963	1.99	55.38
15	95901	2944	1676	1.76	51.45
16	109682	3019	2370	1.27	65.23

Table 5. Data on chip bounding box.

is the average amount of silicon "real estate" *devoted to* each rectangle, and not the *area of* each rectangle; the inverse of this figure gives the density of rectangles (in average number of rectangles per λ^2 area). This ratio remains remarkably consistent across all chips: the median ratio is 63.5, while the mean is 65.7 (with standard deviation 12.4).

The above discussion shows that the "typical" bounding box containing N rectangles is a square of approximately $8N^{1/2}\lambda$ on each side. The next subject to investigate is the placement of rectangles over the bounding box: are they spread uniformly over the area, or are they all clustered in a few dense parts of the chip? Histograms in Appendix V show that the rectangles are indeed spread uniformly over (most of) the bounding box.

2.4 A Probabilistic Model

The previous sections in this chapter have focussed on a statistical study of existing VLSI chips. For use in analyzing potential algorithms for VLSI tasks, we must condense that data into the more useful form of a probabilistic model. In this section we will study two such models (of many possible): one rather simple and the other somewhat complicated. Before discussing the particular models, though, we must address three topics in a more general context: the shape of the bounding box containing the chips, the placement of the rectangles within the bounding box, and the shape of the individual rectangles.

The first issue that a model of rectangles on chips must face is the bounding box in which the rectangles are distributed. The aspect ratios of Section 2.3 show that the bounding boxes are approximately square.[5] The ratio of chip area to the number of rectangles shows that the bounding box should have approximately $64\lambda^2$ area for each rectangle in the set.

The second issue a model must face is the distribution of rectangles over the bounding box. The data of Appendix V shows that the rectangles are indeed uniformly distributed over (most of) the chip's bounding box. This assumption can also be justified by *a priori* arguments based on VLSI design philosophy. One of the aims of a good VLSI design is to make effective use of the silicon area. This implies that the design will fill its bounding box rather uniformly. Although there are typically sparse regions near the edges of the design (containing long wires, bonding pads and pad drivers), the central part of the chip has a much higher (and almost uniform) density of small rectangles that perform most of the computation and data manipulation. The fixed number of layers available for a design (in nMOS there are typically six) and the design practice of not overlapping many rectangles on one layer combine to limit the number of rectangles that cover any given point on the chip; this together with the lower bound on rectangle size implies a constant upper bound on the density of rectangles. Economic arguments provide a lower bound: there is no reason to leave large blank spaces in a design.

The third issue to be faced is the shape of the individual rectangles. Fortunately, we have a great deal of data on rectangle shapes in Section 2.2 and Appendices II through IV.

We can now combine these facts regarding the size and shape of the bounding box, the placement of rectangles, and the shape of rectangles into a probabilistic model. The simplest possible model of an N-rectangle design is that the N rectangles are squares with edge length 7.6λ, uniformly distributed on $[0, 8N^{1/2}\lambda]^2$. (The value 7.6λ is from Table 1, and the 8λ is from Table 5.)

On the other hand, we could postulate a much more complex model for generating a set of N rectangles. We first choose an aspect ratio R uniformly from [1,2], and a sparsity S uniformly from [44,84]. (S is the total amount of silicon real estate per rectangle in the bounding box.) We then set the short side of the bounding box to be $(NS/R)^{1/2}\lambda$, and the long side to be $(RNS)^{1/2}\lambda$. We then distribute the following rectangles uniformly over the bounding box:

- (.72)N *components*, each with side lengths chosen uniformly on $[2\lambda, 10\lambda]^2$. (A more sophisticated method could sample a stored 10-by-10 table giving the probability of realizing each pair of lengths; such a table could be generated from the data in Appendix

[5] A number of engineering and economic reasons dictate that the chip's bounding box should be close to square. This often makes wire-bonding easier, and usually makes more efficient use of the package cavity. The authors regret to have to point out that in the designs we studied, there was motivation for "long and skinny" chips: rumors in the design community reported that such designs were more likely to be placed on multiproject chips! Most of their chips with extremely high aspect ratios were very small and therefore filtered by our criterion of examining only chips with at least 10,000 rectangles.

II.)

- (.26)N *wires*, half vertical and half horizontal, with the short side distributed uniformly between 2λ and 6λ and the long side distributed exponentially with mean 40λ (truncated below 10λ).

- (.02)N *other* rectangles, for which both edge-lengths are chosen uniformly on [10λ,Max], where Max is a typical bonding pad edge size (say, 75λ).

The percentages in the above model are from Table 1, the parameters of each rectangle class are from Tables 2, 3, and 4, and the bounding box distribution is from Table 5.

3. Applications of the Statistics

In this chapter we will consider a set of geometric subproblems that arise in layout, routing and design rule checking called the *geometric intersection problems*. In the abstract problem we are given a set of objects in the plane and must report all pairwise intersections among the objects. In concrete applications the geometric objects are usually components on a chip, and the intersections are potential "trouble areas", such as design rule violations or a place for a crossover.

Much previous work has been done on geometric intersection problems. Baird [1978], Lauther [1978], McCaw [1979], and Wilcox, Rombeek and Caughey [1978] all describe programs for computing geometric intersections that have actually been used in the design of masks. Although all of their programs are much faster than naive geometric intersection programs, they are still slower than desired and lack a solid theoretical basis. Theoretically sound algorithms have been given by Shamos [1978], Bentley and Ottmann [1979] and Bentley and Wood [1980], but they are very complex to code and fail to exploit many of the situations that arise in practice.

In this chapter we will study a particular geometric intersection problem that calls for intersecting rectangles. In Section 3.1 we develop a theoretical algorithm that performs very well for input rectangles drawn from the distributions studied in Chapter 2. Section 3.2 then shows how the theoretical algorithm can be efficiently implemented on the secondary storage media necessitated by the large size of current VLSI designs. We discuss the algorithm's potential for VLSI applications in Section 3.3.

3.1 An Algorithm for Finding Rectangle Intersections

In this section we will examine an algorithm for solving the following problem.

> *The Rectangle Intersection Problem* -- Given a set of N rectangles in the plane, each of which has sides parallel to the coordinate axes, report each intersecting pair of rectangles (by calling some procedure).

The operation of intersecting rectangles is fundamental in many VLSI tasks; we will return to these in Section 3.3. A straightforward way of solving the Rectangle Intersection Problem is to compare all $\binom{N}{2}$ pairs of rectangles. This method is very easy to code and efficient for small values of N, but the $O(N^2)$ running time is prohibitive for large designs. Bentley and Wood [1980] describe an algorithm for solving the problem in $O(N \lg N + k)$ worst-case time, where k is the number of intersecting pairs found. Their algorithm, however, is primarily a theoretical device, being difficult to code and having a large constant factor "hidden" in the O-notation. In this section we will investigate an algorithm that exploits the probabilistic models of Section 2.4 to yield an algorithm with expected running time proportional to N.

The algorithm we will study in this section exploits the probabilistic models of the last chapter to solve the rectangle intersection problem in fast expected time. Before studying that algorithm, it is important for us to summarize the salient points of the probabilistic models *from an algorithmic viewpoint*. The first important fact is the uniformity of the distribution of the rectangles: this uniformity facilitates the use of *bins* to store data. (Bins work very poorly for highly-clustered data, but have excellent performance for uniform distributions). Equally important is the small average edge length of a rectangle (7.6λ) compared to the average chip width ($8N^{1/2}\lambda$). We will find these measurements very helpful as we calculate the optimal size of a bin.

The algorithm for reporting all intersections among a set of rectangles is based on a bottom-to-top scan of the set of rectangles. At each time during the scan, all rectangles currently intersecting the horizontal scan line are represented as a set of one-dimensional line segments. When the bottom of a new rectangle is encountered, the rectangles it intersects are found by checking which line segments its segment intersects; its segment is then added to the set. When the top of a rectangle is found, its segment is deleted. In this way, only the rectangles currently intersecting the scan line need be kept "active" at one time, yet all intersecting pairs are correctly reported.

We can now describe the algorithm more precisely. The input is a set of rectangles, each of which is described by a unique name, four real numbers (giving the extreme left, right, bottom and top coordinates), and any other information needed for the VLSI application. The "output" is a call to procedure REPORT for every intersecting pair of rectangles. We will make use of two primary data structures. The EL (for "Event List") sequence contains two entries for each rectangle; one for its bottom edge and one for its top. This list is sorted by y-coordinate, in increasing order. The other data structure is the set of line segments, called SS for "Segment Set", that intersect the hypothetical scan line (we will discuss the implementation of SS later). With this background, the algorithm is described as follows.

Algorithm A

1. [Build the event list.]
    a. For each rectangle in the input set, create two records in the event list EL: one corresponding to the bottom, one to the top.
    b. Sort EL by increasing y-value. If various rectangles have equal y-values, then place the bottom edges *before* the top edges.

2. [Scan through the event list.]
    a. Initialize SS to represent the empty set.
    b. Scan through EL in increasing order by y-value. As each record R in EL is visited, take one of the following actions.
        i. If R represents the bottom of a rectangle, then first check the line segment corresponding to R (that is, the projection of the rectangle onto the x-axis) for intersection with any other segments in SS (reporting all intersecting pairs), and then insert that segment into SS.

ii. If R represents the top of a rectangle, then delete its segment from SS.

The correctness of Algorithm A is based on the fact that if any two rectangles intersect, then they must be "active" (that is, present in SS) at the same time and will therefore be reported in Step 2.b.i.

We will now briefly analyze the time required by Algorithm A. Step 1.a requires O(N) time, and straightforward implementations of the sort in Step 1.b require O(N lg N) time. We note that the time required to sort can be reduced to O(N) expected time by using the bin method described by Weide [1978]; we will return to the running time of this step in the next section. The initialization cost of Step 2.a is dependent on the implementation of SS, as are the costs of the N insertions and deletions both into and from SS in Steps 2.b.i and 2.b.ii. Our task now is to find an implementation of SS to facilitate rapid insertion and deletion of line segments.

We will implement the SS structure by dividing the portion of the x-axis on which the rectangles fall into a set of *bins*. For instance, if all x-values of rectangles lie between 0λ and 800λ, then we could have one hundred bins, each of width 8λ. The set of bins is implemented in a program as an array of pointers; each pointer points to a linked list of segment names that currently overlap that bin. This situation is illustrated in Figure 1. To insert a new segment into this structure, we merely insert its name into all bins overlapping the segment; to delete it, we traverse the same set of bins and delete the segment name from the linked list of each. To check what segments in SS a new segment X overlaps, we visit each bin in which X falls and compare X against all elements in the bin; careful bookkeeping allows us to be sure that no pair of segments is reported twice.

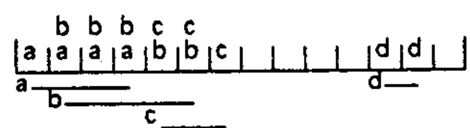


Figure 1. Segments represented in bins.

The final task we face in the implementation of SS is the specification of exactly how many bins there should be, or alternatively, the width of each bin (for given one, we can determine the other). It is at this point that we make crucial use of the data in Section 2.4: we choose the bin width to be (approximately) the average width of a rectangle, so that each segment is placed in two bins, on the average. By the models in Section 2.4, this implies that each bin has width of approximately 8λ, and the total number of bins is the square root of the number of rectangles. In this implementation, the initialization of the bins requires $O(N^{1/2})$ time; we will shortly show that insertion and deletion have only constant cost on the average. The expected total running time of Stage 2 is therefore O(N).

There is one fact that has yet to be established to prove the above claim of linear running time for Stage 2 of algorithm A. We require that the expected number of rectangles in a bin be a constant (independent of N), which implies that insertion and deletion have constant cost on the average. The empirical observations of the program's running time to be presented in the next section supports this claim. We can also show that the claim holds under the models of rectangle distribution and size discussed in Section 2.4. Consider first the simple model of an N-rectangle design: the N rectangles are squares with edge length 7.6λ, uniformly distributed on $[0, 8N^{1/2}\lambda]^2$; we must determine the expected number of rectangles in SS at any time. The probability that a given square intersects a given scan line is $(7.6\lambda)/(8N^{1/2}\lambda)$. (Because the square intersects the scan line defined by $y = S$ if the y value of the lower edge of the square falls within the range $[S-7.6\lambda, S]$). The expected number of squares intersecting the scan line at any given time is simply the total number of squares multiplied by the probability that an individual one intersects the scan line, which is $(7.6\lambda N^{1/2})/(8\lambda)$, or $.95N^{1/2}$. Each of the $N^{1/2}$ bins representing the SS has a width of 8λ; hence each square is entered into two (or possibly just one) bins. Since the squares are uniformly distributed, the bins are evenly filled and each bin contains approximately two squares; this establishes the claim. Now consider the more complicated model for generating the N rectangles which is based on the trichotomy of components, wires and others. The constant expected number of entries in each bin can be proved in a manner similar to the proof for the simple model above. Of critical importance is that (1) other rectangles have an upper bound on their edge length, and that (2) wires have a constant average length (although the observed distribution does not impose an *a priori* upper bound on the length of a wire.)

In preparation for the next section we present a quick summary of algorithm A, with a view toward implementation. The first stage consists primarily of a sort, and the second stage is a linear scan through the sorted output of Stage 1. This scan has the pleasant property that it can perform all necessary processing to report intersections using only $O(N^{1/2})$ storage. Note that the only communication necessary between the two stages of the algorithm is a sequential file giving the sorted event list.

3.2 A Pascal Program

In this section we will describe a set of three computer programs that together implement Algorithm A of the last section. Before going into the details of the individual programs, we will first give a brief overview of the system as a whole. The input to the system is a disk file containing the rectangles to be checked for pairwise intersection, and the "output" is a set of calls to a REPORT procedure (which performs some desired operation) reporting each intersecting pair of rectangles exactly once. Algorithm A is divided into two primary stages: Stage 1 creates a sorted "Event List" named EL, and Stage 2 scans through EL. The EL structure is implemented in this system as a (sorted) disk file. With this overview in mind, we can now describe the computer implementations of the two stages of the algorithm.

