REPORT DOCUMENTATION PAGE

OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	19 Apr 95	

4. TITLE AND SUBTITLE

Automatic Layout of Integrated-Optics Time-Of-Flight Circuits

5. FUNDING NUMBERS

6. AUTHOR(S)

Ruth D. Kennett-Fogg

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

AFIT Students Attending:

Colorado University

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/CI/CIA

95-051

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

DEPARTNEMT OF THE AIR FORCE
AFIT/CI
2950 P STREET, BDLG 125
WRIGHT-PATTERSON AFB OH 45433-7765

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release IAW AFR 190-1
Distribution Unlimited
BRIAN D. GAUTHIER, MSgt, USAF
Chief Administration

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Accesion For	
NTIS CRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

14. SUBJECT TERMS

15. NUMBER OF PAGES

122

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

AUTOMATIC LAYOUT OF INTEGRATED-OPTICS TIME-OF-FLIGHT CIRCUITS

by

Ruth D. Kennett-Fogg

B. S. Electrical Engineering, University of New Hampshire, 1983

M. S. Electrical Engineering, University of New Hampshire, 1985

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Electrical and Computer Engineering

1995

This thesis for the Doctor of Philosophy degree by

Ruth D. Kennett-Fogg

has been approved for the

Department of

Electrical and Computer Engineering

by

Harry F. Jordan

Michael J. Lightner

Date 4/19/95

Kennett-Fogg, Ruth D. (Ph.D., Electrical Engineering)

Automatic Layout of Integrated-Optics Time-of-Flight Circuits

Thesis directed by Professor Harry F. Jordan

This work describes the architecture and algorithms used in the computer-aided design tool developed for the automatic layout of integrated-optic, time-of-flight circuit designs. This is similar to the layout of electronic VLSI circuits, where total wire length and chip area minimization are the major goals. Likewise, total wire length and chip area minimization are also the goals in the layout of time-of-flight circuits. However, there are two major differences between the layout of time of flight circuits and VLSI circuits. First, the interconnection lengths of time-of-flight designs are exactly specified in order to achieve the necessary delays for signal synchronization. Secondly, the switching elements are 120 times longer than they are wide. This highly astigmatic aspect ratio causes severe constraints on how and where the switches are placed. The assumed development of integrated corner turning mirrors allows the use of a parallel, row-based device placement architecture and a rectangular, fixed-grid track system for the connecting paths. The layout process proceeds in two steps. The first step involves the use of a partial circuit graph representation to place the elements in rows, oriented in the direction of the signal flow. After iterative improvement of the placement, the second step proceeds with the routing of the connecting paths. The main problem in the automatic layout of time-of-flight circuits is achieving the correct path lengths without overlapping previously routed paths. This problem is solved by taking advantage of a certain degree of variability present in each path, allowing the use of simple heuristics to circumvent previously routed paths.

DEDICATION

To my loving husband, Jert and children, Rachel and Raymond.

ACKNOWLEDGMENTS

This work was primarily funded by the Air Force Institute of Technology at Wright Patterson Air Force Base, Ohio.

I wish to thank my advisor Dr. Harry F. Jordan for his technical support and very thoughtful guidance. My thanks also to Dr. Michael Lightner who steered me into this subject area. I would also like to thank Dr Vincent Heuring for his comments on the final draft. Special thanks to John Feehrer and Carl Love for their quick email responses to all my questions. Finally, I would like to thank the members of the EE Dept at the Air Force Academy, for providing me the excellent facilities and time to finish this work.

CONTENTS

TABLES

TABLE

FIGURES

FIGURE

CHAPTER 1

INTRODUCTION

The recent design [1] and demonstration [2] of a discrete, stored program optical computer (SPOC) by the Optoelectronic Computing Systems Center at the University of Colorado, has motivated the development of an integrated version of the computer. Instead of using flip flops for synchronization, the sub-circuits of SPOC rely on the delay time or "time-of-flight" of the light in the connections. One of the first steps to achieving an integrated time-of-flight circuit, is to develop a CAD tool for the layout of the circuit components and connections. This dissertation describes the models and algorithms used by the automatic layout tool developed for the layout of integrated-optic, time-of-flight circuits.

The rest of this chapter describes time-of-flight circuits and related work in the development of VLSI layout tools. Chapter Two outlines the system and device models used in the tool. Chapter Three describes the theory that supports the placement and routing algorithms. The details of the algorithms are described in Chapter Four. The algorithms are put to test and the results presented in Chapter Five. Finally, Chapter Six describes how the tool is successful and suggests further work for future improvements.

1.1 Background

SPOC was designed [1] and implemented [2][3] using discrete $LiNbO_3$ electro-optic, directional coupler switches and fixed ratio couplers that are used

as splitters, and combiners. Optical fiber is used for both the interconnections and the delay line memory. The entire system fits onto a large table. A much smaller system can be achieved by integrating the circuit elements and their interconnections onto a single wafer of $LiNbO_3$ and keeping the fiber-optic delay line memory off-chip. The current design tool, XHATCH [4], allows the designer to design and simulate an electro-optic time-of-flight circuit, but, it does not provide a layout capability. Therefore, a tool is needed to assist the designer in the layout of integrated electro-optic components onto a two-dimensional substrate of $LiNbO_3$.

The design and layout of integrated-optic, time-of-flight circuits is similar to the design and layout of electronic VLSI circuits. Typically, when an electronic integrated circuit is designed, the designer first creates a symbolic logic diagram of the circuit, which is simulated to verify the circuit function. Once the capability of the circuit is verified, the designer uses a graphical layout tool to draw the masks to be used during the fabrication of the circuit. During layout, design rule checking is done continuously. This helps the designer stay within the constraints dictated by the fabrication technology.

The layout tool that has been built is similar to a VLSI circuit layout tool with the major exception being the algorithms used and the rules observed during design rule checking. As a consequence of using time-of-flight design, each connection of an integrated opto-electronic circuit must have a particular path length. This is the major difference between the time-of-flight layout tool and the VLSI layout tool.

The design of the time-of-flight circuit layout tool, like VLSI layout tools, revolves around two major steps, the placement of the elements and the

routing of the connections. The main questions that are answered in this work are:

(a) Which switch/coupler placement architecture leads to a routable design?

(b) Given a particular placement architecture, which device placement algorithm produces the best initial placement?

(c) Which iterative improvement algorithms result in the shortest clock periods?

(d) Which connecting path architecture is the best?

(e) Given a particular connecting path architecture, which algorithm produces paths that maintain the desired clock period with the fewest design rule violations?

(f) What design rule checking is done and how is it accomplished?

(g) How does the correction of design rule violations affect the clock period?

1.2 Time-of-Flight Synchronization

All digital computers rely on the synchronization of signals at all points of interaction. In electronic systems, flip-flops are used to synchronize the signals. Using integrated-optics, bistable devices are more difficult to fabricate. However, since signal delays depend on the length of the path, it is possible to do without bistable devices by using a pulsed logic bit stream and incorporating specific path delays for synchronization. Pulsed logic is a form of digital logic where the presence of a high pulse represents a logic one and the absence of a pulse represents a logic zero. Small drifts in timing occurring in feedback loops can be eliminated by gating a correctly timed clock pulse

into the bit stream in place of the original skewed pulse. Multiples of the clock period, represented by lumped delays, are incorporated in the design to achieve sequential circuit operation. An ideal time-of flight design assumes that all path delays are zero except for the paths with a designated lumped delay. Synchronization is made possible by calculating the precise length of all signal paths to ensure that pulsed bit streams arrive at their interaction points simultaneously. Pratt, in [5], describes the algorithms that are used to distribute the ideal lumped delays over all the interconnections, taking into account the real delays of the elements and the connections. The algorithm implemented in XHATCH repeatedly applies synchronization preserving transformations to convert an ideal circuit into a circuit with real distributed delays. In [6], John Feehrer describes the effects of delay uncertainty on the syncronization of time-of-flight circuits and suggests the use of clock gates to minimize these effects. An example of a time-of-flight design is the counter circuit (drawn with the XHATCH tool) shown in Fig. 1.1.

1.3 Description of the Placement and Routing Problem

The goal of the layout tool is to achieve a synchronized, integrated-optic, time-of-flight circuit. The clock period can be specified by the user or the minimum for a particular placement can be determined. This differs from the goal of VLSI placement which, generally, is to minimize total wire length.

The process of laying out time-of-flight circuits is similar to laying out VLSI circuits for which many techniques have been developed. The process is typically split into two interacting phases, placement and routing. In the placement phase, transistors, gates, modules, etc., are placed on a two-dimensional plane with the goal of minimizing the wire lengths of all the connections in

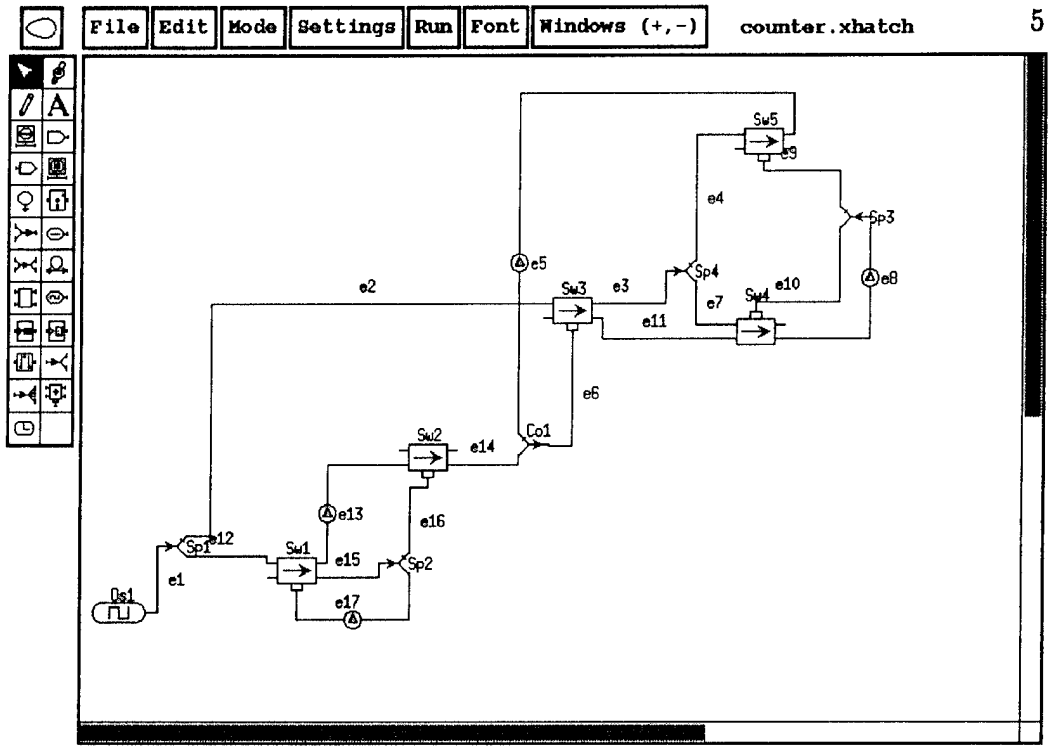


Figure 1.1: Time-Of-Flight Counter Design using the XHATCH Design Tool.

the circuit. In the routing phase, connections are drawn to connect elements together. The success of a particular placement is not known until the routing is complete. Therefore, the two processes are inherently tied together but rely on different algorithms.

Automatic placement is difficult because of the extremely complex problems it must solve [15]. For example, if the wiring is ignored and only placement is considered, the task is simply one of tiling an area optimally, which is NP-complete [16]. When routing is added to this task, every proposed cell placement requires large amounts of time to route. Because of these problems, placement cannot be done optimally; rather it is done with heuristics that produce tolerably good results in small amounts of time[15].

There are also several differences between the placement of time-of-flight design elements and VLSI cells. First, contrary to VLSI designs, in time-of-flight designs, connection lengths are part of the design, and it is possible to

estimate whether or not a connection will be long or short before placement. Using this estimate, it is possible to arrange the elements with short interconnections closer together than the elements with long interconnections. Once placed, the ideal lumped delays can be distributed to account for real device and path delays to achieve synchronization. This produces a set of path lengths which may or may not be scaled up from their minimum lengths. Therefore, in time-of-flight design layout, a set of path lengths is estimated before routing. This differs from VLSI layout where the path lengths are not known till routing is complete. However, the solution space of feasible delay lengths that satisfy synchronization constraints is very large, and calculating a set of connection lengths before routing is equivalent to picking one solution out of many with no guarantee that the resulting clock period will be minimal.

Secondly, the integrated-optic waveguide connections patterned on the $LiNbO_3$ substrate can cross each other with essentially no loss as long as the crossing angle is greater than 6° [8]. Likewise, they can also cross through the switching and coupling elements with very little impact. Because of this, the routing of the integrated optic waveguide connections is simpler than the routing of VLSI connections which cannot cross at all unless there are multiple metalization layers.

Thirdly, while routing the integrated-optic connections, it is possible a path may overlap another previously routed path. To prevent this overlap, a path may have to be adjusted. But adjusting this one path may affect the delay of one or more other connections which will then have to be adjusted as well. When these connections are adjusted, they may end up overlapping another path which will then have to be corrected for. After several connection length

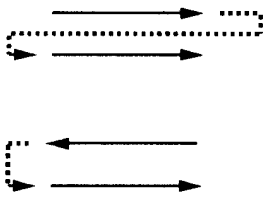


Figure 1.2: Orientation Dependent Connection Lengths.

changes and delay redistribution calculations, it may be possible to converge on a set of routable path lengths, but, there is no guarantee that it will be optimal in terms of clock speed. As can be seen, due to the interdependencies of the path delays, the routing of the connections and the avoidance of overlaps is one of the challenging aspects of the layout problem.

The fourth aspect stems from the fact that the integrated-optic switches are approximately 120 times longer than they are wide. Due to this long aspect ratio, changes in device orientation can produce large variations in the minimum distances between the elements. As shown in Fig. 1.2, two parallel switches can have either a short path between them, or a very long horizontal path, depending on the orientation of the switches relative to each other. Therefore, two elements requiring a long path delay between them can actually be placed close together provided they are oriented the same.

1.4 Related Research in Placement and Routing

It is difficult to perform time-of-flight design layout without an understanding of the major problems and algorithms used in VLSI design layout. A review of the major placement and routing techniques is presented first. Then, specific works which tie more directly to the time-of-flight design problem are described later.

1.4.1 VLSI Placement Placement deals with finding the best geometric coordinates for all circuit elements. The placement problem has deservedly gained considerable attention in the literature and there are many algorithms that produce good results [13]. In most methods, the success of a particular placement is evaluated according to a cost function based on total wire length, wiring density, and total area.

The types of algorithms used for placement fall into two categories, constructive placement and iterative improvement. In a constructive placement approach, elements are progressively assigned to their places based on a partial cost, which is the restriction of the cost function to the subnetwork of elements that have already been placed. Since optimization of this partial cost does not necessarily lead to the optimization of the global cost, this method is frequently used to produce an initial placement [16].

Iterative improvement techniques [13] attempt to improve an initial placement by iteratively producing new, better placements. The chief concern in the development of iterative improvement techniques is to find a metric for the cost function that is computationally simple to calculate at each iteration.

There are several evaluation metrics that have been reported in the literature. One widely used metric is wire length. Since wire length cannot be measured directly until the connections are routed (a time-consuming job), there are several methods used to approximate wire length. One method is to simply calculate the Manhattan distance between the terminals. Other approaches use the half perimeter of the smallest rectangle enclosing the pins of the connection [30] or a minimal Steiner tree length [35] [29]. Another method uses a quadratic form of the square of the distance [17][18] that produces an

objective function which is simpler to minimize.

Two popular placement algorithms are:

(a) Min-cut[26]. This algorithm divides the unplaced cells into two until a tree-structured graph is formed that organizes all cells. The determination of the min-cut division is based on the wiring between the cells. The goal of the best min-cut is found from the division of cells that cuts the fewest wires. It is one method of reducing the size of a problem by breaking it up into smaller problems, however, it doesn't consider cell sizes nor the relative orientation of cells within a group.

(b) Simulated Annealing [27][28][25]. This method consumes large amounts of time but produces very good results. Given an initial placement the annealer makes changes in the positions of the cells and evaluates the results according to a cost function based on wire length. If the cost increases, then the change may or may not be accepted depending on the value of a temperature dependent annealing schedule. If the cost decreases, the change is always accepted. Simulated annealing has been applied to a wide variety of optimization problems and very good results have been reported for the standard cell placement problem [21]. In [22] simulated annealing and a hierarchical approach are used for row-based placement, improving an earlier version reported in[23].

1.4.2 VLSI Routing Techniques The routing step forms the interconnecting paths between the placed circuit elements. Routing algorithms can be categorized into one of three types, channel routing, maze routing, or line search routing [24].

Channel routing has been the main routing algorithm for standard

cells for many years. Each channel consists of several parallel tracks into which the segments of the connecting paths can be routed. It is particularly suited for routing connections where the points to be connected are in parallel rows [19]. Multiple layers of connections result when horizontal tracks are formed on one layer and vertical tracks on another. Since the width of each channel and the number of tracks in each channel is variable, a complete routing is guaranteed.

There are several versions of the maze routing technique. All of them find paths around obstacles by hitting the obstacle and then making a decision to turn left or right. This method may lead to an inefficient routing or it may fail to find a routing at all [24].

The line-search algorithm finds a connection through a maze of obstructions without using a grid. It runs vertical and horizontal expansion lines from the two points to be connected. If either line encounters an obstacle, the router draws a perpendicular line to the original line in order to pass by the obstacle. Two expanding nets are created from both points. The desired connection is created when two expansion lines, one from either point, intersect. This method is not optimal since it does not always yield the shortest path.

1.4.3 Research on Timing Driven Layout of VLSI Circuits

With the increase in VLSI circuit sizes and the decrease in feature sizes, the overall performance of integrated circuits will be increasingly affected by signal propagation time through the interconnects. This has caused increased activity in areas such as timing driven placement, timing driven routing, and clock skew minimization. Also, the incorporation of speed enhancing techniques in VLSI designs, such as wave pipelining, dictates the need for greater timing awareness

in the circuit layout. How each of these areas relates to, but does not solve the time-of-flight layout problem, will be described below.

1.4.4 Timing Driven Placement The work in timing driven placement centers around the efficient use of timing information to form layouts that don't violate the timing constraints of the design specification. Unlike time-of-flight designs, the path delays are not prespecified, so the main difficulty is to derive accurate timing information without spending an enormous time estimating path delays. Also, layout tools can only approximate the routing consequences of a particular placement, so a degree of uncertainty is introduced both in the early timing analysis and in the placement system's assessment of whether the design will satisfy the timing constraints after routing[31]. Several authors [31][32][34] determine, prior to the layout step, a set of feasible net lengths that satisfy timing requirements. Each of these lengths has a lower and an upper bound. The placement algorithm then seeks a solution which keeps the net lengths within these bounds. For example, in [31] an iterative refinement approach is used. The algorithm starts with an initial global placement without the use of timing information, after which the design's timing performance is estimated. A subsequent timing analysis step identifies critical paths which are used to calculate timing margins for every path in the circuit. This information is then used to bias the next iteration of the placement step. In this way, the most accurate placement and routing information is used in calculating timing information for the subsequent layout step. The process ends when all timing constraints have been satisfied.

Another approach that establishes upper and lower bounds on path

lengths for high-speed ECL masterslice LSI's is described in [32]. In this approach, the upper and lower bounds on path lengths for the placement program are calculated from constraints based on several factors such as wire length to minimize clock skew, data path delay, and the resistance of wired-OR logic. An initial placement is performed which uses two-dimensional clustering augmented for optimizing delay. Iterative improvement is then used to adjust each cell location in order to minimize path delays. In each iteration, all path delays are calculated. This process continues until no paths that violate the delay bounds are found.

A third approach in [34] also develops bounds on delays which are converted into length limits and then used as constraints during the placement of the elements on the chip. A zero-slack algorithm is used which traces paths in a circuit and attempts to change wire lengths in order to remove excess slack. The main goal of this approach is to reduce the number of timing violations, therefore it is not necessary that length bounds be satisfied by every path after placement. This differs from integrated, time-of-flight circuit design layout where timing violations cannot be tolerated.

Instead of estimating the bounds before layout, the following methods calculate bounds on the path delays during the layout. In [30] a hierarchical approach is used where at each level of the hierarchy a new set of delay goals is determined and translated into net weights. The dynamically changing net weights change the objective function, which is used to minimize the distance between circuit elements. A constructive approach described in [33] uses an adaptive successive approximation technique, which checks timing constraints

each time a new cell is added to the partial placement. This incremental checking procedure requires less effort than checking constraints repetitively for the whole design. In [29] the path-oriented timing-driven placement problem is transformed into a Lagrange problem and solved using a piecewise linear resistive network method. The problem with these approaches is that placement algorithms are not well suited to honor bounds on net lengths [30]. This is similar to the time-of-flight design layout problem where small changes in the placement can have a large effect on the routing. However, in VLSI layouts, path lengths just need to be within their upper and lower bounds. Timing constraints may still be met if a few paths are not within their bounds. This is not the case in time-of-flight design where all the path lengths must satisfy a set of linear equations with high precision.

1.4.5 Timing Driven Routing In general, routing plays a secondary role in timing driven layout because placement has a more dominant effect than routing on the topography of the layout. In [35] the global routing problem includes estimates of delay limits which are used in the routing and re-routing process to satisfy timing constraints as well as to minimize the routing area. As with the VLSI placement algorithms, the goal is to meet the delay bounds dictated by the timing constraints rather than meeting specific delays as in time-of-flight designs.

1.4.6 Zero Clock Skew Routing The goal in clock net routing is to minimize the clock skew, which is defined as the maximum difference among the delays from the source node, the clock, to a sink node, the clock pin on a latch. There are several different approaches [36] [40] whose goal is to achieve zero skew by equalizing the lengths of the paths from the source to the

sinks. This problem is similar to our problem, since we both wish to ensure the concurrent arrival of signals at their destinations. If all time-of-flight designs resulted in circuits with perfect tree graph structures, then we would have an identical problem. However, time-of-flight circuits contain feedforward paths, reconvergent paths, and feedback paths, which make the problem more difficult than the clock tree routing problem.

An early approach [37] to minimize the clock skew develops a tree delay model where the path distances from the root node to each of the leaf nodes are equal. This delay model is then used to determine the optimal number and placement of buffers within the tree so the clock delay is minimized. A popular clock distribution network is the H-tree [41].

In [39] zero clock skew routing is achieved by equalizing all paths from the root of the clock tree to the leaf nodes. The length of each path is calculated by summing the estimated Manhattan distance between the nodes of the path. The algorithm for routing these links depends on the calculation of a balance point on the wire which will become the root where two sub-trees are merged to form a new single sub-tree. Another algorithm for the routing of a clock net for zero skew hierarchically builds the clock tree by merging two subtrees at each level [38].

1.4.7 Wave Pipelining Another area where path length induced timing is critical is in wave pipelining. In wave pipelining the combinational path delays between clocked logic must be all equal [42] [43]. However, as in VLSI layout, it is difficult to estimate the path delays until the circuit has been laid out. Wong [42] demonstrated the wave pipelining concept on a 63-bit population counter. Where path length constraints were not met after the initial

layout, buffers were added and power levels on gates were adjusted. Even with these modifications, the paths varied more than 10 percent from each other and still the circuit functioned properly at 2-3 times the normal clock frequency.

Joy and Ciesielski [44] have developed a layout method for optimizing the placement of cells in wave-pipelined designs. Their method uses a linear program whose constraints relate the clock period to the maximum and minimum logic path delays. The linear program is run iteratively. During each iteration through the linear program, new constraints become critical due to previous adjustments which increase or decrease path delay. Delay cells are used to increase path delay when desired path lengths can't be fit.

Comparing this to time-of-flight design layout, unless we can find an optical delay cell which is simpler and less lossy than a length of waveguide, our layout problem will still be driven by the specified lengths of the interconnections.

CHAPTER 2

THE SYSTEM AND DEVICE DESCRIPTIONS AND MODELS

2.1 Introduction

In this chapter the entire integrated, electro-optic system, as it is envisioned, is described. This is followed by descriptions of the chip and device models used in the layout tool. Finally, some of the main data structures used in the tool are described as well.

2.2 System Description and Chip Model

As mentioned before, the components of time-of-flight circuits consist of electro-optic switches, fixed-ratio couplers, splitters, and combiners which will be integrated in $LiNbO_3$ as described in [45] and shown in Fig. 2.1. Optical waveguides connect the integrated-optic circuit elements and also serve as memory registers. Main memory, as in the discrete version, is implemented off-chip using fiber delay lines. The electronics necessary to drive the electro-optic switches, such as the electrode driver circuitry, the photo-detector and the amplifier, are integrated onto a separate semiconductor chip. This chip will be flip-chip bonded to the $LiNbO_3$ chip using a solder bump alignment technique described in [7]. The solder provides the electrical connections from the drivers to the electrodes, while surface gratings in the $LiNbO_3$ couple light from the waveguides to photo-detectors. The system clock signal is provided by an external semiconductor laser which generates pulses of the correct duty cycle at the desired bit rate.

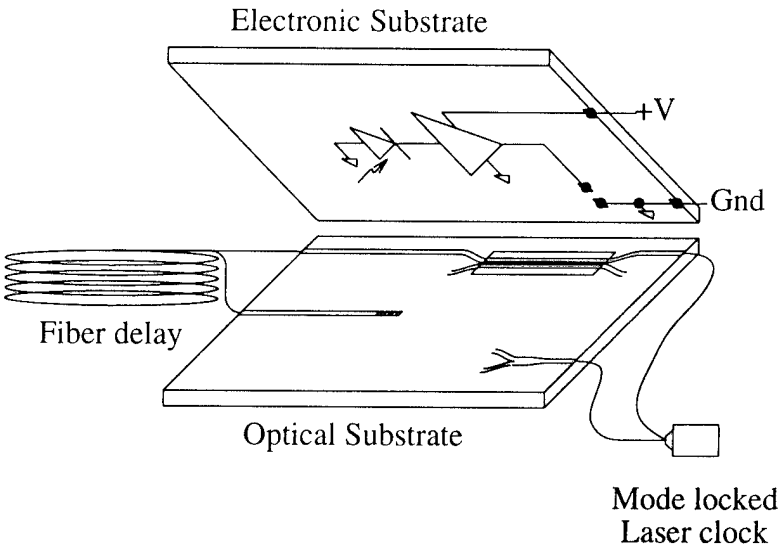


Figure 2.1. The System Design [45]. Note: the element in the upper right corner of the lower chip is a switch.

2.2.1 Switch Description and Model A switch is a five terminal (two inputs, two outputs and a control input) device that models a $LiNbO_3$ optical directional coupler and its associated drive electronics as shown in Fig. 2.2. The switch has two states, known as the "cross" state and the "bar" state. The cross state occurs when the control input is below a specified threshold level. In this state, the optical signal flows to the diagonally opposite output. When the control input signal is above the threshold level the bar state occurs, where the signal flows directly to the corresponding output.

Each switch or directional coupler is fabricated using the proton exchange or titanium in-diffusion process in $LiNbO_3$. A switch is formed by bringing two waveguides together allowing the light to couple between the electrodes. The interaction length of the coupling region is dependent on the strength of the field under the electrodes. The stronger the field, the shorter

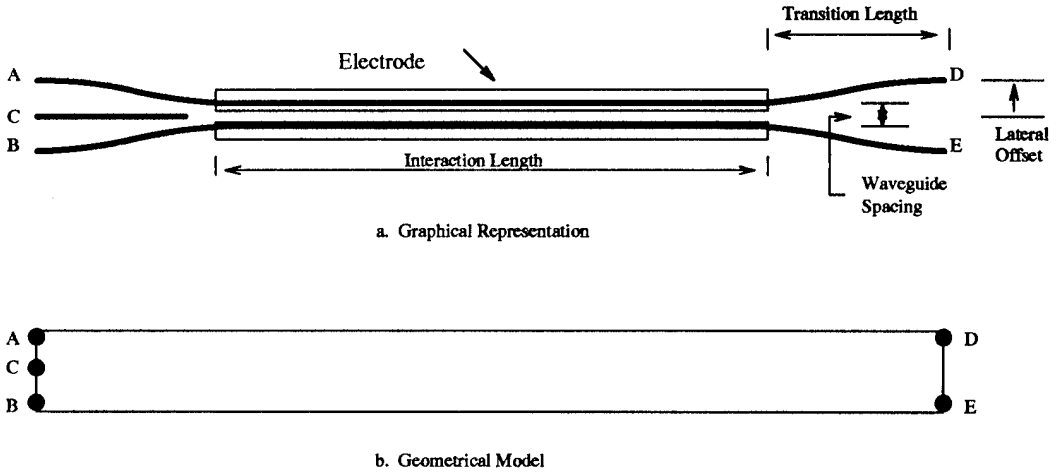


Figure 2.2: Graphical and geometrical models of a switch.

the interaction length. For a voltage of about 4.5 volts, the interaction length is approximately 10 mm [3]. Approximately 1 mm is needed on either end for the waveguides to converge and diverge with a radius of curvature of 10 mm (necessary for negligible loss [8]) into and out of the switch interaction region. Therefore, the total length of the switch is approximately 12 mm. Assuming the width of the electrode structure is approximately 0.1 mm results in a 140:1 aspect ratio which limits the number of suitable placement architectures, as will be discussed later. The entire switch takes approximately 0.012 cm^2, for a packing density of 83 switches per cm^2.

As mentioned before, a third input provides the control signal to drive the switch electrodes. This will be achieved by extending the waveguide which carries the control signal input to a grating coupler which will direct the light to a photo-diode detector on the semi-conductor chip. After detection, amplification, and pulse stretching, the driver signal is connected back to the switch electrode structure via a solder bump.

The switches have several variable parameters as shown in Fig. 2.2.

Each of the model parameters can be changed by the user if the technology changes. All the device dimensions are listed in terms of a scalable unit λ, whose default value is 1 micron, which produces realistic values for the device parameters. The maximum resolution of the layout window is also 1 λ. The origin of the graphical layout diagram is the upper left corner with the x-coordinates increasing to the right and y-coordinates increasing downward. Also, the graphical representation is scalable by the user in both the x and y dimensions. The default scale factors are x/30 and y/5, i.e., each pixel represents 30λ along the x-axis and 5λ along the y-axis.

Each of the switch parameters and their standard values are described below:

- Electrode Width (W_e): This is the width of the metal electrode over the coupling region whose default value is 30 λ. To prevent the electric field from affecting the light in nearby connecting paths, the electrode should not extend beyond the lateral offset (described below) of the diverging waveguides.
- Electrode Length (L_e): This is the length of the metal electrode over the coupling region whose default value is 10,000 λ.
- Long Interaction Length (L_{il}): This is the region under the electrodes where the light couples. The length is the same length as the electrode, whose default value is 10,000 λ.
- Radius of Curvature (R): The radius of curvature determines how much bending loss the light in the waveguide will experience. For negligible loss [8], this value should be greater than 10,000 λ.

- Lateral Offset(O): This parameter depends on the amount of separation needed to prevent the light from coupling between the waveguides. This value is measured from the center of the region between the waveguides to the edge of the upper (or lower) curved waveguide. A nominal value is 30 λ.
- Transition Length (L_t): The transition length depends on the radius of curvature, R, and the lateral offset, O, in the following relationship [8]:

$$L_t = \sqrt{(2RO - O^2)}.$$

  Given a particular lateral offset, the user may specify a transition length and thus derive the radius of curvature or the user may specify a radius of curvature and derive the transition length.
- Waveguide Width (W_w): This is the standard waveguide width, which in practice is between 5 and 10 λ.
- Waveguide Spacing (W_s): The spacing of the waveguides in the coupling region needs to be at most the width of a waveguide. The default value is 10 λ.

For placement purposes, the most important parameters are the length and width of the switch. Assuming that the electrodes do not extend beyond the lateral offset, L, of the outgoing and incoming waveguides, then the device is modeled by a rectangle as shown in Fig. 2.2, whose length and width are respectively,

$$length = L_{il} + 2L_t$$

$$width = 2O + 2W_w + W_s.$$

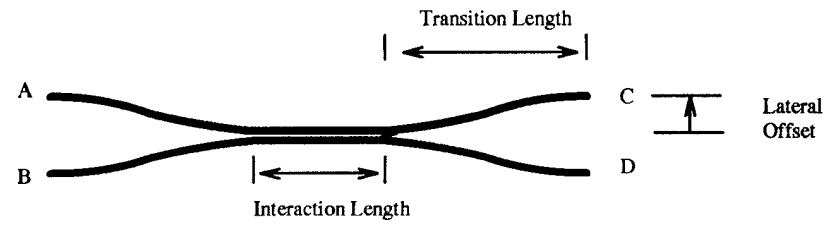
2.2.2 Coupler Description and Model The coupler device, like the switch, has two inputs and two outputs. But, unlike the switch, it does not have a control input. When light enters only one input, the power is split between both outputs. Likewise, when light enters both inputs, the signals are coupled with an overall 3dB power loss. Therefore, depending on how it is connected, the coupler can act as a signal splitter or a combiner.

