ROUTING

Once the designer has floorplanned a chip and the logic cells within the flexible blocks have been placed, it is time to make the connections by routing the chip. This is still a hard problem that is made easier by dividing it into smaller problems. Routing is usually split into global routing followed by detailed routing .

Suppose the ASIC is North America and some travelers in California need advice on how to drive from Stanford (near San Francisco) to Caltech (near Los Angeles). The floorplanner has decided that California is on the left (west) side of the ASIC and the placement tool has put Stanford in Northern California and Caltech in Southern California. Floorplanning and placement have defined the roads and freeways. There are two ways to go: the coastal route (using Highway 101) or the inland route (using Interstate I5, which is usually faster). The global router specifies the coastal route because the travelers are not in a hurry and I5 is congested (the global router knows this because it has already routed onto I5 many other travelers that are in a hurry today). Next, the detailed router looks at a map and gives indications from Stanford onto Highway 101 south through San Jose, Monterey, and Santa Barbara to Los Angeles and then off the freeway to Caltech in Pasadena.

Figure 17.1 shows the core of the Viterbi decoder after the placement step. This implementation consists entirely of standard cells (18 rows). The I/O pads are not included in this example-we can route the I/O pads after we route the core (though this is not always a good idea). Figure 17.2 shows the Viterbi decoder chip after global and detailed routing. The routing runs in the channels between the rows of logic cells, but the individual interconnections are too small to see.

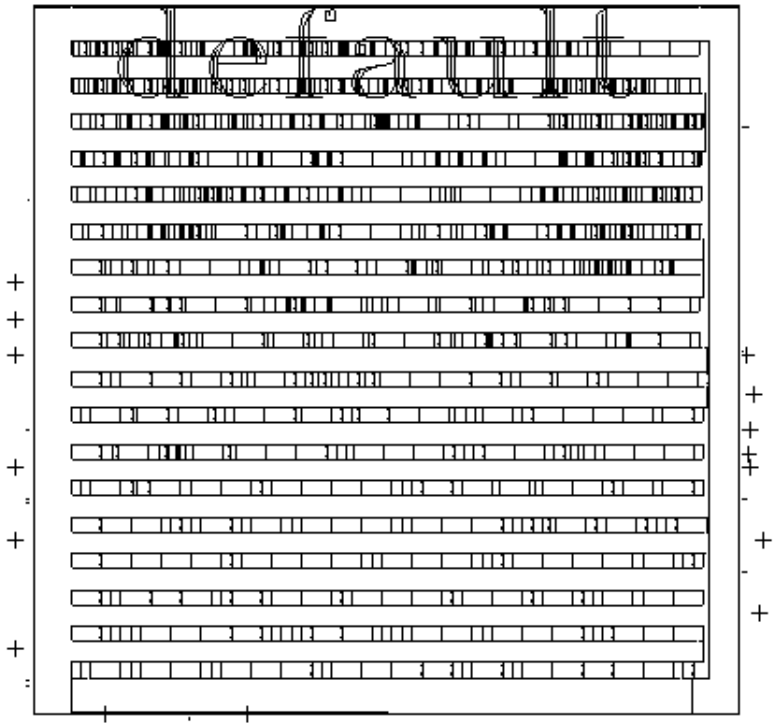


FIGURE 17.1 The core of the Viterbi decoder chip after placement (a screen shot from Cadence Cell Ensemble). This is the same placement as shown in Figure 16.2, but without the channel labels. You

can see the rows of standard cells; the widest cells are the D flip-flops.

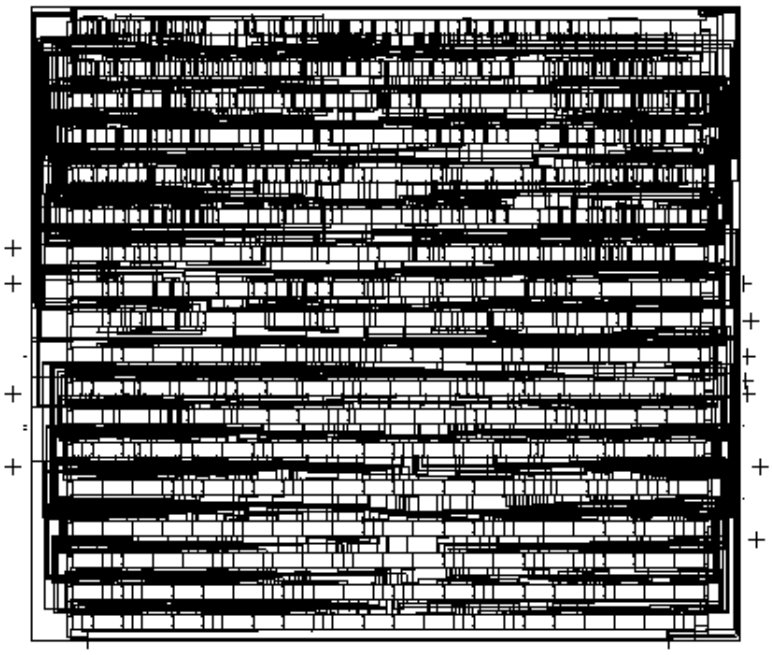


FIGURE 17.2 The core of the Viterbi decoder chip after the completion of global and detailed routing (a screen shot from Cadence Cell Ensemble). This chip uses two-level metal. Although you cannot see the difference, m1 runs in the horizontal direction and m2 in the vertical direction.

17.1 Global Routing

17.2 Detailed Routing

17.3 Special Routing

17.4 Circuit Extraction and DRC

17.5 Summary

17.6 Problems

17.7 Bibliography

17.8 References

17.1 Global Routing

The details of global routing differ slightly between cell-based ASICs, gate arrays, and FPGAs, but the principles are the same in each case. A global router does not make any connections, it just plans them. We typically global route the whole chip (or large pieces if it is a large chip) before detail routing the whole chip (or the pieces). There are two types of areas to global route: inside the flexible blocks and between blocks (the Viterbi decoder, although a cell-based ASIC, only involved the global routing of one large flexible block).