The purpose of Stage 1 is to create the EL file and then sort it. In the first substep (Step 1.a), a simple (40 line) Pascal program creates an unsorted version of the EL file by representing each rectangle in the input file twice: once for the bottom edge and once for the top. The records of this file are then sorted into increasing order by y-coordinate, with a secondary sort key chosen so that events representing bottom edges precede top edges for equal y-values. The sorting program to accomplish this in our system is a standard system sorting program. As mentioned in Section 3.1, we could use a linear expected-time sort based on bins (see Weide [1978]) but this would have been much more difficult to code. On the other hand, it is possible to avoid an explicit sort altogether by generating the rectangles in sorted order from the CIF hierarchical description: at any time, the symbol with the lowest bounding box is expanded first.

The second stage of Algorithm A scans through the sorted event list, performing operations on the segment structure SS. The Pascal program implementing this scan contains about 400 lines of code, of which only 220 are for the algorithm itself, while the remaining 180 are for testing and timing. The number of bins in SS was chosen to be equal to the square root of the number of rectangles, that is, $N^{1/2}$. By the data of Chapter 2, this implies an average bin width of approximately 8λ.

Extensive measurements of the time and space usage of the program were performed. The program corresponding to Step 1.a of the algorithm was I/O bound. Step 1.b was a standard system sort, which has been studied in great detail elsewhere (see, for example, Knuth [1973, Section 5.4]). We therefore concentrated our measurements on the time and space requirements of Stage 2 of the program.

The performance of the program implementing Stage 2 is summarized in Table 6. The Pascal programs were run on a DEC PDP-KL10 (ARPANET Host CMU-10A). The programs were executed on VLSI designs from recent Xerox PARC multiproject chips that contain between 4,000 and 12,000 rectangles.[6] The first row of the table describes the performance of the program on a chip containing 4689 rectangles. There were 219 elements in the segment set SS on the average, and SS was never larger than 475. The ratio of the maximum size of SS to the root of the number of rectangles is 6.94. The running time of the program was 8.9 seconds, and the amortized time per rectangle is 1.9 milliseconds. The number of rectangle intersections found was 14441, and the ratio of the chip's width to its height was 3.1.

As predicted, the running time of the program implementing Stage 2 of our algorithm is proportional to N: the time per rectangle is between 1.73 and 1.93 milliseconds. The rectangle sets used for these measurements include features from all layers of a chip design. Hence an average

[6] For historical reasons (the transfer of VLSI computing at CMU from a PDP-10 to a VAX-11-780), we report the running time of our programs on a set of earlier (and smaller) multiproject chip designs.

# Rects (= N)	Segment Set Size Mean	Max	$Max/N^{1/2}$	Time (sec)	Time/N (msec)	# Ints	Aspect Ratio (Hor/Vert)
4689	219	475	6.94	8.9	1.9	14441	3.1
4960	86	172	2.44	8.6	1.73	11783	1.3
5272	111	246	3.39	9.3	1.76	14441	1.5
7176	150	359	4.24	12.9	1.89	18949	1.3
7416	123	254	2.95	14.3	1.93	22166	.73
11316	55.8	124	1.17	20.7	1.83	30565	.22

Table 6. Measurements for Stage 2 of Algorithm A.

rectangle intersects a relatively large number of other rectangles; in our case this number lies between 3.5 and 6. Many applications need intersections of rectangles only from a subset of the chip layers, and the algorithm will perform even better in that case: the number of intersections per rectangle will be lower, and the SS can be searched more rapidly.

The total number of memory words used in Stage 2 is proportional to the number of bins plus the *maximum* number of rectangles ever in SS at one time. The fourth column of Table 6 shows that this is bounded above by approximately $4RN^{1/2}$ in practice, where R is the aspect ratio of the chip. (The simple probabilistic model of Section 2.4 predicts $RN^{1/2}$; the observed increase is due to the local clustering in real designs not accounted for in our model.) This has a very pleasant implication for the main memory utilization of the algorithm: note that if N is one million and the aspect ratio is unity, then at most four thousand rectangles are ever present in main memory at the same time!

Our program was designed to scan the chip from its bottom to its top, and this led to severe space inefficiency for chips that were "short and fat". Because the rectangles are distributed uniformly over the chip, the number of rectangles intersecting the scan line will be proportional to the line's length, which is less if the chip is scanned in the "tall and skinny direction". (Note that the length of the scan line is less by a factor of the square of the aspect ratio.) Recall that the space used by the algorithm is directly proportional to the number of rectangles intersecting the scan line. This analysis, together with the data of Table 6, suggests strongly that design rule checkers should be designed to "rotate" their inputs, so the chips are always scanned in the beneficial direction.

3.3 Applications in VLSI Tasks

VLSI technology will soon allow several hundred thousand devices in a single IC design. The lowest common denominator for these designs is the geometric specification of the shapes on the mask layers. *Design rule checking* is the process of determining whether certain interrelationships are maintained between those shapes; for example, whether all unrelated polysilicon and diffusion lines are separated by at least 1λ. Design rule checking programs are the "syntax checkers" of the IC world -- they guarantee that the form of the chip is correct (but not, of course, that it does what the designer intended).

Algorithm A of Section 3.1 can be used in design rule checking. Most minimum width, minimum clearance, and enclosure checks can be performed by the combination of programs to expand/shrink rectangles, perform logical operations (for example, forming the logical OR of two layers to produce a third), and an intersection reporter. Such systems are essentially batch oriented; an entire design file is checked at once and all design rule violations reported. The fast rectangle intersector is the workhorse here, processing a large number of rectangles in a single pass. Note that the time required by the rectangle intersector is linear in the number of rectangles, and its space requirements are proportional to the square root of the number of rectangles.[7]

While batch-oriented geometric design rule checkers are in wide use, interactive systems can provide valuable feedback as designs are entered. In an operator-guided placement and routing system, violations can be automatically flagged as each cell is placed. It may also be possible to hide a considerable amount of computation in the operator's "think" time. Straightforward extensions of the algorithm above make such *incremental* checks easy. The chip can be divided into two-dimensional bins, where a rectangle is put into all the bins it overlaps. Once these bins have been constructed, it is simple to find which existing rectangles intersect a given new rectangle. The data of Chapter 2 and Appendix V ensures that this approach will be very fast: the shape data tells us that rectangles are small (and will therefore fall into few bins), and the uniformity of the placement data tells us that only relatively few rectangles will fall into any particular bin.

[7] As appealing as a fast geometric design rule checker may be, we believe that faster design rule checkers will take advantage of the high-level information in hierarchical IC designs. Such designs consist of cells that implement a particular function, connected together by some amount of wiring. Each cell in turn consists of an agglomeration of cells, components, and wires; the lowest-level cells are merely collections of components and wires. Performance improvements may be achieved by taking advantage of the structure in the design. For instance, it is necessary to check any particular cell only once -- subsequent placements of the cell need be checked only if they overlap other shapes. By intersecting the bounding box of a cell before "exploding" the cell into its constituent parts, a quick yes/no determination can be made of whether anything within the cell might intersect previously placed cells. A single intersection check can therefore eliminate many rectangles from consideration. The fast intersection algorithm might be useful for such geometric manipulations.

4. Conclusions

There are two main contributions in the work of the previous chapters. The first contribution is the program for finding geometric intersections: it is easily coded, very fast, and space-efficient. Preliminary comparisons indicate that it is much faster than previous programs for similar tasks. The second, and more fundamental, contribution of this work is the data on the distribution of chip components and the methodology for using such data to design fast programs. We feel that the techniques we have used here will prove to be of broad applicability in constructing design aids for VLSI systems.

A great deal of further work remains to be done. The algorithm of Chapter 3 is being used in a design rule checker currently under development at Carnegie-Mellon. An important open problem is to gather more data on VLSI designs: two particularly interesting questions are whether the conclusions we drew from this data will "scale up" to designs an order-of-magnitude larger, and how our data will compare with industrial chips (as opposed to the Mead-Conway style designs). An interesting problem in VLSI design is to employ more knowledge than the strictly geometric issues that we have studied here; it is therefore important to study the hierarchical structure inherent in the CIF representation of a VLSI design.

Acknowledgements

We are grateful to the chip designers mentioned in Appendix I for allowing their designs to be used in this study, and to Bill Eddy and Bob Sproull for their helpful comments.

References

Baird, H. S. [1978]. "Fast algorithms for LSI artwork analysis," *Journal of Design Automation and Fault-Tolerant Computing* 2, 2, pp. 179-209.

Bechtolsheim, A. and T. Gross [1980], "A Parallel Search Table for Logarithmic Arithmetic," paper presented at the M.I.T. Conference on Advanced Research in Integrated Circuits, January, 1980.

Bentley, J. L. and T. Ottmann [1979]. "Algorithms for reporting and counting geometric intersections," *IEEE Transactions on Computers C-28*, 9, pp. 643-647, September 1979.

Bentley, J. L. and D. Wood [1980]. "An optimal worst-case algorithm for reporting intersections of rectangles," to appear in *IEEE Transactions on Computers*.

Conway, L. , A. Bell, M. Newell, R. Lyon, and R. Pasco [1980]. Implementation Documentation for the MPC79 Multi-University Multiproject Chip-Set, Xerox PARC Report, January 1980.

Clark, J.H. [1980], "A VLSI Geometry Engine for Computer Graphics," paper presented at the M.I.T. Conference on Advanced Research in Integrated Circuits, January, 1980.

Conway, L., A. Bell, M. Newell, R. Lyon, and R. Pasco [1980], "Implementation Documentation for the MPC79 Multi-University Multiproject Chip-Set," Xerox PARC Report, January, 1980.

Holloway, J., G. Steele, G.J. Sussman, and A. Bell [1980], "The SCHEME-79 Chip," paper presented at the M.I.T. Conference on Advanced Research in Integrated Circuits, January, 1980.

Hon, R. W. and C. H. Sequin [1980]. *A Guide to LSI Implementation*, 2nd edition, Xerox PARC TR SSL 79-7, Chapter 7, January 1980.

Lauther, U. [1978]. "4-dimensional binary search trees as a means to speed up associative searches in design rule verification of integrated circuits," *Journal of Design Automation and Fault-Tolerant Computing* 2, 3, pp. 241-247.

McCaw, C. R. [1979]. "Unified shapes checker -- a checking tool for LSI," *Proceedings of the Sixteenth Design Automation Conference*, pp. 81-87, June 1979, ACM.

Mead, C. A. and L. Conway [1980]. *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA., 1980.

Shamos. M. I. [1978]. "Problems in computational geometry," unpublished Ph.D. Thesis, Yale University, 1978.

Weide, B. W. [1978]. "Statistical methods in algorithm design and analysis", Ph.D. Thesis, Carnegie-Mellon University, August 1978.

Wilcox, P., H. Rombeek, and D. M. Caughey [1978]. "Design rule verification based on one dimensional scans", *Proceedings of the Fifteenth Design Automation Conference*, June 1978, ACM.

I. Description of the Chips

Most of the following chips were designed as part of a VLSI course given at Stanford University by Rob Mathews and John Newkirk. The designs were implemented on a multiproject chip (MPC79) managed by Xerox PARC, with data communication provided by the ARPANET, masks by Micro Mask, Inc., and wafer fabrication by Hewlett-Packard. (For more details see Conway et. al. [1980].) In several cases, papers are available that describe the design in detail. Four of the projects have not yet been implemented, as they were not completed by the final MPC79 deadline. Those projects will probably be implemented on future multiproject chips.

1. Bill Frolik and Roderick Young -- Digital Timer.

2. Rob Mathews and John Newkirk -- Firecode Chip.

3. Huang -- Not implemented.

4. Forest Baskett -- Ethernet Synchronizer.

5. Marc Hannah and Peter Eichenberger -- Rectangle Generator.

6. Mike Tarsi and Nagatsugu Yamanouchi -- Multifunction Digital Clock.

7. David Noice and Neil Midkiff -- Multiplier/Divider.

8. Sytwu -- Not implemented.

9. Gluss -- Not implemented.

10. Redford -- Not implemented.

11. Jim Clark -- Graphics ALU (see Clark [1980]).

12. Andy Bechtolsheim and Thomas Gross -- Parallel Search Table for Log Arithmetic (see Bechtolsheim and Gross [1980]).

13. Matt Herndon and Jeff Thorson -- Typesetting Machine.

14. Synth -- Part of a digital speech synthesizer designed by Jim Cherry of MIT.

15. Filters -- Part of a speech recognition system designed by Dick Lyon at Xerox PARC.

16. SChip2 -- The second version of a Lisp microprocessor designed by Sussman, Steele, and Holloway from MIT and Bell from Xerox PARC (see Holloway, et. al. [1980]).

II. Component Histograms

In this appendix we will present histograms that give more detail on the distribution of components. The histograms are contained in Figures 1 through 6; they describe chips 11 through 16, respectively. Figure 1 describes chip 11. The first three lines of that figure summarize data from Table 2 of Section 2.2.2. The next part of the figure gives a two-dimensional histogram of the component distribution. For instance, there were 1059 2λ-by-4λ components (that is, 2λ wide by 4λ high). The five x's in that box indicate that these 1059 components are approximately half the number of the maximum number of components in any of the one hundred sizes (that is, the 2125 4λ-by-4λ components, which have ten x's). The final thirteen lines in the figure "linearize" this two-dimensional data by summarizing the longest edge of the components. For instance, 2139 components had a longest edge of exactly 6λ (or 19.2% of all components), and 81.2% of the components had both edges less than or equal to 6λ in length.

A number of facts are readily apparent from these tables. The first is that there are few components that have an edge length of 1λ. Since a 1λ edge violates Mead and Conway's design rules, all such components were in fact part of logos. The second obvious fact is that most components were 2λ-by-2λ, 2λ-by-4λ, 4λ-by-4λ, or 4λ-by-6λ (or rotations of those by 90°). Finally, we note that most component edge lengths were less than or equal to 6λ. From the summaries at the bottom of the figures, we see that on all chips, 78% or more of the components had longest edge less than or equal to 6λ. Of the remaining 22% with one edge greater than 6λ, very few had both edges greater than 6λ (observe that the upper right 4λ-by-4λ rectangle of each histogram is very sparsely populated). This implies that even if we had changed our definition of component to all rectangles with both edges less than or equal to 6λ, our results would have changed little.