The coupler is fabricated in much the same way as the switch with two exceptions. First, since there is no electronic control, no electrode is required over the coupler. Also, the length of the coupling region, called the interaction length, is much shorter than that of the switch if the two mode interference (TMI) form of the coupler is used [8]. Like the switch, the transition length required for the waveguides to converge into and diverge out of the coupler depends on the radius of curvature and the lateral offset. The resulting length of the coupler is typically less than half the length of a switch.

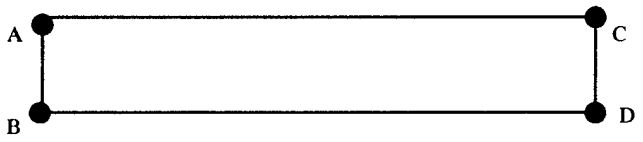
The coupler parameters as shown in Fig. 2.3 are listed below :

- Short Interaction Length (L_{is}): 200 λ.
- Radius of Curvature (R): 10,000 λ.
- Lateral Offset (O): 30 λ.
- Transition Length (L_t): The transition length is calculated from the radius of curvature and lateral offset in the same way as for the switch.
- Waveguide Width (W_w): This is the standard waveguide width.
- Waveguide Spacing (W_s): For the TMI mode the waveguides come together with no space between them.

Again, for placement purposes the most important information is the length and width of the coupler. This is modeled by a rectangle as shown in



a. Graphical Representation



b. Geometrical Model

Figure 2.3: Graphical and geometrical models of a coupler .

Fig. 2.3, whose length and width are respectively,

$$length = L_{is} + 2L_t$$

and

$$width = 2O + 2W_w.$$

2.2.3 Splitter and Combiner Description and Model The splitter device has one input and two outputs. A signal at its input is split between its two outputs where each output signal is half the power of the input signal. The combiner device has two inputs and one output. Signals on both inputs are summed into one output with an overall power loss of 3 dB. Both are special cases of the coupler which is described above. The parameters, as shown in Fig. 2.4, are the same as for the coupler. The splitter and combiner are both modeled by a rectangle, as shown in Fig. 2.4, whose length and width are respectively,

$$length = L_{is} + L_t$$

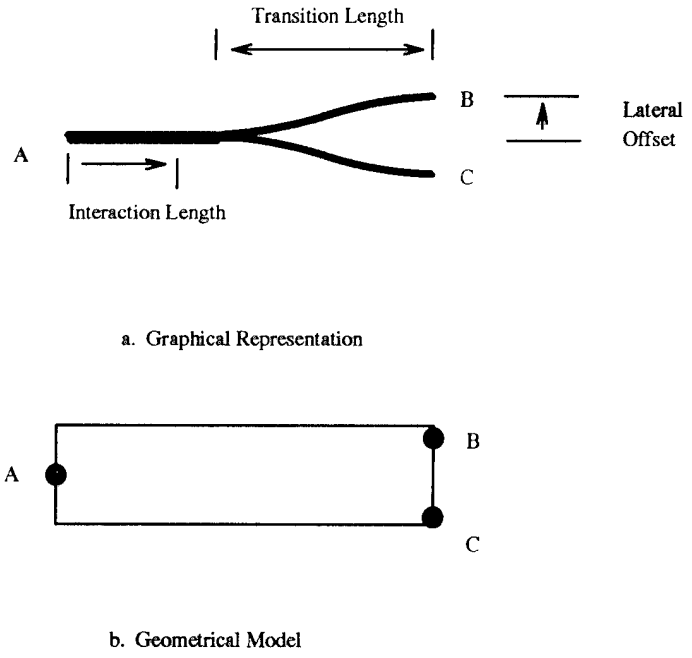


Figure 2.4: Graphical and geometrical models of a splitter or a combiner.

and

$$width = 2O + 2W_w.$$

Whether the device functions as a splitter or combiner depends on its orientation with respect to the signal flow through the device.

2.3 Waveguide and Corner Turning Mirror Description

The connections between the switches, couplers, splitters, and combiners will be formed from proton exchanged lithium niobate PE : $LiNbO_3$ waveguides. These waveguides can intersect with negligible crosstalk, provided the angle formed is greater than 6^o [8]. As with the diverging and converging ends of the switches, the waveguide radius of curvature must be at least 10 mm for negligible bend loss. With this limitation, a full turn will take an area of $4cm^2$, or the entire chip area. This is something that was avoided by anticipating the continued development of corner turning mirrors.

Corner mirrors in GaAs/AlGaAs [9] and in InP [10] have been demonstrated. Using laser ablation techniques in PE : $LiNbO_3$ it may be possible to form 45° turning mirrors to bend the waveguides over a sharp 90° turn. This technique is still under development [11] and the final dimensions of the turning mirror are unknown. Other techniques such as chemically assisted etching [12], or the formation of Bragg grating mirrors using proton exchange methods [11] are being investigated.

Even though the actual implementation is uncertain at this time, it is possible to make some assumptions and model the corner turning mirror as a square whose width, height and orientation are described below.

- Corner Width: The width of the region containing the actual turning mechanism. This parameter is expected be at least the width of a waveguide plus the minimum guide spacing. The default value is 30 λ.
- Corner Height: This is the same as above. Note, in the event that corner turning mirrors turn out to be larger than assumed, then the spacing between parallel connecting paths will also need to be increased.

The direction of the in-coming and out-going path segments determines the orientation of each mirror as shown in Fig. 2.5. For each orientation, the directions of the incoming path and outgoing segments correspond to one of two cases. These are listed as follows:

- Southwest: Increasing y followed by increasing x or, decreasing x followed by decreasing y.
- Southeast: Increasing y followed by decreasing x or, increasing x followed by decreasing y.

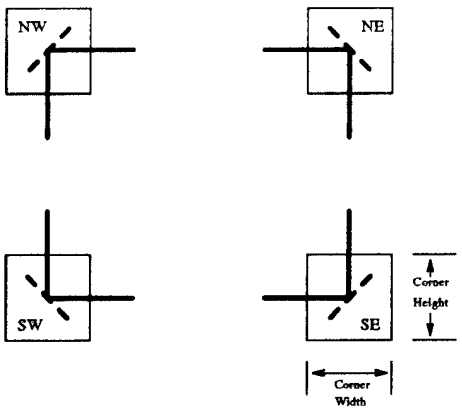


Figure 2.5: Geometrical model of a corner turning mirror.

- Northeast: Decreasing y followed by decreasing x or, increasing x followed by increasing y.
- Northwest: Decreasing y followed by increasing x or, decreasing x followed by increasing y.

2.4 Row Placement Architecture

Due to the highly astigmatic aspect ratio of the switches, the problem of choosing a suitable placement architecture is like choosing the best way to arrange uncooked spaghetti on a tray, where no piece may touch another. If they are not arranged in parallel, the space on the tray will be inefficiently used. Likewise, integrated-optic switches may not touch or cross. Also, to minimize the clock period, the path lengths between elements must be minimal. As with the spaghetti on the tray, the most practical solution is place the elements in parallel rows.

2.4.1 Possible Row Configurations There are several row configurations, depending on the number of elements per row and their relative positions such as:

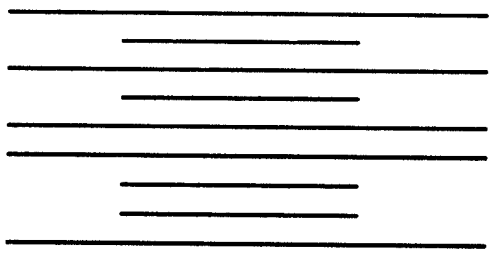


Figure 2.6: Place one element per row.

(a) Place only one device per row as shown in Fig. 2.6. The advantage of this architecture is that the algorithm to place the devices is very simple. However, since every connection must bend 180 degrees to complete its path, at least two corner turning mirrors are required per connection. It is likely that the corner turning mirrors will be imperfect reflectors, therefore, this option will be more lossy than a row configuration which does not require a pair of corner turning mirrors for each connection. Also, since the couplers, splitters, and combiners are so much shorter than switches, placing one of these elements per row is inefficient in terms of area usage.

(b) Place the elements in a fixed grid as shown in Fig. 2.7. This architecture forms two parallel banks of elements, one for the long switches and the other for the shorter couplers, splitters, and combiners. In this configuration, not every connection will need to bend, so corner turning mirrors are not always required. This allows for short intra-row connections, however, the fixed locations of the elements is not conducive to achieving a short clock period.

(c) Place at most one long and one short device, or two or three short devices per row, each in one of several discrete positions as shown in Fig.

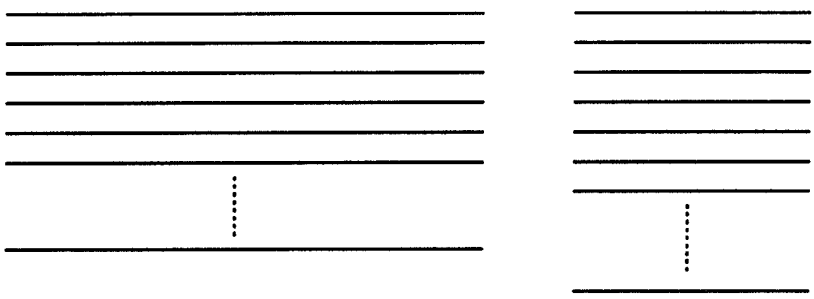


Figure 2.7: Place elements in a grid.

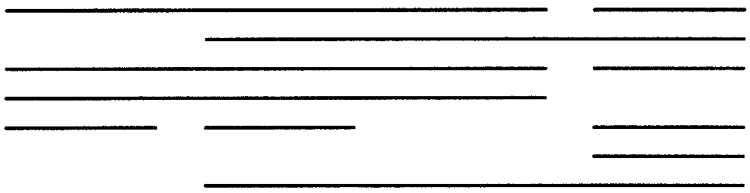


Figure 2.8: Place elements in one of several discrete positions.

2.8. Choosing one of just a few discrete positions simplifies the placement problem. The relative positions of the devices are determined by the direction of the signal flow in the circuit. As in the previous option, this option also allows for short, intra-row connections.

(d) Place zero, one, two, or three devices per row, each in one of several discrete positions. This is the same as above, but a row can be devoid of elements to allow space for several more horizontal tracks for the connections.

(e) Place zero, one, two or three devices per row, each in a variable position. Instead of placing the elements in one of a few discrete locations, the devices are placed as close together as possible. The previous two configurations can be optimized to this configuration by shrinking the shortest intra-row connections till they are as short as possible.

I have chosen the fourth option as the default row architecture in order to allow zero, one, two, or three devices in a row, each in one of several

discrete positions. Each row has a maximum of one switch. It is possible to place a shorter device such as a coupler, splitter or a combiner on either the right or left side of the switch. In the event a switch does not occupy a row, two or three shorter devices can be placed in a row. The spacing between rows is fixed or can change to accommodate a varying number of tracks (in the optimal case). With fixed spacing between the rows, it is possible to skip a row (i.e. not place any elements in it) and use the freed space for additional horizontal tracks.

Now that the row architecture is chosen, the parameters which affect the geometry need to be identified and defined. There are two parameters which affect the row geometry, which are:

(a) Row Spacing. This establishes the spacing between the rows and is measured from the top of one device row to the top of the next device row.

(b) Intra-row Device Spacing: This is the spacing between devices in a row. This value is determined by the maximum number of corner turning mirrors which could possibly be needed for the connections into and out of two consecutive elements in a row. The worst case occurs when a splitter is followed by a switch in the same row. If all the outputs of the splitter and all the inputs of the switch are used, then space for at least five corner turning mirrors (i.e. five tracks) will be needed. Adding two more tracks allows room for pass through connections. The resulting 7-track inner vertical channel is shown in Fig. 2.9.

2.4.2 Placement map A two-dimensional array data structure called the placement map is used to store the device placement information

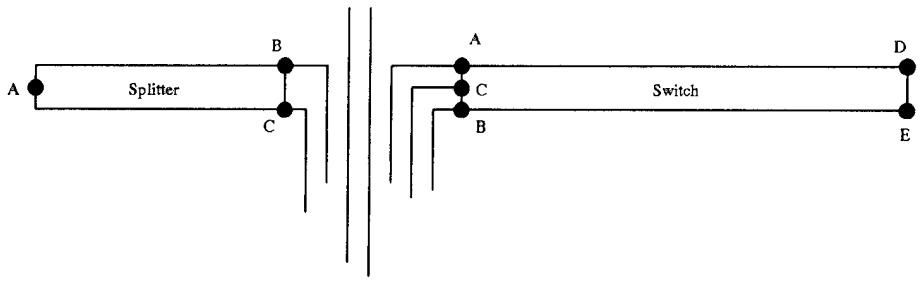


Figure 2.9: Worst case vertical tracks usage between devices.

such as the device position, identification and orientation. Each row of the array becomes a row in the layout and each column represents one of the several discrete positions in which the device may be placed. For the default row configuration, there are three columns in the two-dimensional array. If one or more additional horizontal channels are needed between rows, then the desired rows of the placement map can be cleared and the remaining elements shifted down the desired number of rows.

2.5 Waveguide Channel Model

Since a Manhattan geometry is assumed, the waveguide connections consist of vertical and horizontal path segments. In order to manage the routing of the connections, a grid of horizontal and vertical channels is used. Each channel contains several waveguide tracks. Vertical and horizontal path segments can cross each other arbitrarily, but connecting them requires a corner turning mirror. The track width is equal to the waveguide width plus the waveguide spacing. The maximum number of tracks per channel is a function of the channel width and the width of the tracks. After the user specifies the particulars of the geometry, the tracks can be initialized with their corresponding horizontal or vertical position.

2.5.1 Horizontal Channels The horizontal waveguide path segments are routed in tracks in horizontal channels which are formed in and between the rows of devices and extend through the margins to the border of the chip as shown in Fig. 2.10. Each channel has a fixed number of tracks. For example in Fig. 2.10 the channel consists of eight tracks. Three of these tracks (represented by dashed lines) are used in the device rows and five more fit between the rows. Each track is wide enough to hold a single waveguide. Note, one or more segments from different paths can be placed in a single track.

There are three sets of channels:

- Top Channel. This single wide channel contains the tracks which are formed in the top margin of the chip. The number, N_T, of tracks is determined by the amount of space between the first row of devices and the border of the chip. The tracks are numbered from 0 to $N_T - 1$, starting with 0 closest to the first row of devices.
- Middle Channels. The middle channels are the set of channels which are formed between the device rows as well as in the device rows. There are three tracks in each device row, one each for the segments to the upper, middle and lower terminals of the devices. The number of tracks, k, between each row is determined by the amount of space between rows and the width of the waveguide, as well as the required waveguide spacing. The tracks are numbered from 0 to $N_M - 1$, where

$$N_M = (k + 3) * \text{number of rows in placement map.}$$

- Bottom Channel. This is the opposite of the top channel and contains the tracks which are formed in the bottom of the chip. The number of tracks, N_B, is determined by the amount of space between the last row

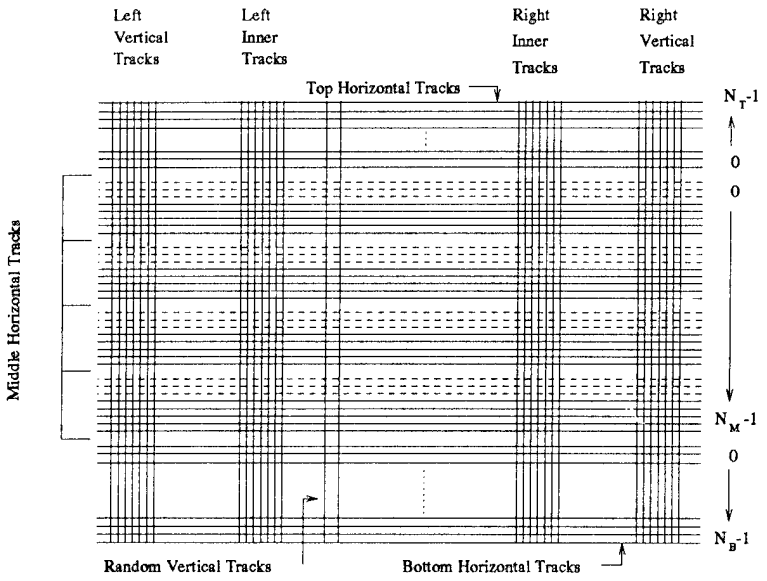


Figure 2.10: Horizontal and vertical tracks.

of devices and the bottom edge of the chip. The tracks are numbered from 0 to $N_B - 1$, starting with 0 closest to the last row of devices.

If more tracks are needed in either top or bottom channels, the chip area will need to be increased. The user will be notified about the change in the geometry and be queried if he/she would like to continue with the layout.

2.5.2 Vertical Channels There is also a set of four single vertical channels and a set of random tracks as shown in Fig. 2.10. These are:

- Left Channel. This channel contains the tracks which are formed along the left margin of the chip. The number of tracks is determined by the amount of space between the leftmost devices and the left edge of the chip. The tracks are numbered from 0 to $N_L - 1$, starting with 0 closest to the left side of the devices.
- Left-inner Channel. The inner channel consists of a minimum of seven tracks to allow at most five terminal tracks, i.e. tracks used to feed into and out of the device terminals, and two pass-through tracks for

long vertical segments which may span several rows.

- Right-Inner Channel. This is the same as the left-inner channel but on the right-inner side of the chip .
- Right Channel. This channel contains the tracks which are formed along the right margin of the chip. The number of tracks is determined by the amount of space between the rightmost position of the devices and the right edge of the chip. The tracks are numbered from 0 to $N_R - 1$, starting with 0 closest to the rightmost position of the devices.

Since connections can cross over devices, vertical tracks are formed as needed between the vertical channels as shown in Fig. 2.10. The spacing of these tracks is monitored to prevent overlapping path segments.

2.6 Waveguide Connection Model

Integrated-optic waveguides will form the connections between the circuit elements. There are two parameters which affect the layout of the waveguides. These are the:

- Connection Width: This is the width of the connecting waveguide and is the same as the width of the guiding regions of the devices. The default value is 8 λ.
- Connection Spacing: This value is measured from the edge of one connecting waveguide to the edge of the nearest connecting waveguide. This spacing must be large enough to fit a corner turning mirror and to prevent light from coupling between the connections.

Several terms are used to describe the various lengths associated with a connection and its path. These are:

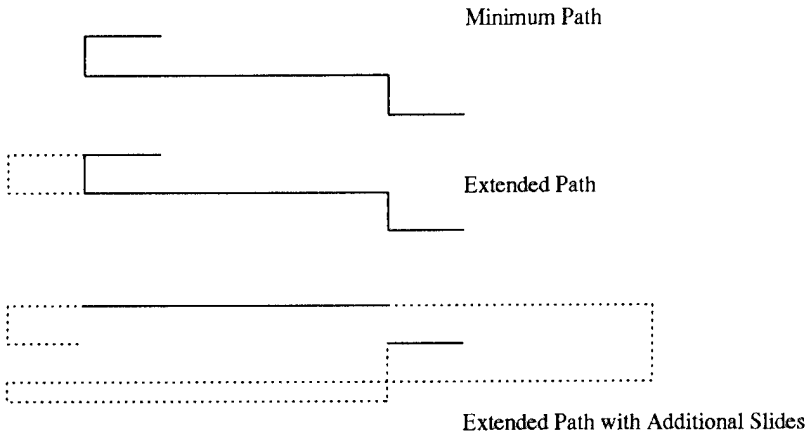


Figure 2.11: Path length variability is achieved by using slides.

(a) Minimum length. This is the minimum path length the connection can have. This value is estimated after device placement and before running the XHATCH delay distribution algorithm.

(b) Connection length. This is the distributed length calculated using the XHATCH delay distribution algorithm. This is the length necessary to achieve circuit synchronization at either a specified clock period or the minimum clock period for the particular set of estimated lengths.

(c) Slack Length. This is the difference between the distributed connection length and the minimum length. After each invocation of the XHATCH delay distribution algorithm, the slack length must be updated.

The success of the routing phase depends on the amount of variability each path has. To achieve this variability, extendible slides are used as shown in 2.11. There are basically two types of slides. The first type are the slides formed by pulling out the vertical segments of the minimum length path like a rubber-band. The second type are slides which are added to the extended minimum path.

2.7 The Chip Geometry

The parameters of the chip geometry are shown in Fig. 2.12 and described below:

(a) Outer dimensions: The chip width and chip height define the outer dimensions of the substrate. Both values are initially set at 2 *cm* or 20,000λ.

(b) Inside geometry parameters: It is necessary to allow some area between the chip edge and the devices. The top, bottom, and side margin regions provide room for this and for the waveguide connections.

(c) Border Width: To prevent handling damage to the circuit, a narrow border region of one track width is formed along the edge where no devices or connections can be formed.

(d) Input/Output ports: Within the lower left and right border there exists a set of ports for the off-chip connections. The number of ports and their relative spacing is specified by the user.

Note, the chip parameters provide a starting point for the layout algorithms. If the chip area needs to be increased to complete the routing, the user will be notified. After routing is complete, the layout tool can provide the user with the actual chip size.

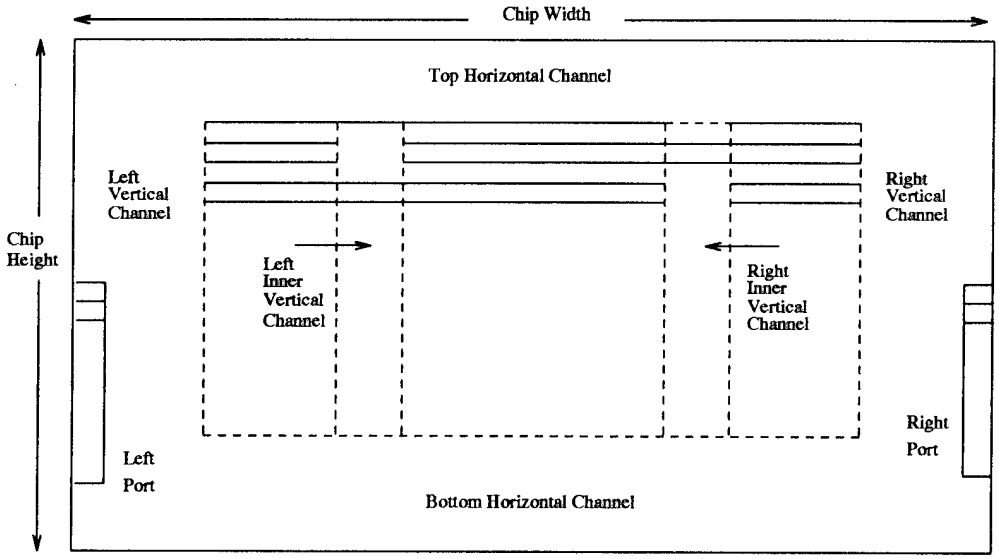


Figure 2.12: Chip geometry.

CHAPTER 3

THE OVERALL APPROACH FOR AUTOMATIC LAYOUT

As mentioned before, the task of laying out a time-of-flight circuit is split into two phases, placement and routing. From a placement perspective, let us assume one switch and one short device will be placed per row. Given there are N switches there will be N! possible patterns in which the N switches can be placed. Additionally, each switch can be oriented in one of two directions, either to the right or to the left. This brings the total number of possible switch placement patterns to $2^N N!$. Likewise, if we assume we have k couplers which also have two orientations and 2N possible positions (either to the left or right of a switch), the number of possible short element configurations is

$$2^k \begin{pmatrix} k \\ 2N \end{pmatrix}.$$

At this stage of development, N is on the order of 100 which is small compared to the millions of electronic devices which are laid out on a single chip using VLSI tools. Nevertheless, it is still advantageous to reduce the complexity of the time-of-flight circuit placement and routing problem.

3.1 Time-of-Flight Circuit Representation

In XHATCH, a time-of-flight circuit is represented as a directed circuit graph $G(V, E)$ where V is the set of vertices representing the elements of the circuit, and E is the set of edges representing the connections between elements [46]. For synchronization purposes, the vertices in the graph are treated as

idealized points of signal interaction and the edges represent directed paths along which the signals of the system travel. For layout purposes, the vertices of G are partitioned into 2 sets V_l and V_s where V_l forms the set of long elements, the switches, and V_s is the set of short elements such as the couplers, combiners and splitters. If an edge $e_i \in E$ then $e_i = (v_s, v_t)$ where $v_s, v_t \in V$.

3.1.1 Definitions The following definitions apply for each edge e_i in E:

- M = the set of minimum Manhattan path lengths where $m_i \in M$ and m_i is the minimum path length for edge e_i. If the two endpoints of edge e_i are x_{si}, y_{si} and x_{ti}, y_{ti} then

$$m_i = |x_{si} - x_{ti}| + |y_{si} - y_{ti}|.$$

- R = the set of required connection lengths calculated using the XHATCH Distribution algorithm where $r_i \in R$ is the required length of edge e_i.
- S = the set of slack lengths where $s_i \in T$ is the difference between r_i and m_i, i.e.

$$s_i = r_i - m_i.$$

- D = the set of drawn interconnection lengths where $d_i \in D$ is the drawn waveguide length for edge e_i. When all the paths have been routed and all the conflicts resolved, then for every path, $d_i = r_i$. If there is an edge e_i where $d_i \neq r_i$, then the path has not been routed correctly.
- The edges of the circuit graph are partitioned into two edge sets, one set E_l for the edges representing the long connections and another set E_s for the edges representing the short connections. Long edges are initially assumed to be those connections marked with an ideal

lumped delay and the short edges are without a lumped delay. This is a reasonable assumption because each lumped delay value is a multiple of the clock period which is typically greater than the time-of-flight through the longest element, a switch. Even though not always correct, the assumption provides a starting point which can be corrected later. Using the long and short edge designations, it is possible to simplify the circuit graph as described below.

3.1.2 The Partial Directed Circuit Graph The simpler the circuit graph, the simpler the placement task. Since the short edges dictate the relative positions of the elements, the long edges can be removed from $G(V, E)$ without any deleterious effect on the relative positions of the placed elements. The resulting graph is called a "partial directed circuit graph" or $PG(V, E)$ where V is the set of vertices, as before, and E_s is the set of short edges representing the connections without a lumped delay. As before, the vertices of G can be partitioned into 2 sets V_l and V_s. If an edge $e_i \in E_s$ then $e_i = (v_s, v_t)$ where $v_s, v_t \in V$. The root vertex v_r is the device which has an input from the oscillator. In the event a sub-graph becomes disconnected from $PG(V, E)$ after removing a long edge, the sub-graph becomes a new $PG(V, E)$ with its own root vertex v_{sr}.

3.2 Placement Theory

The placement process involves three steps which are summarized below.

(a) The first step is to form the initial placement of the devices from the initial $PG(V, E)$ and estimate the minimum connection lengths to allow distribution of the lumped delays.

(b) After the delays are distributed, correct the partition of E and redo the initial placement if the $PG(V, E)$ changed.

(c) If a shorter clock period is desired, the third step is to iteratively improve the placement to shorten the clock period.

3.2.1 Initial Placement using a Partial Circuit Graph The partial circuit graph, $PG(V, E)$, can be traversed using either a breadth-first or depth-first traversal algorithm. Contrary to the breadth-first algorithm, which acts on all the children of a node before going to the next node, the depth-first algorithm searches for the deepest nodes which are acted on before acting on the rest of its children. The breadth-first search traversal algorithm was chosen because the depth-first traversal algorithm may place parallel children far apart. The traversal algorithm can prefer the left-child over the right-child (or vice versa) by placing the preferred child first. Once the circuit graph has been traversed and the elements placed in rows, the physical position of each input and output terminal can be calculated. This provides the start point, $(x_{si}, y_{si}$, and target point, $x_{ti}, y_{ti})$, from which the minimum path length for each connection can be estimated. These estimates are the minimum connection lengths over which the lumped delays must be distributed in order to synchronize the circuit and determine the minimum clock period. The details of the placement algorithms are discussed in the next chapter.

3.2.2 Correct the Edge Partition After estimating the minimum connection lengths and distributing the lumped delays, it is necessary to identify any incorrect assumptions on the initial partition of E into the subsets E_l and E_s. If the partition changes, implying a change to $PG(V, E)$, then a new placement of the devices must be determined. The partitions are checked using

the following two heuristics. If any connections that were initially assumed to be long whose distributed length is

$$r_i < 2m_i$$

must be reinserted into $PG(V, E)$. Likewise, those edges that were initially assumed to be short and whose length r_i is

$$\text{clock period} - 10000\lambda < r_i$$

must be removed from $PG(V, E)$. The bounds on both of the above equations were chosen somewhat arbitrarily after observing the results of the delay distribution algorithm and noticing that the short connections are typically less than $2m_i$. Moreover, the long connections can range in length from a switch length less than the length corresponding to a clock period to several clock periods. As mentioned before, since these devices have such large aspect ratios, the partition of the edges is not critical. Also, these bounds can be changed by the user if desired.

3.2.3 Iterative Improvement of the Placement Since the initial placement does not necessarily produce the shortest clock period, iterative improvement techniques are used to find a placement that produces a shorter clock period. Also, iterative improvement may be necessary if the user specifies a clock period which wasn't achieved after the initial placement. Some possible moves are:

(a) Change device orientation. If a device is oriented to the right, after the change it will be oriented to the left. This type of move will produce large variations in the minimum connection lengths.

(b) Sequential row swaps. Starting from the top of the placement map, successively exchange the row positions of the next two rows of the placement map. At each exchange, calculate the new cost function.

(c) Parallel pair row exchange. Scan the placement map for two successive rows which have their devices oriented in the same direction. For each pair of rows found, exchange them, and calculate the new cost function.

In general, the initial placement using the partial circuit graph, is much better than a random device placement, since the devices are oriented in the direction of signal flow. Changing the orientation of a single device without regard to the respective orientation of its children is not beneficial. The only devices whose orientation can be changed are the roots of those trees which became independent upon formation of $PG(V, E)$. If the children of the sub-root have already been placed, then the sub-root is placed close to its children. Otherwise, it is placed in the next available position.

The cost function at each iteration is defined as

$$C_n = \text{minimum clock period.}$$

The goal of any iterative improvement algorithm is to find a cost function which can be calculated quickly. Determining the minimum clock period involves several iterations of the distribution algorithm, which is not a simple task. Since there is a high degree of correlation between the minimum connection lengths and the minimum clock period, the cost function can be simplified by summing the minimum lengths of the short connections i.e,

$$\hat{C}_n = \sum_{i=1}^{q} m_i.$$

where

$$e_i \in E_s,$$

and q is the number of short connections.

Whether the move is acceptable or not depends on the acceptance philosophy. If a monotonic decrease in cost is preferred, then any move such that $C_n < C_{n-1}$, where C_{n-1} is the cost of the previous placement, will be acceptable. If a simulated annealing schedule is used, then once again, if the cost C_n of the new placement is less than the cost C_{n-1} of the previous placement, the change is kept. However, if $C_n > C_{n-1}$ then the change will be accepted only if

$$r < exp^{\frac{-(C_n - C_{n-1})}{kT}},$$

where

- r is a random number between 0 and 1,
- k is Boltzmann's constant (which is a simple constant in simulated annealing), and
- T is the temperature of the system.

The exponential cooling schedule from [21] is used. It is given by

$$T_n = 0.95 * T_{n-1}$$

where T_n is the temperature of the next step and T_{n-1} is the temperature of the previous step.

3.3 Routing and the Loop Basis Equations

The interdependency of the connection delays is best described by a set of loop basis equations [45]. Any circuit graph can be represented by a set of loop equations which form a basis set or set of fundamental circuits [48] for the graph. Each loop equation represents a loop of the original circuit graph.

As the loop is traversed, the element and connection delays are summed. If the element or connection is aligned with the loop flow, its delay will be positive. Likewise, if the element or connection is oriented opposite to the loop direction, its delay will be negative.

L is the loop basis for the circuit graph where $\mathcal{L}_j \in L$ and l_j is the length of loop $\mathcal{L}_j$.

For each undirected loop $\mathcal{L}_j \in L$ the total length of edges in the loop is

$$l_j = \sum_{k=1}^{m} f_k r_k$$

where m is the number of edges which form the loop and

- $f_k = \ 1$ if edge k is in the loop circulation direction.
- $f_k = -1$ if edge k is against the loop circulation direction.