17.1.1 Goals and Objectives

The input to the global router is a floorplan that includes the locations of all the fixed and flexible blocks; the placement information for flexible blocks; and the locations of all the logic cells. The goal of global routing is to provide complete instructions to the detailed router on where to route every net. The objectives of global routing are one or more of the following:

- Minimize the total interconnect length.
- Maximize the probability that the detailed router can complete the routing.
- Minimize the critical path delay.

In both floorplanning and placement, with minimum interconnect length as an objective, it is necessary to find the shortest total path length connecting a set of terminals . This path is the MRST, which is hard to find. The alternative, for both floorplanning and placement, is to use simple approximations to the length of the MRST (usually the half-perimeter measure). Floorplanning and placement both assume that interconnect may be put anywhere on a rectangular grid, since at this point nets have not been assigned to the channels, but the global router must use the wiring channels and find the actual path. Often the global router needs to find a path that minimizes the delay between two terminals-this is not necessarily the same as finding the shortest total path length for a set of terminals.

17.1.2 Measurement of Interconnect Delay

Floorplanning and placement need a fast and easy way to estimate the interconnect delay in order to evaluate each trial placement; often this is a predefined look-up table. After placement, the logic cell positions are fixed and the global router can afford to use better estimates of the interconnect delay. To illustrate one method, we shall use the Elmore constant to estimate the interconnect delay for the circuit shown in Figure 17.3 .

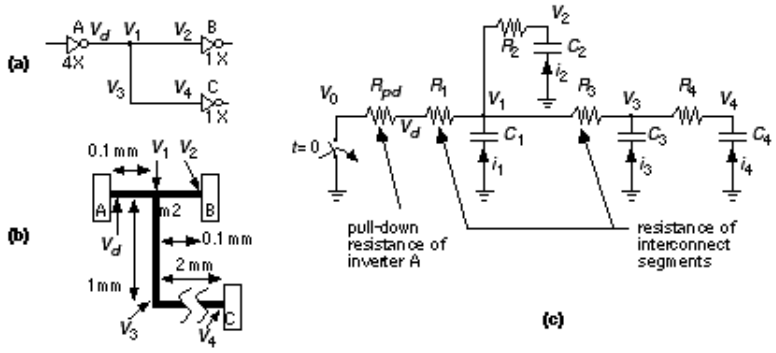


FIGURE 17.3 Measuring the delay of a net. (a) A simple circuit with an inverter A driving a net with a fanout of two. Voltages V_1, V_2, V_3, and V_4 are the voltages at intermediate points along the net. (b) The layout showing the net segments (pieces of interconnect). (c) The RC model with each segment replaced by a capacitance and resistance. The ideal switch and pull-down resistance R_{pd} model the inverter A.

The problem is to find the voltages at the inputs to logic cells B and C taking into account the parasitic resistance and capacitance of the metal interconnect. Figure 17.3 (c) models logic cell A as an ideal switch with a pull-down resistance equal to R_{pd} and models the metal interconnect using resistors and capacitors for each segment of the interconnect.

The Elmore constant for node 4 (labeled V_4) in the network shown in Figure 17.3 (c) is

$$t_{D4} = \sum_{k=1}^{4} R_{k4} C_k \qquad (17.1)$$

$$= R_{14} C_1 + R_{24} C_2 + R_{34} C_3 + R_{44} C_4 ,$$

where,

$$R_{14} = R_{pd} + R_1 \qquad (17.2)$$
$$R_{24} = R_{pd} + R_1$$
$$R_{34} = R_{pd} + R_1 + R_3$$
$$R_{44} = R_{pd} + R_1 + R_3 + R_4$$

In Eq. 17.2 notice that $R_{24} = R_{pd} + R_1$ (and not $R_{pd} + R_1 + R_2$) because R_1 is the resistance to V_0 (ground) shared by node 2 and node 4.

Suppose we have the following parameters (from the generic 0.5 m m CMOS process, G5) for the layout

shown in Figure 17.3 (b):

- m2 resistance is 50 m W /square.
- m2 capacitance (for a minimum-width line) is 0.2 pFmm $^{-1}$.
- 4X inverter delay is 0.02 ns + 0.5 C_L ns (C_L is in picofarads).
- Delay is measured using 0.35/0.65 output trip points.
- m2 minimum width is 3 l = 0.9 m m.
- 1X inverter input capacitance is 0.02 pF (a standard load).

First we need to find the pull-down resistance, R_{pd} , of the 4X inverter. If we model the gate with a linear pull-down resistor, R_{pd} , driving a load C_L , the output waveform is exp - t /(C_L R_{pd}) (normalized to 1V). The output reaches 63 percent of its final value when t = C_L R_{pd} , because exp (-1) = 0.63. Then, because the delay is measured with a 0.65 trip point, the constant 0.5 nspF $^{-1}$ = 0.5 k W is very close to the equivalent pull-down resistance. Thus, $R_{pd} \approx 500$ W .

From the given data, we can calculate the R 's and C 's:

$$R_1 = R_2 = \frac{(0.1 \text{ mm})(50 \times 10^{-3} \text{ W})}{0.9 \text{ m m}} = 6 \text{ W}$$

$$R_3 = \frac{(1 \text{ mm})(50 \times 10^{-3} \text{ W})}{0.9 \text{ m m}} = 56 \text{ W}$$

$$R_4 = \frac{(2 \text{ mm})(50 \times 10^{-3} \text{ W})}{0.9 \text{ m m}} = 112 \text{ W}$$

(17.3)

C_1 = (0.1 mm) (0.2 ¥ pFmm $^{-1}$) = 0.02 pF

C_2 = (0.1 mm) (0.2 ¥ pFmm $^{-1}$) + 0.02 pF = 0.04 pF

C_3 = (1 mm) (0.2 ¥ pFmm $^{-1}$) = 0.2 pF

C_4 = (2 mm) (0.2 ¥ pFmm $^{-1}$) + 0.02 pF = 0.42 pF

(17.4)