16194 rectangles
11128 components (68.7% of total)
Component edge length : Mean 4.0, Standard deviation 2.0

Distribution of component sizes

```
     |
  10 |           x     x     x           x     x                 x
     |         132     1    31          12     1                 1
     |
   9 |           x           x           x                 x
     |         176          86          16                 2
     |
   8 |           x     x     x           x           x
     |          48     3   116         314           3
     |
   7 |           x     x     x           x     x
     |         133     7    14           4    23
     |                       x
   6 |           x     x    xx     x     x     x     x     x     x
     |         299     2   558    26   311     7   133    28    15
     |
   5 |           x     x     x     x     x
     |         171     2    40    31    16
     |          xx     x xxxxx           x
   4 |         xxx     x xxxxx     x    xx     x     x     x     x
     |        1059   331  2125    38   625    36    38    77    12
     |
   3 |           x     x     x     x     x     x     x     x     x
     |          22    40   295     3     1     6     1     2     1
     |        xxxx          xx
   2 |        xxxx     x    xx     x     x     x     x     x     x
     |        1707    47   790   164   301   146    96   307    64
     |
   1 |           x
     |          32
      ------------------------------------------------------------
           1     2     3     4     5     6     7     8     9    10
```

Component size breakdown

Longest edge	Count		Cumulative
1	0	0.0%	0.0%
2	1739	15.6%	15.6%
3	109	1.0%	16.6%
4	4600	41.3%	57.9%
5	449	4.0%	62.0%
6	2139	19.2%	81.2%
7	376	3.4%	84.6%
8	752	6.8%	91.3%
9	694	6.2%	97.6%
10	270	2.4%	100.0%

Figure 1. Components of chip 11.

17565 rectangles
10219 components (58.2% of total)
Component edge length : Mean 3.5, Standard deviation 1.6

Distribution of component sizes

```
      |
   10 |           x           x
      |          44          24
      |
    9 |           x           x           x
      |           1          10          20
      |
    8 |           x           x           x           x
      |           5          11          36          20
      |
    7 |           x           x                                   x
      |          68          66                                  66
      |
    6 |           x     x     x     x     x           x     x
      |          23     1   292    22     2          26    11
      |
    5 |           x     x     x     x     x
      |          86     2     1     4    20
      |           x       xxxxx
    4 |          xx     x xxxxx     x     x           x           x
      |         962   233  3138    12   222          90          10
      |
    3 |           x     x     x     x     x     x           x     x
      |           6    17   177    11    11     3           4     2
      |        xxxx          xx
    2 |        xxxx     x    xx     x     x     x     x     x     x
      |        2583   115  1193    60   134    97    88    44   143
      |
    1 |           x
      |           2
      -------------------------------------------------------------
            1     2     3     4     5     6     7     8     9    10
```

Component size breakdown

Longest edge	Count		Cumulative
1	0	0.0%	0.0%
2	2586	25.3%	25.3%
3	138	1.4%	26.7%
4	5703	55.8%	82.5%
5	176	1.7%	84.2%
6	727	7.1%	91.3%
7	234	2.3%	93.6%
8	276	2.7%	96.3%
9	90	.9%	97.2%
10	289	2.8%	100.0%

Figure 2. Components of chip 12.

18056 rectangles
13731 components (76.0% of total)
Component edge length : Mean 3.9, Standard deviation 1.8

Distribution of component sizes

```
      |
   10 |          x           x           x
      |        295           4         128
      |
    9 |          x           x           x
      |         44           2          90
      |
    8 |          x           x           x
      |         74          18         233
      |
    7 |          x           x
      |        303           1
      |
    6 |          x           x     x     x           x     x     x
      |         96         509     2    39          33    46     8
      |
    5 |          x           x           x
      |         90           1           8
      |                  xxxxx
    4 |    x     x     x xxxxx           x     x     x     x     x
      |   28   705   538  4862         580    28   197     4     9
      |
    3 |          x           x
      |         28         436
      |          x           x
    2 |    x    xx     x     x     x     x     x     x     x     x
      |   29  1700    96  1083   134   354   129   517    35   159
      |
    1 |                      x
      |                     56
      -------------------------------------------------------------
           1     2     3     4     5     6     7     8     9    10
```

Component size breakdown

Longest edge	Count		Cumulative
1	0	0.0%	0.0%
2	1729	12.6%	12.6%
3	124	.9%	13.5%
4	7708	56.1%	69.6%
5	225	1.6%	71.3%
6	1588	11.6%	82.8%
7	461	3.4%	86.2%
8	1072	7.8%	94.0%
9	221	1.6%	95.6%
10	603	4.4%	100.0%

Figure 3. Components of chip 13.

33387 rectangles
25431 components (76.2% of total)
Component edge length : Mean 3.7, Standard deviation 2.0

Distribution of component sizes

```
      |
   10 |    x       x               x                                       x
      |   58      47             590                                       2
      |
    9 |    x       x               x               x       x       x
      |  177      31              51              25       8       1
      |
    8 |    x       x               x               x
      |  156      70              43              77
      |
    7 |    x       x               x
      |   65     294              32
      |
    6 |    x       x               x               x               x       x       x
      |   50     393            1098             132            1062       1       1
      |
    5 |    x       x       x       x                               x
      |   28     233       2     168                              21
      |                    x      xx               x
    4 |    x       x       x     xxx       x       x               x       x       x
      |   39     884    1187    3825       1    1782              85       4      46
      |
    3 |    x       x       x       x               x
      |   12     670      17     870               1
      |          xxxxx
    2 |    x     xxxxx     x       x       x       x       x       x       x       x
      |    8    7569     347    1097     579      93     146     265     770     204
      |
    1 |    x       x               x
      |    2      11               1
       --------------------------------------------------------------------------------
           1       2       3       4       5       6       7       8       9      10
```

Component size breakdown

Longest edge	Count		Cumulative
1	2	.0%	.0%
2	7588	29.8%	29.8%
3	1046	4.1%	34.0%
4	7903	31.1%	65.0%
5	1011	4.0%	69.0%
6	3549	14.0%	83.0%
7	537	2.1%	85.1%
8	1779	7.0%	92.1%
9	1068	4.2%	96.3%
10	948	3.7%	100.0%

Figure 4. Components of chip 14.

95901 rectangles
76116 components (79.4% of total)
Component edge length : Mean 4.0, Standard deviation 2.1

Distribution of component sizes

```
   |
10 |           x     x     x     x           x     x           x
   |        1597     2  1506     4           1    29           2
   |
 9 |           x     x     x           x     x           x
   |         433    37   195          56    36        1392
   |
 8 |           x     x     x           x     x     x
   |        1108    31    59         255    29     4
   |
 7 |           x     x     x           x     x     x           x
   |        2408     8    78          59  1420     7           1
   |                       x
 6 |           x     x     x     x     x     x     x     x     x
   |        2013    29  2811     1   214     9  1449   289     1
   |                             x
 5 |           x           x     x     x           x
   |         869         247  2864     2          36
   |           x     x xxxxx           x
 4 |           x     x xxxxx     x    xx     x     x     x     x
   |        3357  3365 16059    52  4092     1   260   208    98
   |
 3 |           x     x     x           x           x           x
   |         293   127  2129          58          25           2
   |        xxxx           x
 2 |     x xxxxx     x     x     x     x     x     x     x     x
   |    14 14947   576  3898  1230   491   493   999  1040   683
   |
 1 |           x                                   x
   |           4                                  24
   ------------------------------------------------------------
         1     2     3     4     5     6     7     8     9    10
```

Component size breakdown

Longest edge	Count		Cumulative
1	0	0.0%	0.0%
2	14965	19.7%	19.7%
3	996	1.3%	21.0%
4	28808	37.8%	58.8%
5	5262	6.9%	65.7%
6	9711	12.8%	78.5%
7	4476	5.9%	84.4%
8	4286	5.6%	90.0%
9	3686	4.8%	94.8%
10	3926	5.2%	100.0%

Figure 5. Components of chip 15.

```
109682 rectangles
73939 components (67.4% of total)
Component edge length : Mean 3.8, Standard deviation 1.7

Distribution of component sizes
     |
  10 |        x     x     x           x     x
     |     1193     2   183          33     1
     |
   9 |        x     x     x           x
     |      937     2    83          64
     |
   8 |  x     x     x     x           x
     |  9   550    49   189         395
     |
   7 |        x     x     x           x           x           x
     |      369    34   489          49          34          49
     |
   6 |        x     x     x     x     x     x     x     x     x
     |      441     4  3136   135   425     1   365   342   312
     |
   5 |        x           x           x
     |      550         414           1
     |                xxxxx
   4 |        x     x xxxxx     x     x     x     x     x     x
     |     2448   999 31891   128  2004    64   735   174   249
     |
   3 |        x     x     x     x     x     x     x     x     x
     |     1363     8   881    18    48     1    51    78    28
     |       xx
   2 |  x    xx     x     x     x     x     x     x     x     x
     | 74 13954   512  2860   333  1126   294   683  1557   522
     |
   1 |        x           x
     |       14           2
 ------------------------------------------------------------------
     1     2     3     4     5     6     7     8     9    10
```

```
Component size breakdown
Longest        Count        Cumulative
 edge
   1          0    0.0%         0.0%
   2      14042   19.0%        19.0%
   3       1883    2.5%        21.5%
   4      39081   52.9%        74.4%
   5       1443    2.0%        76.3%
   6       7320    9.9%        86.2%
   7       1301    1.8%        88.0%
   8       3060    4.1%        92.1%
   9       3237    4.4%        96.5%
  10       2572    3.5%       100.0%
```

Figure 6. Components of chip 16.

III. Wire Histograms

The histograms of this appendix give details on the distributions of the long edges of the rectangles classified as wires. Figures 7 through 12 give the data for chips 11 through 16. The first five lines of each figure repeat data from Table 3 of Section 2.2.3. The remaining part of the figure is a histogram of the distribution of long edges, giving both the absolute number in each range, and the cumulative percentages of all wires. For example, on the wires of chip 11 (Figure 7) there were 111 long edges between 30λ and 34λ, and 77.9% of all long edges were less than or equal to 34λ.

The following table summarizes the distribution of wires on the six chips studied in this appendix.

Chip #	< 25λ	< 50λ	< 100λ	< 150λ	< 195λ
11	68.1	82.8	96.9	98.5	98.9
12	95.0	97.7	99.9	99.9	99.9
13	68.2	95.1	98.5	99.3	99.5
14	60.0	90.1	97.2	98.9	99.1
15	69.1	87.2	97.0	99.2	99.5
16	84.9	96.4	98.2	99.1	99.3

This says that 68.1% of the wires on chip 11 had a long edge less than 25λ, and 98.9% had a long edge of less than 195λ. To summarize the above data, the "typical" chip has 70% of its edges less than 25λ in length, 90% less than 50λ, 97% less than 100λ, and 99% less than 150λ.

The above summary shows that the number of wires with long edges decreases very rapidly. As a first approximation, we might assume that the wire lengths are distributed exponentially; however, there is too much "activity" in the tails of the distributions for this to be the case. Determination of the exact character of this distribution remains an open problem.

```
16194 rectangles
4668 wires (28.8% of total)
2565 vertical, 2103 horizontal
Wire width  : Mean 3.0, Standard deviation 1.2
Wire length : Mean 32.1, Standard deviation 40.5, Max 791

Distribution of wire lengths
  length  count  % of
                 total
----------------------
  0 -   4     0    0.0
  5 -   9     0    0.0
 10 -  14  1198   25.7 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 15 -  19  1149   50.3 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 20 -  24   832   68.1 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 25 -  29   345   75.5 xxxxxxxxxxxxxx
 30 -  34   111   77.9 xxxx
 35 -  39    87   79.7 xxx
 40 -  44   106   82.0 xxxx
 45 -  49    35   82.8 x
 50 -  54    36   83.5 x
 55 -  59    64   84.9 xx
 60 -  64    22   85.4 x
 65 -  69    46   86.4 x
 70 -  74   287   92.5 xxxxxxxxxxx
 75 -  79    85   94.3 xxx
 80 -  84    16   94.7 x
 85 -  89     8   94.8 x
 90 -  94    36   95.6 x
 95 -  99    40   96.5 x
100 - 104    22   96.9 x
105 - 109    41   97.8 x
110 - 114     4   97.9 x
115 - 119     4   98.0 x
120 - 124     2   98.0 x
125 - 129     2   98.1 x
130 - 134     3   98.1 x
135 - 139     1   98.2 x
140 - 144     9   98.4 x
145 - 149     5   98.5 x
150 - 154     2   98.5 x
155 - 159     2   98.5 x
160 - 164     2   98.6 x
165 - 169     6   98.7 x
170 - 174     4   98.8 x
175 - 179     2   98.8 x
180 - 184     0   98.8
185 - 189     2   98.9 x
190 - 194     0   98.9
195 - max    52  100.0 xx
```

Figure 7. Wires of chip 11.