For time-of-flight circuit synchronization, the total delay for any loop is constant and equal to the number of clock cycles specified for that loop in an initial lumped delay design. Since the element delays are constant and do not affect the routing, the sum of the connection delays around a loop is also constant. To maintain a constant clock period as well as signal synchronization, the individual connection lengths r_k may change so long as the total loop length l_j remains constant. In other words, if an edge in a loop is shortened, another edge in the same loop (and in the same circulation direction) must be lengthened by the same amount. The maximum amount an edge can be shortened is its slack s_i. A loop basis for the counter circuit of Fig. 1.1 is shown in Fig. 3.1. The vertices with the single circle represent the long elements, the switches, while the vertices with the double circle represent the short elements such as a coupler, splitter, or combiner. The resulting loop equations are listed

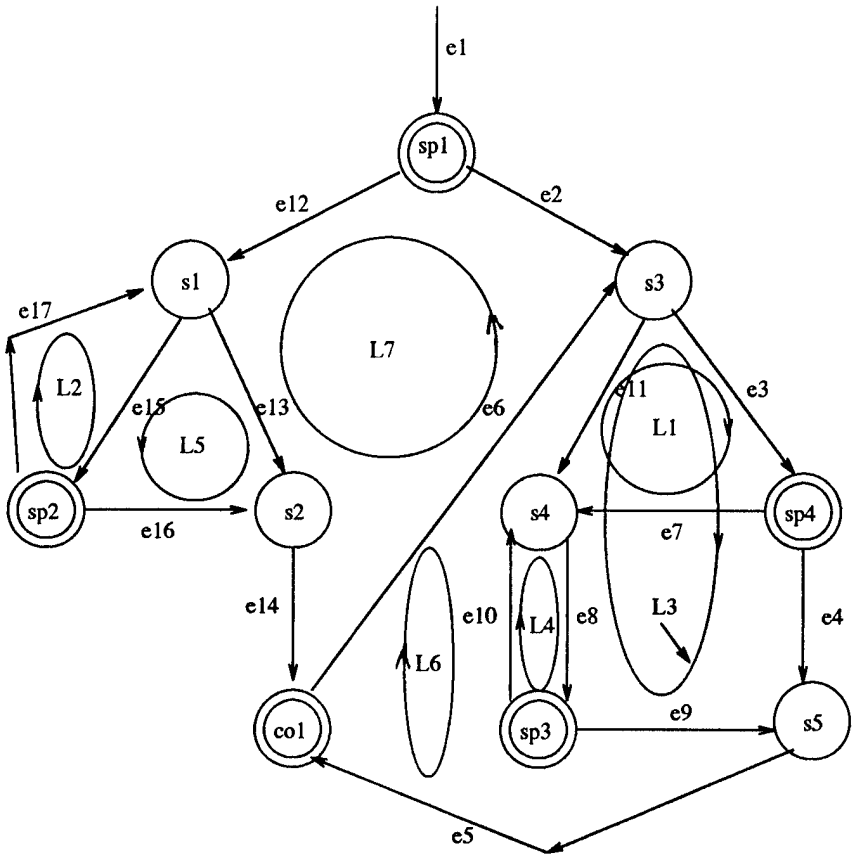


Figure 3.1: A loop basis for the 4-bit word counter circuit.

below. Note, the signs on the lengths r_i correspond to the direction of the loops as drawn in Fig. 3.1.

$$
\begin{aligned}
l_1 &= r_3 + r_7 - r_{11} \\
l_2 &= r_{15} + r_{17} \\
l_3 &= r_3 + r_4 - r_8 - r_9 - r_{11} \\
l_4 &= r_8 + r_{10} \\
l_5 &= -r_{13} + r_{16} - r_{17} \\
l_6 &= r_5 + r_6 + r_8 + r_9 + r_{11} \\
l_7 &= -r_2 + r_6 + r_{13} + r_{14} + r_{12}
\end{aligned}
$$

3.3.1 Forming the Loop Basis Equations The algorithm for finding a loop basis of a circuit graph G(V,E) [48] is as follows:

(a) Find a spanning-tree T and the corresponding co-tree CT of G(V,E).

(b) Clear the set of loops.

(c) For each edge $e_i = (v_s, v_t) \in CT$ do

(1) Find the path from v_s to v_t in T and denote it by P_i.

(2) $C_i \leftarrow P_i \cup e_i$

(3) $L \leftarrow L \cup C_i$

The path from v_s to v_t in T is found by starting at v_s and performing a breadth-first search of T, labeling each vertex $n + 1$, where n is the label of the parent vertex. The search stops when v_t is reached. The path from v_t to v_s is traced by starting at v_t and backtracking through vertices labeled one-less than the current label. This algorithm can be executed in $O(V^3)$ time, since there are $|V| - |E| + 1$ edges in the co-tree, and determining the path for each edge is at most $O(V^2)$.

The loop basis for the circuit graph depends on the spanning tree, and there may be many spanning trees for a particular circuit graph. For layout purposes, the routing problem can be simplified if there is at least one adjustable connection per loop of the loop basis. Using the above algorithm, it is observed that the connections that occur in the loop equations most frequently are the connections that form the spanning tree T. Since the connections with slack are the most adjustable, a spanning tree incorporating these connections would be preferable. Therefore, Kruskal's maximum weight spanning tree algorithm [49] is used to form the spanning tree, where the edge weight is the slack in the connection. A spanning tree for the 4-bit word counter circuit is

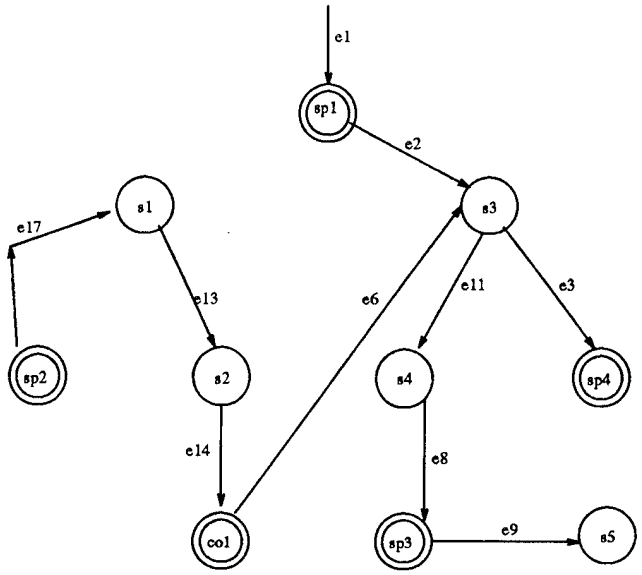


Figure 3.2: A spanning tree for the 4-bit word counter circuit.

shown in Fig. 3.2. To quickly identify other connections which may be affected when there is a conflict, a loop/edge matrix is formed. This a two dimensional matrix where each row represents a loop and each column represents an edge of $G(V, E)$. If an edge is a member of a loop, then the matrix entry will consist of a "1" or a "-1" depending on the orientation of the edge terminal components with respect to the signal flow of the loop. If an edge has only one entry in the loop/edge matrix, it is called an "independent edge". The corresponding loop/edge matrix for Fig. 3.2 is shown in Table 3.1.

3.4 Outline of the Automatic Layout Process

In time-of-flight circuit design, the operating clock speed of the circuit may or may not be initially specified. For instance, if the circuit is to be a subcircuit of an existing design whose clock period is already established, then the subcircuit must operate at the established clock speed. On the other hand, if the circuit is independent, then by minimizing the clock period, the

Table 3.1: The Loop/Edge Matrix for Fig. 3.1.

Loop	Edge															
	e8	e13	e2	e17	e11	e9	e3	e7	e6	e15	e4	e10	e16	e14	e5	e12
l_1	0	0	0	0	-1	0	1	1	0	0	0	0	0	0	0	0
l_2	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
l_3	-1	0	0	0	-1	-1	1	0	0	0	1	0	0	0	0	0
l_4	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
l_5	0	-1	0	-1	0	0	0	0	0	0	0	0	1	0	0	0
l_6	1	0	0	0	1	1	0	0	1	0	0	0	0	0	1	0
l_7	0	1	-1	0	0	0	0	0	1	0	0	0	0	1	0	1

connection lengths and the chip area will be reduced. Therefore, the flow of the automatic layout algorithm depends on whether or not the clock period is initially specified.

3.4.1 Unspecified Clock Period When the clock period is unspecified, the goal is to achieve a routed, synchronized, circuit layout that operates at a minimal clock period. An initial placement is used to estimate the minimum path lengths of each connection. Using Feehrer's algorithm [47], the minimum clock period for this placement can be calculated. This initial placement is iteratively improved until the clock period can no longer be shortened. As the routing of the connections proceeds and conflicts are encountered, effort is made to avoid increasing this clock period. However, it may become necessary to increase the clock period in order to route a path around an existing path.

3.4.2 Specified Clock Period When the clock period is specified, an initial placement is used to form the estimates of the minimum connection lengths. So long as the specified clock period is greater than or equal to the minimum (as would be found above), then the circuit can be routed. If conflicts during routing cause an increase in the clock period, then an alternate

placement should be used.

The steps in the layout process are outlined below and described in more detail in the next chapter. Assume a circuit, with ideal lumped delays and specified device delays, has already been designed using the XHATCH design tool.

(a) Form a partial directed circuit graph $PG(V, E)$.

(b) Do an initial placement of circuit elements based on $PG(V, E)$.

(c) Determine which devices need to be flipped vertically about their horizontal axis.

(d) Estimate the minimum length of each connection. These minimum lengths will be used as a starting point in the XHATCH delay distribution algorithm.

(e) Run the XHATCH delay distribution algorithm to find the required lengths r_i of each connection and the minimum clock period.

(f) If a clock period was specified, determine if it is still satisfiable.

(g) Check if any long and short edge designations change after the delay distribution. If so, repeat the previous steps.

(h) Optimize the placement if a minimal clock period is needed.

(i) Route the connections, shortest lengths first. After establishing each path, identify and resolve design-rule violations. Recalculate the delay distribution if necessary.

(j) If routing is not successful, modify the placement and reroute.

CHAPTER 4

THE ALGORITHMS

This chapter describes the details of the placement, routing, and conflict identification and resolution algorithms.

4.1 Placing the Devices

The goal during the placement phase depends on whether the clock period is specified or not. If the clock period is unspecified, the goal is to place elements connected by short connections as close together as possible in order to reduce the clock period. If the clock period is specified, the goal is to place the devices so the clock period can be achieved. Given the above goals, the major task of the placement phase is to determine the relative position of the devices and their orientation in the placement map.

4.1.1 Formation of the Directed Partial Circuit Graph The directed circuit graph $G(V, E)$ is formed from the connection information obtained using the XHATCH time-of-flight design tool. The root of the graph is the device which is connected to the output of the clock, the laser oscillator. Starting from the root, the symbolic circuit diagram is traversed in a breadth-first manner. Each element is represented by a vertex in the circuit graph and each connection is represented by a directed edge from the start vertex v_s to the target vertex v_t. The more elements in the path originating from the root, the greater the level number of the vertex in the circuit graph, as shown in Fig. 4.1.

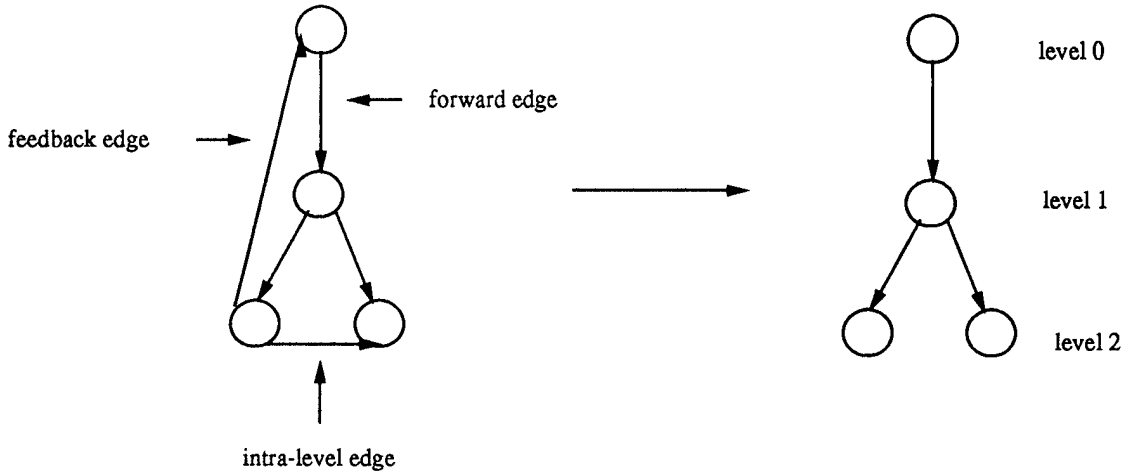


Figure 4.1: Formation of the partial circuit graph.

The major connectivity information (for layout purposes) is contained in the short connections. Therefore, the initial placement is based on the traversal, in the direction of signal flow, of the partial circuit graph $PG(V, E)$ formed from the short edges. This partial circuit graph can be formed directly from $G(V, E)$ by doing a breadth-first search of $G(V, E)$ and ignoring all edges marked with an ideal lumped delay. Marking each vertex the first time it is encountered and not revisiting it after it is marked results in a $PG(V, E)$ which has a tree structure. Therefore, the feedback edge and the intra-level edge shown in the left graph of Fig. 4.1 will be ignored producing the tree structure on the right.

It is possible that the connectivity of the graph is broken upon removal of one or more long edges. When a portion of $PG(V, E)$ becomes independent in this way, a new sub-tree with its own root is created. The vertex which was the target vertex of the long edge, becomes a new root which is stored in an array of roots. The new root and its sub-tree elements are placed after the previous root and its sub-tree elements are placed.

4.1.2 Automatic Initial Placement Once $PG(V, E)$ has been formed, it is traversed using a standard breadth-first traversal algorithm found in [49]. The algorithm uses a queue structure with two simple supporting functions to enqueue and dequeue elements. The enqueue function places node pointers at the bottom of the queue whereas dequeue removes node pointers from the top of the queue. The top and the bottom of the queue are indicated by a pair of top and bottom pointers which are shifted at each enqueue and dequeue operation. The following algorithm places the left child first. It may be beneficial to place the right child first. Therefore the standard breadth first algorithm has been modified to check a Boolean variable called "`left_first`" in order to determine if the left child or the right child should be placed first.

The breadth-first algorithm is as follows:

```
for each root element
   clear the queue
   parent = root
   place parent
   while parent is not NULL
      if(left_first)
         get left child
         if left child is not NULL, place left child
         enqueue left child

         get right child
         if right child is not NULL, place right child
         enqueue right child
```

```
        else
            get right child
            if right child is not NULL, place right child
            enqueue right child

            get left child
            if left child is not NULL, place left child
            enqueue left child

        parent = next element dequeued from the top of the queue
    end while loop
end for loop
```

The algorithm begins by placing the first root node at the upper left corner of the placement map. Once the root has been placed, it becomes a parent. The placement of the children of any parent is determined by the end point of the parent and the length of children. After a child is placed, it is put on the queue where it remains until it is pulled off to become a parent. The process of removing elements from the queue and placing their children continues until the queue is empty.

As mentioned above, for any parent, the placement of its children depends on the end-point position of the parent and the length of the child. In general, a child will be placed as close to its parent's end-point as possible with the same orientation as the parent unless it occupies a new row. How the children are placed is summarized in Fig. 4.2. The top of the figure shows all the possible children configurations (not including the no-child case).

The dashed circle represents a parent, the solid, filled circle represents a short child, and the empty circle represents a long child. Using the default row configuration described in Chapter Two, there are three possible end-point positions for any parent device. If the parent is oriented from left to right, then it can terminate at positions one, two, or three. Likewise, if the parent is oriented from right to left, it can terminate at positions two, one or zero. Each of these conditions forms a case as shown in Fig. 4.2 and described below:

When the parent is oriented from right to left, the three cases are:

- Case 1: If the parent terminates at point 1, there is enough space remaining in the row for a switch. Therefore, regardless of the left-child or right-child first preference, if one or the other of the children is a switch, it will be placed in the same row as the parent. Otherwise the left-child or right-child preference is adhered to.
- Case 2: If the parent terminates at point 2, there is only room left in the row for a short device. If either of the children is short and the other is long, then the short device will be placed in the same row as the parent. If the children are the same length, then the left-child or right child preference is adhered to. If both of the children are short, they will both have the same orientation as the parent.
- Case 3: Here the parent ends at the right side of the row. The children will be placed according to the left or right-child first preference starting a point 3 in the next available row. The orientation of the children is opposite their parent's orientation.

Note that case 0, when the parent is pointed to the right and terminates at point 0, is not allowed.

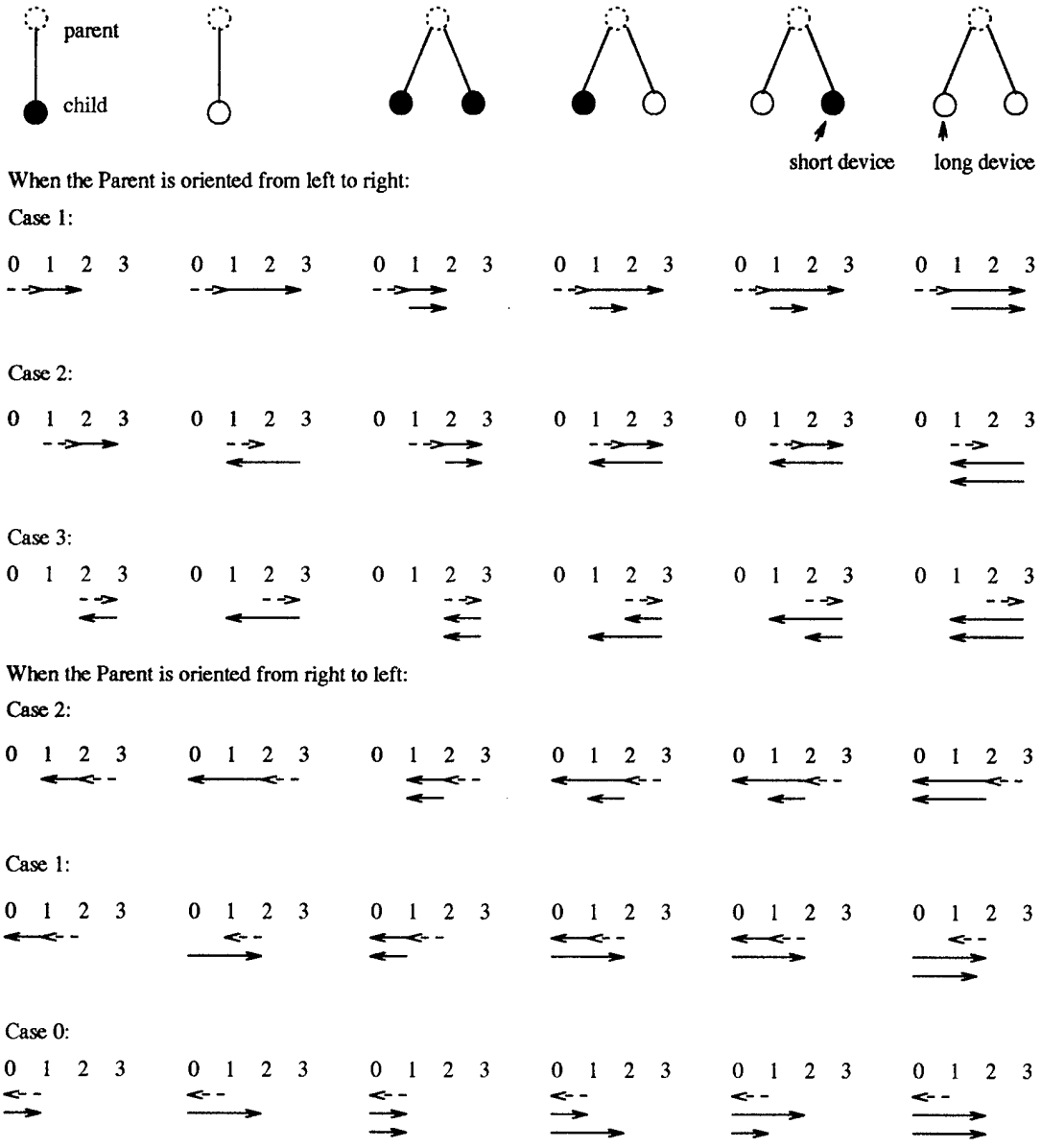


Figure 4.2: Placing the children.

Similarly, when the parent is oriented from right to left, there are again three cases, which are:

- Case 2: This is similar to case 1 above with reversed orientations.
- Case 1: This is similar to case 2 above with reversed orientations.
- Case 0: This is similar to case 3 above with reversed orientations.

4.1.3 Flipping Elements Each device has a horizontal and a vertical axis around which it can be flipped. Since flipping a device causes the position of the input/output terminals to change, all element flips must occur before the minimum path length of each connection is estimated. Also, because the devices are oriented in the direction of signal flow when they are placed, flipping around the vertical axis is not beneficial. However, it may be advantageous to flip the devices around their horizontal axis. First, the long switches are flipped and, if necessary, then the short elements such as splitters, combiners or couplers are flipped. A switch is flipped by exchanging the y-position of the A and B input terminals and the y-position of the D and E output terminals. The C input terminal remains the same. Likewise, for the coupler and combiner, the y-position of the two input signal terminals are exchanged and for the coupler and splitter the y-position of the two output terminals are exchanged.

The algorithm to determine whether or not to flip a device depends on the basic path of the connection into each terminal. If the path into or out of one terminal crosses in front of the opposing terminal then a counter is incremented. When the value of the counter exceeds half the number of connected input/output terminals for the device, then the element is flipped. The resulting pseudocode for determining whether or not to flip a switch or

coupler is presented below:

```
Scan through device list
  if a switch is found
      for each input (except the C input for a switch) and output
 if the basic path passes by the other input or output, then
    increment a counter.
      if the counter value is >= half the number of connected I/O
      terminals,
 interchange the y-positions of the inputs and
 interchange the y-positions of the outputs.
```

The algorithm to determine whether or not to flip the inputs or outputs of a short device is the same as above, except only the inputs of a combiner or the outputs of a splitter are checked.

4.1.4 Estimate Minimum Path Lengths Estimating the minimum path lengths involves the straightforward calculation of the Manhattan distance between the start and target terminals of each connection. Then, a certain amount of length, depending on the number of 90 degree turns in the basic path, is added to provide space to connect into and out of the corner turning mirrors.

4.1.5 First Distribution After the the minimum path lengths have been calculated, Feehrer's version of the distribution algorithm is used to calculate the minimum clock period for the placement. This results in a set of connection lengths which will be close to the final results. This is assumed because the anticipated changes to the lengths during routing are

typically small, just enough to clear one or more corner turning mirrors (which is 20-30 microns in either dimension). Therefore, compared to the length of a switch (approximately 12000 microns), the changes are relatively small. If the user provided a clock period and the ideal lumped delays were successfully distributed, then the layout process can proceed directly to the routing phase.

4.1.6 Final Placement of Devices If the goal is to achieve a minimum clock period, then the first step after the initial placement, is to correct for any wrong initial assumptions about the designation of long and short edges. Therefore, after the first distribution, the distributed length of each connection is compared with the original assumption. If there are any long edges (those with lumped delays) which actually end up being short, or vice versa, it is necessary to reform the partial circuit graph with the updated list of short connections. The placement map is cleared before the updated circuit graph is traversed and the elements placed once again. The flipping algorithm is run before the minimum paths are estimated. After calculating a new set of delay lengths and minimum clock period, the placement is ready to be optimized.

4.1.7 Iterative Improvement of the Placement After trying both the left-first and right-first child options, the placement of the devices may be further improved by swapping rows of the placement map. This simple algorithm begins by exchanging the y-position of the top two rows. After calculating the new positions of the device terminals, the minimum length of each connection is estimated and the minimum clock period is found. If this clock period is less than the previous clock period, the exchange is kept, otherwise, the rows are restored to their original positions. The above series

of steps is repeated for the second and third rows, and so on, till all sequential pairs of rows have been tested.

After the last delay distribution, when the slack of each connection is updated, if a path has a slack which is less than the minimum slide length, then the delay of the path is locked to it's minimum path length. If this occurs, it is necessary to recalculate the delay distribution once again to ensure the system will synchronize, even though one or more connections are locked to their minimum length.

4.2 Forming the Connecting Paths

The routing of the waveguide connections between the circuit elements is the next major step in the layout of integrated-optic time-of-flight circuits. Since connecting paths can cross each other as well as other devices, obstacle avoidance is not a problem. However, path segments may not overlap. Luckily, due to the use of corner turning mirrors and a Manhattan geometry, the connections are very structured entities and have a versatile representation which promotes the identification and repositioning of the segments.

The routing of the path segments defines a series of numbered points, starting with the start point and terminating at the target point of the connection. The start point is point numbered zero and the target terminal point is numbered either 3 or 5 depending on whether the connection has a minimum of 3 or 5 segments. For generality, the target terminal is labeled x_f, y_f, and the point next to the last point is labeled x_{f-1}, y_{f-1} see Fig. 4.3.

Some terms which are used describe the connecting path are presented below.

- Basic Path. This is the minimum length path of the connection before

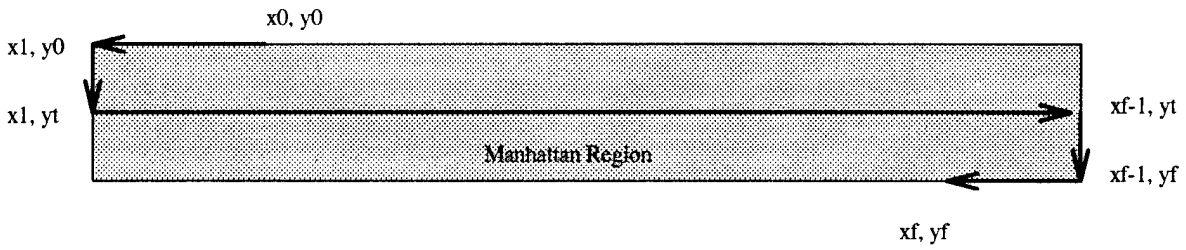


Figure 4.3: Basic points and Manhattan region of a path.

any additional length is added to increase the delay for signal synchronization.

- Path Segments. These are the horizontal and vertical sections which form the path of the connection. The basic path contains either 3 or 5 segments depending on the type of the connection. More on connection types will follow.
- Start Point. This defines the coordinates of the output terminal of the source device. This is always point zero in the Point Array.
- Target Point. The coordinates of the input terminal of the sink device.
- Point Array. This is the series of points which determine the basic path, starting with the start point and ending with the target point.
- Even Point. This is a point in the point array with an even index, which indicates the start of a horizontal segment.
- Odd Point. This is a point in the point array with an odd index, which indicates the start of a vertical segment.
- Manhattan Region. The area formed by all possible minimum length paths is called the Manhattan region, see Fig. 4.3.
- Basic Extension. A small horizontal extension formed at the output or input of a device when the path needs to change direction by 180 degrees, as shown in Fig. 4.4. This extension provides room for a pair

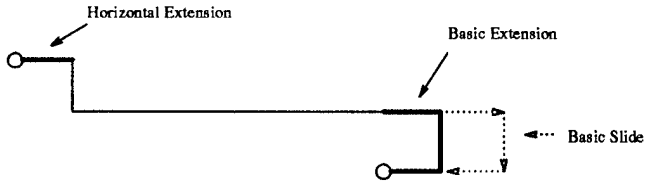


Figure 4.4: The Basic Extension.

Figure 4.5: Left and Right Adjustable Slides.

of corner turning mirrors. The length of this extension is added to the estimate of the minimum connection length before the first delay distribution.

- Basic Slide. This is an adjustable slide which occurs along the basic path within the Manhattan region as shown in Fig. 4.4.
- Designated Point. The point in the point array after which the additional slides are inserted. This point is predetermined for each of the connection types to be defined in the next section.
- Additional Slide(s). To create long path delays, additional length is added to the basic path in the form of slides. An additional slide is a series of three segments, connected like a trombone slide extension (i.e. two horizontal segments and a connecting vertical segment), represented by a series of four points (0,1,2, and 3). These slides are connected to the basic path after the designated point in the point array and can extend beyond the Manhattan region as shown in Fig. 4.6. The length of a slide can be varied by repositioning the x-coordinate of points 1 and 2 as shown in Fig. 4.5.

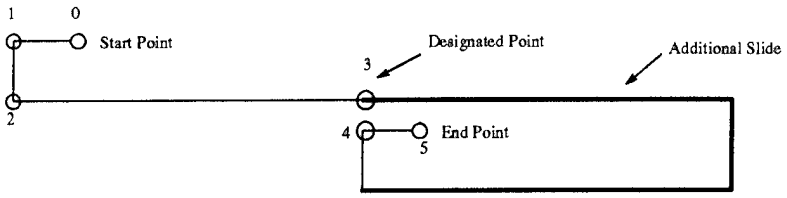


Figure 4.6: An additional slide added to the basic path.

- Slide Array. If more than one slide is needed to create the required path delay, then an array of additional slides is formed after the designated point. The path formed by the slide array continues from the designated point through the three segments of the slide. The last point of the previous slide becomes the first point of the next slide and so on till the path returns to the basic path at the point after the designated point.

4.2.1 The Six Path Types If all possible basic paths between all possible start and target point positions are drawn, a set of six types emerges. Each path type has a characteristic shape which is shown in Fig. 4.7.

The details of each path type are described below:

(a) Type 1: This path has 6 points which determine 5 basic segments, three horizontal and two vertical. For this type of path and the subsequent types, there is a short (the length of a basic extension) horizontal segment originating at the start point and another short horizontal segment leading into the target point. An additional horizontal track segment must be found for the segment between points 2 and 3. The connection has two basic extensions, and the potential to form zero, one or two basic slides in the Manhattan region depending on whether there are any terminals on the far left or far right sides of the layout.

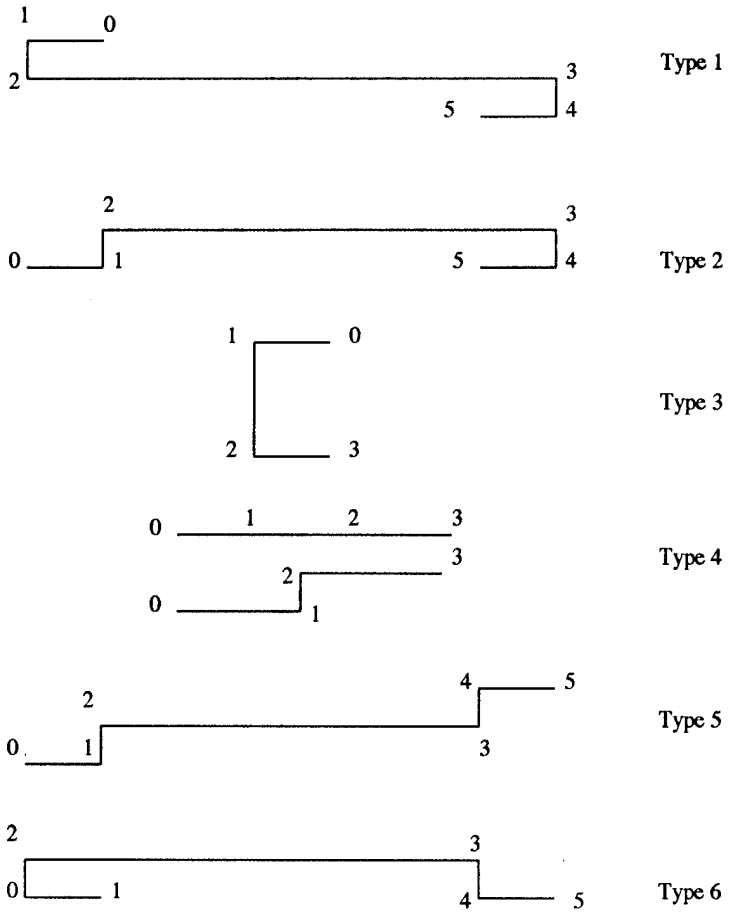


Figure 4.7: Connection types.

Any slack in the connection needed for signal synchronization, is taken up by slides formed after point 3.

(b) Type 2: This path, like type 1, also has 5 segments determined by 6 points. Again, a horizontal track segment must be found for the segment between points 2 and 3. The connection has a single basic extension between points 3 and 4, which may be extended to form a basic slide if the terminals of the path are either on the far left or far right sides of the layout. Any slack in the connection is taken up by additional slides formed after the designated point, point 3.

(c) Type 3: This path has basic 3 segments determined by 4 points. No additional horizontal track segments need to be found. The connection has one basic extension which can be extended to form a basic slide if the path is formed in either the left or right margin. Any slack in the connection is taken up by additional slides formed after point 1.

(d) Type 4: This path type forms the short, cross-channel, intra-row paths. This type of path is formed by 3 segments determined by 4 points. Like the type 3 path, no additional horizontal track segments are needed for the basic path. Any slack in the connection needed for additional delay is taken up by slides formed after point 1.

(e) Type 5: Like path types 1 and 2, these paths have 5 basic segments determined by 6 points. An additional horizontal track segment needs to be found for the segment between points 2 and 3. The path has no room any basic slides, nor does it require additional length for the corner turning mirrors. Any slack in the connection needed for additional delay is taken up by slides formed after point 3.

Table 4.1: The path type for each start and target position.