Now we can calculate the path resistance, R_{ki} , values (notice that $R_{ki} = R_{ik}$):

$$R_{14} = 500\ W + 6\ W = 506\ W$$
$$R_{24} = 500\ W + 6\ W = 506\ W$$
$$R_{34} = 500\ W + 6\ W + 56\ W = 562\ W$$
$$R_{44} = 500\ W + 6\ W + 56\ W + 112\ W = 674\ W \quad (17.5)$$

Finally, we can calculate Elmore's constants for node 4 and node 2 as follows:

$$t_{D4} = R_{14}C_1 + R_{24}C_2 + R_{34}C_3 + R_{44}C_4 \quad (17.6)$$
$$= (506)(0.02) + (506)(0.04) + (562)(0.2) + (674)(0.42)$$
$$= 425\ ps\ .$$
$$t_{D2} = R_{12}C_1 + R_{22}C_2 + R_{32}C_3 + R_{42}C_4 \quad (17.7)$$
$$= (R_{pd} + R_1)(C_2 + C_3 + C_4) + (R_{pd} + R_1 + R_2)C_2$$
$$= (500 + 6 + 6)(0.04) + (500 + 6)(0.02 + 0.2 + 0.2)$$
$$= 344\ ps\ .$$

and $t_{D4} - t_{D2} = (425 - 344) = 81$ ps.

A lumped-delay model neglects the effects of interconnect resistance and simply sums all the node capacitances (the lumped capacitance) as follows:

$$t_D = R_{pd}(C_1 + C_2 + C_3 + C_4) \quad (17.8)$$
$$= (500)(0.02 + 0.04 + 0.2 + 0.42)$$
$$= 340\ ps\ .$$

Comparing Eqs. 17.6 - 17.8 , we can see that the delay of the inverter can be assigned as follows: 20 ps (the intrinsic delay, 0.2 ns, due to the cell output capacitance), 340 ps (due to the pull-down resistance and the output capacitance), 4 ps (due to the interconnect from A to B), and 65 ps (due to the interconnect from A to C). We can see that the error from neglecting interconnect resistance can be important.

Even using the Elmore constant we still made the following assumptions in estimating the path delays:

- A step-function waveform drives the net.
- The delay is measured from when the gate input changes.
- The delay is equal to the time constant of an exponential waveform that approximates the actual

output waveform.
- The interconnect is modeled by discrete resistance and capacitance elements.

The global router could use more sophisticated estimates that remove some of these assumptions, but there is a limit to the accuracy with which delay can be estimated during global routing. For example, the global router does not know how much of the routing is on which of the layers, or how many vias will be used and of which type, or how wide the metal lines will be. It may be possible to estimate how much interconnect will be horizontal and how much is vertical. Unfortunately, this knowledge does not help much if horizontal interconnect may be completed in either m1 or m3 and there is a large difference in parasitic capacitance between m1 and m3, for example.

When the global router attempts to minimize interconnect delay, there is an important difference between a path and a net. The path that minimizes the delay between two terminals on a net is not necessarily the same as the path that minimizes the total path length of the net. For example, to minimize the path delay (using the Elmore constant as a measure) from the output of inverter A in Figure 17.3 (a) to the input of inverter B requires a rather complicated algorithm to construct the best path. We shall return to this problem in Section 17.1.6 .

17.1.3 Global Routing Methods

Global routing cannot use the interconnect-length approximations, such as the half-perimeter measure, that were used in placement. What is needed now is the actual path and not an approximation to the path length. However, many of the methods used in global routing are still based on the solutions to the tree on a graph problem.

One approach to global routing takes each net in turn and calculates the shortest path using tree on graph algorithms-with the added restriction of using the available channels. This process is known as sequential routing . As a sequential routing algorithm proceeds, some channels will become more congested since they hold more interconnects than others. In the case of FPGAs and channeled gate arrays, the channels have a fixed channel capacity and can only hold a certain number of interconnects. There are two different ways that a global router normally handles this problem. Using order-independent routing , a global router proceeds by routing each net, ignoring how crowded the channels are. Whether a particular net is processed first or last does not matter, the channel assignment will be the same. In order-independent routing, after all the interconnects are assigned to channels, the global router returns to those channels that are the most crowded and reassigns some interconnects to other, less crowded, channels. Alternatively, a global router can consider the number of interconnects already placed in various channels as it proceeds. In this case the global routing is order dependent -the routing is still sequential, but now the order of processing the nets will affect the results. Iterative improvement or simulated annealing may be applied to the solutions found from both order-dependent and order-independent algorithms. This is implemented in the same way as for system partitioning and placement: A constructed solution is successively changed, one interconnect path at a time, in a series of random moves.

In contrast to sequential global-routing methods, which handle nets one at a time, hierarchical routing handles all nets at a particular level at once. Rather than handling all of the nets on the chip at the same time, the global-routing problem is made more tractable by dividing the chip area into levels of hierarchy. By considering only one level of hierarchy at a time the size of the problem is reduced at each level. There are two ways to traverse the levels of hierarchy. Starting at the whole chip, or highest level,

and proceeding down to the logic cells is the top-down approach. The bottom-up approach starts at the lowest level of hierarchy and globally routes the smallest areas first.

17.1.4 Global Routing Between Blocks

Figure 17.4 illustrates the global-routing problem for a cell-based ASIC. Each edge in the channel-intersection graph in Figure 17.4 (c) represents a channel. The global router is restricted to using these channels. The weight of each edge in the graph corresponds to the length of the channel. The global router plans a path for each interconnect using this graph.