17565 rectangles
6722 wires (38.3% of total)
3598 vertical, 3124 horizontal
Wire width : Mean 3.2, Standard deviation 1.1
Wire length : Mean 17.0, Standard deviation 10.4, Max 322

Distribution of wire lengths

length	count	% of total	
0 - 4	0	0.0	
5 - 9	0	0.0	
10 - 14	3521	52.4	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
15 - 19	2309	86.7	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
20 - 24	558	95.0	xxxxxxx
25 - 29	120	96.8	x
30 - 34	28	97.2	x
35 - 39	11	97.4	x
40 - 44	10	97.5	x
45 - 49	10	97.7	x
50 - 54	31	98.2	x
55 - 59	0	98.2	
60 - 64	0	98.2	
65 - 69	22	98.5	x
70 - 74	11	98.6	x
75 - 79	45	99.3	x
80 - 84	0	99.3	
85 - 89	0	99.3	
90 - 94	34	99.8	x
95 - 99	11	100.0	x
100 - 104	0	100.0	
105 - 109	0	100.0	
110 - 114	0	100.0	
115 - 119	0	100.0	
120 - 124	0	100.0	
125 - 129	0	100.0	
130 - 134	0	100.0	
135 - 139	0	100.0	
140 - 144	0	100.0	
145 - 149	0	100.0	
150 - 154	0	100.0	
155 - 159	0	100.0	
160 - 164	0	100.0	
165 - 169	0	100.0	
170 - 174	0	100.0	
175 - 179	0	100.0	
180 - 184	0	100.0	
185 - 189	0	100.0	
190 - 194	0	100.0	
195 - max	1	100.0	x

Figure 8. Wires of chip 12.

```
18056 rectangles
4132 wires (22.9% of total)
2381 vertical, 1751 horizontal
Wire width  : Mean 3.2, Standard deviation 1.4
Wire length : Mean 26.8, Standard deviation 23.5, Max 303

Distribution of wire lengths
  length   count  % of
                  total
----------------------
  0 -   4      0   0.0
  5 -   9      0   0.0
 10 -  14    953  23.1 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 15 -  19   1234  52.9 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 20 -  24    630  68.2 xxxxxxxxxxxxxxxxxxxxxxxxx
 25 -  29    215  73.4 xxxxxxxx
 30 -  34    289  80.4 xxxxxxxxxxx
 35 -  39     44  81.4 x
 40 -  44     12  81.7 x
 45 -  49    552  95.1 xxxxxxxxxxxxxxxxxxxxxx
 50 -  54     13  95.4 x
 55 -  59     16  95.8 x
 60 -  64      4  95.9 x
 65 -  69     22  96.4 x
 70 -  74      9  96.6 x
 75 -  79     38  97.6 x
 80 -  84      8  97.7 x
 85 -  89      4  97.8 x
 90 -  94     12  98.1 x
 95 -  99     14  98.5 x
100 - 104     22  99.0 x
105 - 109      0  99.0
110 - 114      1  99.0 x
115 - 119      1  99.1 x
120 - 124      1  99.1 x
125 - 129      2  99.1 x
130 - 134      2  99.2 x
135 - 139      2  99.2 x
140 - 144      0  99.2
145 - 149      2  99.3 x
150 - 154      2  99.3 x
155 - 159      2  99.4 x
160 - 164      1  99.4 x
165 - 169      1  99.4 x
170 - 174      1  99.4 x
175 - 179      2  99.5 x
180 - 184      0  99.5
185 - 189      0  99.5
190 - 194      0  99.5
195 - max     21 100.0 x
```

Figure 9. Wires of chip 13.

33387 rectangles
7706 wires (23.1% of total)
3078 vertical, 4628 horizontal
Wire width : Mean 3.2, Standard deviation 1.5
Wire length : Mean 28.9, Standard deviation 40.5, Max 1284

Distribution of wire lengths

length	count	% of total	
0 - 4	0	0.0	
5 - 9	0	0.0	
10 - 14	3456	44.8	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
15 - 19	586	52.5	xxxxxxxx
20 - 24	582	60.0	xxxxxxxx
25 - 29	779	70.1	xxxxxxxxxxx
30 - 34	114	71.6	x
35 - 39	601	79.4	xxxxxxxx
40 - 44	537	86.4	xxxxxxx
45 - 49	290	90.1	xxxx
50 - 54	255	93.4	xxx
55 - 59	91	94.6	x
60 - 64	35	95.1	x
65 - 69	20	95.3	x
70 - 74	10	95.5	x
75 - 79	77	96.5	x
80 - 84	9	96.6	x
85 - 89	12	96.7	x
90 - 94	27	97.1	x
95 - 99	8	97.2	x
100 - 104	33	97.6	x
105 - 109	38	98.1	x
110 - 114	25	98.4	x
115 - 119	1	98.4	x
120 - 124	1	98.5	x
125 - 129	23	98.8	x
130 - 134	0	98.8	
135 - 139	3	98.8	x
140 - 144	5	98.9	x
145 - 149	2	98.9	x
150 - 154	1	98.9	x
155 - 159	1	98.9	x
160 - 164	2	98.9	x
165 - 169	3	99.0	x
170 - 174	0	99.0	
175 - 179	1	99.0	x
180 - 184	0	99.0	
185 - 189	1	99.0	x
190 - 194	7	99.1	x
195 - max	70	100.0	x

Figure 10. Wires of chip 14.

```
95901 rectangles
18881 wires (19.7% of total)
6773 vertical, 12108 horizontal
Wire width  : Mean 3.1, Standard deviation 1.3
Wire length : Mean 28.3, Standard deviation 40.2, Max 1771

Distribution of wire lengths
  length   count  % of
                  total
----------------------
  0 -   4      0   0.0
  5 -   9      0   0.0
 10 -  14   6584  34.9 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 15 -  19   3909  55.6 xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 20 -  24   2562  69.1 xxxxxxxxxxxxxxxxxxx
 25 -  29    860  73.7 xxxxxx
 30 -  34    634  77.1 xxxx
 35 -  39   1356  84.2 xxxxxxxxxx
 40 -  44    478  86.8 xxx
 45 -  49     90  87.2 x
 50 -  54   1018  92.6 xxxxxxx
 55 -  59    145  93.4 x
 60 -  64    110  94.0 x
 65 -  69     50  94.3 x
 70 -  74      8  94.3 x
 75 -  79    365  96.2 xx
 80 -  84     52  96.5 x
 85 -  89      6  96.5 x
 90 -  94     72  96.9 x
 95 -  99     17  97.0 x
100 - 104     91  97.5 x
105 - 109    145  98.3 x
110 - 114     60  98.6 x
115 - 119      6  98.6 x
120 - 124      3  98.6 x
125 - 129    100  99.2 x
130 - 134      2  99.2 x
135 - 139      6  99.2 x
140 - 144      0  99.2
145 - 149      7  99.2 x
150 - 154      0  99.2
155 - 159      0  99.2
160 - 164      1  99.2 x
165 - 169      1  99.2 x
170 - 174      0  99.2
175 - 179      1  99.2 x
180 - 184      4  99.3 x
185 - 189      1  99.3 x
190 - 194     35  99.5 x
195 - max    102 100.0 x
```

Figure 11. Wires of chip 15.

```
109682 rectangles
34712 wires (31.6% of total)
17838 vertical, 16874 horizontal
Wire width  : Mean 2.9, Standard deviation 1.1
Wire length : Mean 22.2, Standard deviation 45.6, Max 1523

Distribution of wire lengths
  length  count  % of
                 total
----------------------
   0 -   4      0   0.0
   5 -   9      0   0.0
  10 -  14  26265  75.7 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  15 -  19   1261  79.3 xx
  20 -  24   1954  84.9 xxx
  25 -  29   1300  88.7 xx
  30 -  34    454  90.0 x
  35 -  39     63  90.2 x
  40 -  44   1765  95.2 xxx
  45 -  49    389  96.4 x
  50 -  54     93  96.6 x
  55 -  59    128  97.0 x
  60 -  64     97  97.3 x
  65 -  69    116  97.6 x
  70 -  74     20  97.7 x
  75 -  79     18  97.7 x
  80 -  84     22  97.8 x
  85 -  89     13  97.8 x
  90 -  94    123  98.2 x
  95 -  99     12  98.2 x
 100 - 104     15  98.3 x
 105 - 109      5  98.3 x
 110 - 114    128  98.6 x
 115 - 119      5  98.7 x
 120 - 124      5  98.7 x
 125 - 129    105  99.0 x
 130 - 134      7  99.0 x
 135 - 139     10  99.0 x
 140 - 144      9  99.0 x
 145 - 149     11  99.1 x
 150 - 154      8  99.1 x
 155 - 159     10  99.1 x
 160 - 164     11  99.2 x
 165 - 169     10  99.2 x
 170 - 174      8  99.2 x
 175 - 179     11  99.2 x
 180 - 184      8  99.3 x
 185 - 189      5  99.3 x
 190 - 194      5  99.3 x
 195 - max    243 100.0 x
```

Figure 12. Wires of chip 16.

IV. Histograms of Other Rectangles

The histograms of this appendix describe the rectangles classified as "others". Figures 13 through 18 describe chips 11 through 16, respectively. The first three lines of each figure duplicate information from Table 4 of Section 2.2.4. The remaining part of the figure is a histogram summarizing the sizes of (most of) the other rectangles. For instance, Figure 13 shows that chip 11 had 15 rectangles with width from 70λ to 79λ and height between 10λ and 19λ; the x's play the same role as in Appendix II. The number of rectangles that were truncated is noted on each figure (that is, some of the rectangles really had edges from 100λ to 109λ, and longer edges were truncated to that length).

Note that most of the other rectangles are clustered in the lower left corner of the histograms. Relatively few rectangles were truncated; chips 14 and 16 both had one-sixth of their rectangles truncated, all the rest had less than ten percent truncated. In each histogram one can observe two clusters of large squares; in Figure 13 they have sides between 40λ and 50λ between 50λ and 60λ. These clusters correspond to the bonding pads of each design: there is a large square for the large metal pad itself, and a slightly smaller square for the overglassing cut. Chip 11 has 37 pads, chip 12 has 24, chip 13 has 20, chip 14 has 16, chip 15 has 38, and chip 16 has 49.

These figures underline the robustness of the trichotomy presented in Section 2.2.1. We defined wires as those rectangles with one edge greater than 10λ and the other edge less than or equal to 6λ. The figures of this section allow us to see what would change if we said that the short edge of a rectangle could be up to 9λ in length; the other rectangles that would become wires can be found in the leftmost columns and bottommost rows of Figures 13 through 18. The chip most dramatically affected is chip 11; the number of wires increases by approximately 4.3%. On all other chips, the percentage change imposed by this redefinition is less than 4% (and only 0.7% on chip 16).

```
16194 rectangles
398 others (2.5% of total)
Other edge length : Mean 44.3, Standard deviation 75.8, Max 968

Distribution of other rectangle's sizes
19 truncated edges
     |  xxx
max  |  xxx     x     x     x
     |   24     3     5     1
     |
 99  |
     |
     |   xx
 89  |   xx
     |   16
     |    x     x     x
 79  |    x    xx     x
     |   10    11     8
     |
 69  |    x     x
     |    4     1
     |                               xxxx
 59  |    x           x             xxxxx
     |    1           5                37
     |                         xxxx
 49  |                        xxxxx
     |                           37
     |                      x
 39  |    x     x           x
     |    6     4           9
     |    x                                         x
 29  |    x     x                       x           x                 x
     |   10     5                       4          10                 6
     |xxxxx    xx     x                            xx
 19  |xxxxx   xxx     x     x     x           x    xx                 x
     |   42    21     8     3     1           1    15                 1
     |        xxx                                        xx         xxx
  9  |        xxx     x     x                 x         xxx         xxx
     |         26     6     5                 5          20          27
     -------------------------------------------------------------------
       0- 9 10-19 20-29 30-39 40-49 50-59 60-69 70-79 80-89 90-99   max
```

Figure 13. Other rectangles of chip 11.

```
17565 rectangles
624 others (3.6% of total)
Other edge length : Mean 24.4, Standard deviation 28.0, Max 460

Distribution of other rectangle's sizes
4 truncated edges
     |
max  |    x     x     x
     |   25     1    11
     |
  99 |
     |
     |
  89 |    x
     |   23
     |
  79 |          x
     |         11
     |
  69 |
     |
     |
  59 |    x     x     x                 x
     |    1     1    11                24
     |
  49 |          x                 x
     |         11                24
     |
  39 |    x     x           x
     |    1     1           2
     |          x
  29 |          x                                   x                 x
     |         55                                  22                 1
     |    x xxxxx
  19 |   xx xxxxx                             x                       x
     |   76   247                             1                       2
     |          x
   9 |          x     x     x                                   x     x
     |         49    10    11                                   2     1
     -------------------------------------------------------------------
       0- 9 10-19 20-29 30-39 40-49 50-59 60-69 70-79 80-89 90-99  max
```

Figure 14. Other rectangles of chip 12.

```
18056 rectangles
193 others (1.1% of total)
Other edge length : Mean 56.1, Standard deviation 104.9, Max 1068

Distribution of other rectangle's sizes
20 truncated edges
    |                x
max |    x     x     x
    |    2     2     5
    |
 99 |
    |
    |
 89 |                x
    |                1
    |                x
 79 |               xx
    |                8
    |
 69 |    x
    |    1
    |    x                           xxx
 59 |   xx                          xxxx
    |    7                            20
    |                          xxx
 49 |                         xxxx
    |                           20
    |    x
 39 |    x
    |    5
    |                                                                x
 29 |                                  x                            xx
    |                                  4                             9
    |xxxxx          xx                            xx
 19 |xxxxx          xx     x                     xxx
    |   27          12     2                      14
    |          x                                       xxx       xxxxx
  9 |         xx                                      xxxx       xxxxx
    |          8                                        19          27
    ------------------------------------------------------------------
      0- 9 10-19 20-29 30-39 40-49 50-59 60-69 70-79 80-89 90-99  max
```

Figure 15. Other rectangles of chip 13.

```
33387 rectangles
250 others (.7% of total)
Other edge length : Mean 46.1, Standard deviation 88.5, Max 1675

Distribution of other rectangle's sizes
42 truncated edges
     |               xx
max  |          x   xxx
     |          4    16
     |
 99  |
     |
     |
 89  |
     |                                             xx
 79  |                                            xxx
     |                                             15
     |                                       xx
 69  |                                      xxx
     |                                       15
     |
 59  |
     |
     |
 49  |
     |
     |    x
 39  |   xx     x
     |    8     1
     |                                                    x           x
 29  |          x           x                            xx          xx
     |          4           1                             8           8
     | xxxx   xxx  xxxx   xxx    xx                                 xxx
 19  | xxxx   xxx  xxxx   xxx    xx           x                     xxx
     |   25    18    24    19    13           1                      18
     |        xxx                                                 xxxxx
   9 |        xxx           x                                     xxxxx
     |         19           2                                        31
     ------------------------------------------------------------------
       0- 9 10-19 20-29 30-39 40-49 50-59 60-69 70-79 80-89 90-99  max
```