Start	Target							
	LE	LS	Li-L	Li-R	Ri-L	Ri-R	RS	RE
LE	4	4	2	5	2	5	2	5
LS	4	3	1	6	1	6	1	6
Li-L	6	1	3	4	2	5	2	5
Li-R	5	2	4	3	1	6	1	6
Ri-L	6	1	6	1	3	4	2	5
Ri-R	5	2	5	2	4	3	1	6
RS	6	1	6	1	6	1	3	4
RE	5	2	5	2	5	2	4	4

(f) Type 6: This path is the mirror image of path type 2. It has one basic extension between points 1 and 2, which may also form a slide if it extends into either left or right margins. Again, point 3 is the designated point from which the additional slides can be formed.

The abbreviations for the different start and target positions are:

- LE: left edge.
- LS: left side.
- Li-L: left-inner column, left side.
- Li-R: left-inner column, right side.
- Ri-L: right-inner column, left side.
- Ri-R: right-inner column, right side.
- RS: right side.
- RE: right edge.

Table 4.1 lists all the possible combinations of start and target point positions and the resulting path type. Note that the edges which start or terminate at either the left or right edges are the I/O connections. After determining their path type, I/O connections are treated as any other connection.

Once the path type for a given connection is determined, other information about the path is simple to deduce based on the position of the start and target points. The data accompanying each path is summarized below:

- The type identification code, for the path types 1-6 as shown in Fig. 4.7.
- The number of points in the connection (either 4 or 6), depending on the number of segments in the basic path.
- The number of basic extensions, which depends on the number of 180 degree turns the basic path makes.
- The number of basic slides, which is determined by the number of basic extensions which have room to extend even further to form an adjustable slide.
- The first basic slide position. This is the point which identifies the start of the first basic extension which can be used as a basic slide.
- The second basic slide position. If the path has two basic slides, this point identifies the start of the second basic extension which can be used as a basic slide.
- The index of the designated point after which additional slides are formed.
- Is a horizontal track needed for the basic path? This variable is true if the basic path has a long horizontal segment which needs to be placed in a horizontal track such as for path types 1, 2, 5 and 6.

For example, for a type 1 path with a start terminal on the left-most side of the devices and a target terminal on the left side of the right vertical channel, the above variables are set as follows:

- The type identification code is 1 for a type 1 path.
- There are 6 points in the basic path.
- There are 2 basic extensions. The first is needed for the 180 turn between points 0, 1, and 2 and the second for the 180 turn between points 3, 4, and 5.
- Since the basic extension in the right vertical channel can not be extended beyond the right side of the channel without coinciding with the inputs of the next device, only the left-most basic extension has room to extend further. Therefore, there is only 1 basic extension.
- The first basic slide position. The start point of the only basic extension is point 1.
- The second basic slide position. This is zero since there is only one basic extension.
- The index of the designated point after which additional slides are formed is point 2. This is the best point after which to form additional slides because if a horizontal track is needed for the basic path, this track segment can be extended along the path direction to form the first slide.
- Is a horizontal track needed for the basic path? Yes, one horizontal track is needed between points 2 and 3.

The data for the other paths is listed in Appendix A.

4.2.2 The Routing Algorithm Assuming all the devices have been entered into the placement map, the overall flow of the routing algorithm is described below.

(a) Sort the connections according to their distributed lengths.

(b) Route the connections starting with the shortest lengths first.

(1) Based on the type of the connection, form its basic path.

(2) If the slack of the connection > 0:

(a) Extend the basic slide(s).

(b) While there is still unrouted slack length, form additional slides till all remaining slack has been routed.

(3) Identify and resolve conflicts with other connections.

The details of each of the above steps are explained in the following sections.

4.2.3 Sort the connections by length During routing, as more connections are routed, more length is needed to go around the previously placed connections. A natural routing order is one which routes the short connections first and the long connections last. Therefore, the connections are sorted using the quicksort algorithm in [49]. Quicksort was chosen because it sorts in place and runs in $O(n \lg n)$ time on average. This time is no worse than that of other sorting algorithms such as merge sort, which does not sort in place, or heap sort, which quicksort generally outperforms [49].

4.2.4 Route the Paths The goal is to draw each connection so that its drawn path length equals its required length. The routing of each connection in the sorted list is accomplished in three steps which are:

(a) Form the basic path. The basic path consists of the minimum Manhattan path from the start point to the target point. Once the path type is known, the routing of the basic path is straightforward since there is essentially only one unknown coordinate of the basic path. The reason for this is that the first and last segments of each path are always short

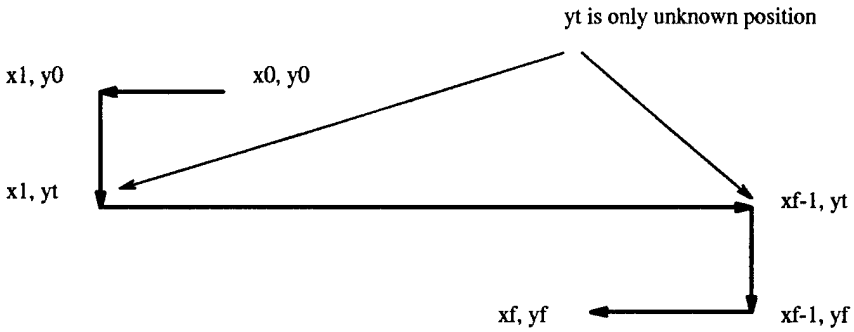


Figure 4.8: One unknown position in the basic path.

horizontal segments the length of a basic extension. Since this length is constant, points x_1, y_1 and x_{f-1}, y_{f-1} are easily determined. The remaining point in the basic path to be found is the y-position of the horizontal track segment for those connections requiring a horizontal track segment. Therefore, the only unknown in the routing of the basic connection, is the y-position of the middle horizontal segment, see Fig. 4.8. The first available horizontal track in the Manhattan region is used. However, if no tracks are available in the Manhattan region, the search continues outside the region stopping when a track which can store the desired length segment, is found.

(b) Extend the basic slide(s) if necessary. If the slack, which is the difference in length between the distributed and minimum delays, is greater than zero, then the path delay needs to be increased. This is done by extending the basic slide(s) to the chip border.

(c) Form additional slides. If there is still unrouted slack, s_r, after extending the basic slide(s), then additional slides must be added to the path. The three segments of each additional slide are stored in a four point data structure which is inserted in a slide array which is scanned after

the designated point in the basic path. The four points of the slide array are determined as follows:

- Point 0: The x_0 is equal to the x-position of the designated point. If this is the first slide and if there is space on the track, then y_0 is the same as the y-position of the designated point. If a new track is used, then y_0 is the position of the first available horizontal track.
- Point 1: x_1 is determined by the magnitude of s_r. The distance from x_0 to the furthest edge is estimated and multiplied by two. If the remaining unrouted slack is greater than this, then x_1 is the x-position of the furthest edge, otherwise $x_1 = x_0 + s_r/2$, and $y_1 = y_0$
- Point 2: $x_2 = x_1$ and y_2 is the position of the next available track.
- Point 3: x_3 is the x-position of the designated point if the first point of the first slide is within the Manhattan region. Otherwise, x_3 is shifted over to prevent the return path from coinciding with the the vertical segments between the slides, and $y_3 = y_2$.

After the points of the slide of length l_s are found,

$$s_r = s_r - l_s - 2 * l_d$$

where l_d is the y-deviation from the Manhattan region which is determined by the position of y_3 relative to the target point of the connection.

(1) If y_3 is within the Manhattan region, $l_d = 0$.

(2) If this is the first slide which has gone outside the Manhattan region, then $l_d = y_3 - y_t$, where y_t is the target point of the path.

(3) If this is the second or any other succeeding slide which has gone outside the Manhattan region, then

$$l_d = y_3 - y_3^{'}$$

where $y_3^{'}$ is the y-position of the last point of the previous slide.

4.3 Resolving Conflicts

While forming the basic path, the basic extension(s), and the additional slides, the path may run into another path. If the new path crosses over another path at right angles, there is no problem. However, when two segments overlap and share the same portion of a track, there is a design-rule violation or a conflict. As each successive path is routed, there is an increasing chance a new path will overlap a previously routed path. For example, in the counter layout, connections e9 and e10 overlap between points 1 and 2 as shown in Fig. 4.9. The two connections which overlap form what is called a "conflicting pair". Conflicts can occur with either horizontal or vertical path segments. While checking for conflicts, a new segment may partially or completely overlap an existing segment or the existing segment may lie within the new segment. When an overlap is detected, either the path of the new connection or the path of the previously routed connection must be altered to avoid the conflict. There are two possible approaches to solving this problem which are described below.

4.3.1 Resolve Conflicts Before Routing One approach is to try and predict where the conflicts may arise before the connections are actually routed. The basic path of the connection can be predicted once the position of the start and target points are known. If the basic path of one connection has

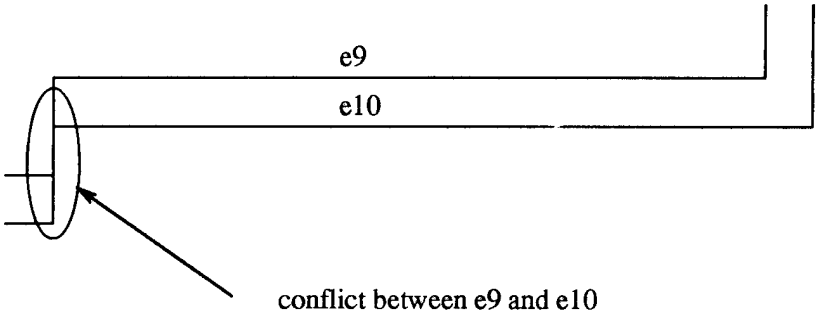


Figure 4.9: Example of a conflict between two connections.

a segment which will overlap the basic path of another connection, then there is a high chance there will be an overlap during routing. A certain amount of length needed to avoid the conflict can be added to the estimate of the minimum path length and accounted for in the delay distribution. The problem with this approach is that it impossible to accurately predict the entire path, including the horizontal track segments as well as the slides, before it is actually routed. This is because the availability of space in the tracks depends on the paths of the previously routed connections. The actual path of any connection is not entirely known till the connection has been completely routed.

4.3.2 Resolve Conflicts During Routing A more direct approach, and the approach used in this work, is to resolve conflicts during the routing stage. There are three ways this can be done.

(a) The simplest approach uses the most basic form of conflict resolution. Conflicts are identified while the path is being formed. If one occurs, the conflicting segment of the new path is simply repositioned to avoid the conflict. After the conflict free path has been formed, the total drawn length is compared to the required length. If they differ, the drawn length of the new path is entered as its minimum length and the delays of the system are redistributed. Any previously routed paths

whose required length changed as a result of the redistribution, are adjusted.

(b) Another approach is to check for and resolve conflicts as the path is formed. If a conflict is encountered, there are two choices, either adjust the new path or else adjust the old path. If either segment in the conflict pair can be repositioned without affecting the total path length, or causing a new conflict with a horizontal segment, then resolving the conflict is simple. If neither path can be adjusted without changing its path length, then extend one connection beyond the conflict and redistribute the system. It is possible the clock period may increase. Usually, the path length increase is on the order of a few tens of microns corresponding to a delay change on the order of 0.1 ps, so the overall effect on the clock period will be small. Once the conflict is resolved, routing proceeds where it left off.

The major advantage of this approach is that it is not necessary to make adjustments on existing portions of the path. However, it may be necessary to make adjustments on a previously routed path.

(c) The third approach is essentially the same as the previous approach except that the vertical segments are scanned for conflicts only after the entire path and slides have been formed. Horizontal conflicts are identified and resolved during the routing of the basic path. The advantage of this approach is that the algorithms for adjusting the current path are the same as for adjusting an existing path.

Note, in all three cases, if a conflict occurs with a horizontal segment, the path is simply shifted to the next available track. As long as the track is

within the Manhattan region, the path length remains constant. The choice of horizontal track is made without checking for conflicts within the vertical tracks. This tactic loads the task of conflict resolution onto the vertical segments. Therefore, the rest of the discussion involves the identification and resolution of conflicts between vertical path segments.

4.3.3 Path Variability In order to resolve a conflict, the path of one connection in the conflicting pair needs to be adjusted. This adjustment can be simple if the conflict occurs with a segment which can be repositioned without affecting the path length. Therefore, how each conflict is resolved depends on the relative position of the conflict in the path. When a conflict is identified, three segments are stored: the segment just prior to the conflict, the conflicting segment, and the segment following the conflict. There are essentially two different configurations these three segments may have, depending on whether or not the signal changes its direction after passing through the vertical segment, as shown in Fig. 4.10. In the first case (as shown in the upper half of the figure), the direction of signal flow in the two horizontal segments is the same. Whereas, in the lower half of the figure, the direction of signal flow in the upper horizontal segment is different than in the lower segment. In the first case, repositioning the vertical segment, a "variable segment," has no affect on the path length, while in the second case it does. Several path types (types 2, 4, 5 and 6) have one or or two variable segment(s) within their basic path as shown in bold in Fig. 4.11. Portions of the basic path routed between slides may also be variable. If a path has two or more slides (either basic or additional) then it is considered to have variable segments.

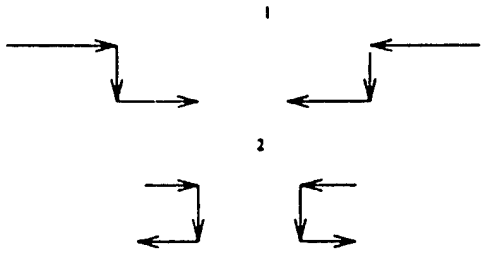


Figure 4.10: Two Vertical Path Configurations.

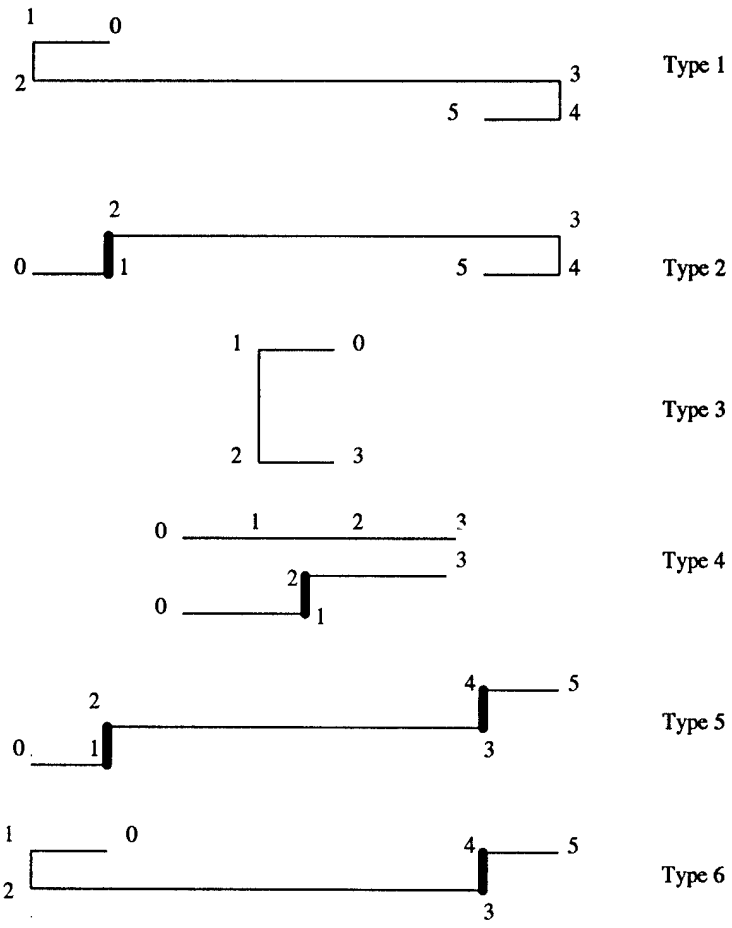


Figure 4.11: Variable segments in each Connection type.

4.3.4 Conflicts Between Connections With Invariable Paths

If both of the segments in the conflicting pair are invariable, then it will be necessary to increase the length of a path in order to circumvent the conflict. The loop basis equations are used to determine which connection of the conflict pair will have the least impact on the other connections when its length is changed. Once a path length is increased, it may be necessary to adjust several connections in other loops in order to achieve synchronization. If this situation results in a set of connection delays for which the circuit cannot be synchronized, then a new clock period must be found.

4.4 The Steps to Identify and Resolve Conflicts

Regardless of which phase of the routing process, either during path formation or after, the steps for conflict identification and resolution are the same. These two steps are described below:

4.4.1 Conflict Identification While the path is formed, or after it has been established and before routing the next connection, the coordinates of the point and slide arrays are scanned. As the segments are scanned, the vertical segments, identified by an odd numbered start index, are compared to the other segments in the track associated with the particular y-position of the segments. Conflicts are identified by comparing the endpoints of the new segment with the endpoints of the existing path segments and checking if either one overlaps the other. The identity of the connection containing the other conflicting segment is determined at the time of conflict identification, but not the indices of the conflicting segment in the other connection. Therefore, in order to determine the exact position of the conflict in the existing path, the point and slide arrays of the other connection are scanned until the position of

the segment matches the position of the conflict.

4.4.2 Conflict Resolution As mentioned above, how a conflict is resolved depends on whether or not the conflict occurred with a variable segment. The simplest case is one where the conflict occurs with a variable slide segment as described below. The situation becomes slightly more complex when the conflict occurs on a variable segment of the basic path. Even though repositioning the segment in question will not affect the path length, it is not always obvious in which direction to shift the segment. This depends on the path type and the specific position of its basic extensions. Finally, if the conflict occurs on an invariable segment, it may be possible to reposition the segment if the connection had a slide which could counteract the change in length. Therefore, the first step is to determine whether the conflict is in the slides, or part of the basic path (including between the slides). Once this is determined, then the following algorithm is used to route one or the other of the paths around the conflict. In the following discussion the term "new connection" refers to the most recently routed path and "old connection" refers to the previously routed connection of the conflicting pair.

4.4.3 When the Conflict is in the Slides The simplest conflict resolution case is when the conflict occurs with the vertical segment of a slide. When this is true, and there is another slide which can offset the path length change, the slide is extended beyond the conflict and another slide is pulled in to compensate for the delay change. In paths with many slides, all the slides, except the last, will be extended to the far left or right side of the chip. Thus, a conflict will most likely occur in a slide prior to the last slide in the array. Therefore, in most cases, a conflict in a slide segment can be resolved

by reducing the length of the slide containing the conflicting segment and increasing the last slide an equal amount to maintain a constant path length. If the conflicting segment is not at either edge, then it can be repositioned out towards the edge, while the last slide is decreased in length. If the conflict occurs in the last slide, then the last slide is increased and a previous slide decreased. Note, any path adjustment is preceded by a check of the horizontal tracks to ensure the adjustment will not cause a conflict there.

4.4.4 When the Conflict is in the Basic Path If the conflict occurs in the basic path, then the course of action depends on the relative position of the conflict within the path and the path type. For each connection in the conflicting pair, try the following:

(a) Check if the conflicting segment is variable. If it is, reposition it in a direction depending on the path type.

(b) If the segment is not variable, check if the path contains a basic or additional slides. If it does, try repositioning the segment and adjusting the length with the slide.

(c) If there are no slides, check the other connection.

As with the slide adjustments, adjustments are only executed if there is no conflict with a horizontal segment.

4.4.5 When Neither Connection is Adjustable If neither connection in the pair can be adjusted without changing its path length, then it is necessary to increase the length of one path. The loop basis equations, which are represented by the loop/edge matrix, are used to determine which connection of the conflicting pair occurs in the fewest loops. This is the one which will cause the fewest changes on the other path delays. When the length

of a connection is extended to clear a conflict, care must be taken to ensure enough length is provided to clear all other possible conflicts along its path. After distribution of the path delays, the path lengths of the previously routed connections are adjusted as necessary. If these adjustments cause additional conflicts which can not be resolved, start over with a new placement.

4.5 Data Structures

The ultimate purpose of the layout tool is to allow the production of masks. In order to facilitate the production of these masks, the layout tool will produce a data file containing all the graphical information about each device and connection in the layout. This information is gathered by scanning through the list of layout elements and the list of connections and reading the desired graphical information from the data structures.

In order to minimize interaction with the original XHATCH data structures, a separate device data structure is used to store the graphical information used by the layout tool. In particular there are the:

- Placement map coordinates.
- Graphics coordinates of upper left corner of the enclosing rectangle.
- Graphics coordinates of lower right corner of the enclosing rectangle.
- Array of graphics coordinates for the input terminals.
- Array of graphics coordinates for the output terminals.
- Pointer to circuit device definition structure.

CHAPTER 5

RESULTS

How the automatic layout tool performs on three successively larger time-of-flight circuits is described next.

5.1 Four Bit Word Counter Circuit

The 4-bit word counter circuit, shown schematically again in Fig 5.1, is one of the many subcircuits of the SPOC [2]. Input pulses to be counted are generated by a 1300 nm laser (represented by Os1 in the Figure). These input pulses are split at Sp1 and sent to Sw1 and Sw3. From Sw1, only every fourth clock pulse is passed through to Sw2 to arrive at the control input of Sw3. After arriving at Sw3, the control pulse switches the arriving input clock pulse to e3 which is then split at Sp4 and sent to Sw4 and Sw5. Accumulated bits circulate in the e8 and e10 memory loop. With a memory loop delay of four bits, this circuit can count up to a maximum of 16, 4-bit binary words. The output generated by Sw4, is a string of pulses with a binary bit pattern representing the present 4-bit word count.

5.1.1 Placement of the Devices The first step in the layout process is to form an initial placement of the devices. Since edges e13, e17, e5 and e8 have lumped delays of 2, 1, 1, 4, clock cycles respectively, they were initially assumed to be long edges. These edges were removed from the circuit graph of Fig. 3.1 resulting in the partial circuit graph of Fig. 5.2. Note that edges e5, e6, e7, and e9 were subsequently removed, since only one edge is

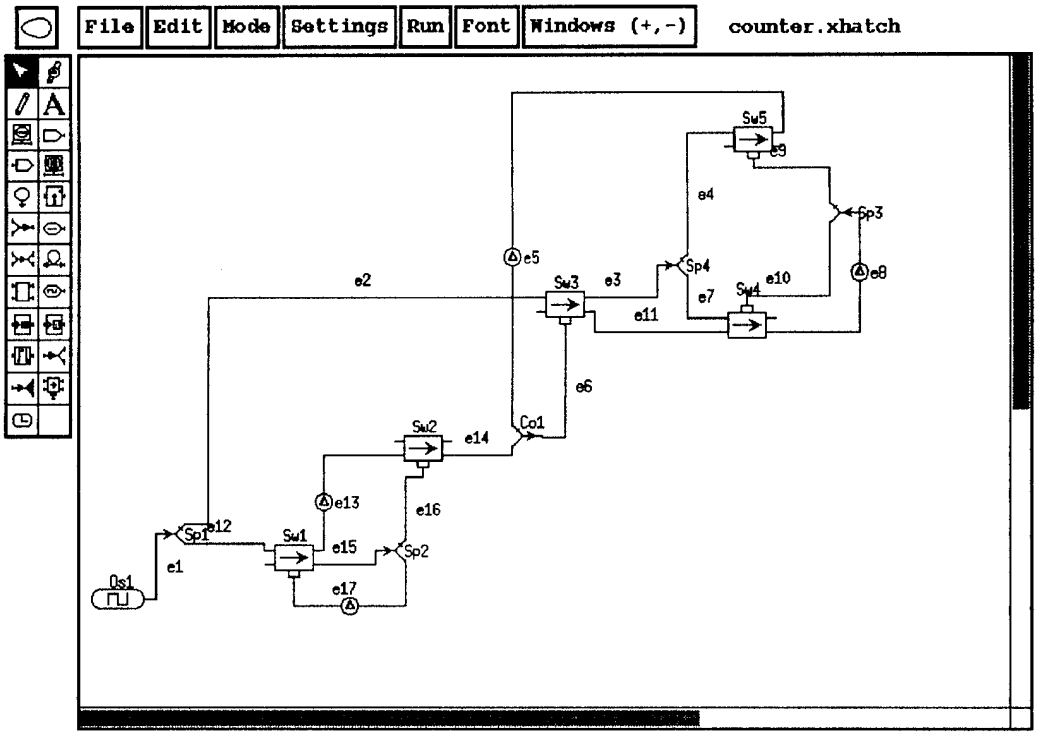


Figure 5.1: Four-bit word counter design.

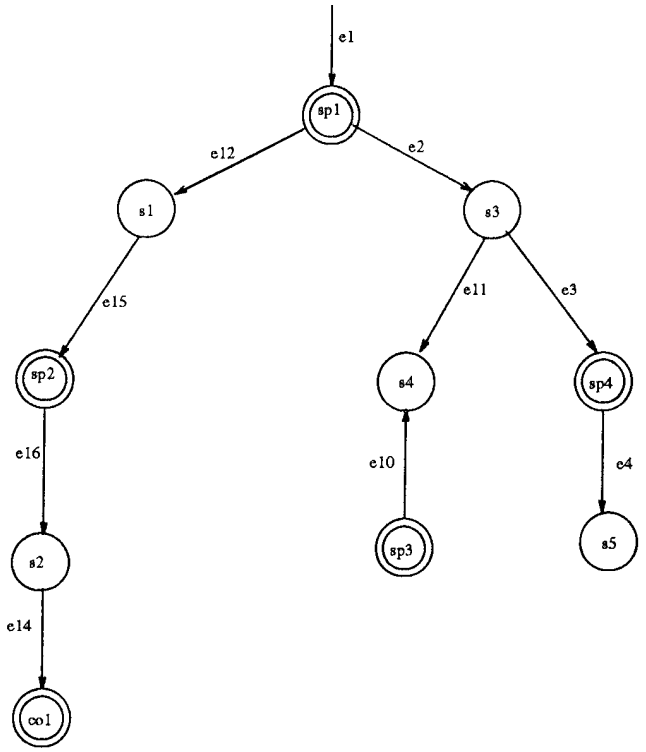


Figure 5.2: Initial partial circuit graph of the 4-bit word counter circuit.

required between vertices.

Using this partial-circuit graph (which now resembles a tree) and a left-child first preference, results in the initial placement of the devices as shown in Fig. 5.4. After estimating the minimum connection lengths, the distribution algorithm was run and the minimum clock cycle length of 55850 λ was found. The next step was to correct the partition of the long and short edge sets. After comparing the estimated minimum lengths with the actual delay lengths needed for circuit synchronization, it turns out that path e5, which was originally assumed to be long, is actually short and path e2 is long. After correcting the edge set partition, the partial circuit graph of Fig. 5.3 is formed. Notice, with e2 removed, it is not necessary to add e5, since co1 is reached via e6. With these changes, Sw3 is no longer in parallel with Sw1,

resulting in the placement of Fig. 5.5, which has a minimum clock cycle length of 55150λ (a reduction of 1.25%). The clock period can be calculated using nl/c where l is the clock cycle length (with $\lambda = 1\mu m$), $n = 2.2$ and c = speed of light. This placement has the potential of using a laser clock with 0.4044 ns period corresponding to a maximum operating frequency of 2.473 Ghz.

Fig. 5.6 shows the completely routed circuit using the corrected initial placement of Fig. 5.5. During the routing phase, e9 conflicted with e10. Since e10 is a type 1 path with a little slack, the routine first tried to shift the conflicting segment and extend e10's basic slide, however, doing so would cause the vertical segment to overlap the terminal points. Thus, this attempt to adjust e10 failed, so e9 was tried. Even though e9 does not have any slack, it does have two variable vertical segments because it is a type 5 path. Therefore, the conflicting segment of e9 was successfully adjusted without effecting e9's overall path length. Path e13 also ran into a conflict. After forming several additional slides which had ventured outside the Manhattan region, the return path to the target point overlapped a segment of the basic path. This conflict was avoided by shifting e13's return path over to the next vertical path and reducing the length of an additional slide to keep the path length constant. Likewise, the return path of e8 was adjusted in a similar manner.

The next figure shows the effect of using an iterative improvement, sequential row swapping algorithm on the initial placement of Fig. 5.5. Only the exchanges which decreased the major clock cycle time, were kept. In this case, rows 2 and 3 were exchanged, then rows 3 and 4 were also exchanged. This resulted in a placement which is shown in Fig. 5.7 with a minimum clock cycle length of 54450λ corresponding to a period of 0.3993 ns and a

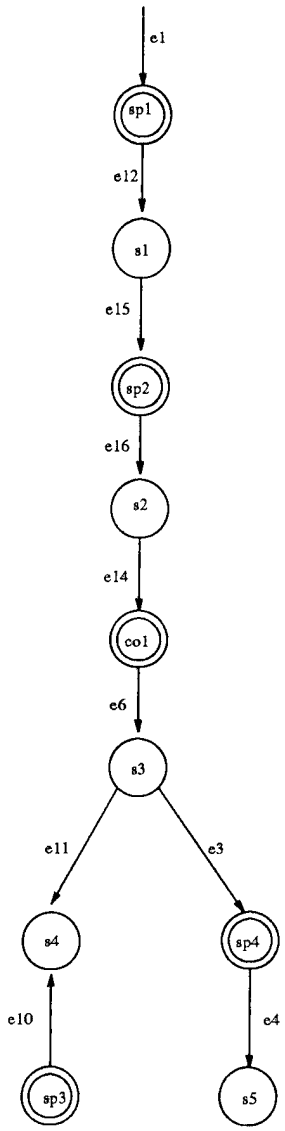


Figure 5.3: Corrected partial circuit graph of the 4-bit word counter circuit.

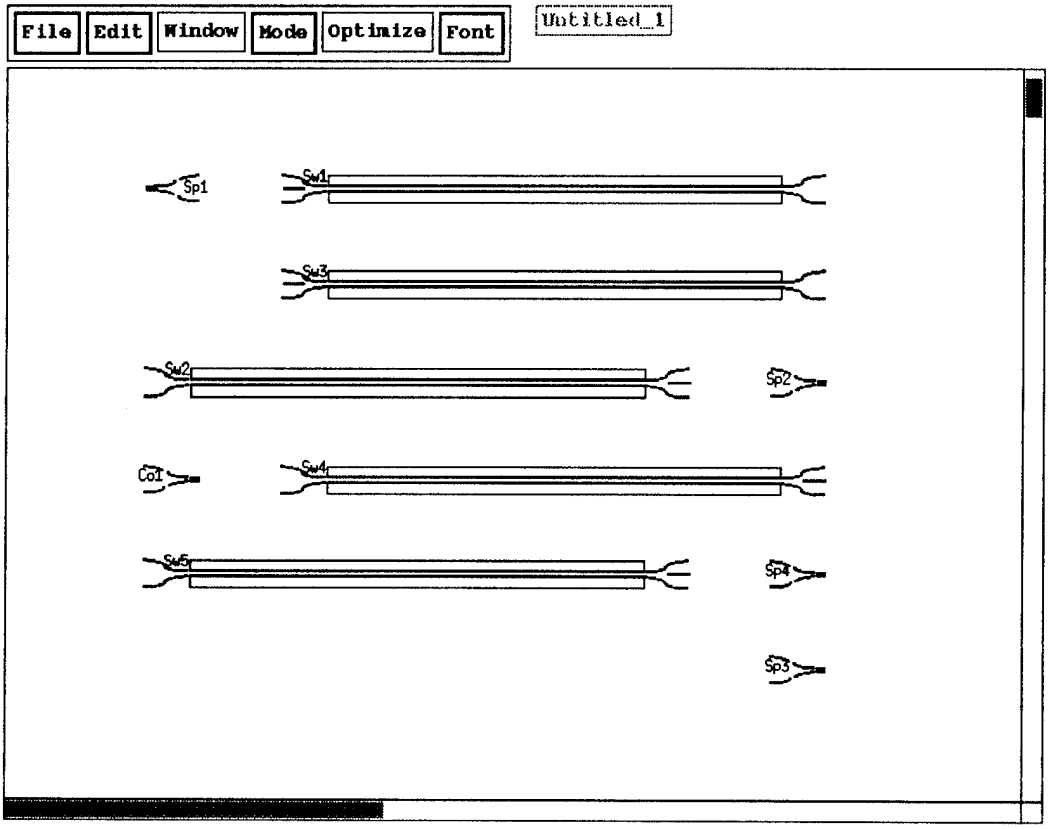


Figure 5.4: Initial placement (left-child first) of the 4-bit word counter.