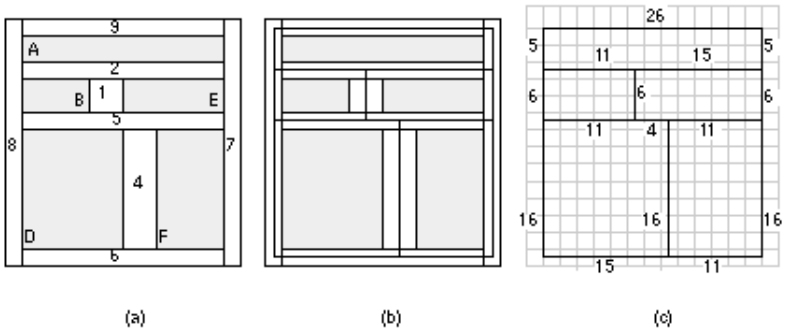


FIGURE 17.4 Global routing for a cell-based ASIC formulated as a graph problem. (a) A cell-based ASIC with numbered channels. (b) The channels form the edges of a graph. (c) The channel-intersection graph. Each channel corresponds to an edge on a graph whose weight corresponds to the channel length.

Figure 17.5 shows an example of global routing for a net with five terminals, labeled A1 through F1, for the cell-based ASIC shown in Figure 17.4 . If a designer wishes to use minimum total interconnect path length as an objective, the global router finds the minimum-length tree shown in Figure 17.5 (b). This tree determines the channels the interconnects will use. For example, the shortest connection from A1 to B1 uses channels 2, 1, and 5 (in that order). This is the information the global router passes to the detailed router. Figure 17.5 (c) shows that minimizing the total path length may not correspond to minimizing the path delay between two points.

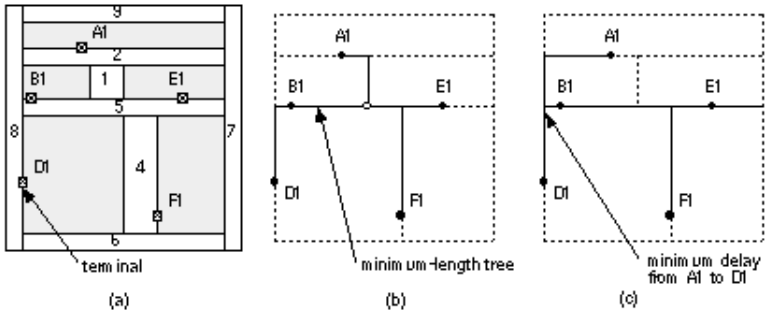


FIGURE 17.5 Finding paths in global routing. (a) A cell-based ASIC (from Figure 17.4) showing a

single net with a fanout of four (five terminals). We have to order the numbered channels to complete the interconnect path for terminals A1 through F1. (b) The terminals are projected to the center of the nearest channel, forming a graph. A minimum-length tree for the net that uses the channels and takes into account the channel capacities. (c) The minimum-length tree does not necessarily correspond to minimum delay. If we wish to minimize the delay from terminal A1 to D1, a different tree might be better.

Global routing is very similar for cell-based ASICs and gate arrays, but there is a very important difference between the types of channels in these ASICs. The size of the channels in sea-of-gates arrays, channelless gate arrays, and cell-based ASICs can be varied to make sure there is enough space to complete the wiring. In channeled gate-arrays and FPGAs the size, number, and location of channels are fixed. The good news is that the global router can allocate as many interconnects to each channel as it likes, since that space is committed anyway. The bad news is that there is a maximum number of interconnects that each channel can hold. If the global router needs more room, even in just one channel on the whole chip, the designer has to repeat the placement-and-routing steps and try again (or use a bigger chip).

17.1.5 Global Routing Inside Flexible Blocks

We shall illustrate global routing using a gate array. Figure 17.6 (a) shows the routing resources on a sea-of-gates or channelless gate array. The gate array base cells are arranged in 36 blocks, each block containing an array of 8-by-16 gate-array base cells, making a total of 4068 base cells.

The horizontal interconnect resources are the routing channels that are formed from unused rows of the gate-array base cells, as shown in Figure 17.6 (b) and (c). The vertical resources are feedthroughs. For example, the logic cell shown in Figure 17.6 (d) is an inverter that contains two types of feedthrough. The inverter logic cell uses a single gate-array base cell with terminals (or connectors) located at the top and bottom of the logic cell. The inverter input pin has two electrically equivalent terminals that the global router can use as a feedthrough. The output of the inverter is connected to only one terminal. The remaining vertical track is unused by the inverter logic cell, so this track forms an uncommitted feedthrough.

You may see any of the terms landing pad (because we say that we "drop" a via to a landing pad), pick-up point , connector , terminal , pin , or port used for the connection to a logic cell. The term pick-up point refers to the physical pieces of metal (or sometimes polysilicon) in the logic cell to which the router connects. In a three-level metal process, the global router may be able to connect to anywhere in an area-an area pick-up point . In this book we use the term connector to refer to the physical pick-up point. The term pin more often refers to the connection on a logic schematic icon (a dot, square box, or whatever symbol is used), rather than layout. Thus the difference between a pin and a connector is that we can have multiple connectors for one pin. Terminal is often used when we talk about routing. The term port is used when we are using text (EDIF netlists or HDLs, for example) to describe circuits.

In a gate array the channel capacity must be a multiple of the number of horizontal tracks in the gate-array base cell. Figure 17.6 (e) shows a gate-array base cell with seven horizontal tracks (see Section 17.2 for the factors that determine the track width and track spacing). Thus, in this gate array, we can have a channel with a capacity of 7, 14, 21, ... horizontal tracks-but not between these values.