Figure 16. Other rectangles of chip 14.

```
95901 rectangles
904 others (.9% of total)
Other edge length : Mean 38.2, Standard deviation 114.5, Max 2902

Distribution of other rectangle's sizes
95 truncated edges
    |                x
max |    x     x    xx     x
    |    1     4    36     1
    |
 99 |          x
    |          1
    |
 89 |    x     x
    |    1     1
    |                                              x
 79 |                                             xx
    |                                             38
    |                                        x
 69 |    x                                  xx
    |    1                                  38
    |
 59 |                                  x
    |                                  1
    |
 49 |    x
    |   10
    |   xx
 39 |   xx                 x
    |   45                 2
    |    x
 29 |    x     x     x     x                             x           x
    |   27     5     6     2                            18          18
    |  xxx   xxx  xxxx     x                                         x
 19 |  xxx   xxx xxxxx     x     x     x     x                      xx
    |   71    78   112    21     1     1     1                      39
    |      xxxxx  xxxx                                             xxx
  9 |      xxxxx  xxxx     x     x     x     x     x           x  xxxx
    |        127    99     6     2     2     1     1           1    85
    ------------------------------------------------------------------
      0- 9 10-19 20-29 30-39 40-49 50-59 60-69 70-79 80-89 90-99  max
```

Figure 17. Other rectangles of chip 15.

```
109682 rectangles
1031 others (.9% of total)
Other edge length : Mean 46.6, Standard deviation 95.1, Max 1416

Distribution of other rectangle's sizes
173 truncated edges
    |               xx           x
max |    x         xxx     x     x     x                             x
    |    3          71     2    31     2                             2
    |
 99 |
    |
 89 |                x           x
    |               19           1
    |                                             xx
 79 |                x                            xx
    |                2                            49
    |    x                                  xx
 69 |    x                                  xx
    |   30                                  49
 59 |    x                                                            x
    |    1                                                            3
    |                                                                xx
 49 |    x                 x     x                       x          xxx
    |    3                13     2                       1           74
    |
 39 |                x           x
    |                2          19
    |    x xxxxx                                         x           xx
 29 |    x xxxxx     x     x     x     x                 x           xx
    |   21  135      4     2     2     1                25           57
    |   xx  xxxx   xxx
 19 |  xxx xxxxx  xxxx                                                x
    |   62  120    95                                                 6
    |        xxx
  9 |        xxx     x           x           x                 x      x
    |         85    14           1          13                 1      8
    --------------------------------------------------------------------
       0- 9 10-19 20-29 30-39 40-49 50-59 60-69 70-79 80-89 90-99  max
```

Figure 18. Other rectangles of chip 16.

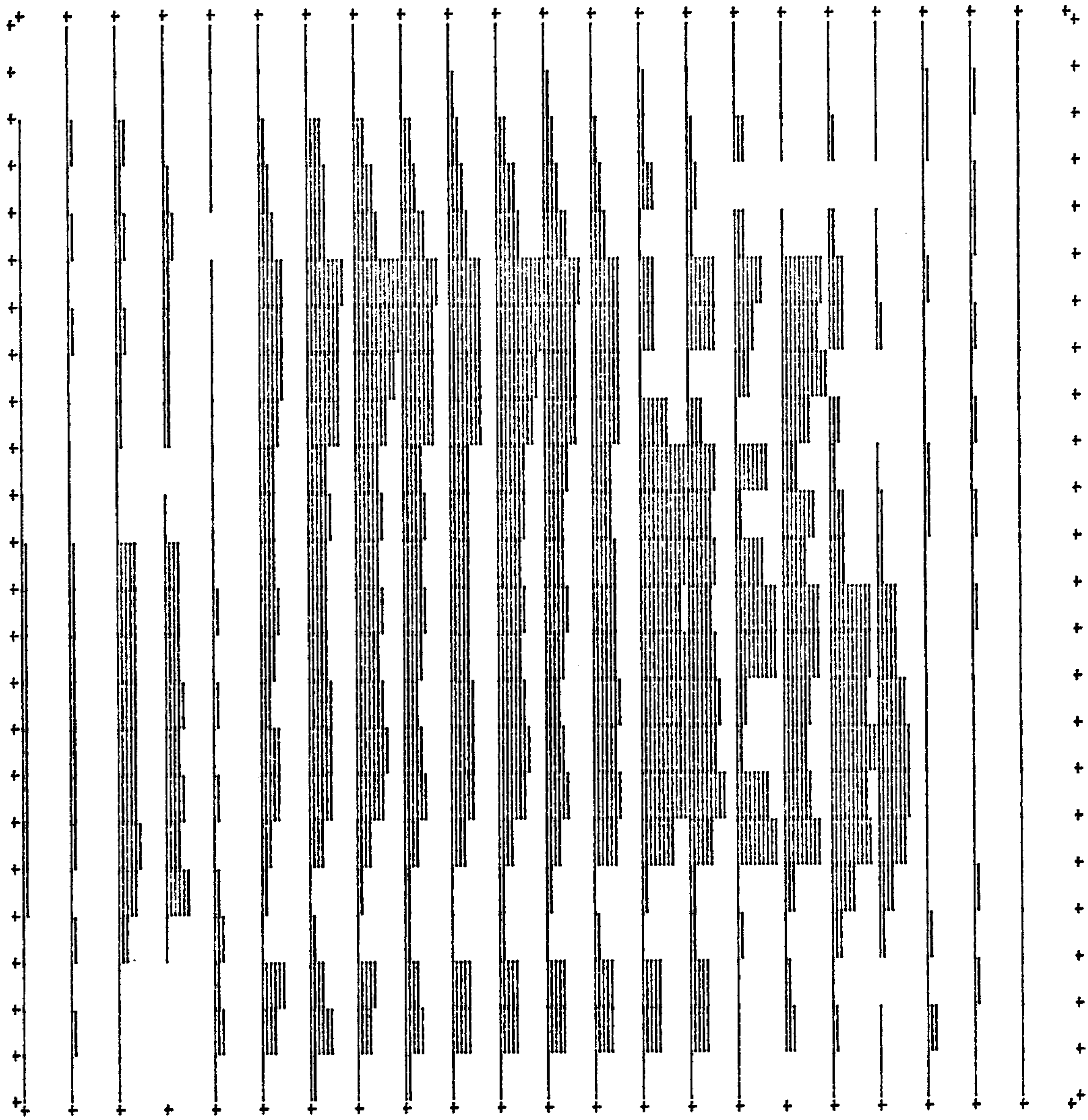
V. Placement Histograms

In this appendix we present histograms that describe the placement of rectangles over the chip area. The primary histograms are contained in Section V.1; these histograms describe the two-dimensional structure of the data. One-dimensional histograms in Section V.2 provide a view of the data useful in the analysis of the algorithm in Section 3.1.

V.1 Two-dimensional Histograms

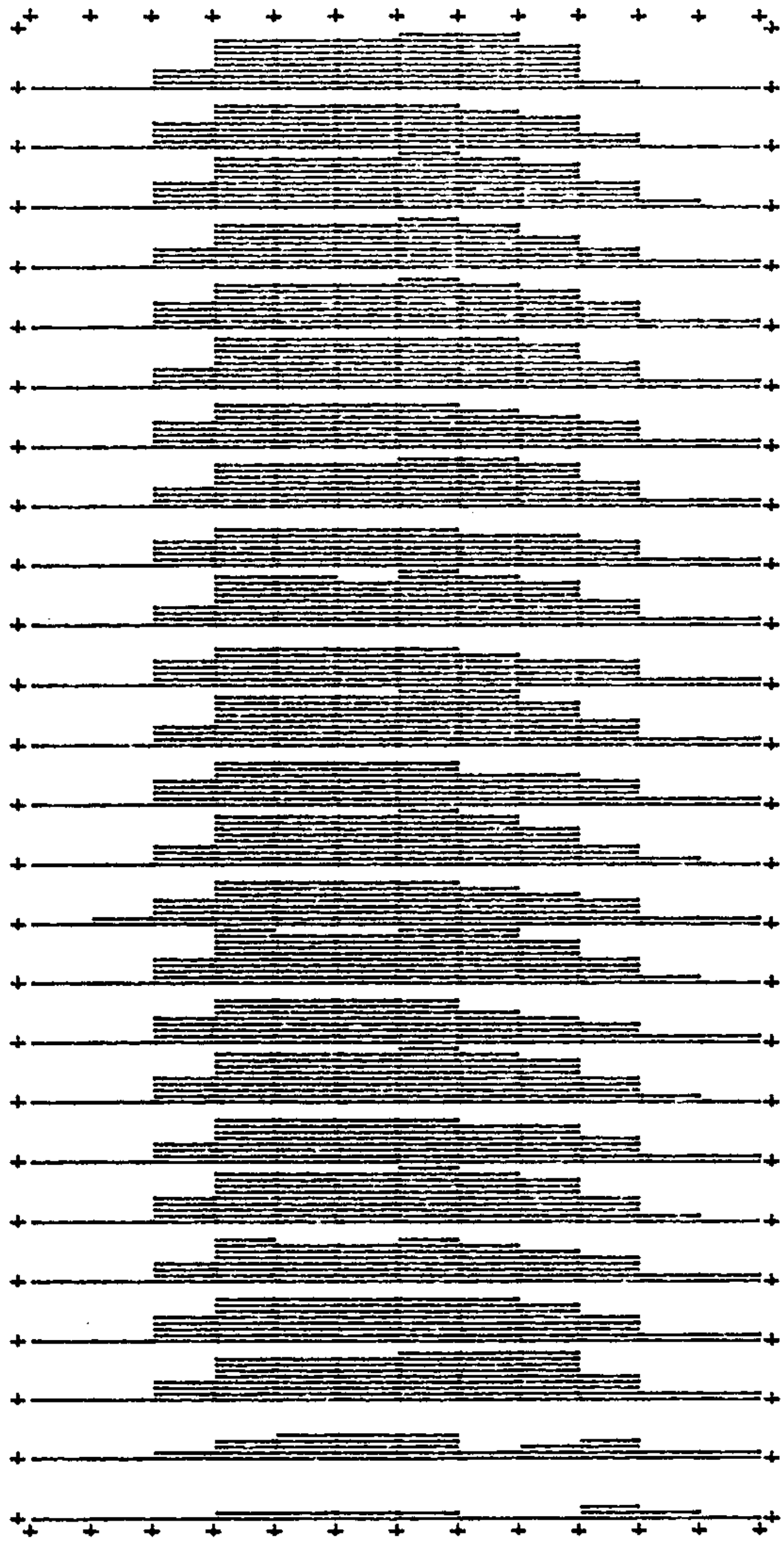
The histograms of this section describe the two-dimensional placement of the rectangles on the chip. Figure 19, for example, describes chip 11. As stated on the figure, that chip contains 1614 rectangles. It was divided into 50λ-by-50λ squares (from the "50L" in the figure, and the number of rectangles overlapping each square is denoted by the height of the horizontal lines in the square. The maximum number of rectangles overlapping any square was 147 (so denoted on the figure); a square half-full of horizontal lines therefore overlapped approximately 75 rectangles.

The histograms of Figures 19 through 24 show that the rectangles are definitely *not* distributed uniformly over the chip: there are large sparse regions around the edges (where there were usually pads or blank space), and very dense portions in the center of the chip. Indeed, the rectangles are "more than uniform" over the large, dense portions of the chip -- that is, the variance is less than expected for uniform random distributions. This observation is quite consistent across all chips: there is a great deal of sparsity around the edges, and most of the rectangles are distributed uniformly over portions of the internal part of the chip.



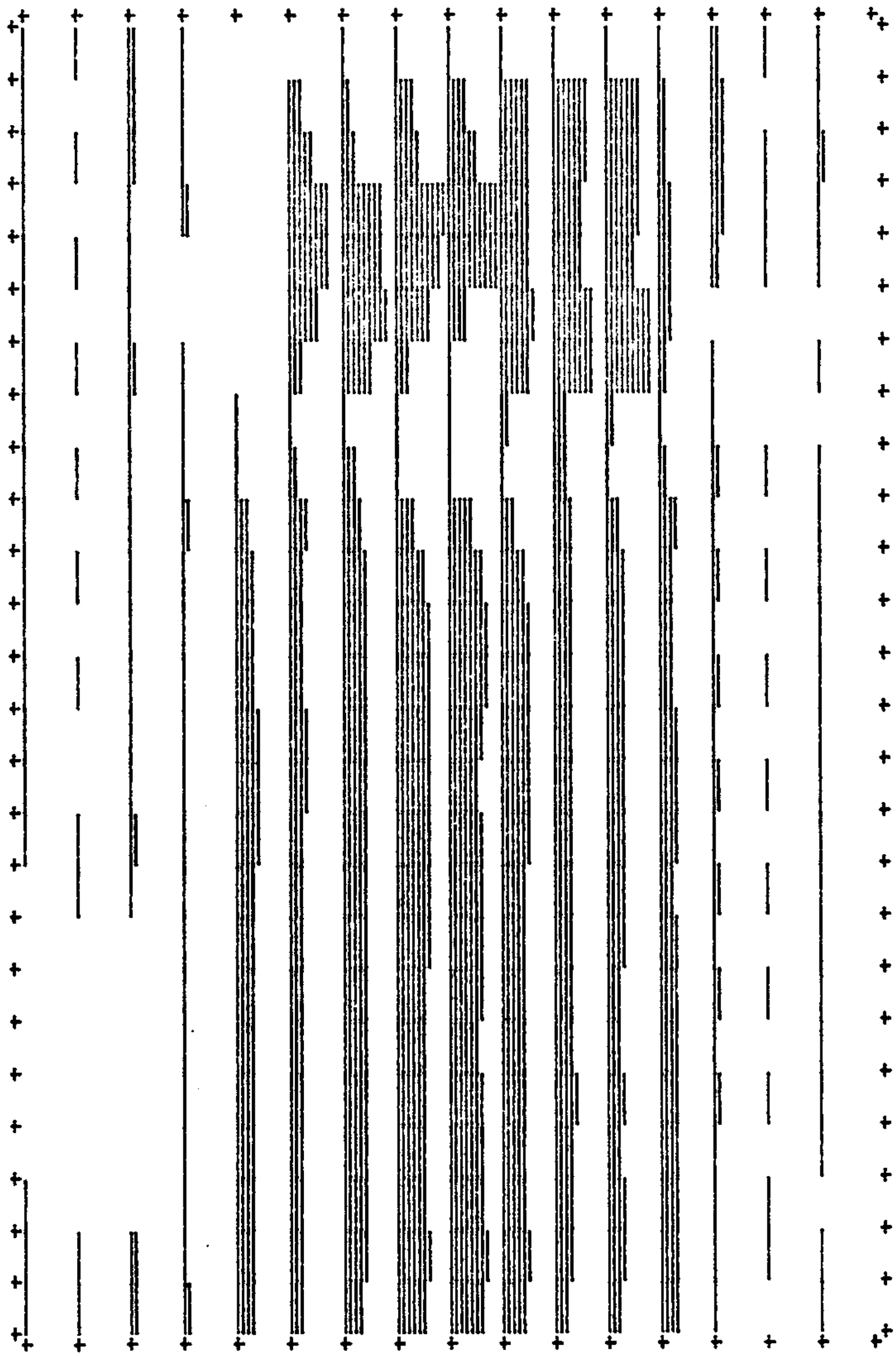
Bin size: 50λ; 16194 rectangles; Max in bin: 147

Figure 19. Placement histogram of chip 11.