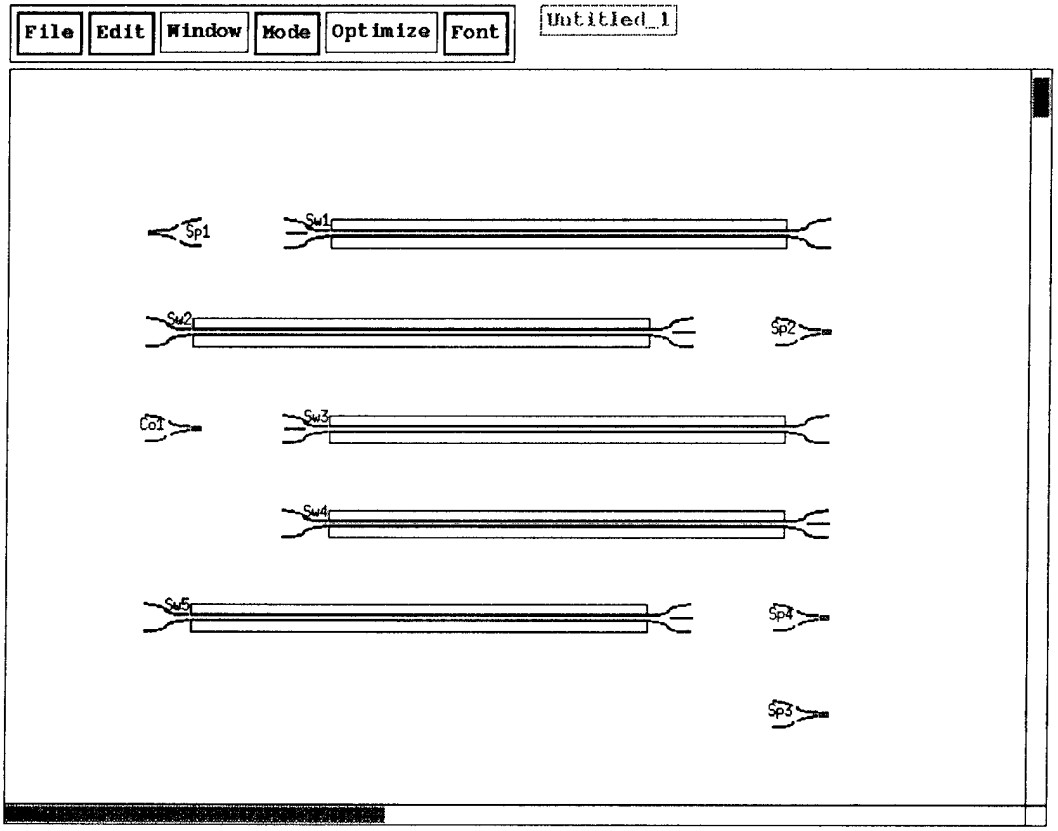


Figure 5.5: Corrected placement (left-child first) of the 4-bit word counter.

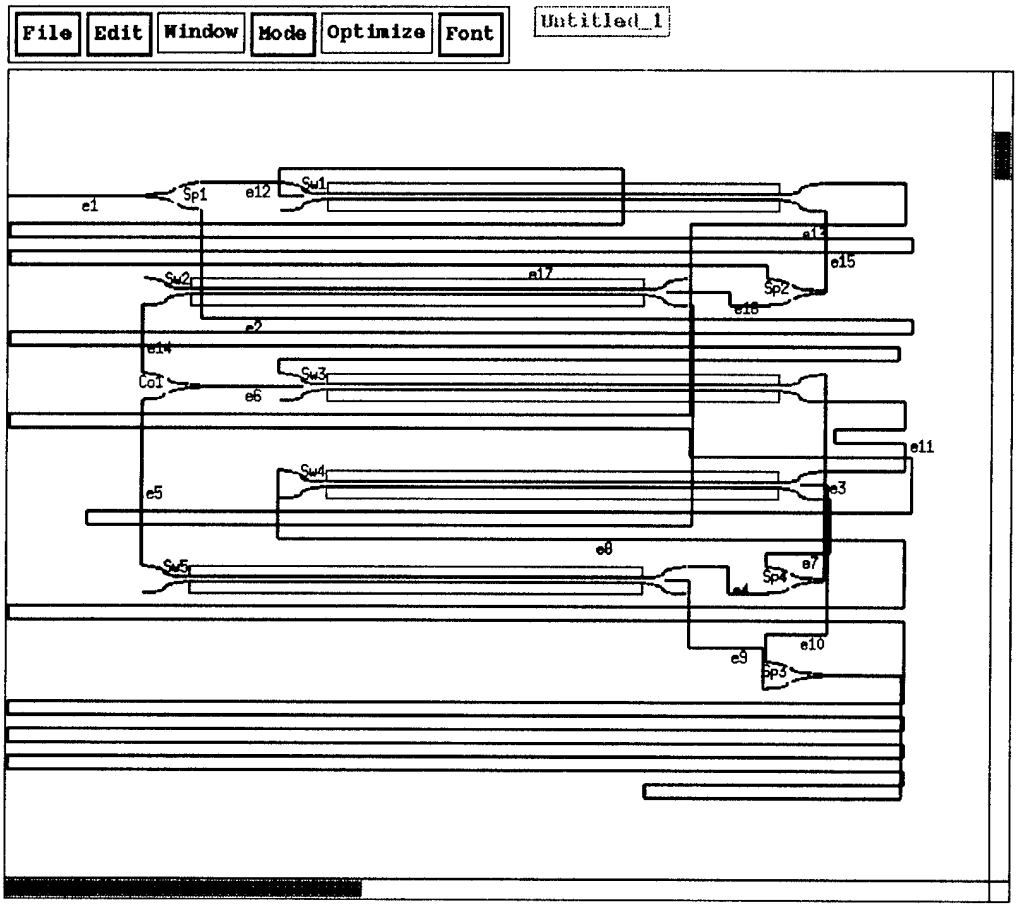


Figure 5.6: Routed 4-bit word counter circuit (left-child first placement).

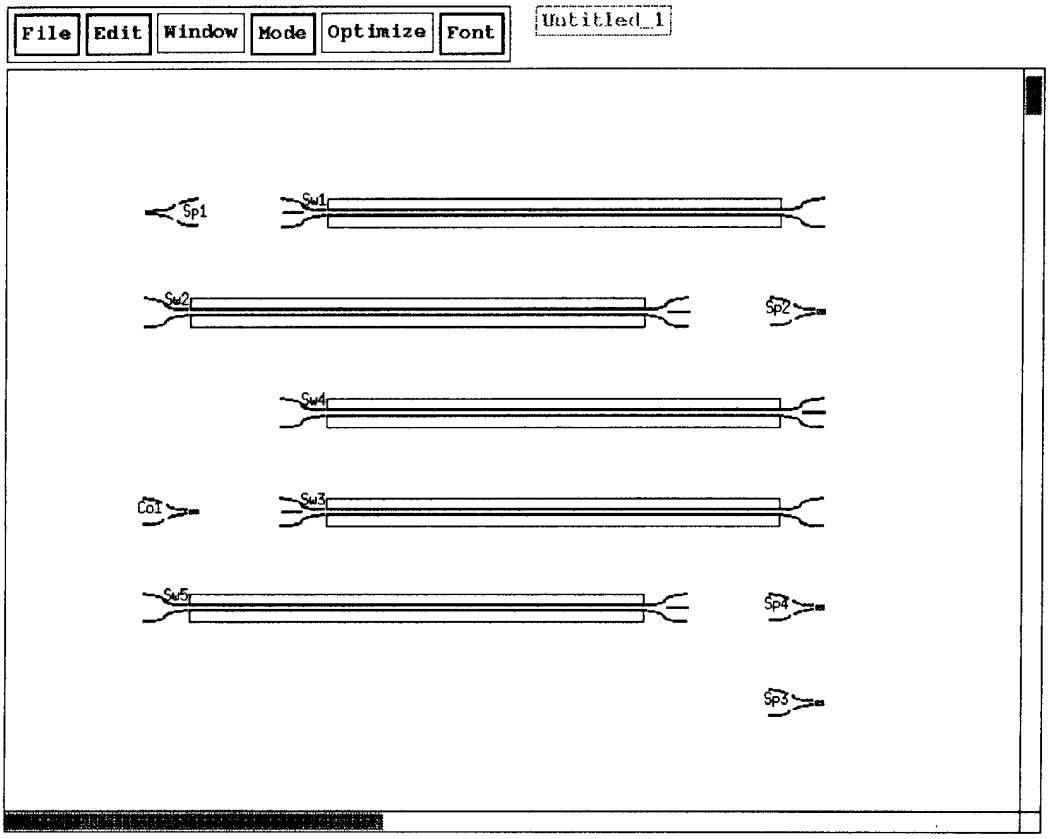


Figure 5.7: Improved placement (left-child first) of 4-bit word counter.

maximum frequency of 2.504 Ghz (which is 1.27% less than the unoptimized initial placement).

The complete layout is shown in Fig. 5.8. During the routing phase, e7 which is a type 1 path with no slack, conflicted with e3 which is type 3 path, also without slack. Since neither path could be adjusted without changing its length, the loop-edge matrix was checked to determine which of the two paths is a member of the fewest loops. Since e3 is an independent edge of the loop-edge matrix for this placement, its length was increased by 100 λ and the delays were redistributed without any change to the clock period. As subsequent paths were routed, e10 encountered several conflicts First with e9, and as in the previous layout, the conflicting segment of e9 was extended.

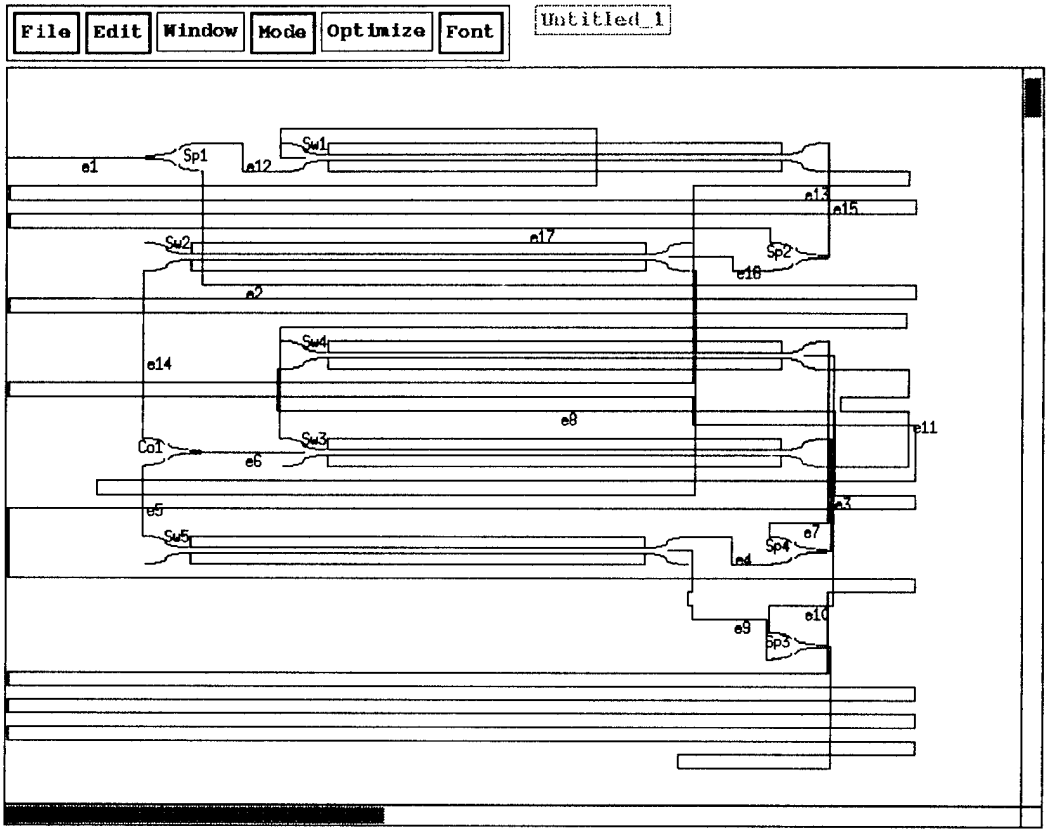


Figure 5.8. Complete layout of 4-bit word counter (improved placement, left-child first).

Then e10 conflicted with e7. Since e10 is a type 1 connection with no slack, it cannot be adjusted without changing its length. After checking the loop-edge matrix, it was determined that e10 should be extended. Since a vertical segment occupies the next track, e10 was extended past e3 incurring a 200λ path increase. After redistribution of the path delays, the path length of e9 needed to be increased to maintain synchronization. This was done by adding a slide to its path. Finally, after adjusting the return path of e13, the rest of the paths were routed without conflicts. The entire circuit was laid out with 4 calls to the distribution algorithm with no change to the clock period during the routing phase.

Notice the above placements preferred the left child first during breadth

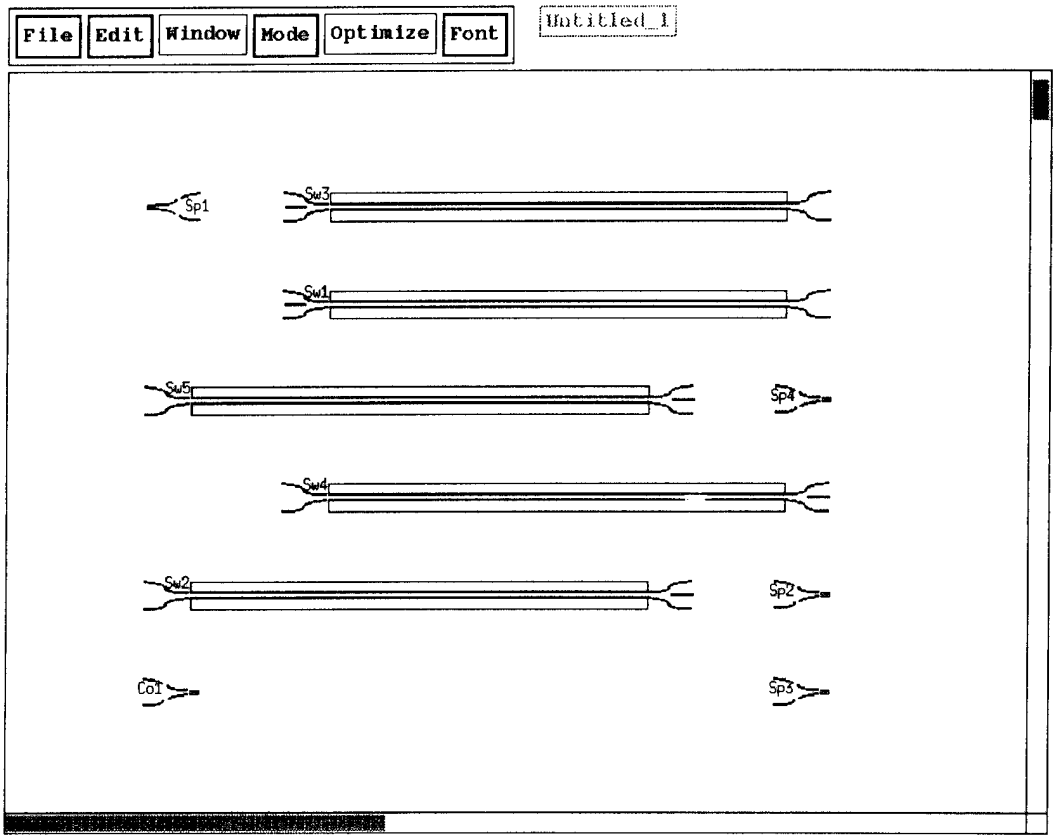


Figure 5.9: Initial placement (right-child first) of 4-bit word counter.

first traversal of the partial circuit graph. Therefore, when there were two parallel, similar length children, the left child was placed before the right child. A variation of the placement algorithm is simply to choose the right child first. Using this option, the devices are initially placed as shown in Fig. 5.9. The minimum length of the clock period increased by 2.24% to 57962λ, or 0.4251 ns. Note how this small change in the algorithm produced a large variation in the placement, compared with the initial placement of Fig. 5.4. Not only are the parallel children, Sw1 and Sw3, switched but also the placement of the children's subtrees, causing sp4 and Sw5 to move up two rows and Sw2 to move down two rows.

After correcting the edge partition, the difference between using the

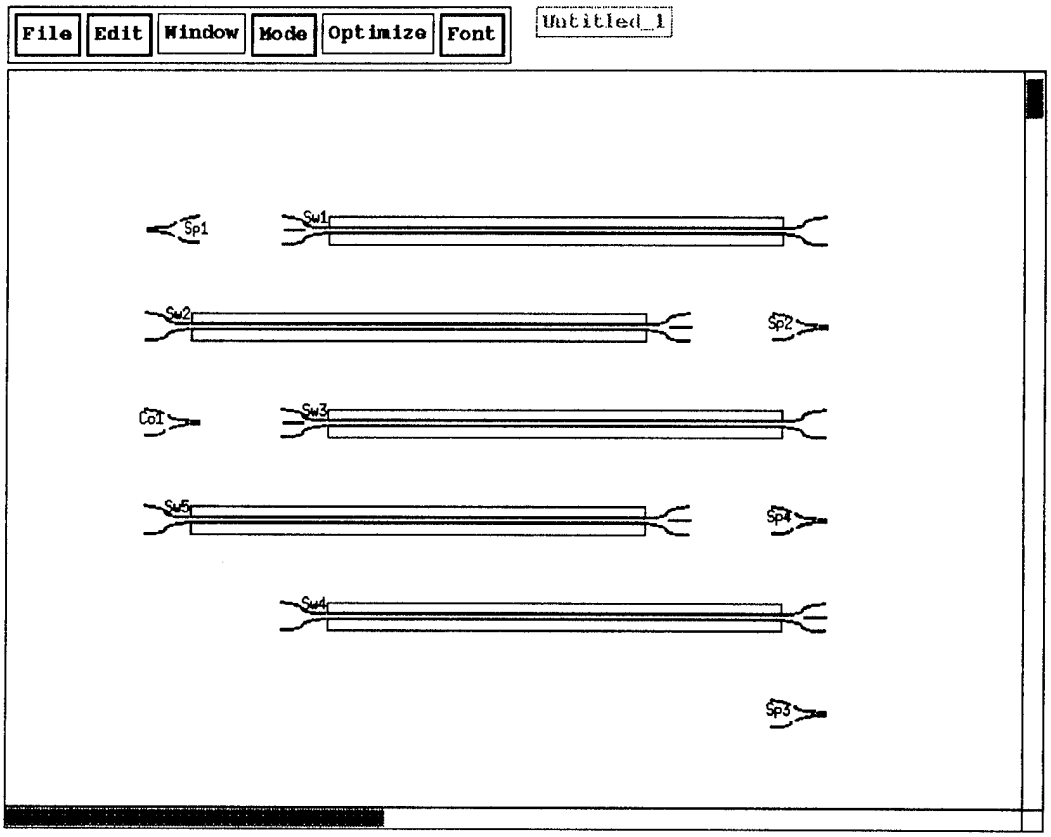


Figure 5.10: Corrected placement of 4-bit word counter (right-child first).

left or right child first is less dramatic in this case, because after removing e2, Sw1 and Sw3 are no longer in parallel. This corrected placement is shown in Fig. 5.10.

Running the sequential row swap, iterative improvement algorithm on the above placement produces the placement which is shown in Fig. 5.11. This placement has the potential to operate with a minimum clock length of 56618λ (0.4152 ns), an improvement of 2.32% over the unoptimized, right-first, placement. However, this is still greater than the minimum clock period possible with the optimized left-first placement.

The completely routed layout of the above placement is shown in Fig. 5.12. Even though two redistributions were required, first for a conflict between

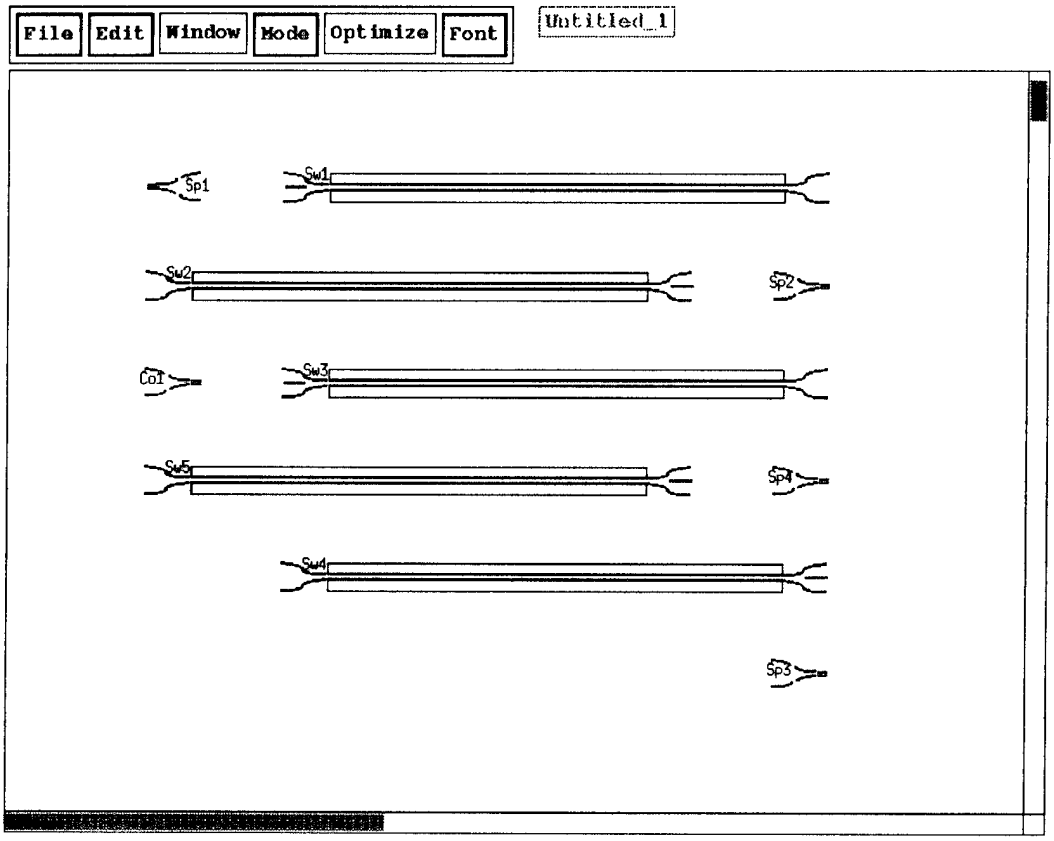


Figure 5.11: Improved placement of 4-bit word counter (right-child first).

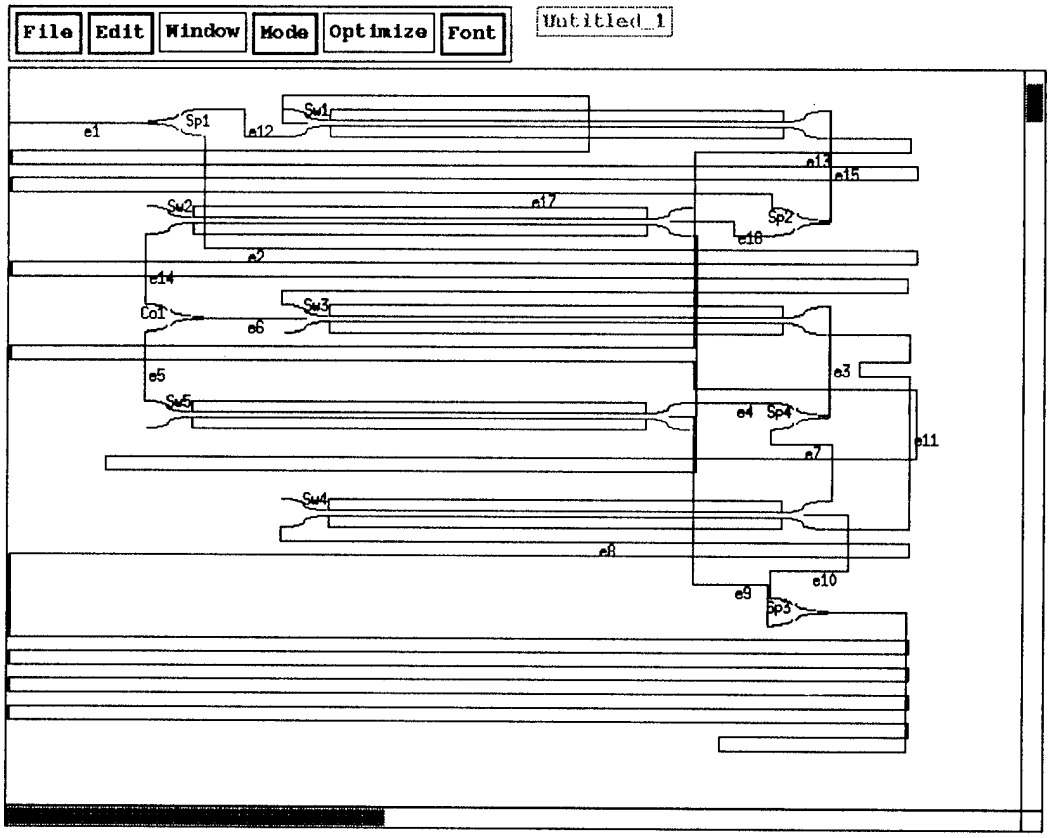


Figure 5.12. Layout of 4-bit counter (improved placement and right-child first).

e7 and e15, then again between e3 and e7, the clock period didn't change.

5.1.2 Layout of Circuits with a Specified Clock Period An example of a layout where the clock period is specified is shown in Fig. 5.13. The 4-bit counter clock period length was specified as 64000λ or 0.4693 ns. Since this period is greater than the minimum found before optimization, it was not necessary to iteratively improve the initial placement.

5.1.3 Summary of Results for the Counter Circuit Table 5.1 summarizes the placement and routing results of the 4-bit word counter circuit.

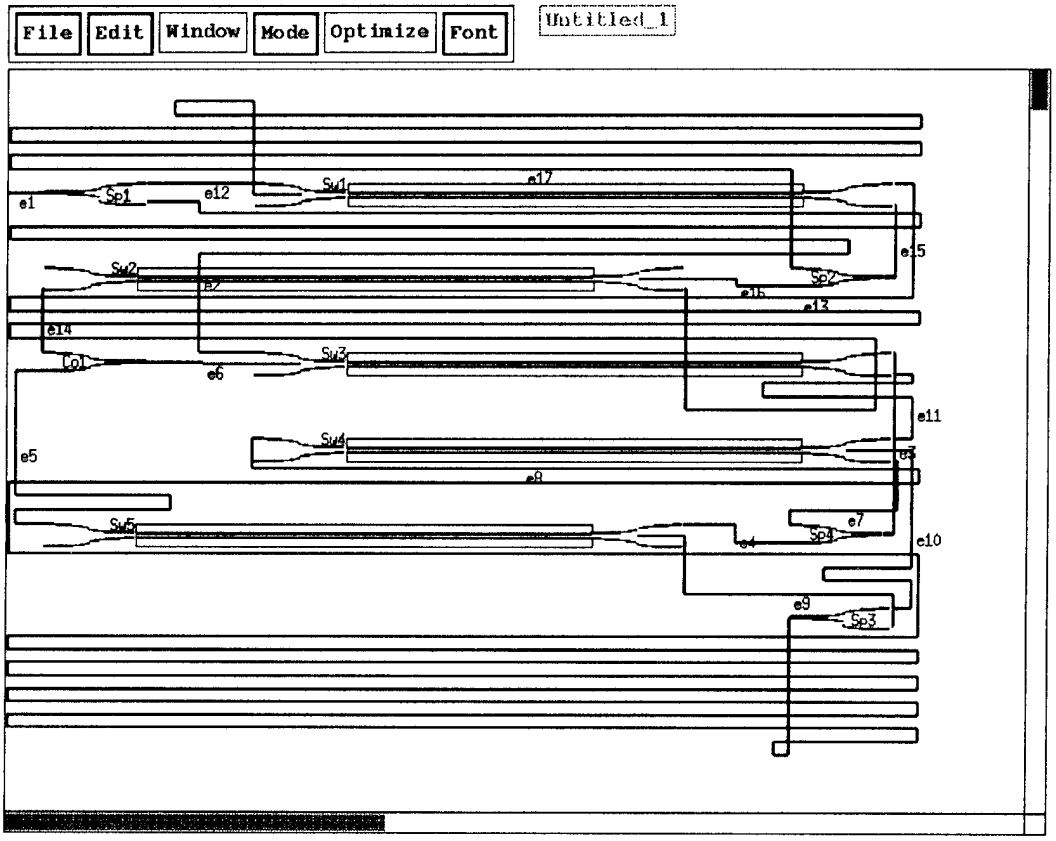


Figure 5.13. Layout of 4-bit counter using a specified clock period (improved placement, left-child first).

Table 5.1. Summary of the minimum clock delay length, period and frequency for each layout.

Placement	Figure	Minimum Clock		
		Length (λ)	Period (ns)	Freq. (Ghz)
Initial Pl. (lt)	5.4	55850	0.4096	2.442
Corrected Pl.(lt)	5.5	55150	0.4044	2.473
Layout of Correct. Pl. (lt)	5.6	55150	0.4044	2.473
Improved Pl. (lt)	5.7	54450	0.3993	2.504
Layout of Impr. Pl. (lt)	5.8	54450	0.3993	2.504
Initial Pl. (rt)	5.9	57350	0.4206	2.378
Corrected Pl. (rt)	5.10	54350	0.3986	2.509
Improved Pl. (rt)	5.11	54350	0.3986	2.509
Layout of Impr. Pl. (rt)	5.12	54350	0.3986	2.509

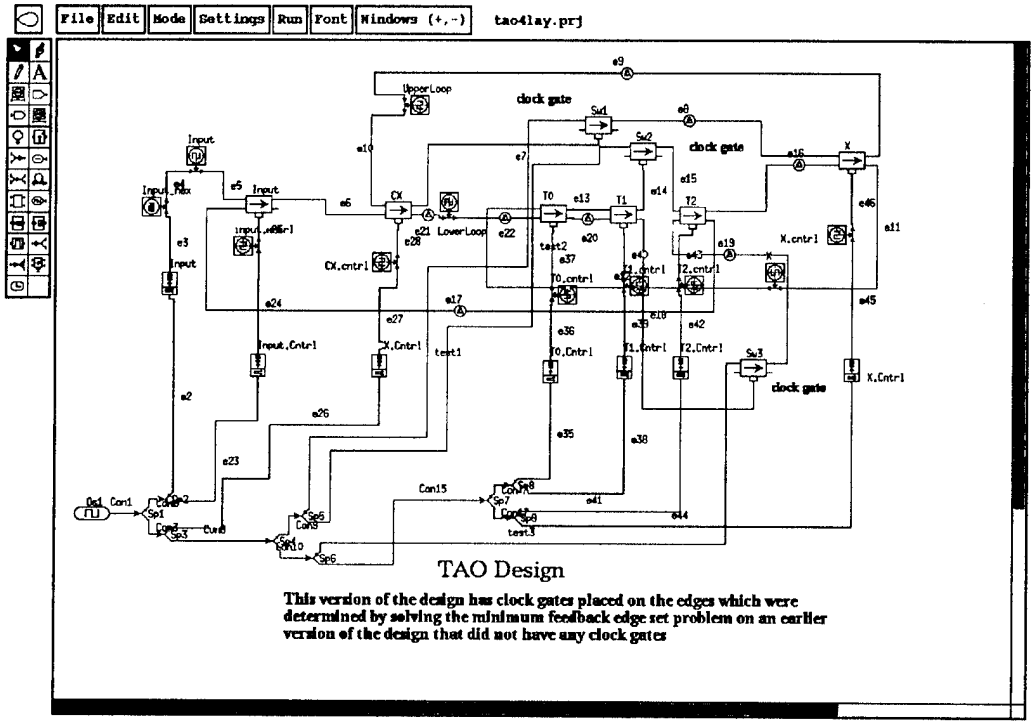


Figure 5.14: Schematic of the simple sorter.

5.2 A Simple Sorter

An example of a slightly larger circuit is the simple sorting circuit shown in Fig. 5.14. This circuit, which has 9 switches and 9 splitters, was designed by Neil Coletti of the Supercomputing Research Center and John Feehrer from the Optoelectronic Computing Systems Center at the University of Colorado, Boulder.

The automatic placement of the elements (without iterative improvement) is shown in Fig. 5.15. The minimum clock length for this placement is 29121λ, corresponding to a period of 0.2136 ns and a maximum frequency of 4.683 Ghz.