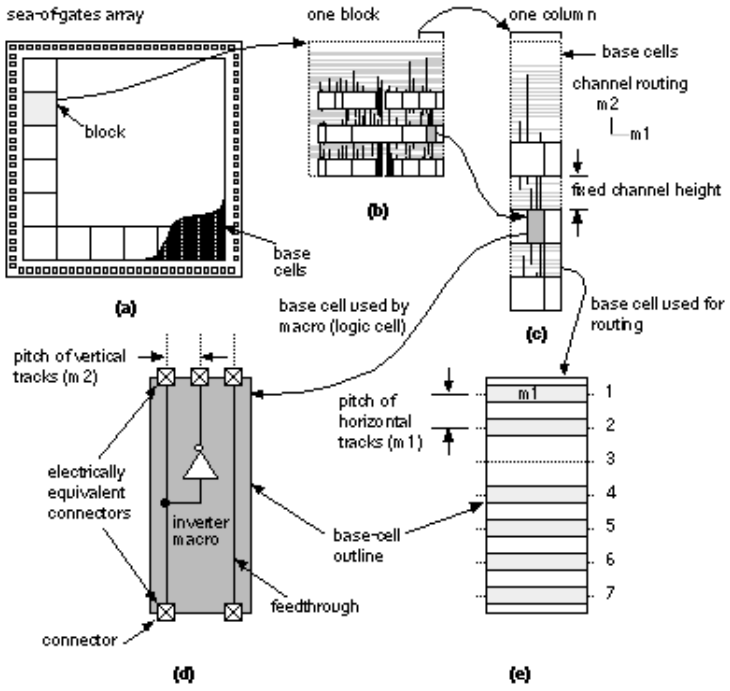


FIGURE 17.6 Gate-array global routing. (a) A small gate array. (b) An enlarged view of the routing. The top channel uses three rows of gate-array base cells; the other channels use only one. (c) A further enlarged view showing how the routing in the channels connects to the logic cells. (d) One of the logic cells, an inverter. (e) There are seven horizontal wiring tracks available in one row of gate-array base cells-the channel capacity is thus 7.

Figure 17.7 shows the inverter macro for the sea-of-gates array shown in Figure 17.6 . Figure 17.7 (a) shows the base cell. Figure 17.7 (b) shows how the internal inverter wiring on m1 leaves one vertical track free as a feedthrough in a two-level metal process (connectors placed at the top and bottom of the cell). In a three-level metal process the connectors may be placed inside the cell abutment box (Figure 17.7 c). Figure 17.8 shows the global routing for the sea-of-gates array. We divide the array into nonoverlapping routing bins (or just bins , also called global routing cells or GRCs), each containing a number of gate-array base cells.

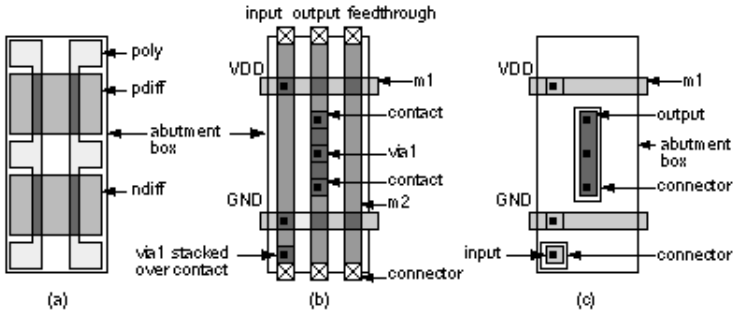


FIGURE 17.7 The gate-array inverter from Figure 17.6 d. (a) An oxide-isolated gate-array base cell, showing the diffusion and polysilicon layers. (b) The metal and contact layers for the inverter in a 2LM (two-level metal) process. (c) The router's view of the cell in a 3LM process.

We need an aside to discuss our use of the term cell . Be careful not to confuse the global routing cells with gate-array base cells (the smallest element of a gate array, consisting of a small number of n -type and p -type transistors), or with logic cells (which are NAND gates, NOR gates, and so on).

A large routing bin reduces the size of the routing problem, and a small routing bin allows the router to calculate the wiring capacities more accurately. Some tools permit routing bins of different size in different areas of the chip (with smaller routing bins helping in areas of dense routing). Figure 17.8 (a) shows a routing bin that is 2 -by-4 gate-array base cells. The logic cells occupy the lower half of the routing bin. The upper half of the routing bin is the channel area, reserved for wiring. The global router calculates the edge capacities for this routing bin, including the vertical feedthroughs. The global router then determines the shortest path for each net considering these edge capacities. An example of a global-routing calculation is shown in Figure 17.8 (b). The path, described by a series of adjacent routing bins, is passed to the detailed router.

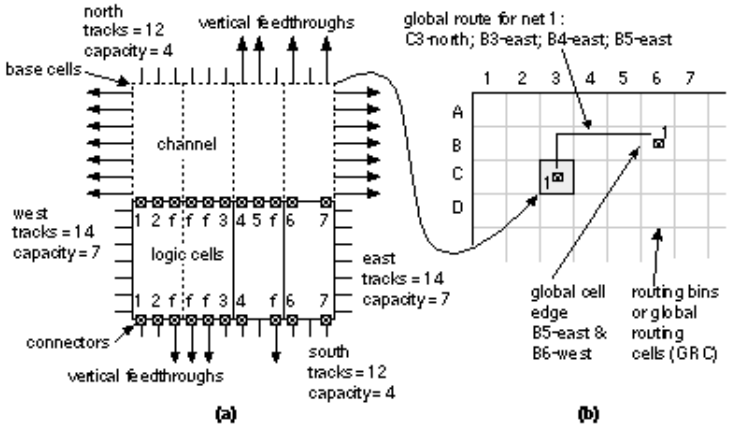


FIGURE 17.8 Global routing a gate array. (a) A single global-routing cell (GRC or routing bin) containing 2-by-4 gate-array base cells. For this choice of routing bin the maximum horizontal track capacity is 14, the maximum vertical track capacity is 12. The routing bin labeled C3 contains three logic cells, two of which have feedthroughs marked 'f'. This results in the edge capacities shown. (b) A view of the top left-hand corner of the gate array showing 28 routing bins. The global router uses the edge capacities to find a sequence of routing bins to connect the nets.