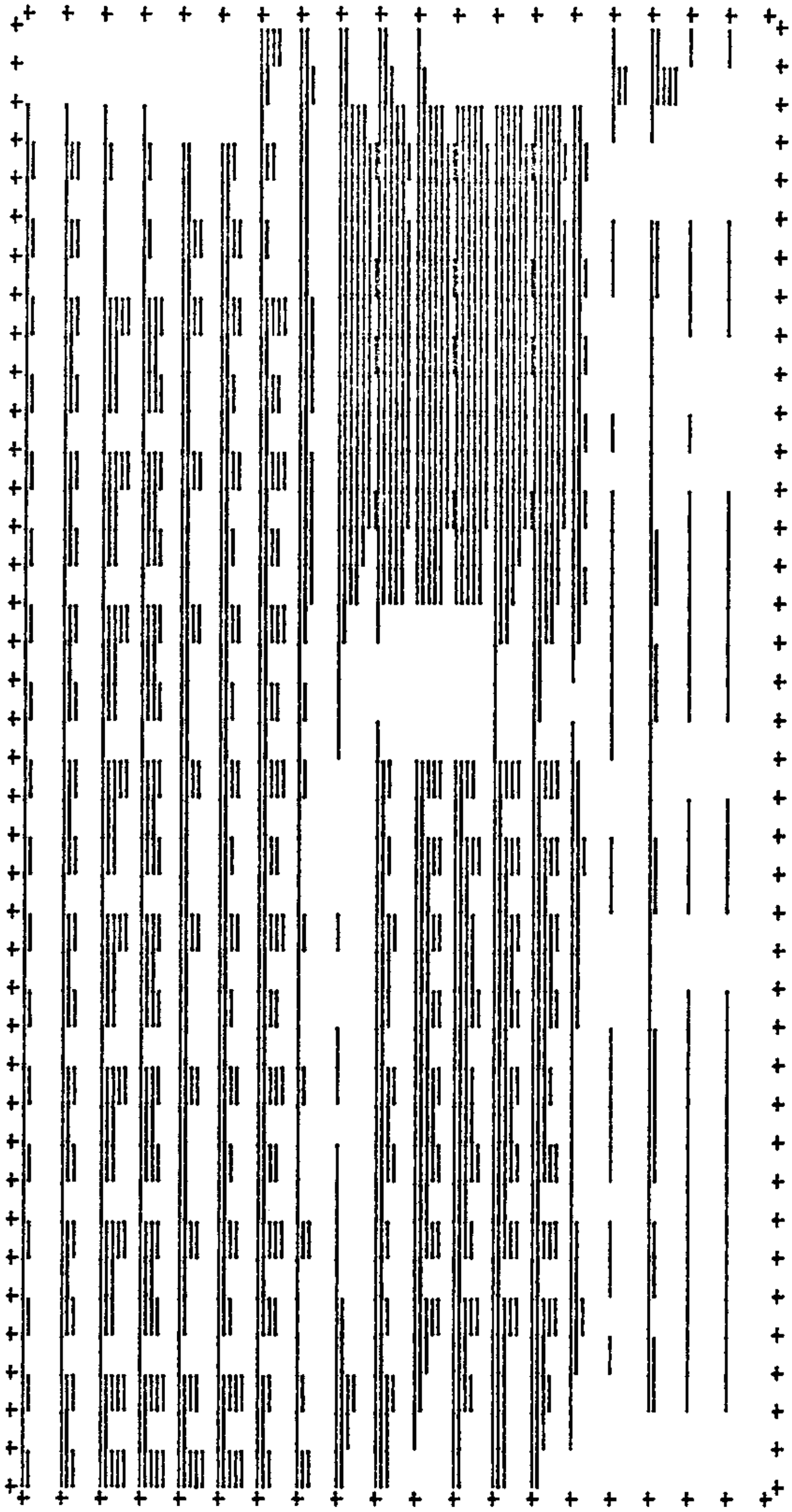
Bin size: 50λ; 17565 rectangles; Max in bin: 157

Figure 20. Placement histogram of chip 12.



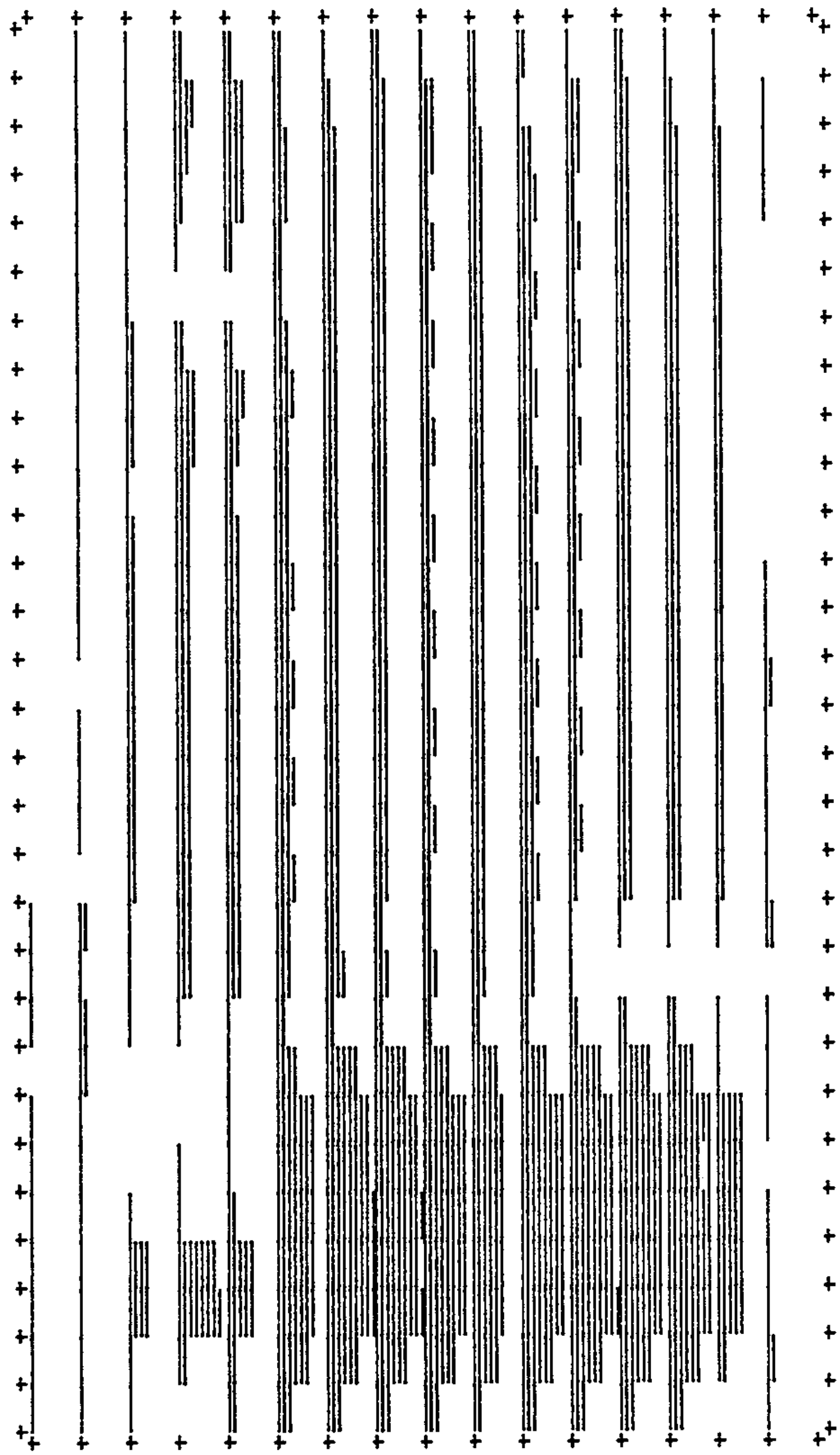
Bin size: 50λ; 18056 rectangles; Max in bin: 224

Figure 21. Placement histogram of chip 13.



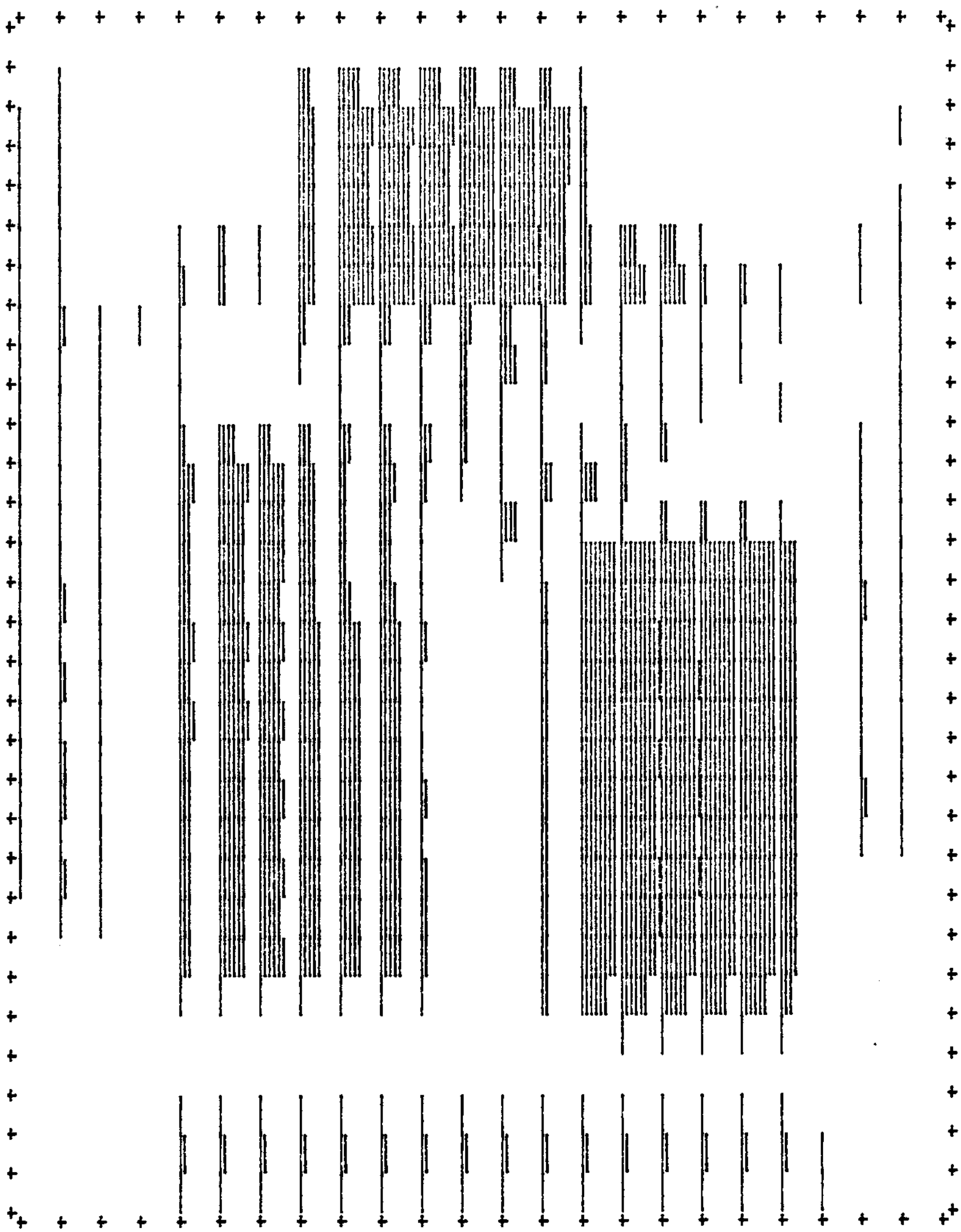
Bin size: 50λ; 33387 rectangles; Max in bin: 173

Figure 22. Placement histogram of chip 14.



Bin size: 100λ; 95901 rectangles; Max in bin: 768

Figure 23. Placement histogram of chip 15.



Bin size: 100λ; 109682 rectangles; Max in bin: 740

Figure 24. Placement histogram of chip 16.

V.2 One-dimensional Histograms

The algorithm of Section 3.1 assumes that at any point in the bottom-to-top scan of the chip, only a relatively small number of rectangles will intersect the current scan line. We showed in that section that under the probabilistic models of Section 2.4 the expected number of intersecting rectangles is proportional to $N^{1/2}$, and this conjecture was supported by the data of Section 3.2. In this appendix we will give histograms that give more of a feel of the one-dimensional distribution of the data.

Figure 25 gives the one-dimensional histograms of chip 11. The bottom figure describes the projection of the chip onto the x axis. That axis was divided into bins of size 50λ (denoted on the figure), and the height of each bin is proportional to the ratio of the number of rectangles in that 50λ slice of the chip. The maximum number of rectangles in any bin is noted (1309 on chip 11's x axis) to facilitate conversion of the relative measures to absolute measures.