The layout of the unoptimized placement is shown in 5.16. During

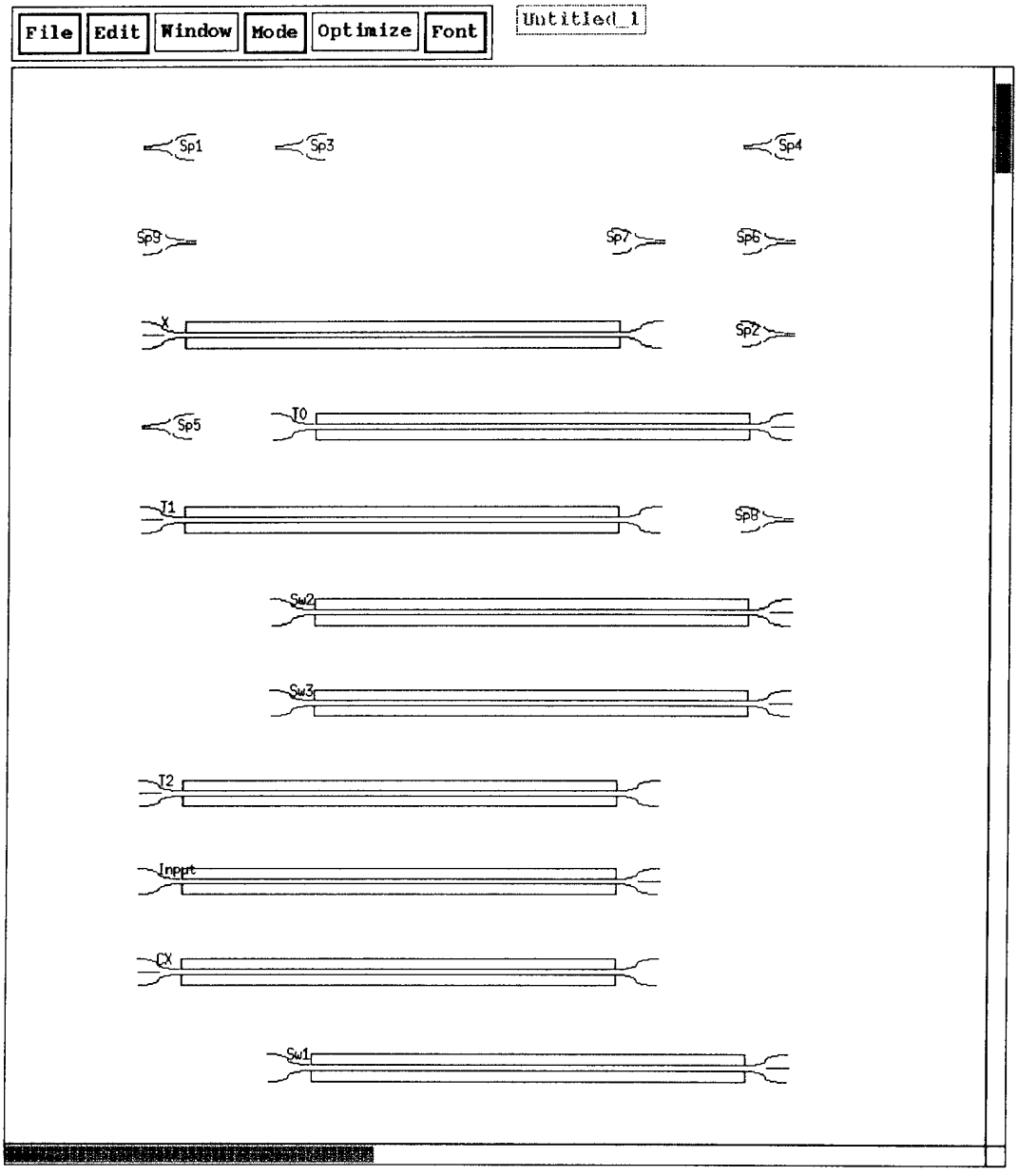


Figure 5.15: Unimproved placement of the sorter elements.

Table 5.2. Summary of delay redistributions for the unoptimized sorter layout.

Num.	Curr. Path	Conflict. Path	Extended Path	Clock (λ)
1	e15	e19	e15	29121
2	e16	e11	e16	29121
3	e16	e44	e16	29121
4	e16	e19	e16	29321
5	e16	e13	e13	29421
6	e38	e16	e38	29421
7	test2	Con16	test2	29421
8	Con9	e16	Con9	29421
9	Con2	e11	Con2	29421

the layout, the conflicts in Table 5.2 resulted in 9 redistributions causing the clock delay length to increase by 1% to 29421λ for a period of 0.2158 ns and a frequency of 4.635 Ghz. Looking at the Table, e16, which is a type 1 path with no slack, ran into the most conflicts requiring the extension of the other path and subsequent redistribution of the path delays.

Running the sequential row swap algorithm caused the exchanges of rows 3 and 4, and rows 7 and 8, resulting in the placement of Fig. 5.17. This placement has a clock delay length 28721λ corresponding to a period of 0.2106 ns and a frequency of 4.478 GHz which is a decrease of 1.37% compared to the unoptimized placement.

After optimization of the placement, the connecting paths were successfully routed as shown in 5.18. The final clock delay length increased by 100λ to 28821λ. The final clock period is 0.211 ns for a maximum frequency of 4.731 Ghz, which is an increase of 2.04% from the maximum clock frequency using the unoptimized placement. The entire layout has a width of 2 cm and a length of 0.6 cm for a total area of 1.2 $\mathbf{cm^2}$. The paths encountering conflicts which required a delay redistribution are listed in Table 5.3. Notice, two less

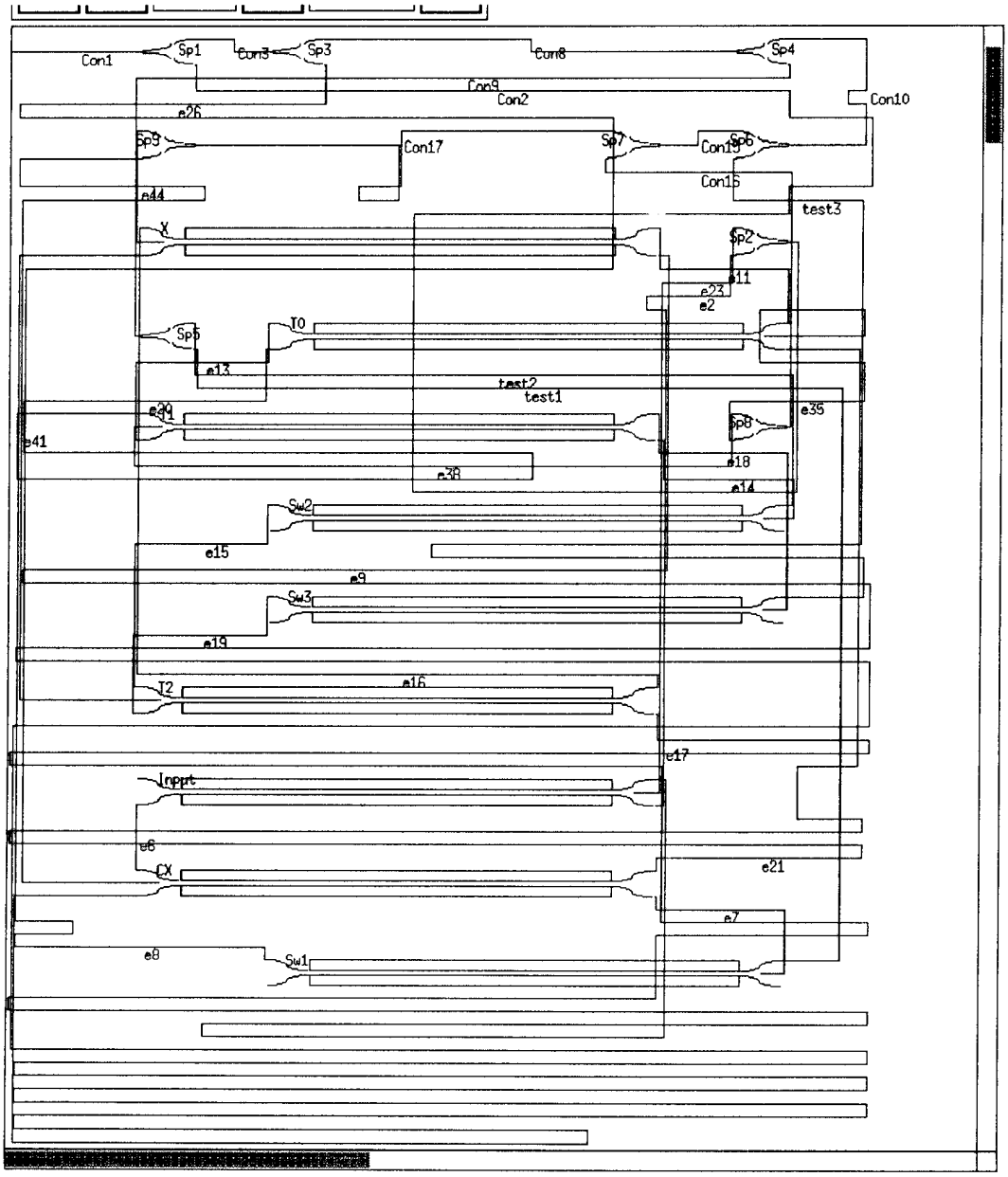


Figure 5.16: Unoptimized layout of the sorter.

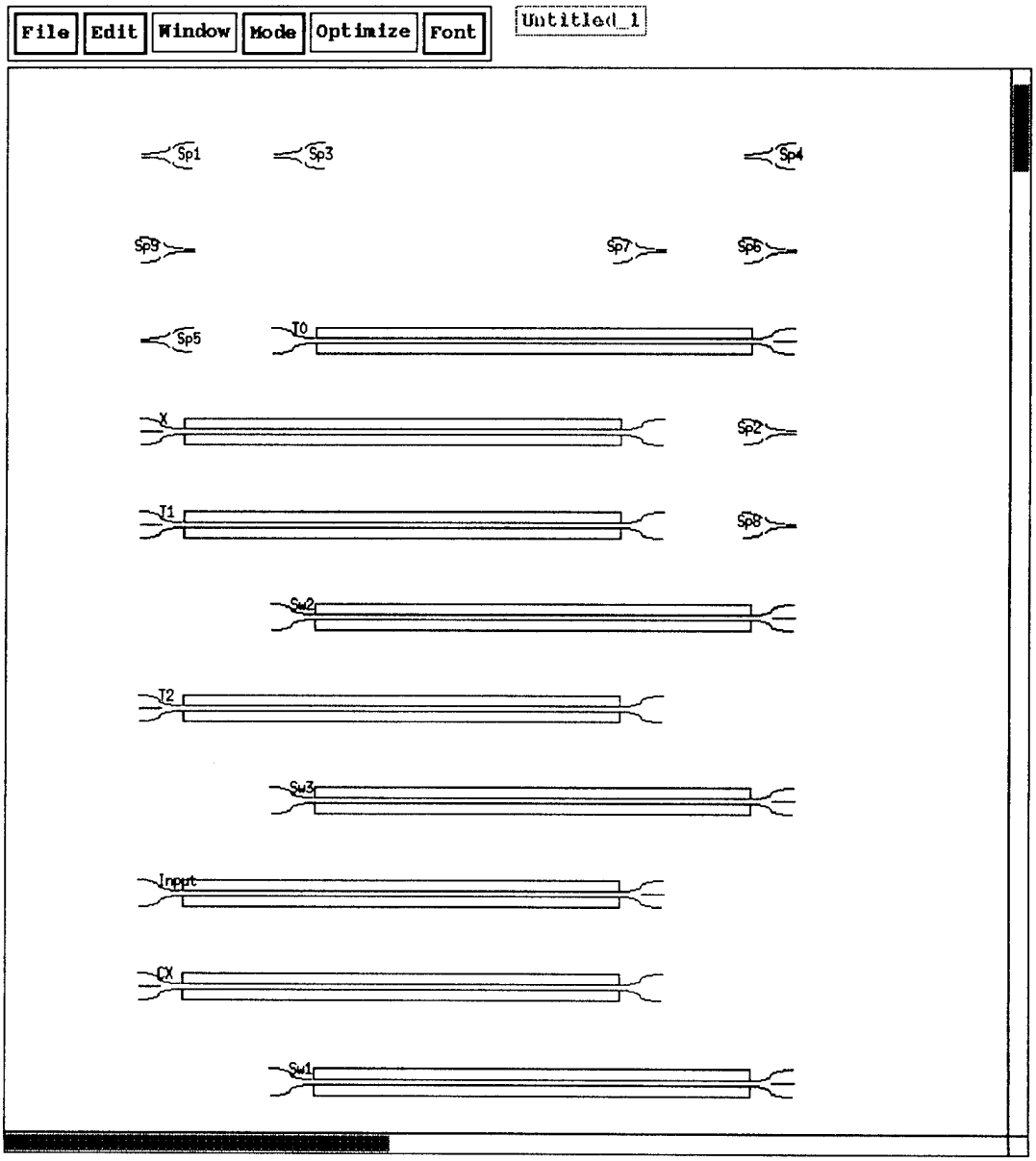


Figure 5.17: Optimized placement of the sorter elements.

Table 5.3: Summary of delay redistributions for the optimized sorter layout.

Num.	Curr. Path	Conflict. Path	Extended Path	Clock (λ)
1	e13	e44	e15	28821
2	Con16	e11	Con16	28821
3	e16	e44	e44	28821
4	e16	e15	e15	28821
5	e16	e38	e38	28821
6	test2	e11	test2	28821
7	Con2	e11	Con2	28821

delay distributions were required than for the unoptimized layout.

5.3 Logarithmic Delay Line

The next circuit is an example of a much larger circuit than the previous two. It consists of 116 internal connections, 30 switches, 25 splitters, 4 combiners and 4 couplers. The logarithmic delay line is an architecture for the temporal alignment of two incoming packets [50]. The delay line uses a logarithmic number of switches or nodes to align two signals. If two packets arrive at different times at a node, k, it synchronizes them by delaying the first of the two packets by an amount given by $(T/2)^k$, where T is the length of a packet plus the separation between packets.

The circuit schematic is shown in Appendix B. After invocation of the automatic placement algorithm, the devices are placed as shown in Fig 5.19. The upper portion of the completely routed layout (scaled by 30 in both x and y-directions) is shown in Fig 5.20. This figure shows everything but two delay lines, whose lengths are greater than a meter, which may be routed better off-chip. The entire layout (scaled by a factor of 30 in the x-direction and 5 in the y-direction) is shown in Fig. 5.21 (folded, in the back pocket). It turns out that the shortest clock period was achieved after using the complete circuit

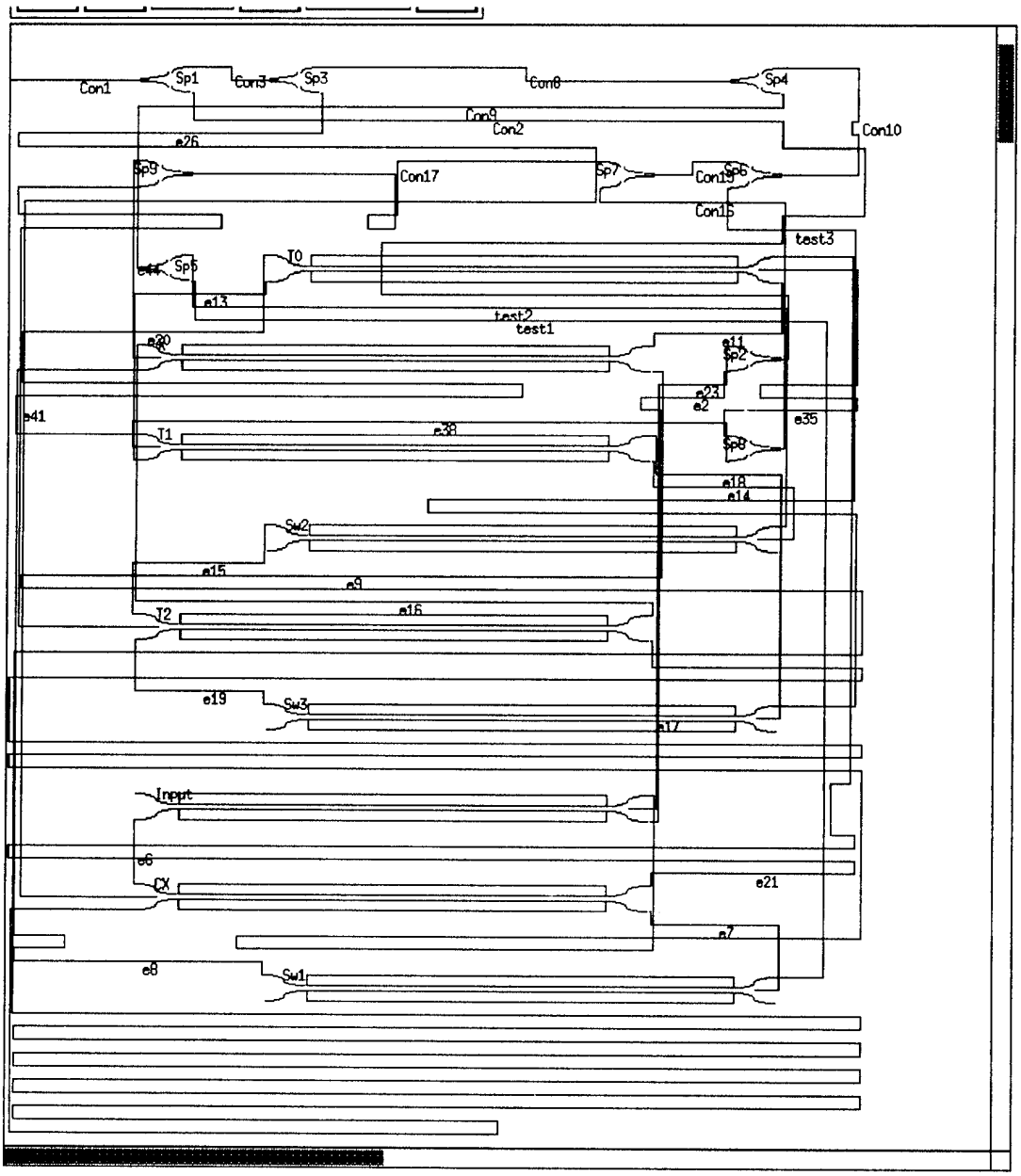


Figure 5.18: Layout of the sorter using the optimized placement.

Table 5.4. Summary of the minimum clock delay length, period and frequency for each placement of the logarithmic delay line.

Circuit Configuration	Minimum Clock		
	Length (λ)	Period (ns)	Freq. (Ghz)
Init. Placement (Partial Graph)	40503	0.2970	3.367
Init. Placement (Complete Graph)	38153	0.2797	3.574
Layout of Init. Placement	38753	0.2829	3.535

graph. When the partial circuit graph was used, the new subtrees, created when the long edges were removed, ended up being placed farther away from their parents, causing the clock period to increase. The results (all using the left-child first preference) are shown in Table 5.4. The integrated chip size will be 2 cm wide by 2.3 cm long for a total area of 4.6 $\mathbf{cm^2}$.

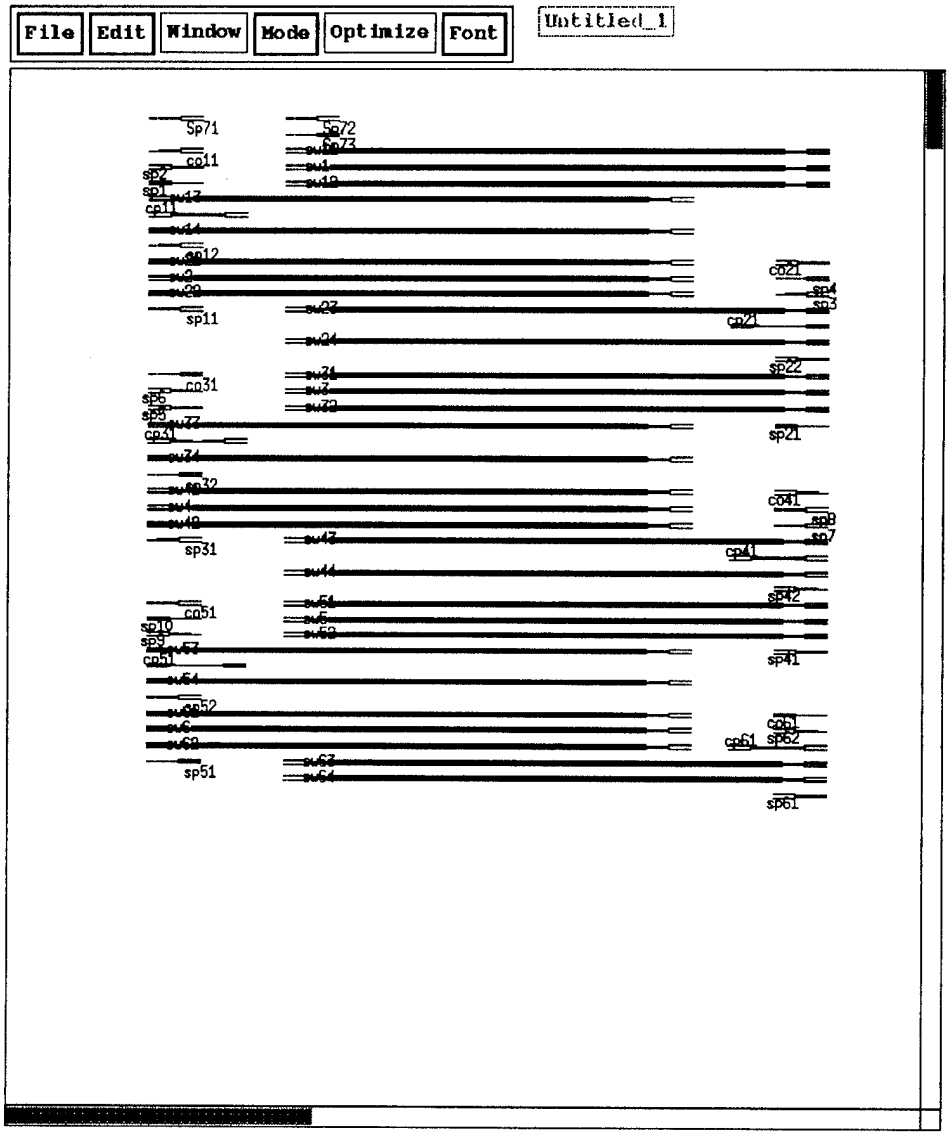


Figure 5.19. Placement of the logarithmic delay line (scaled by 30 in both directions).

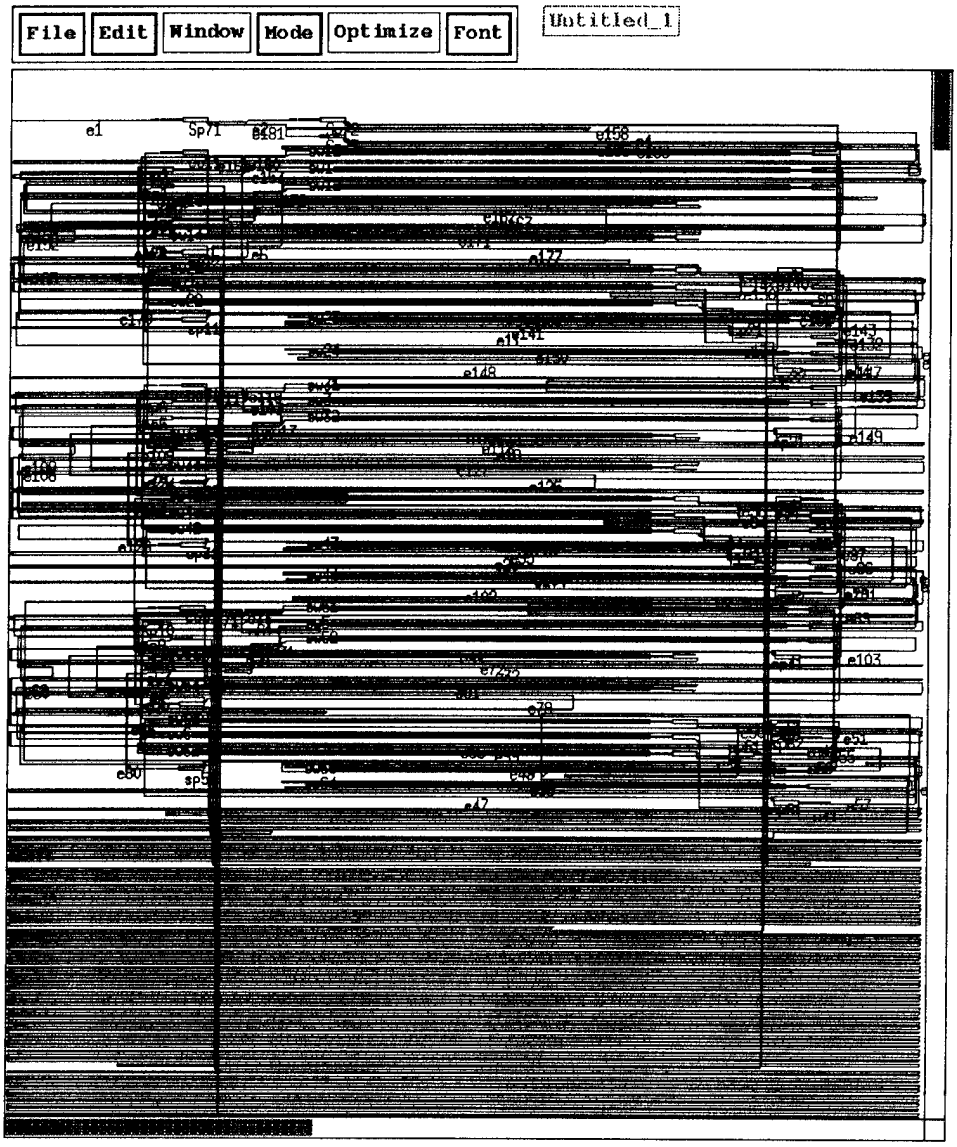


Figure 5.20. Layout of the logarithmic delay line.(scaled by 30 in both directions).

CHAPTER 6

SUMMARY OF RESULTS AND FUTURE WORK

This final chapter concludes by summarizing the results of the previous chapters and providing suggestions for future research.

6.1 Summary

The problem of time-of-flight circuit layout has two unique aspects which drive a new, yet simple, approach for automatic circuit layout. These two factors are the highly astigmatic aspect ratio of the circuit elements and the necessity to route the connecting paths to specified lengths. Despite the large volume of literature on VLSI layout, this problem is not addressed.

Before any algorithms were developed, preliminary effort was devoted to the choice of a suitable device placement architecture. After considering several potential alternatives, the only viable placement architecture is one based on placing the elements in parallel rows. Using the connectivity information provided by the short connections, it is possible to produce an initial placement which results in a clock period which is within a few percent of the minimum achieved after iterative improvement.

One major advantage of integrated-optic circuit layout over VLSI circuit layout is the fact that optical waveguides can cross other waveguides and devices at right angles without any negative effect on the signals. This simplified the routing algorithms considerably. The connections have essentially

unrestricted usage of the chip space, provided that they do not overlap or coincide with each other. After establishing a regular track grid and a connecting path data structure, the paths are routed in a straightforward manner. Finally, by taking advantage of path variability, any potential overlapping of paths is resolved by shifting a segment of one path around the other path, usually without any change in the path length and clock period. If a path has no variability, then it is simply extended past the other path and the delays are redistributed with little or no effect on the clock period.

6.2 Results

This work shows it is possible to automatically generate a layout of a synchronized, integrated-optic, time-of-flight circuit with no user intervention. The user can choose from several different options in search of the most desirable layout. If a minimum clock period is the goal, the user can use either a full or partial circuit graph representation from which to perform the placement. There is also the choice between placing the left child or the right child first, while traversing of the circuit graph in the placement phase. Most noteworthy, is the observation that all of these options produce layouts whose minimum clock period varies less than 6 % between any two layouts. If the circuit must be synchronized with a specified clock period, then so long as the specified clock period is greater than the minimum, any option which allows the connections to be routed without changing the clock period is acceptable.

6.3 Areas for Further Study

(a) The use of a grid-based track system, results in path lengths which may deviate as much as the width of a track from the desired length.

Though this difference is very small, percentage wise, it may be possible to use a different conflict identification and avoidance scheme which does not rely on a grid-based track scheme, which would allow routing the path to the exact length.

(b) The main problem with the tool is that it is not possible to directly determine the minimum clock period either analytically from the loop equations or from the initial placement. Only after trying several different placement options can the user select one which provides the best clock period. Also, the option which produces the best clock period for one circuit is not guaranteed to produce the best for another circuit. The tool should be modified to test the various placement options automatically and then use the desired placement for the final layout.

(c) Currently, once a particular set of chip and device parameters are chosen, the spacing and thus the number of tracks between rows is fixed. However, the number of tracks needed between rows is not constant. Depending on the length of the paths, there may be a need for more or less tracks between any pair of rows. By scanning through the horizontal tracks, it is possible to identify empty tracks which can be removed to reduce the width of a channel. Likewise, it is possible to determine which tracks are fully occupied and thus identify those channels which could benefit from being widened. After the horizontal channels are adjusted, it is a simple matter to adjust the vertical position of the devices accordingly.

(d) As seen in the logarithmic delay line, at least two paths are longer than a meter. An off-chip fiber delay line may be less lossy than the multitude of corner turning mirrors which are required for the on-chip connection. If this is the case, the program should be altered to flag these long paths and create two I/O paths to route the path off and back onto the chip.

(e) Currently, the task of resolving conflicts for minimal impact on the clock period results in the repositioning of just the vertical path segments. There may be occasions where the optimal resolution of a conflict would require the repositioning of both horizontal and vertical segments. These situations are currently ignored under the assumption that changes in placement have a much greater effect on the overall layout than the small alterations on path length resulting from the resolution of a conflict. However, this option may need investigation in the future.

(f) Currently, the number of random vertical tracks is so few that it is unlikely that they may end up closer than the waveguide spacing. However, for larger circuits this may become a problem requiring a full set of vertical tracks all across the chip or some method to check the relative distance between random tracks to ensure their spacing is large enough.

(g) The I/O demands for the circuits laid out in Chapter Five are minimal. The counter and the sorter both had one input whose path was routed directly to the edge of the chip. The logarithmic delay line had one laser input and two outputs. However, if the chip is to be one of several

in a system, the I/O paths need to be considered more carefully. The use of a managed set of I/O ports as discussed in Chapter Two may be necessary.

(h) Also, it was assumed that the layout of the electronic circuits for driving the control inputs to the switches can be done far more easily than the layout of the integrated-optic chip. However, the placement of the optical detectors depends on the placement of switches and the position of the control input and will need some consideration before full-scale system integration.

(i) Finally, one by four splitters must be manually converted to three, one by two splitters. The manual changing of the circuit is tedious and error prone and should be done automatically.

BIBLIOGRAPHY

[1] V. P. Heuring, H. F. Jordan, and J. P. Pratt, "Bit-serial Architecture for Optical Computing," **Applied Optics**, Vol. 31, No. 17, 10 June 1992, pp. 3213-3224.

[2] P. T. Main, R. J. Feuerstein, V. P. Heuring, H. F. Jordan, J. R. Feehrer, and C. E. Love, "Implementation of a General Purpose Stored Program Digital Optical Computer," **Applied Optics**, Vol. 33, No. 8, 10 March 1994, pp. 1619-1628.

[3] P. T. Main, "Implementation of a General Purpose Stored Program Digital Optical Computer," Optoelectronic Computing Systems, University of Colorado, OCS Technical Report 93-04, May 1993.

[4] M. Salerno, "XHATCH: User's Manual," Optoelectronic Computing Systems, University of Colorado, OCS Technical Report 91-25, Oct 1991.

[5] J. P. Pratt, and V. P. Heuring, "Delay Synchronization in Time-of-Flight Optical Systems," **Applied Optics**, Vol. 31, No. 14, 10 May 1992, pp. 2430-2437.

[6] J. R. Feehrer, "Propagation Delay Uncertainty in Time-of-Flight Systems," Ph.D Dissertation, University of Colorado, Boulder, April 1995.

[7] H. F. Jordan, A. R. Mickelson, B. VanZeghbroek, and I. Januar, "An Integrated Optics, Stored Program Computer",Topical Meeting on Optical Computing, Optical Society of America, March 16-19, 1993.

[8] K. Katsura, T. Hayashi, F. Ohira, S. Hata, and K. Iwashita, "A Novel Flip-Chip Interconnection Technique Using Solder Bumps for High Speed Photoreceivers," **J. Lightwave Technology**, Vol. 8, No. 9, Sept. 1990.

[9] I. P. Januar, "Characteristics of Integrated Optical Devices and Structures," Guided Wave Optics Laboratory, Report No. 49 (based on the author's Ph.D. Thesis), Nov. 1992.

[10] H. Appelman, J. Levy, M. Pion, D. Krebs, C. Harding, and M. Zediker,

"Self-Aligned Chemically Assisted Ion-Beam Etched GaAs/AlGaAs Turning Mirrors for Photonic Applications," **J. Lightwave Technology**, Vol. 8, No. 1, pp. 39-41, (Jan. 1991).

[11] E. Gini, G. Guekos, and H. Melchoir, "Low Loss Corner Mirrors with 45° Deflection Angle for Integrated Optics," **Electronic Letters**, Vol. 28, No. 5, 27 Feb 1992, pp. 499.

[12] Private communication with A. Mickelson.

[13] K. Noguchi, O. Mitomi, K. Kawano and M. Yanagibshi, "Highly Efficient 40-GHz Bandwidth $Ti : LiNbO_3$ Optical Modulator Employing Ridge Structure," **IEEE Photonics Technology Letters**, Vol. 5, No. 1, Jan. 1993, pp. 52-54.