17.1.6 Timing-Driven Methods

Minimizing the total pathlength using a Steiner tree does not necessarily minimize the interconnect delay of a path. Alternative tree algorithms apply in this situation, most using the Elmore constant as a method to estimate the delay of a path (Section 17.1.2). As in timing-driven placement, there are two main approaches to timing-driven routing: net-based and path-based. Path-based methods are more sophisticated. For example, if there is a critical path from logic cell A to B to C, the global router may increase the delay due to the interconnect between logic cells A and B if it can reduce the delay between logic cells B and C. Placement and global routing tools may or may not use the same algorithm to estimate net delay. If these tools are from different companies, the algorithms are probably different. The algorithms must be compatible, however. There is no use performing placement to minimize predicted delay if the global router uses completely different measurement methods. Companies that

produce floorplanning and placement tools make sure that the output is compatible with different routing tools-often to the extent of using different algorithms to target different routers.

17.1.7 Back-annotation

After global routing is complete it is possible to accurately predict what the length of each interconnect in every net will be after detailed routing, probably to within 5 percent. The global router can give us not just an estimate of the total net length (which was all we knew at the placement stage), but the resistance and capacitance of each path in each net. This RC information is used to calculate net delays. We can back-annotate this net delay information to the synthesis tool for in-place optimization or to a timing verifier to make sure there are no timing surprises. Differences in timing predictions at this point arise due to the different ways in which the placement algorithms estimate the paths and the way the global router actually builds the paths.

17.2 Detailed Routing

The global routing step determines the channels to be used for each interconnect. Using this information the detailed router decides the exact location and layers for each interconnect. Figure 17.9 (a) shows typical metal rules. These rules determine the m1 routing pitch (track pitch , track spacing , or just pitch). We can set the m1 pitch to one of three values:

1. via-to-via (VTV) pitch (or spacing),
2. via-to-line (VTL or line-to-via) pitch, or
3. line-to-line (LTL) pitch.

The same choices apply to the m2 and other metal layers if they are present. Via-to-via spacing allows the router to place vias adjacent to each other. Via-to-line spacing is hard to use in practice because it restricts the router to nonadjacent vias. Using line-to-line spacing prevents the router from placing a via at all without using jogs and is rarely used. Via-to-via spacing is the easiest for a router to use and the most common. Using either via-to-line or via-to-via spacing means that the routing pitch is larger than the minimum metal pitch.

Sometimes people draw a distinction between a cut and a via when they talk about large connections such as shown in Figure 17.10 (a). We split or stitch a large via into identically sized cuts (sometimes called a waffle via). Because of the profile of the metal in a contact and the way current flows into a contact, often the total resistance of several small cuts is less than that of one large cut. Using identically sized cuts also means the processing conditions during contact etching, which may vary with the area and perimeter of a contact, are the same for every cut on the chip.

In a stacked via the contact cuts all overlap in a layout plot and it is impossible to tell just how many vias on which layers are present. Figure 17.10 (b-f) show an alternative way to draw contacts and vias. Though this is not a standard, using the diagonal box convention makes it possible to recognize stacked vias and contacts on a layout (in any orientation). I shall use these conventions when it is necessary.

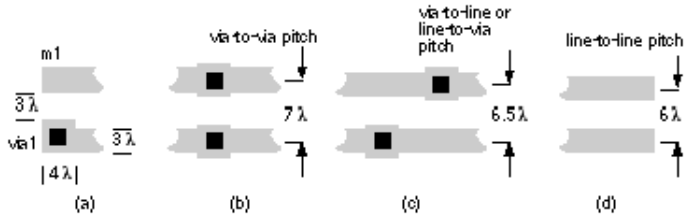


FIGURE 17.9 The metal routing pitch. (a) An example of l -based metal design rules for m1 and via1 (m1/m2 via). (b) Via-to-via pitch for adjacent vias. (c) Via-to-line (or line-to-via) pitch for nonadjacent vias. (d) Line-to-line pitch with no vias.

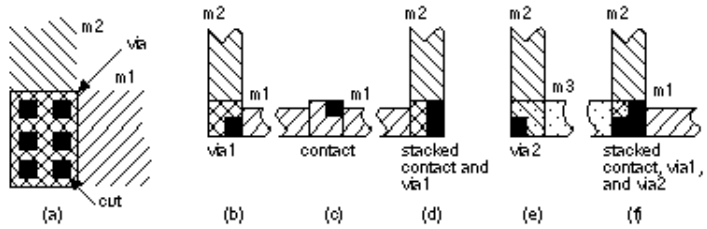


FIGURE 17.10 (a) A large m1 to m2 via. The black squares represent the holes (or cuts) that are etched in the insulating material between the m1 and 2 layers. (b) A m1 to m2 via (a via1). (c) A contact from m1 to diffusion or polysilicon (a contact). (d) A via1 placed over (or stacked over) a contact. (e) A m2 to m3 via (a via2) (f) A via2 stacked over a via1 stacked over a contact. Notice that the black square in parts b-c do not represent the actual location of the cuts. The black squares are offset so you can recognize stacked vias and contacts.

In a two-level metal CMOS ASIC technology we complete the wiring using the two different metal layers for the horizontal and vertical directions, one layer for each direction. This is Manhattan routing , because the results look similar to the rectangular north-south and east-west layout of streets in New York City. Thus, for example, if terminals are on the m2 layer, then we route the horizontal branches in a channel using m2 and the vertical trunks using m1. Figure 17.11 shows that, although we may choose a preferred direction for each metal layer (for example, m1 for horizontal routing and m2 for vertical routing), this may lead to problems in cases that have both horizontal and vertical channels. In these cases we define a preferred metal layer in the direction of the channel spine. In Figure 17.11 , because the logic cell connectors are on m2, any vertical channel has to use vias at every logic cell location. By changing the orientation of the metal directions in vertical channels, we can avoid this, and instead we only need to place vias at the intersection of horizontal and vertical channels.