The conclusions that can be drawn from these histograms are similar to those we drew from the two-dimensional histograms: there are some sparse areas in the projections (usually corresponding to the edges of the chip), but most of the rectangles are quite uniformly distributed over a large portion of the projection.

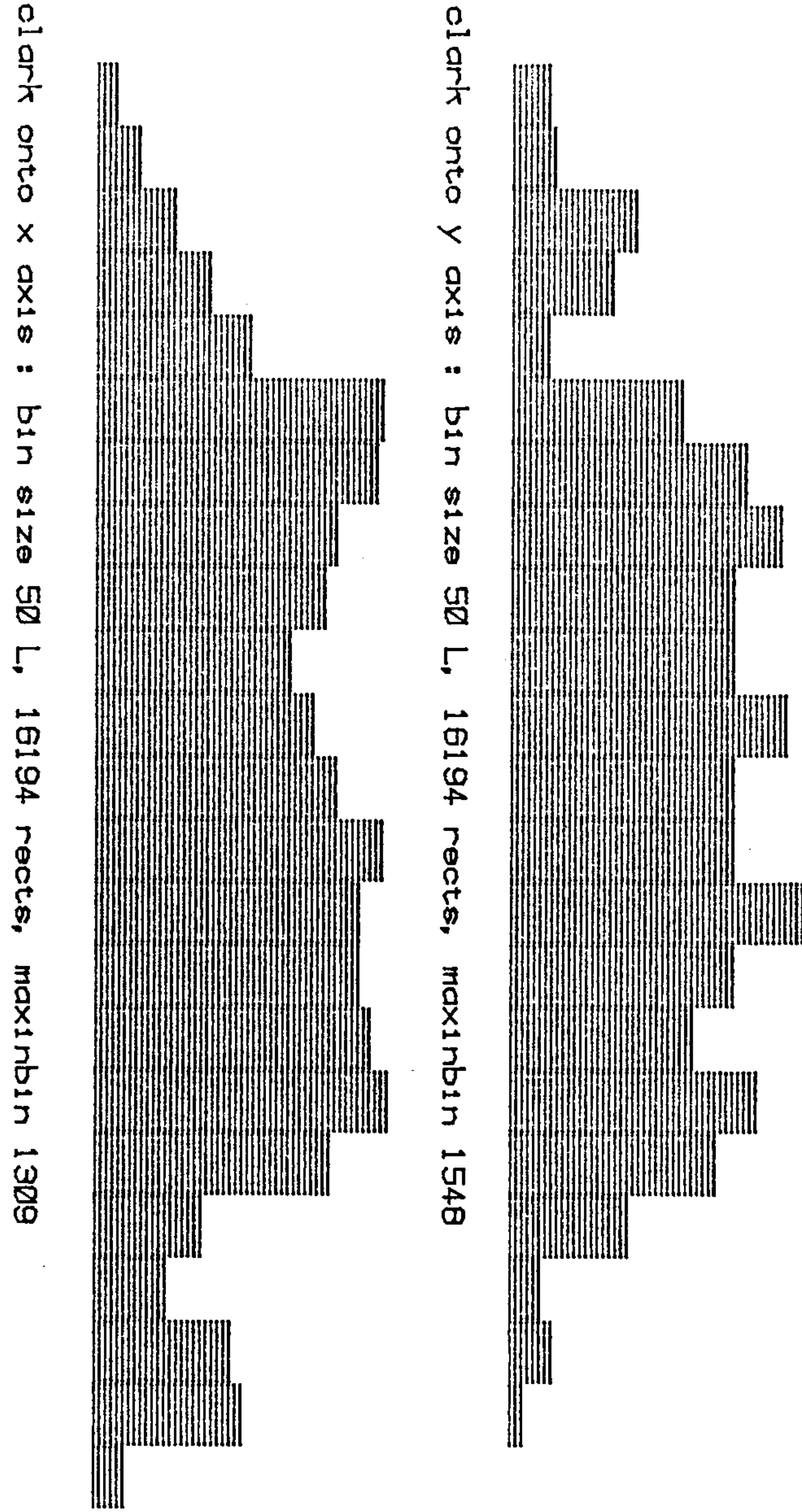


Figure 25. Projected placement histograms of chip 11.

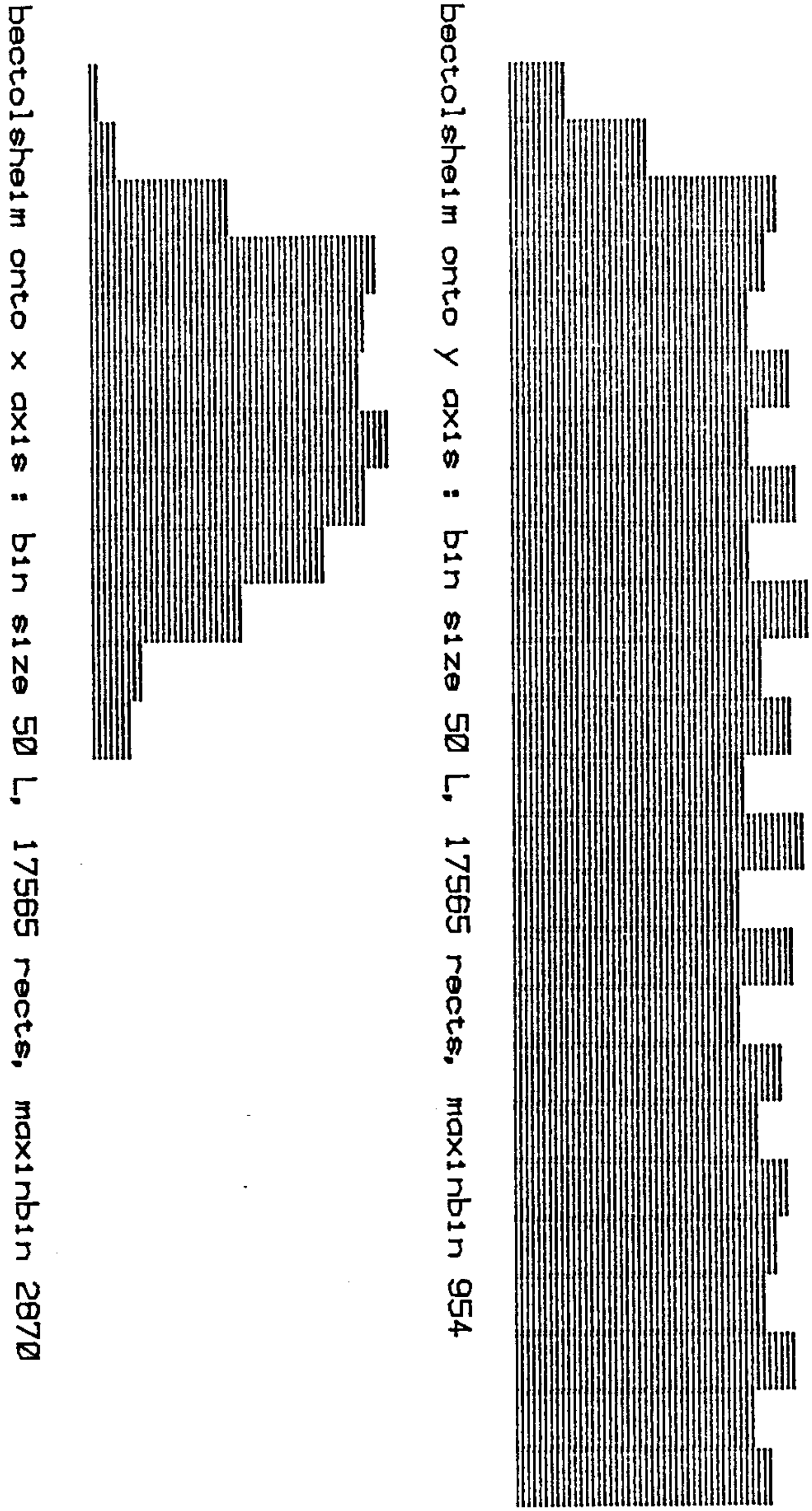


Figure 26. Projected placement histograms of chip 12.

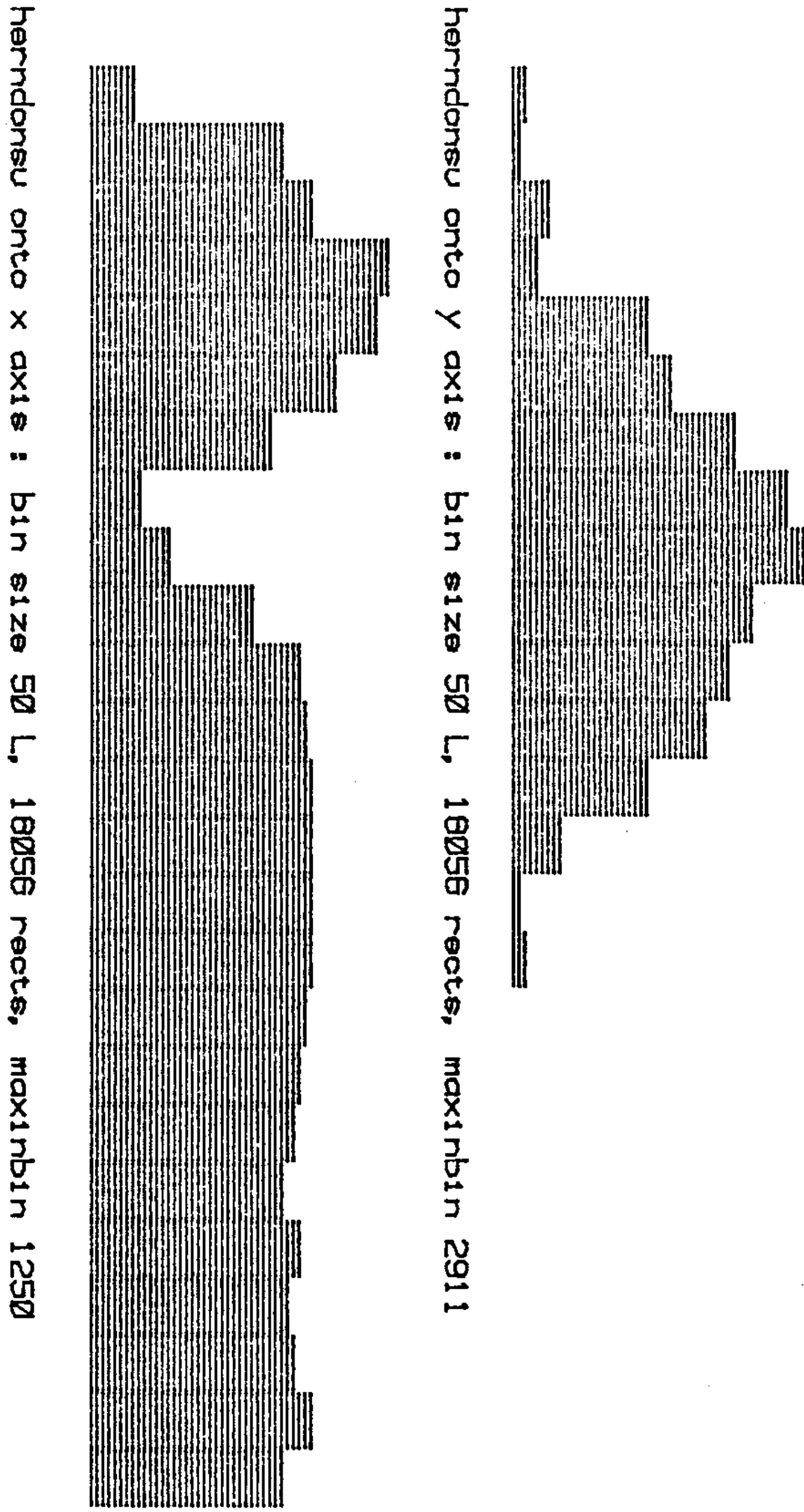


Figure 27. Projected placement histograms of chip 13.

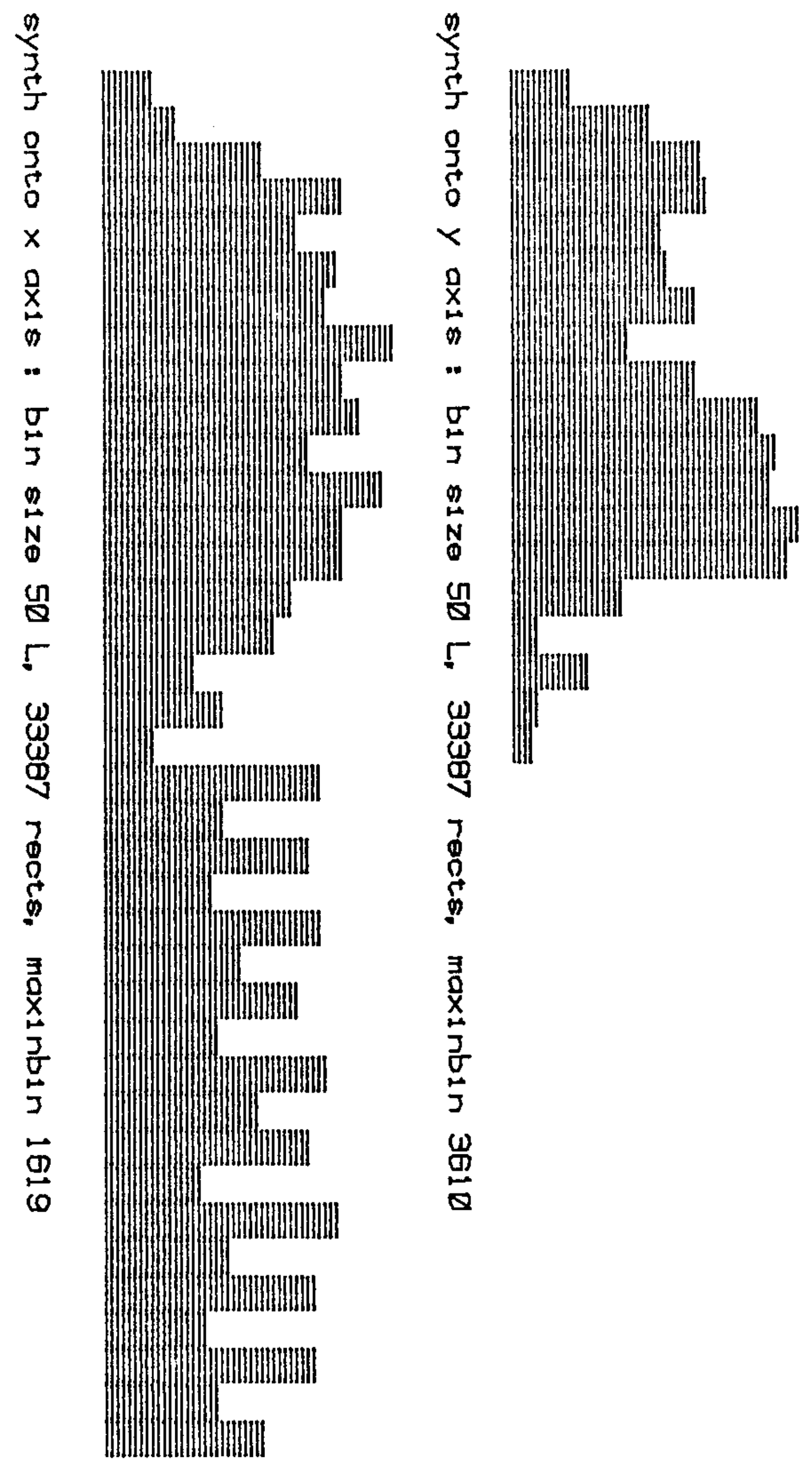


Figure 28. Projected placement histograms of chip 14.