[14] T. C. Hu and Ernest S. Kuh, "Theory and Concepts of Circuit Layout," **VLSI Circuit Layout: Theory and Design**, T. C. Hu and Ernest S. Kuh eds., IEEE Press, 1985.

[15] Steven M. Rubin, **Computer Aids for VLSI Design**, Addison-Wesley Pub., 1987.

[16] M. R. Garey and D. S. Johnson, **Computers and Intractability**, San Francisco, CA, Freeman, 1979.

[17] G. Saucier, A. Bellon, J. F. Paillotin, J. Tsitsimis, A. Janati, P. Chaisemartin, "Classification and Comparison of Different Placement Strategies," **IEEE International Conference on Computer Design: VLSI in Computers (ICCD-84)**, 1984, pp. 745-750.

[18] J. M. Kleinhans, G. Sigl, and F. M. Johannes, "GORDIAN: A New Global Optimization / Rectangle Dissection Method For Cell Placement," **Proc. Intl. Conf. on Computer-Aided Design**, 1988, pp. 506-509.

[19] J. M. Kleinhans, G. Sigl, F. M. Johannes and K. J. Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization," **IEEE Trans. on Computer-Aided Design**, Vol. 10, No. 3, Mar., 1991, pp. 356-365.

[20] S. B. Newell, A. J. DeGeus, and R. A Rohrer, "Design Automation for Integrated Circuits," **Science**, 29 April 1983, Vol. 220, N0. 4596, pp. 465.

[21] B. T. Preas and P. G. Karger, "Automatic Placement: A Review of Current Techniques," **Proceedings of 23rd Design Automation Conference**, 1986, pp. 622-629.

[22] M. R. Hartoog, "Analysis of Placement Procedures for VLSI Standard Cell Layout," **Proceedings of 23rd Design Automation Conference**, 1986, pp. 314.

[23] W.-J. Sun and C. Sechen, "Efficient and Effective Placement for Very Large Circuits," **Proc. Intl. Conf. on Computer-Aided Design**, 1993, pp. 170-177.

[24] C. Sechen and K. W. Lee, "An Improved Simulated Annealing Algorithm for Row-Based Placement", **IEEE/ACM Proc. Intl. Conf. on Computer-Aided Design**, 1987, pp. 478-481.

[25] M. Mogati and Y. Shiraishi, "Cooperative Approach to a Practical Analog LSI Layout System," **Proceedings of 30th Design Automation Conference**, 1993, pp. 544-549.

[26] W. E. Dickinson, "Automatic Placement and Packed Wiring," **IEEE International Conference on Computer Design: VLSI in Computers (ICCD-84)**, 1984, pp. 742-744.

[27] M. A. Breuer, "Min-Cut Placement," **Jour. Design Automation Fault-Tolerant Computing**, Vol. 1, pp. 343-362, 1977.

[28] S. Kirkpatrick, C. D. Gelatt. Jr., M. P. Vecchi, "Optimization by Simulated Annealing," **Science**, 13 May 83, Vol. 220, No. 4598, pp. 671.

[29] J. W. Greene and K. J. Supowit, "Simulated Annealing Without Rejected Moves," **ICCD**, 1984.

[30] T. Hamada, C. K. Cheng, and P. M Chau, "Prime: A Timing-Driven Placement Tool using A Piecewise Linear Resistive Network Approach," **Proceedings of 30th Design Automation Conference**, 1993, pp. 531-536.

[31] M. Marek-Sadowska and S. Lin, "Timing Driven Placement," **Proc. Int. Conf. on Computer-Aided Design**, pp. 94-97, 1989.

[32] S. Teig, R. L. Smith, and J. Seaton, "Timing Driven Layout of Cell-Based IC's," **VLSI Systems Design**, May 1986, pp. 62-73.

[33] Y. Ogawa, T. Ishi, et.al., "Efficient Placement Algorithms Optimizing Delay for High-Speed ECL Masterslice LSI's," **Proceedings of 23rd Design Automation Conference**, 1986, pp. 404-410.

[34] S. Sutanthavibul, E. Shragowitz, and R.-B. Lin, "An Adaptive Timing-Driven Placement for High Performance Vlsi's," **IEEE Trans. on Computer-Aided Design**, Vol. 12, No. 10, Oct., 1993, pp.1488-1498.

[35] P. S. Hauge, R. Nair, and E. J. Yoffa, "Circuit Placement for Predictable Performance," **Proc. Int. Conf. on Computer-Aided Design**, 1987, pp. 88-91.

[36] J. Huang, X. -L. Hong, C.-K. Cheng, and E. S. Kuh, "An Efficient Timing-Driven Global Routing Algorithm," **Proceedings of 30th Design Automation Conference**, 1993, pp. 596-600.

[37] T. -H. Chao and Y. -C. Hsu, "A Clock Net Routing Algorithm for High Performance VLSI," **Proc. European Design Automation Conf.**, 1992, pp. 343-347.

[38] S. Dhar, M. A. Franklin, and D. F Wann, "Reduction of Clock Delays in VLSI Structures," **IEEE International Conference on Computer Design: VLSI in Computers (ICCD-84)**, 1984, pp. 778-783.

[39] W. Khan, M. Hossain, and N. Sherwani, "Zero Clock Skew Routing in Multiple-Clock Synchronous Systems", **IEEE/ACM Proc. Intl. Conf. on Computer-Aided Design**, 1992, pp. 464-472.

[40] Y. -M. Li, M. A. Jabri, "A Zero-Skew Clock Routing Scheme for Vlsi Circuits", **IEEE/ACM Proc. Intl. Conf. on Computer-Aided Design**, 1992, pp. 458-463.

[41] R.-S. Tsay, "An Exact Zero-Skew Clock Routing Algorithm," **IEEE Trans. on Computer-Aided Design**, Vol. 12, No. 2, Feb., 1993, pp.242-249.

[42] D. F. Wann, and M. A. Franklin , "Asynchronous and Clocked Control Structures for Vlsi Based Interconnection Networks," **IEEE Trans. on Computers**, Vol. , No. , Mar., 1983, pp. .

[43] D. C. Wong, **Techniques for Designing High-Performance Digital Circuits Using Wave-Pipelining**, PhD Thesis, Electrical Engineering Dept., Stanford University, August 1991.

[44] D. C. Wong, G. De Micheli, M. J. Flynn, and R. E Huston, " A Bipolar Population Counter Using Wave Piplining to Achieve 2.5 x Normal Clock Frequency," **IEEE Jour. of Solid-State Circuits**, Vol. 27, No. 5, May 1992, pp. 745-753.

[45] D. A. Joy and M. J. Ciesielski, "Placement for Clock Period Minimization With Multiple Wave Propagation," **Proc. 28th Design Automation Conference**, 1991, pp. 640-643.

[46] H. F. Jordan, V. P. Heuring, and R. Feuerstein, "Optoelectronic time-of-Flight Design and the Demonstration of an All-Optical, Stored Program, Digital Computer," **Proc. IEEE**, Vol. 82, No. 11, pp. 1678-1689 (November 1994).

[47] J. P. Pratt, and V. P. Heuring, "Synchronization, Crosstalk and Loss in Optical Systems," Optoelectronic Computing Systems, University of Colorado, OCS Technical Report 89-35.

[48] J. R. Feehrer, "Minimizing the Major Clock Cycle in Bit-Serial Time-of-Flight Synchronized Digital Circuits," OCS Technical Report 93-03, Optical Computing Group, University of Colorado, Boulder, March 1993.

[49] A. Gibbons, **Algorithmic Graph Theory**, Cambridge University Press, 1985.

[50] T. H. Cormen, C. E. Leiserson, R. R. Rivest, **Introduction to Algorithms**, MIT Press, The MIT Electrical Engineering and Computer Science Series, 1990.

[51] C. E. Love, and H. F. Jordan, "Time of Flight Packet Synchronizers," **Optical Computing Technical Digest** (Optical Society of America, Washington, D.C.), Vol. 7, pp. 326-329, 1993.

APPENDIX A

THE CONNECTION TYPE DATA

The following tables show the constants established for every possible path in the layout.

Table A.1: The path dependent constants for the left-edge start position.

Start: LE	Target							
	LE	LS	Li-L	Li-R	Ri-L	Ri-R	RS	RE
Type	4	4	2	5	2	5	2	5
Points	4	4	6	6	6	6	6	6
Basic Ext.	0	0	1	0	1	0	1	0
Basic Slides	0	0	4	0	0	0	1	0
First Sl. Pt	0	0	0	0	0	0	3	0
Sec. Sl. Pt.	0	0	0	0	0	0	0	0
Desig. Pt	1	1	3	3	3	3	3	3
Track	F	F	T	T	T	T	T	T

Table A.2: The path dependent constants for the left-side start position.

Start: LS	Target							
	LE	LS	Li-L	Li-R	Ri-L	Ri-R	RS	RE
Type	4	3	1	6	1	6	1	6
Points	4	4	6	6	6	6	6	6
Basic Ext.	0	1	2	1	2	1	2	1
Basic Slides	0	1	1	1	1	1	2	1
First Sl. Pt	0	1	1	1	1	1	1	1
Sec. Sl. Pt.	0	0	0	0	0	0	3	0
Desig. Pt	1	1	3	3	3	3	3	3
Track	F	F	T	T	T	T	T	T

Table A.3. The path dependent constants for the left-inner-left start position.

Start: Li-L	Target							
	LE	LS	Li-L	Li-R	Ri-L	Ri-R	RS	RE
Type	6	1	3	4	2	5	2	5
Points	6	6	4	4	6	6	6	6
Basic Ext.	1	2	1	0	1	0	1	0
Basic Slides	0	1	0	0	0	0	1	0
First Sl. Pt	0	3	0	0	3	0	3	0
Sec. Sl. Pt.	0	0	0	0	0	0	0	0
Desig. Pt	3	3	1	1	3	3	3	3
Track	T	T	F	F	T	T	T	T

Table A.4: The path dependent constants for left-inner-right start position.

Start: Li-R	Target							
	LE	LS	Li-L	Li-R	Ri-L	Ri-R	RS	RE
Type	5	2	4	3	1	6	1	6
Points	6	6	4	4	6	6	6	6
Basic Ext.	0	1	1	0	2	1	2	1
Basic Slides	0	1	0	0	0	0	1	0
First Sl. Pt	0	3	0	0	0	0	3	0
Sec. Sl. Pt.	0	0	0	0	0	0	0	0
Desig. Pt	3	3	1	1	3	3	3	3
Track	T	T	F	F	T	T	T	T

Table A.5. The path dependent constants for the right-inner-left start position.

Start: Ri-L	Target							
	LE	LS	Li-L	Li-R	Ri-L	Ri-R	RS	RE
Type	6	1	6	1	3	4	2	5
Points	6	6	6	6	4	4	6	6
Basic Ext.	1	2	1	2	1	0	1	0
Basic Slides	0	1	0	0	0	0	1	0
First Sl. Pt	0	3	0	0	3	0	3	0
Sec. Sl. Pt.	0	0	0	0	0	0	0	0
Desig. Pt	3	3	3	3	1	1	3	3
Track	T	T	T	T	F	F	T	T

Table A.6. The path dependent constants for the right-inner-right start position.

Start: Ri-R	Target							
	LE	LS	Li-L	Li-R	Ri-L	Ri-R	RS	RE
Type	5	2	5	2	4	3	1	6
Points	6	6	6	6	4	4	6	6
Basic Ext.	0	1	0	1	0	1	2	1
Basic Slides	0	1	0	0	0	0	1	0
First Sl. Pt	0	3	0	3	0	0	3	0
Sec. Sl. Pt.	0	0	0	0	0	0	0	0
Desig. Pt	3	3	3	3	1	1	3	3

Table A.7: The path dependent constants for the right-side start position.

Start: RS	Target							
	LE	LS	Li-L	Li-R	Ri-L	Ri-R	RS	RE
Type	6	1	6	1	6	1	3	4
Points	6	6	6	6	6	6	4	4
Basic Ext.	1	2	1	2	1	2	1	0
Basic Slides	1	2	1	1	1	1	1	0
First Sl. Pt	1	1	1	3	1	3	1	0
Sec. Sl. Pt.	0	3	0	0	1	0	0	0
Desig. Pt	3	3	3	3	0	3	1	3
Track	T	T	T	T	T	T	F	F

Table A.8: The path dependent constants for the right-edge start position.

Start: RE	Target							
	LE	LS	Li-L	Li-R	Ri-L	Ri-R	RS	RE
Type	5	2	5	2	5	2	4	4
Points	6	6	6	6	6	6	4	4
Basic Ext.	0	1	0	1	0	1	0	0
Basic Slides	0	1	0	0	0	0	0	0
First Sl. Pt	0	3	0	0	0	0	0	0
Sec. Sl. Pt.	0	0	0	0	0	0	0	0
Desig. Pt	3	3	3	3	3	3	1	1
Track	T	T	T	T	T	T	F	F

APPENDIX B

THE SCHEMATIC OF THE LOGARITHMIC DELAY LINE

The following figures show the circuit schematic of the logarithmic delay line.

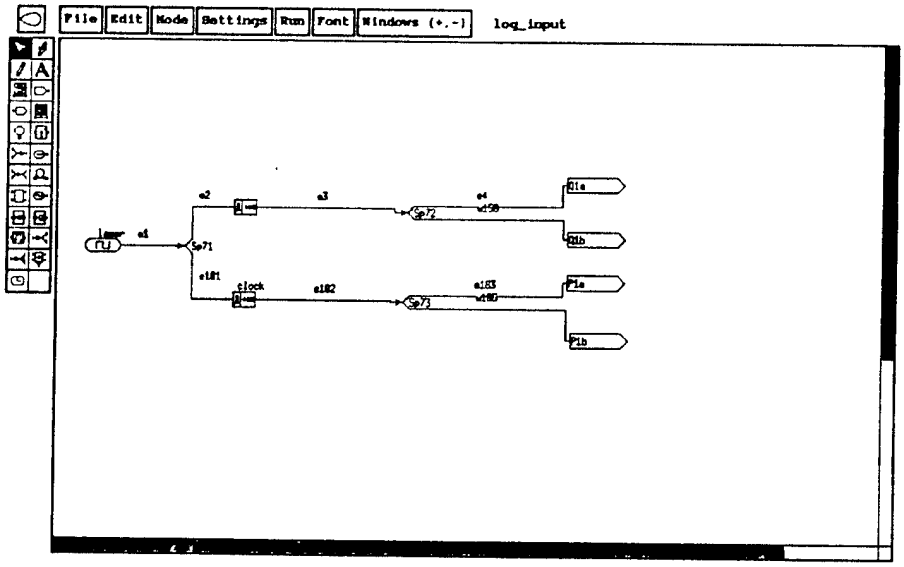


Figure B.1: Input to the logarithmic delay line.

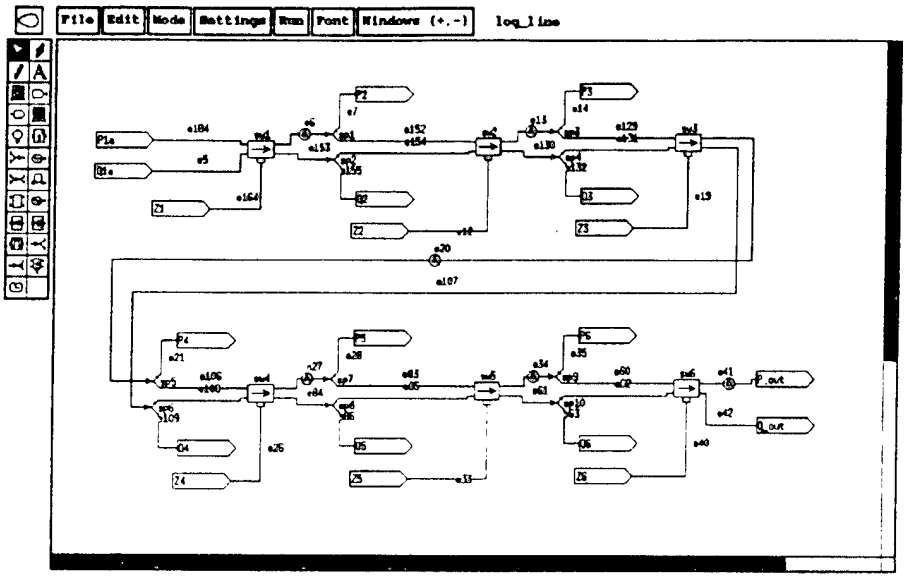


Figure B.2: The main circuit of the logarithmic delay line.

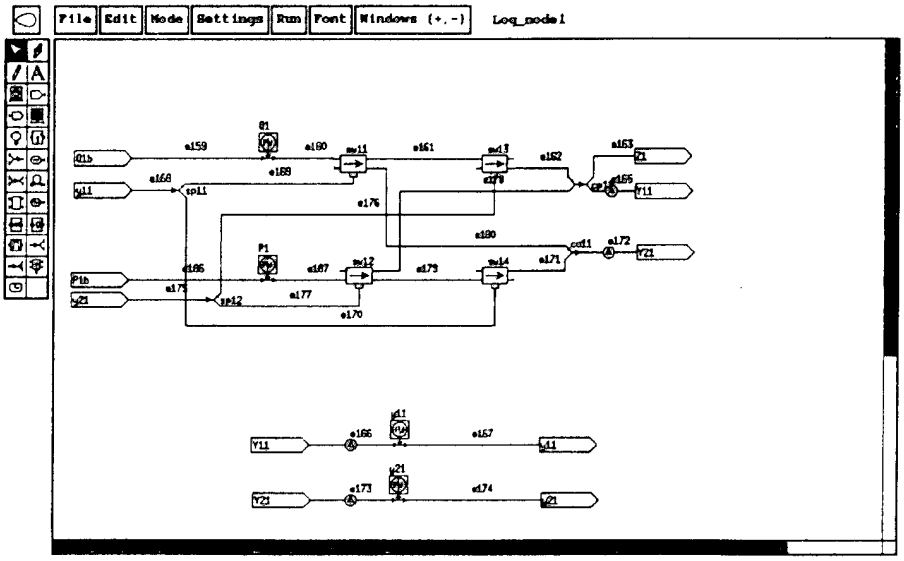


Figure B.3: Subcircuit 1.

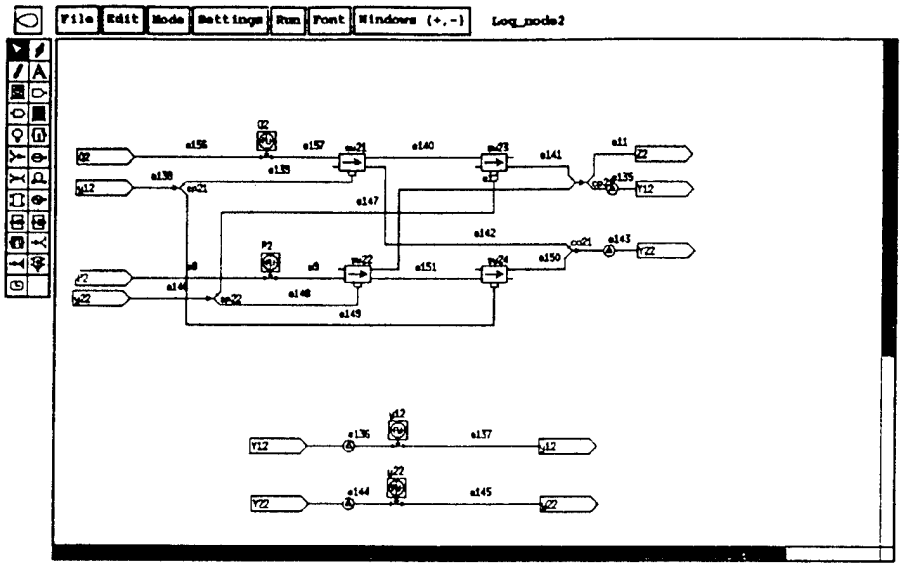


Figure B.4: Subcircuit 2.

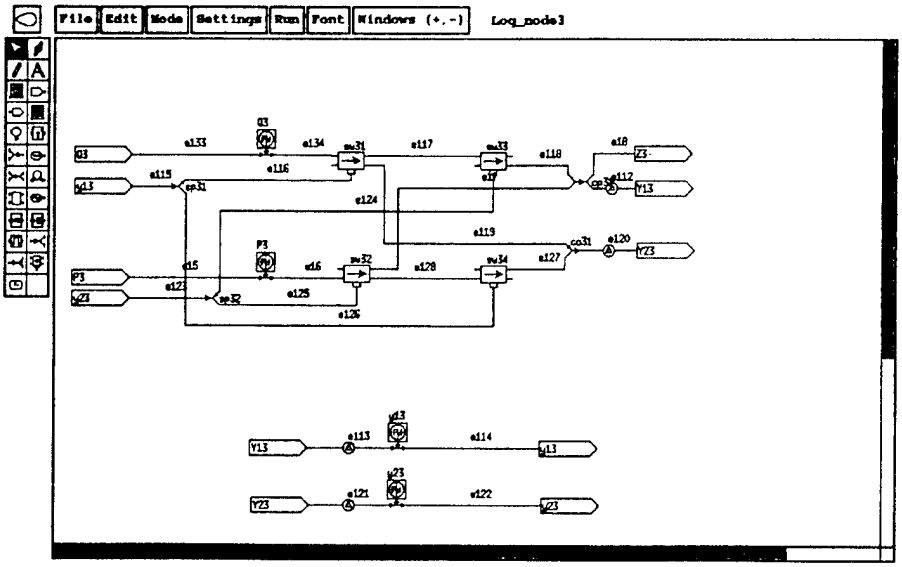


Figure B.5: Subcircuit 3.

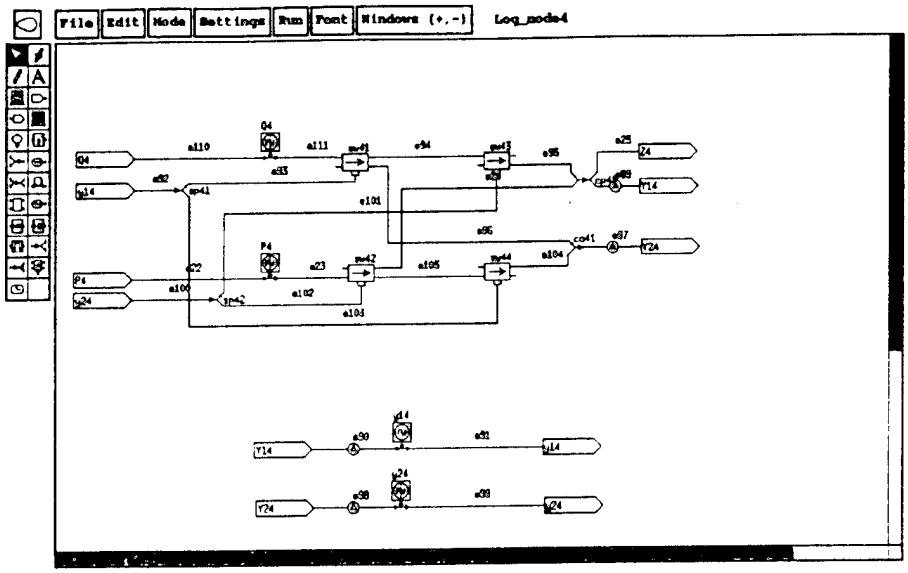


Figure B.6: Subcircuit 4.

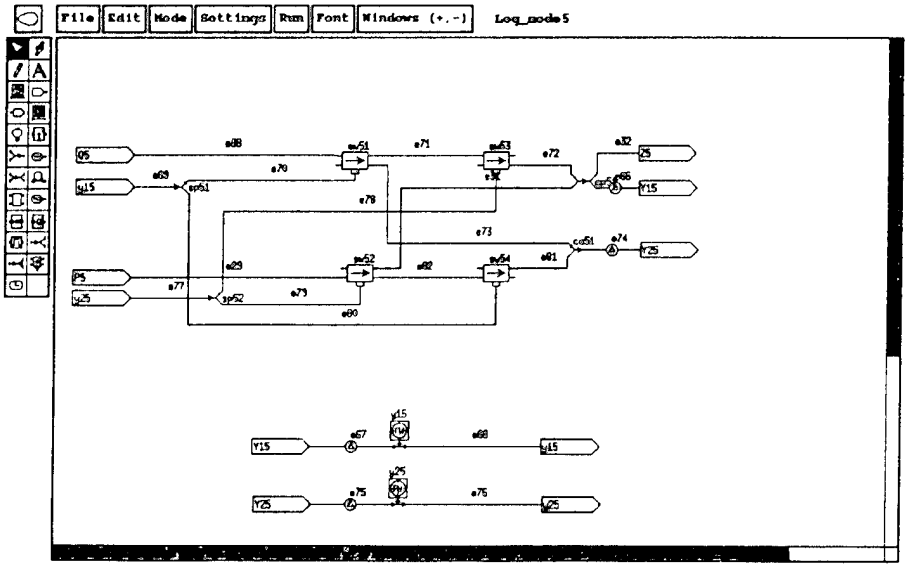


Figure B.7: Subcircuit 5.