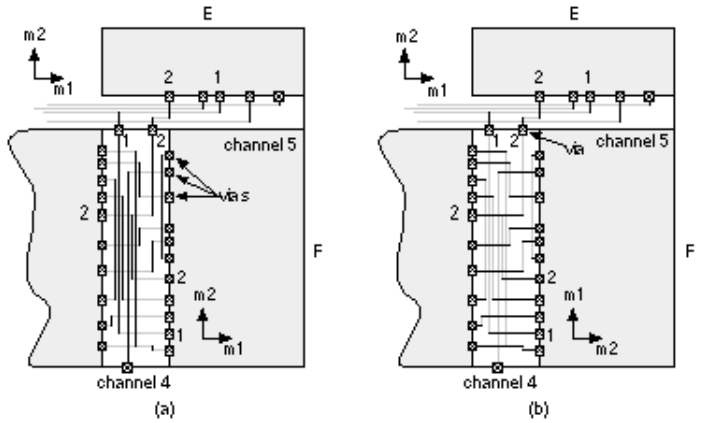


FIGURE 17.11 An expanded view of part of a cell-based ASIC. (a) Both channel 4 and channel 5 use m1 in the horizontal direction and m2 in the vertical direction. If the logic cell connectors are on m2 this requires vias to be placed at every logic cell connector in channel 4. (b) Channel 4 and 5 are routed with m1 along the direction of the channel spine (the long direction of the channel). Now vias are required only for nets 1 and 2, at the intersection of the channels.

Figure 17.12 shows an imaginary logic cell with connectors. Double-entry logic cells intended for two-level metal routing have connectors at the top and bottom of the logic cell, usually in m2. Logic cells intended for processes with three or more levels of metal have connectors in the center of the cell, again usually on m2. Logic cells may use both m1 and m2 internally, but the use of m2 is usually minimized. The router normally uses a simplified view of the logic cell called a phantom . The phantom contains only the logic cell information that the router needs: the connector locations, types, and names; the abutment and bounding boxes; enough layer information to be able to place cells without violating design rules; and a blockage map -the locations of any metal inside the cell that blocks routing.

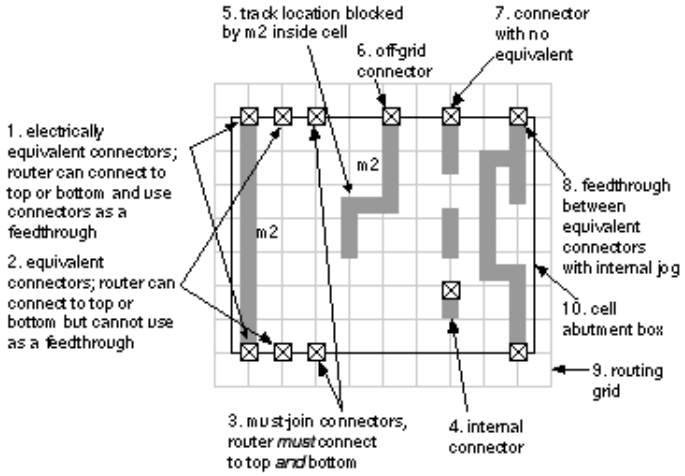


FIGURE 17.12 The different types of connections that can be made to a cell. This cell has connectors at the top and bottom of the cell (normal for cells intended for use with a two-level metal process) and internal connectors (normal for logic cells intended for use with a three-level metal process). The interconnect and connections are drawn to scale.

Figure 17.13 illustrates some terms used in the detailed routing of a channel. The channel spine in Figure 17.13 is horizontal with terminals at the top and the bottom, but a channel can also be vertical. In either case terminals are spaced along the longest edges of the channel at given, fixed locations. Terminals are usually located on a grid defined by the routing pitch on that layer (we say terminals are either on-grid or off-grid). We make connections between terminals using interconnects that consist of one or more trunks running parallel to the length of the channel and branches that connect the trunk to the terminals. If more than one trunk is used, the trunks are connected by doglegs . Connections exit the channel at pseudoterminals .

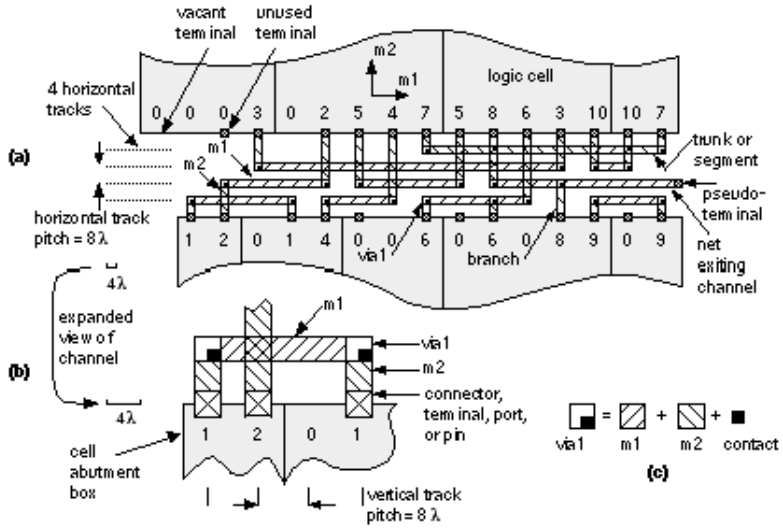


FIGURE 17.13 Terms used in channel routing. (a) A channel with four horizontal tracks. (b) An expanded view of the left-hand portion of the channel showing (approximately to scale) how the m1 and m2 layers connect to the logic cells on either side of the channel. (c) The construction of a via1 (m1/m2 via).

The trunk and branch connections run in tracks (equispaced, like railway tracks). If the trunk connections use m1, the horizontal track spacing (usually just called the track spacing for channel routing) is equal to the m1 routing pitch. The maximum number of interconnects we need in a channel multiplied by the horizontal track spacing gives the minimum height of a channel (see Section 17.2.2 on how to determine the maximum number of interconnects needed). Each terminal occupies a column . If the branches use m2, the column spacing (or vertical track spacing) is equal to the m2 routing pitch.