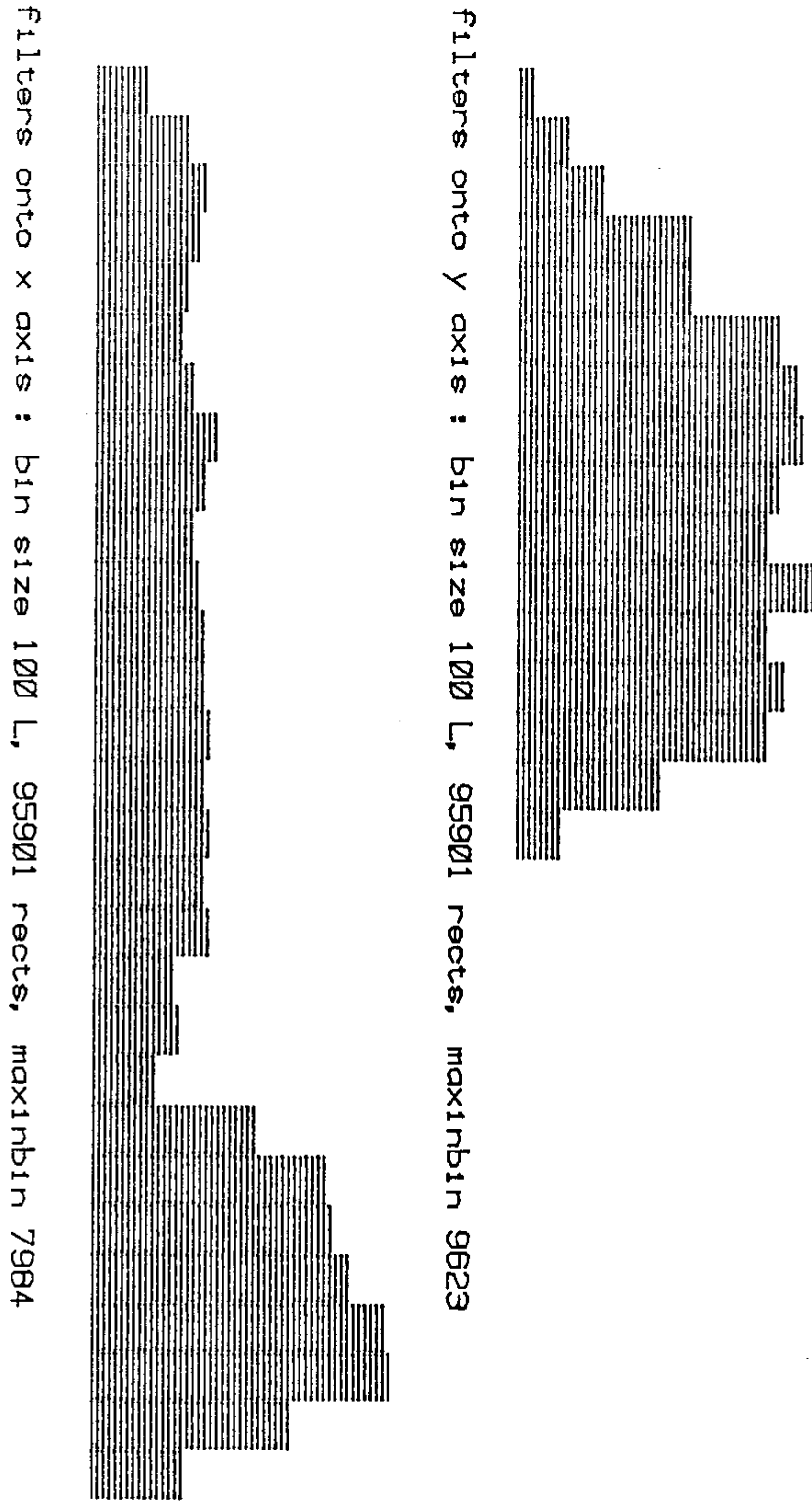


Figure 29. Projected placement histograms of chip 15.

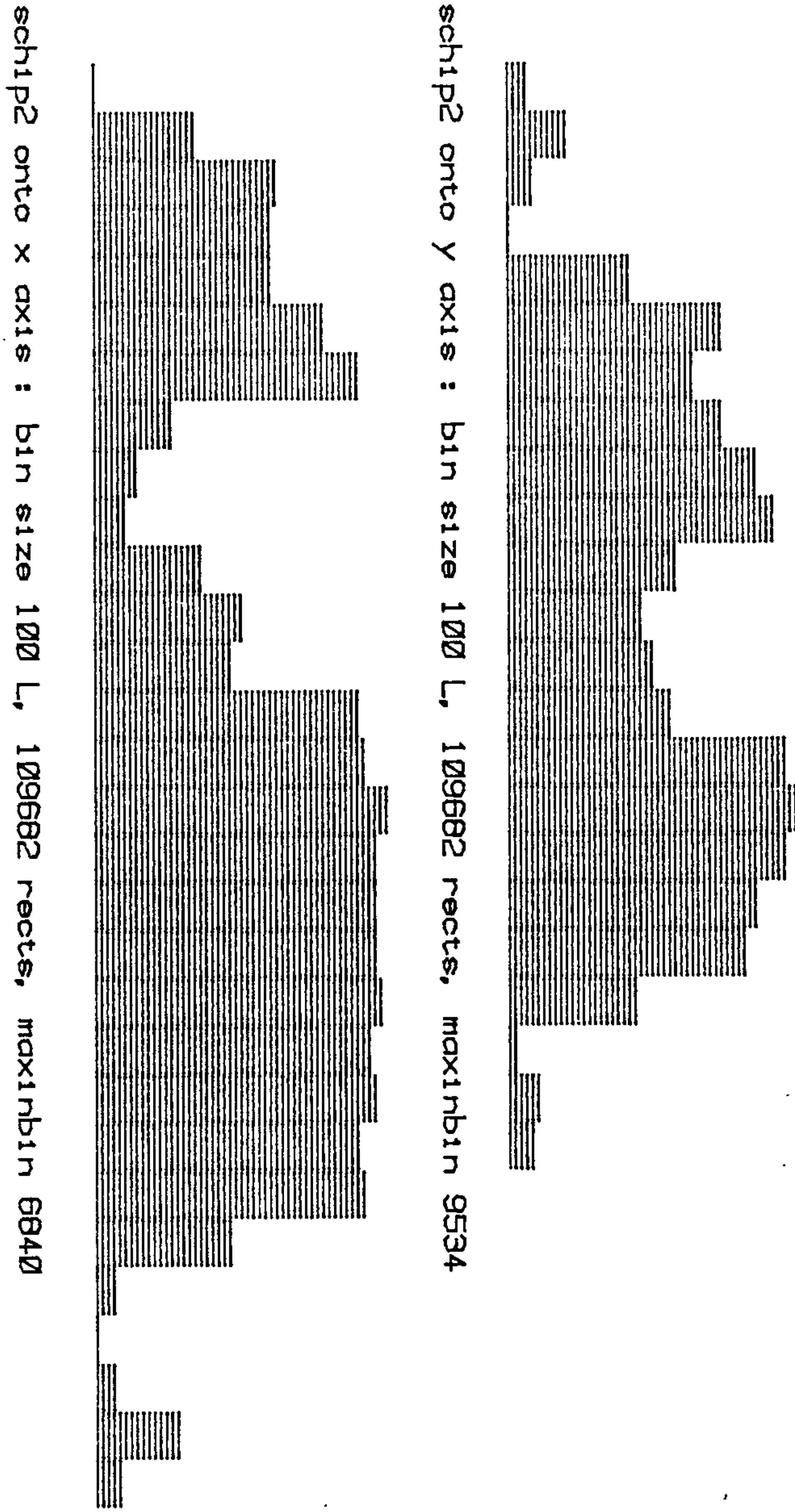


Figure 30. Projected placement histograms of chip 16.

VI. Data on Rectangle Area

The measurements presented in Section 2.2 focussed on the *number* and the *edge lengths* of rectangles of certain types. These statistics were used in designing the algorithm of Chapter 3. Different applications will require additional statistics; in this appendix we present measurements of more general interest. One of the most important aspects of a layout is *area* usage. How does the space devoted to computation compare to the space devoted to communication? What percentage of the area is used for pads to communicate with the outside world? It is the aim of a good layout to minimize communication costs; how well can this be achieved?

Table 7 contains area-usage measurements for the 16 chips we studied. The first two columns are repeated from Table 1 of Section 2.2. The next three columns describe the area used in the chip as a whole. The column labeled 'BB' gives the total area of the chip's bounding box, in units of kilo-λ^2. The next column, labeled 'Rects', gives the total area occupied by the rectangles, also in kilo-λ^2 units. This column is just the sum of the areas of all rectangles on the chip; thus, if a particular λ-by-λ square of the chip is covered by five rectangles, it is counted five times in this measure. The ratio of these two total area measures is given in the fifth column. For example, columns three through five of row one say that chip 1 has a bounding box area of $909{,}000\lambda^2$ and a total rectangle area of $841{,}000\lambda^2$, making the ratio of bounding box area to rectangle area .93. The remainder of the table is devoted to an area breakdown according to the rectangle trichotomy introduced in Section 2.2. The 'Area' column under 'Components' lists the total area occupied by components (sum of the areas of all individual components); the next column gives the percentage of the total rectangle area this represents. The columns for wires and others are similar.

The following table summarizes the percentage columns (columns seven, nine and eleven) in Table 7.

	Median	Mean	S.D.
Component area %	19.0	19.6	5.0
Wire area %	38.5	37.4	6.1
Other area %	43.5	43.1	9.2

That is, of the sixteen component-area percentages, the median is 19.0%, the mean is 19.6% and the standard deviation is 5.0%. The summary shows that the area percentage breakdowns are quite consistent across the sixteen designs, with low standard deviations and good agreement between means and medians. This consistency is rather surprising, given that the chips all had different designers and that they implemented designs for widely varying applications.

The above data demonstrates the high cost of communication on a chip. On the average, less than 20% of the total rectangle area was devoted to the active components; close to 40% was used for on-chip communication; slightly over 40% was used for off-chip communication (pads). The designs

Chip #	# Rects	Total Area BB	Rects	BB/Rects	Components Area	%	Wires Area	%	Others Area	%
1	10803	909	841	.93	102	12.1	269	32.0	471	55.9
2	11265	939	600	.64	118	19.8	235	39.3	244	40.8
3	11566	672	877	1.31	141	16.1	255	29.0	481	54.9
4	11853	975	967	.99	130	13.4	272	28.2	565	58.4
5	11915	817	658	.81	120	18.3	222	33.7	316	48.0
6	12063	777	689	.89	119	17.2	338	49.2	232	33.6
7	12800	728	685	.94	135	19.7	262	38.2	288	42.1
8	14186	1132	1006	.89	173	17.1	354	35.2	479	47.6
9	14423	836	735	.88	170	23.2	203	27.6	362	49.2
10	15097	944	740	.78	168	22.7	274	37.1	297	40.2
11	16194	1315	1092	.83	179	16.4	423	38.8	490	44.9
12	17565	770	866	1.13	127	14.7	350	40.4	389	44.9
13	18056	1014	884	.87	210	23.7	351	39.7	323	36.5
14	33387	1849	1400	.76	355	25.4	677	48.4	367	26.3
15	95901	4934	3900	.79	1286	33.0	1614	41.4	1000	25.7
16	109682	7155	5472	.76	1098	20.1	2158	39.4	2216	40.5

Table 7. Data on Rectangle Area.

we studied were relatively small; hence the large relative area devoted to the pads is not surprising. Moving down the table from the smallest to the largest designs there is a trend toward lower percentage of the area used for 'other' rectangles. Chip 16 is an exception to this rule; on the other hand, it has an unusually large number of pads (49). Larger designs, which perform more computation, can amortize the high cost of the pads more effectively.

The ratio of bounding box area to total rectangle area has mean and median both about .88, with a standard deviation of .15. These chips were designed with little emphasis on optimal space utilization, and yet there is substantial regularity in this ratio.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CMU-CS-80-111	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) STATISTICS ON VLSI DESIGNS		5. TYPE OF REPORT & PERIOD COVERED Interim
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) JON LOUIS BENTLEY DOROTHEA HAKEN & ROBERT W. HON		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0370
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Computer Science Department Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA 22217		12. REPORT DATE APRIL 1980
		13. NUMBER OF PAGES 61
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)