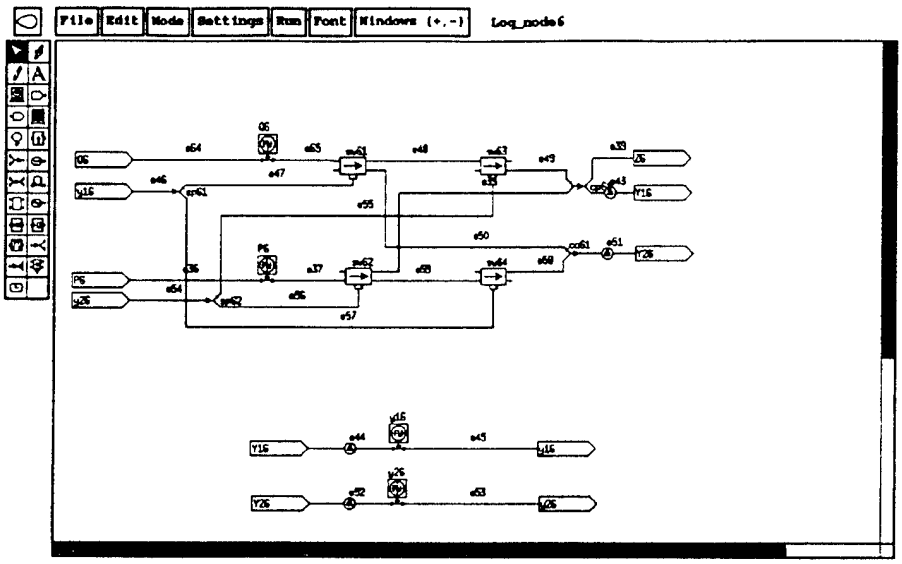
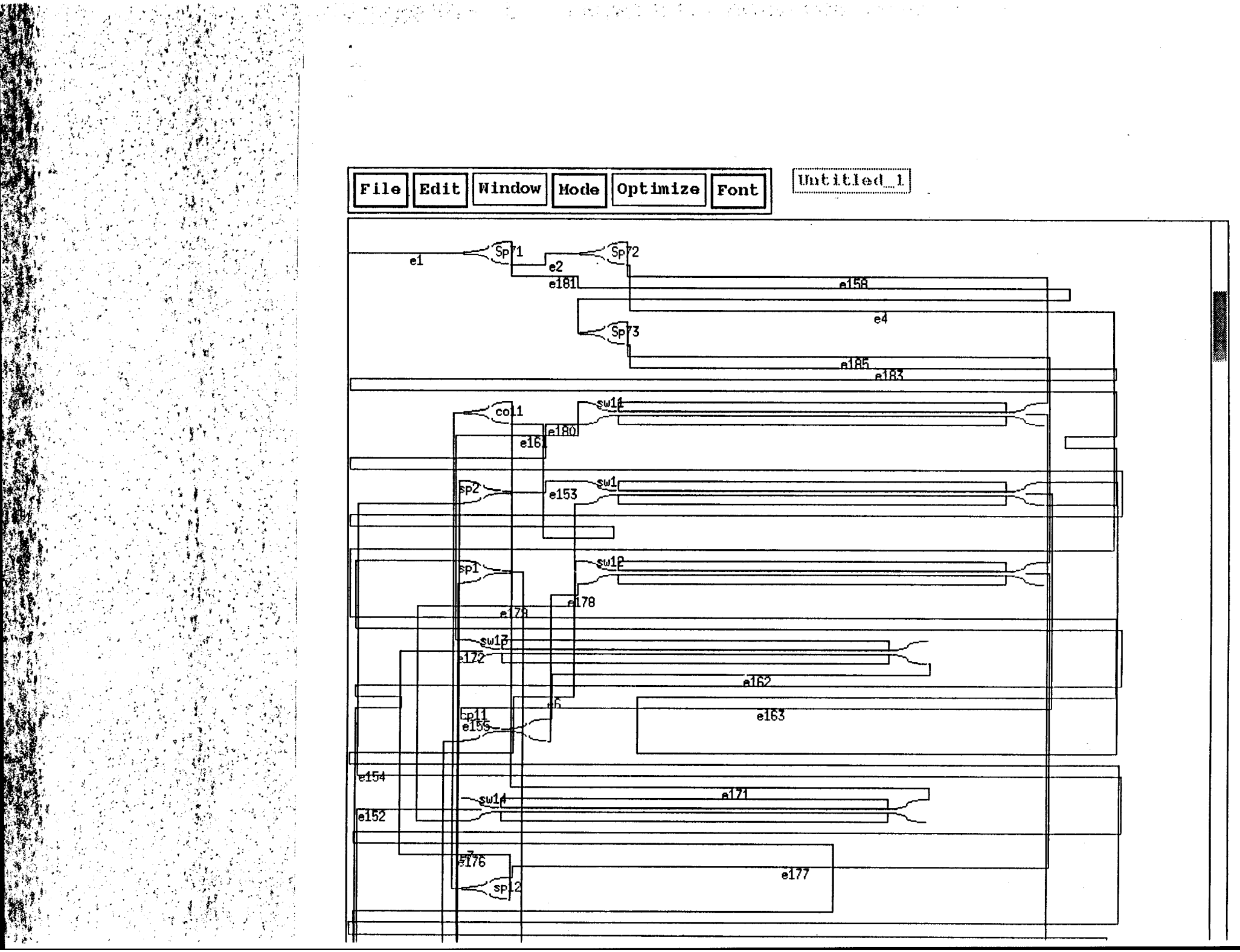
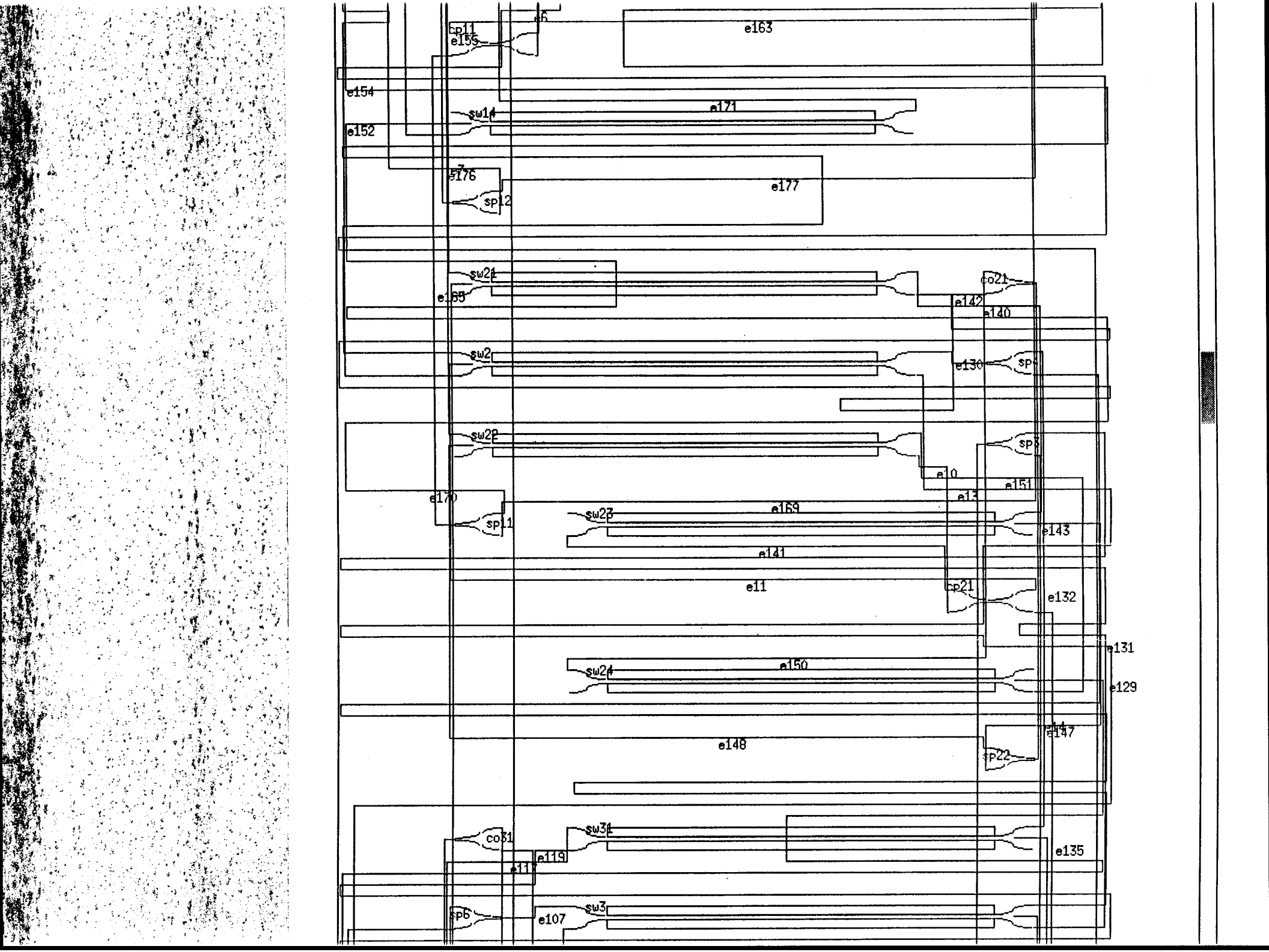
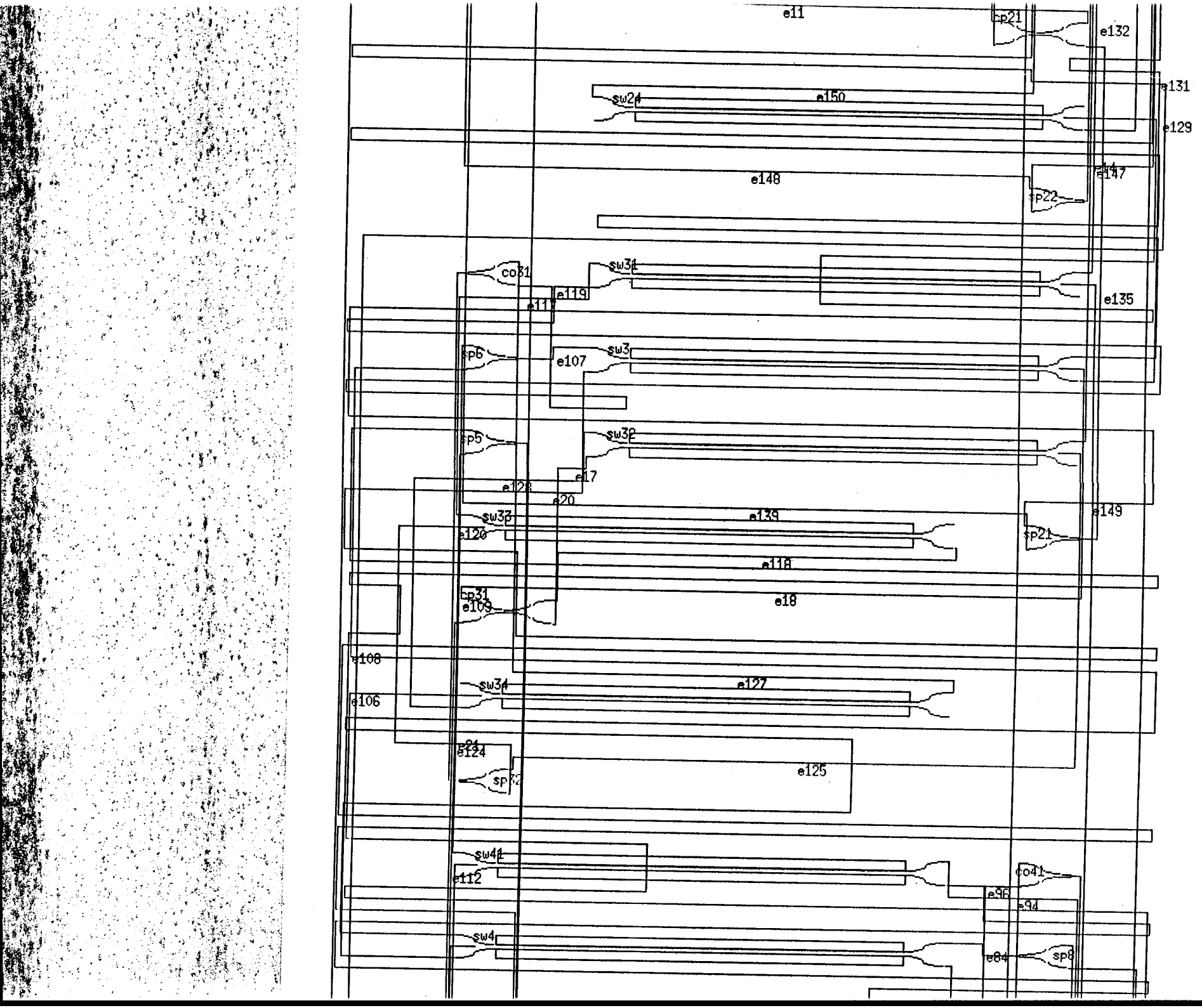
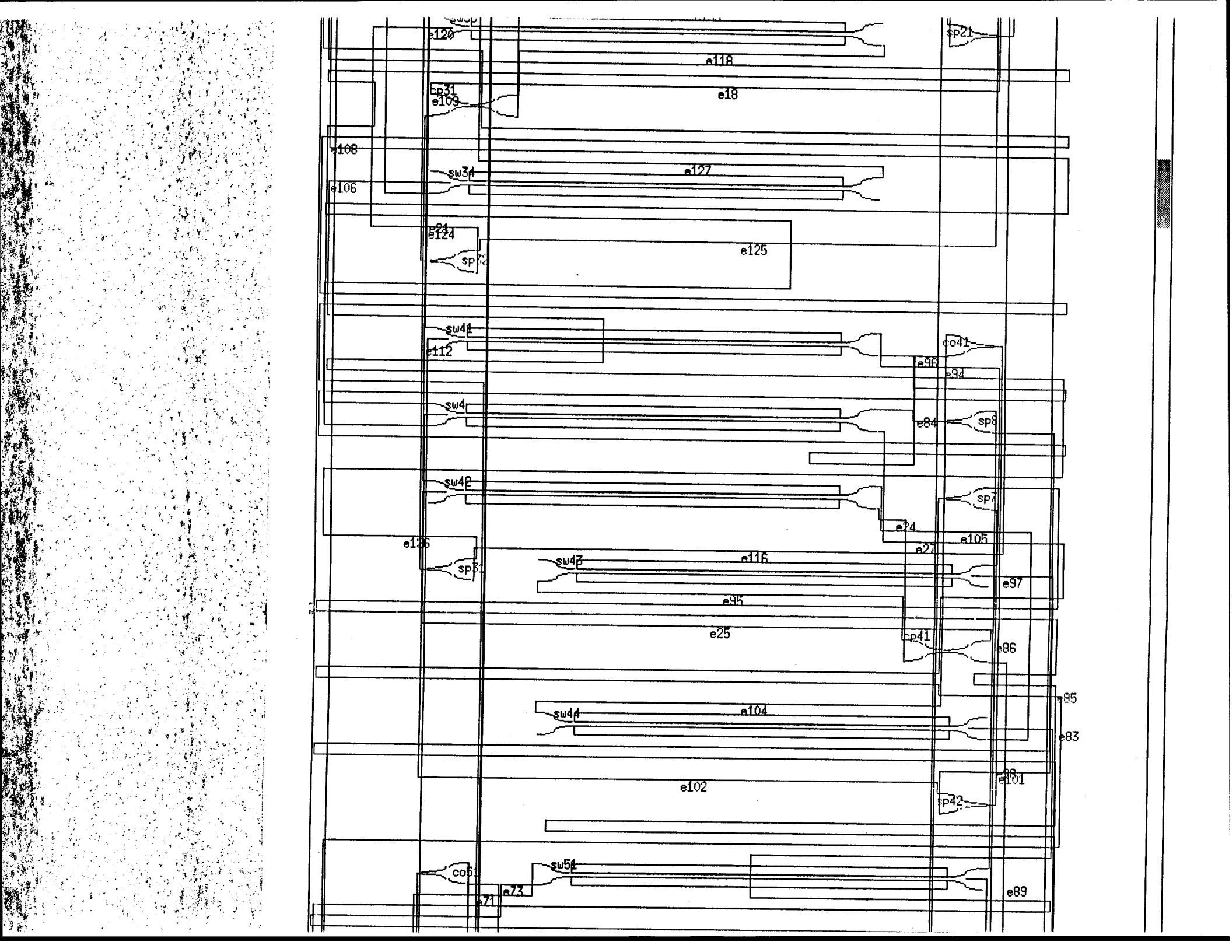


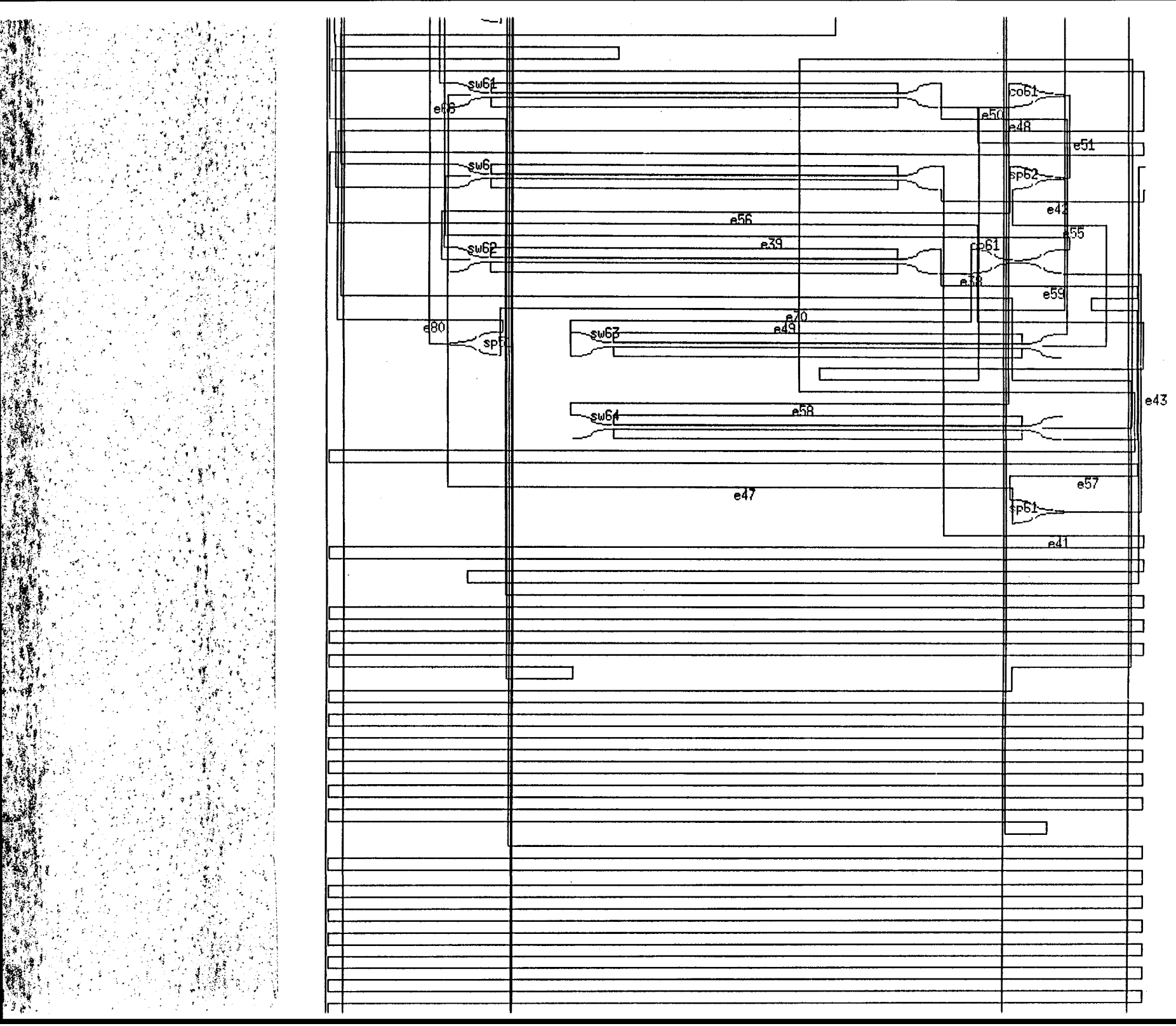
Figure B.8: Subcircuit 6.

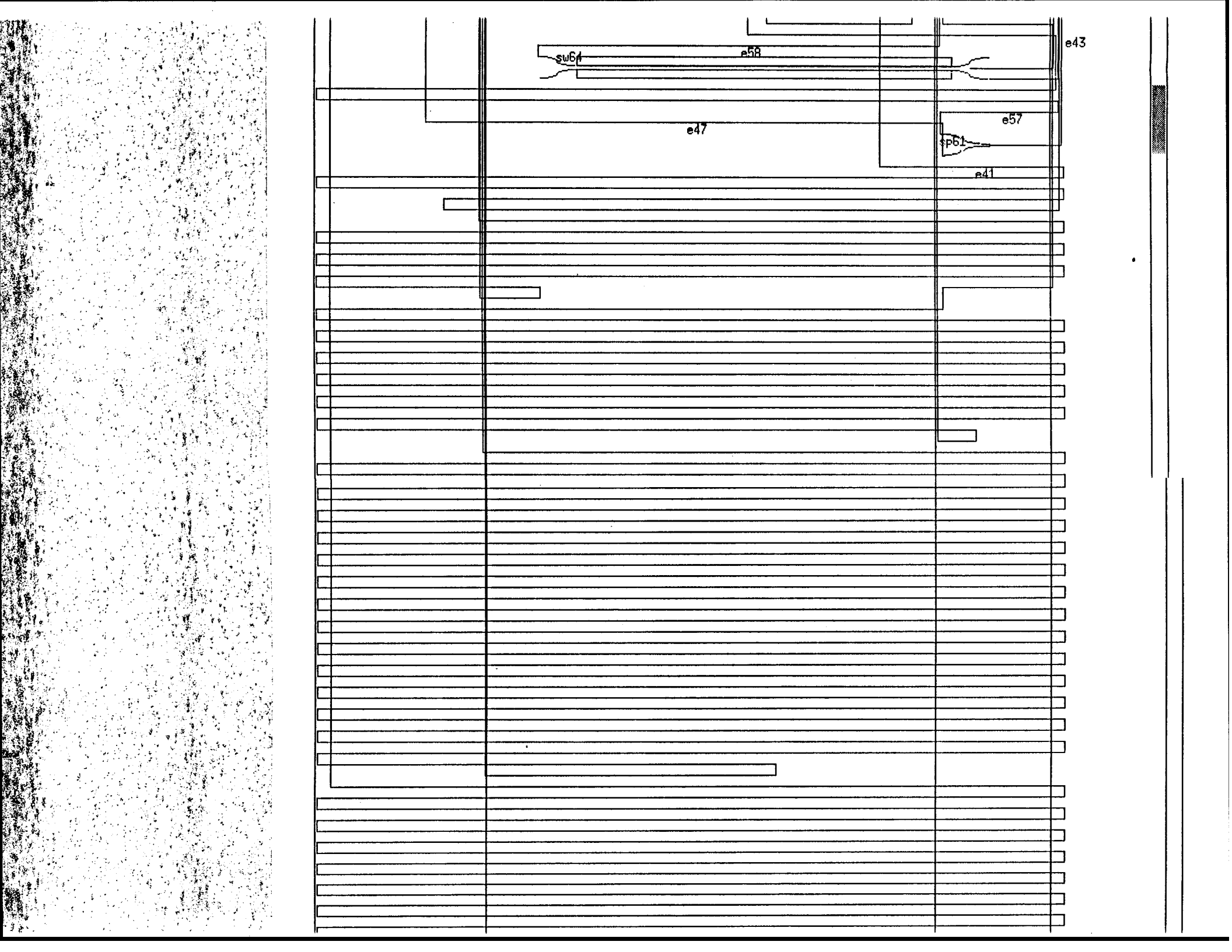

File
Edit
Window
Mode
Optimize
Font
Untitled_1
e1
Sp71
e2
Sp72
e181
e158
e4
Sp73
e185
e183
col1
sw14
e180
e161
sp2
e153
sw1
sp1
sw12
e178
e179
sw13
e172
e162
e163
sp11
e155
e154
sw14
e171
e152
e176
sp12
e177

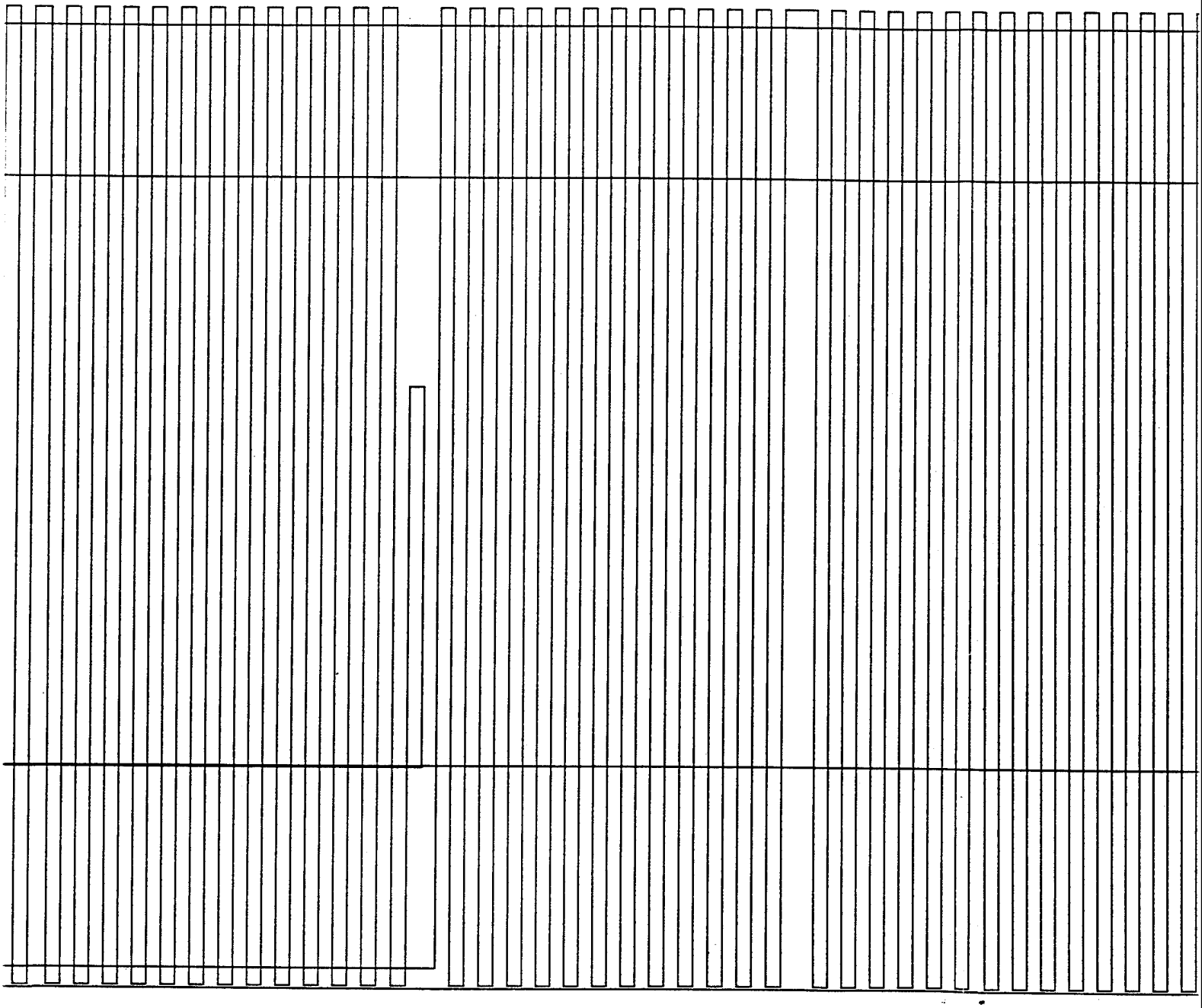

e163
e154
e171
sw14
e152
e176
e177
sp12
sw21
co21
e142
e140
sw2
e130
sp4
sw22
sp3
e151
e170
sp11
sw23
e169
e143
e141
e11
sp21
e132
e131
sw24
e150
e129
e148
sp22
e147
co31
sw31
e119
e135
e117
sp6
e107
sw3

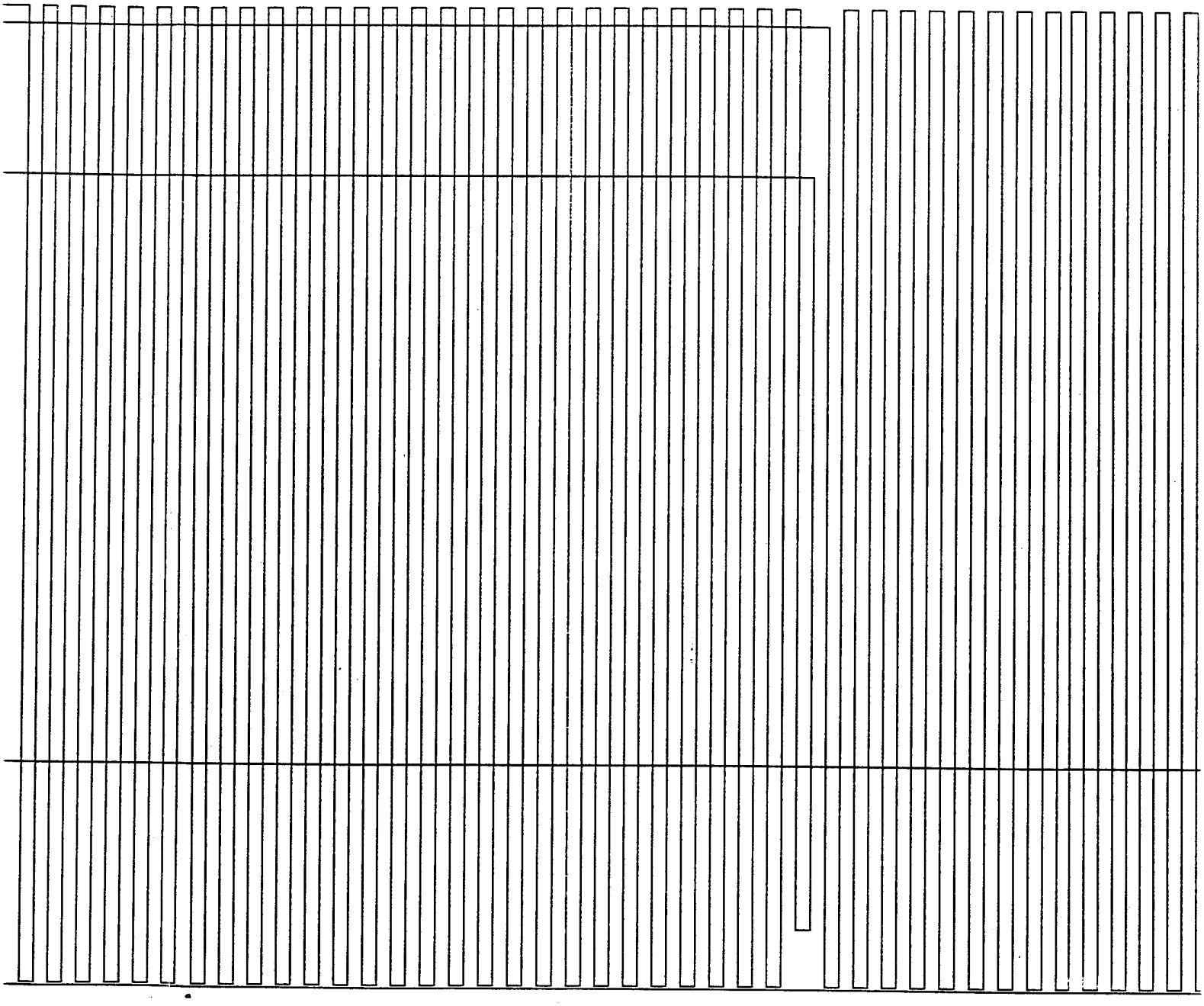

e11
sp21
e132
e131
sw24
e150
e129
e148
e147
sp22
co31
sw31
e119
e135
sp6
e107
sw3
sp5
sw32
e17
e20
e149
sw33
e139
sp21
e120
e118
sp31
e109
e18
e108
sw34
e127
e106
e124
e125
sw41
co41
e112
e96
e94
sw4
e84
sp8

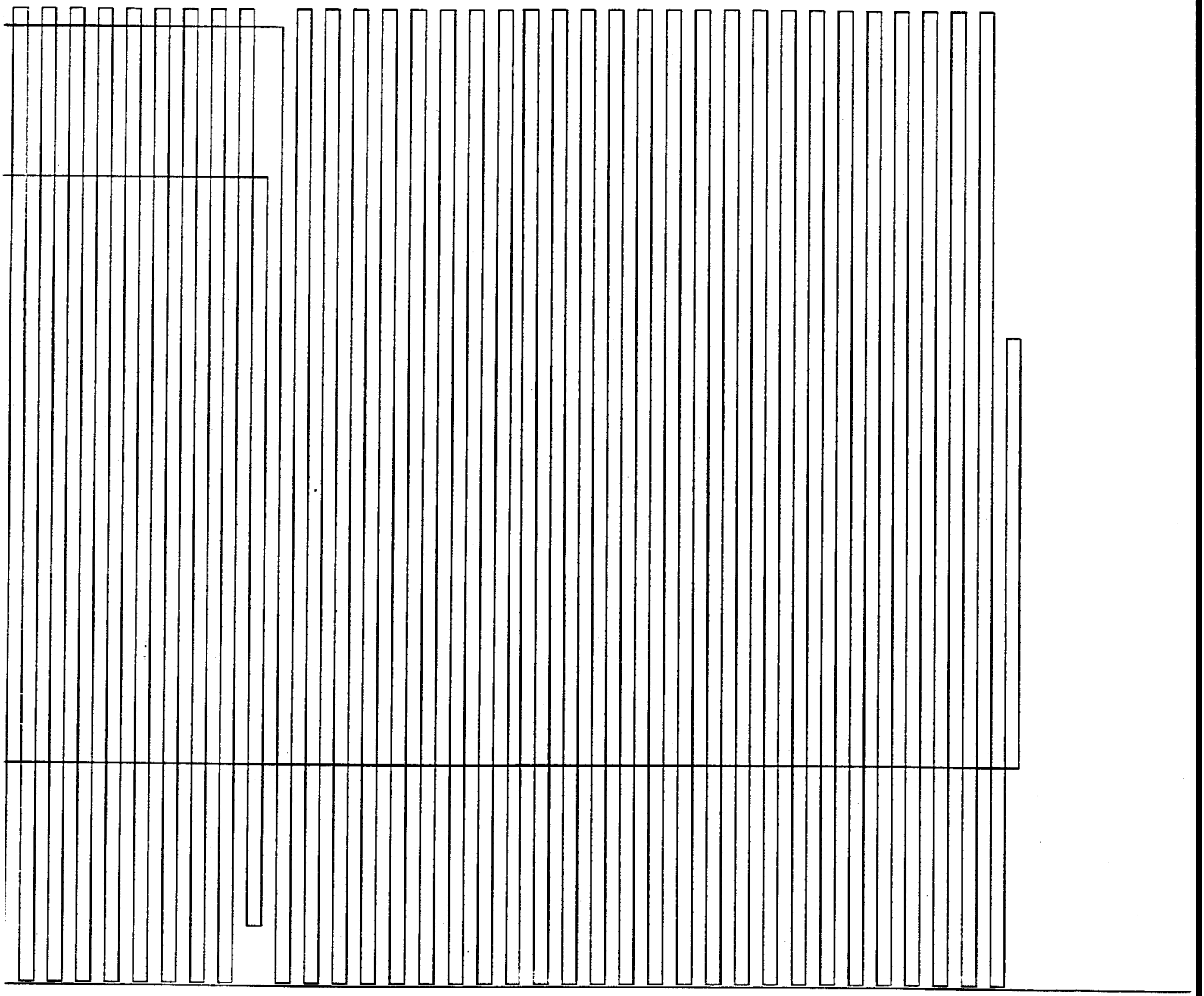

e120
e118
e18
e109
e108
e127
e106
e124
e125
e112
e84
e94
e74
e105
e116
e97
e95
e25
e86
e104
e85
e83
e102
e101
e73
e89
sp21
sp31
sw34
sw41
sw4
sw42
sw43
sw44
sw51
co41
sp8
sp41
sp42
co51


sw61
co61
e50
e48
e51
sw6
sp62
e42
e56
e55
sw62
e39
sp61
e38
e59
e70
e80
e49
sw63
e43
sw64
e58
e57
e47
sp61
e41


e43
sw64
e58
e57
e47
sp61
e41







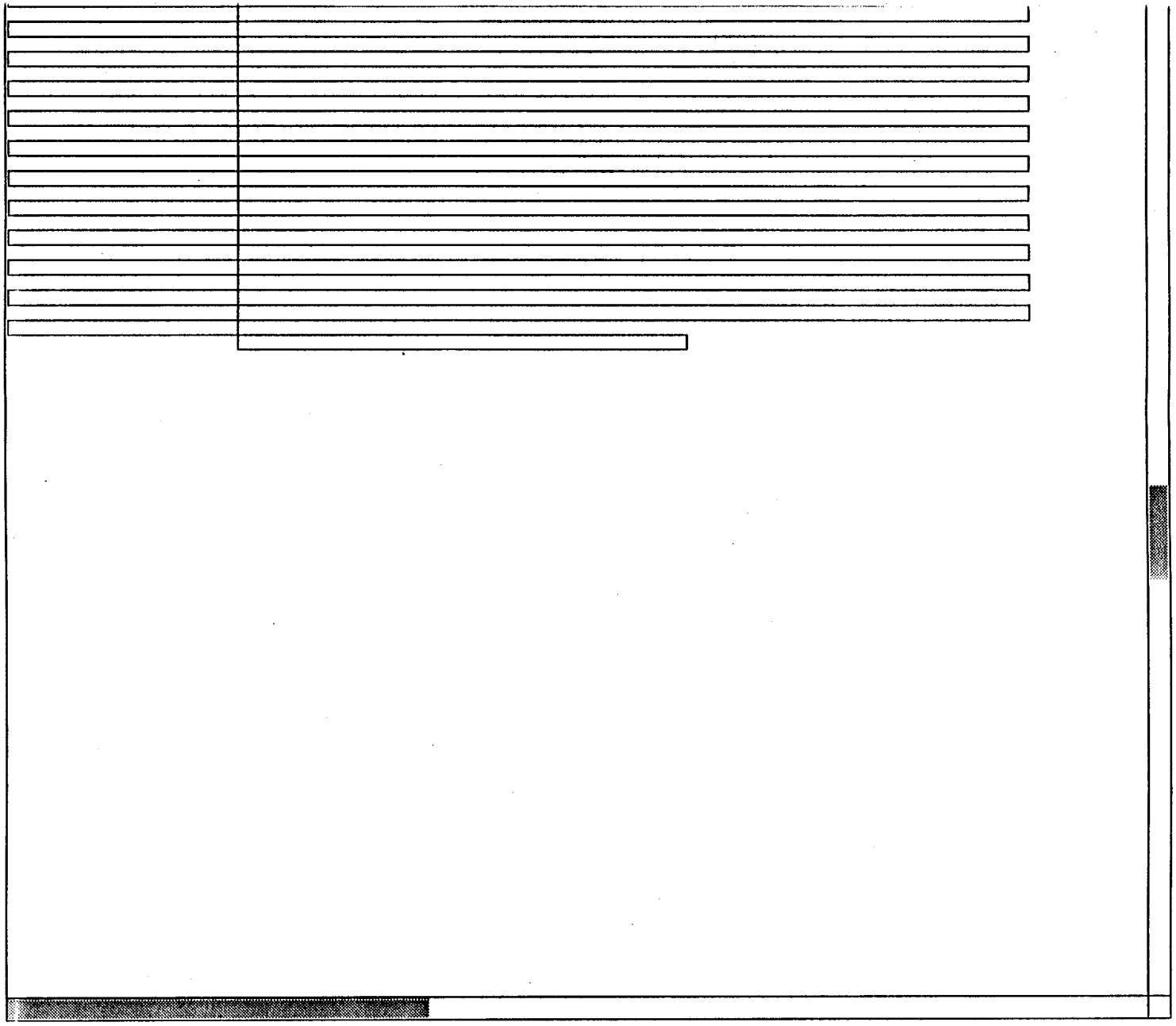


Figure 5.21: Entire Layout of the Logarithmic Delay Line