17.2.1 Goals and Objectives

The goal of detailed routing is to complete all the connections between logic cells. The most common objective is to minimize one or more of the following:

- The total interconnect length and area
- The number of layer changes that the connections have to make
- The delay of critical paths

Minimizing the number of layer changes corresponds to minimizing the number of vias that add

parasitic resistance and capacitance to a connection.

In some cases the detailed router may not be able to complete the routing in the area provided. In the case of a cell-based ASIC or sea-of-gates array, it is possible to increase the channel size and try the routing steps again. A channeled gate array or FPGA has fixed routing resources and in these cases we must start all over again with floorplanning and placement, or use a larger chip.

17.2.2 Measurement of Channel Density

We can describe a channel-routing problem by specifying two lists of nets: one for the top edge of the channel and one for the bottom edge. The position of the net number in the list gives the column position. The net number zero represents a vacant or unused terminal. Figure 17.14 shows a channel with the numbered terminals to be connected along the top and the bottom of the channel.

We call the number of nets that cross a line drawn vertically anywhere in a channel the local density . We call the maximum local density of the channel the global density or sometimes just channel density . Figure 17.14 has a channel density of 4. Channel density is an important measure in routing-it tells a router the absolute fewest number of horizontal interconnects that it needs at the point where the local density is highest. In two-level routing (all the horizontal interconnects run on one routing layer) the channel density determines the minimum height of the channel. The channel capacity is the maximum number of interconnects that a channel can hold. If the channel density is greater than the channel capacity, that channel definitely cannot be routed (to learn how channel density is calculated, see Section 17.2.5).

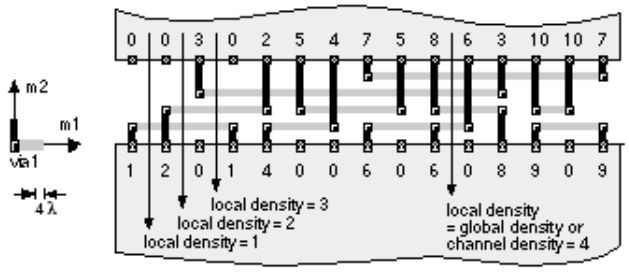


FIGURE 17.14 The definitions of local channel density and global channel density. Lines represent the m1 and m2 interconnect in the channel to simplify the drawing.

17.2.3 Algorithms

We start discussion of routing methods by simplifying the general channel-routing problem. The restricted channel-routing problem limits each net in a channel to use only one horizontal segment. In other words the channel router uses only one trunk for each net. This restriction has the effect of minimizing the number of connections between the routing layers. This is equivalent to minimizing the number of vias used by the channel router in a two-layer metal technology. Minimizing the number of vias is an important objective in routing a channel, but it is not always practical. Sometimes constraints will force a channel router to use jogs or other methods to complete the routing (see Section 17.2.5).

Next, though, we shall study an algorithm that solves the restricted channel-routing problem.

17.2.4 Left-Edge Algorithm

The left-edge algorithm (LEA) is the basis for several routing algorithms [Hashimoto and Stevens, 1971]. The LEA applies to two-layer channel routing, using one layer for the trunks and the other layer for the branches. For example, m1 may be used in the horizontal direction and m2 in the vertical direction. The LEA proceeds as follows:

1. Sort the nets according to the leftmost edges of the net's horizontal segment.
2. Assign the first net on the list to the first free track.
3. Assign the next net on the list, which will fit, to the track.
4. Repeat this process from step 3 until no more nets will fit in the current track.
5. Repeat steps 2-4 until all nets have been assigned to tracks.
6. Connect the net segments to the top and bottom of the channel.

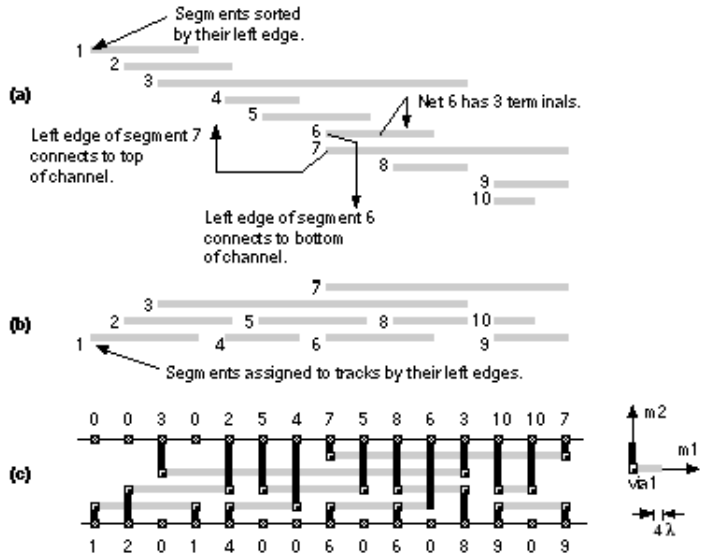


FIGURE 17.15 Left-edge algorithm. (a) Sorted list of segments. (b) Assignment to tracks. (c) Completed channel route (with m1 and m2 interconnect represented by lines).

Figure 17.15 illustrates the LEA. The algorithm works as long as none of the branches touch-which may occur if there are terminals in the same column belonging to different nets. In this situation we have to make sure that the trunk that connects to the top of the channel is placed above the lower trunk. Otherwise two branches will overlap and short the nets together. In the next section we shall examine this situation more closely.

17.2.5 Constraints and Routing Graphs

Two terminals that are in the same column in a channel create a vertical constraint . We say that the terminal at the top of the column imposes a vertical constraint on the lower terminal. We can draw a

graph showing the vertical constraints imposed by terminals. The nodes in a vertical-constraint graph represent terminals. A vertical constraint between two terminals is shown by an edge of the graph connecting the two terminals. A graph that contains information in the direction of an edge is a directed graph . The arrow on the graph edge shows the direction of the constraint-pointing to the lower terminal, which is constrained. Figure 17.16 (a) shows an example of a channel, and Figure 17.16 (b) shows its vertical constraint graph.

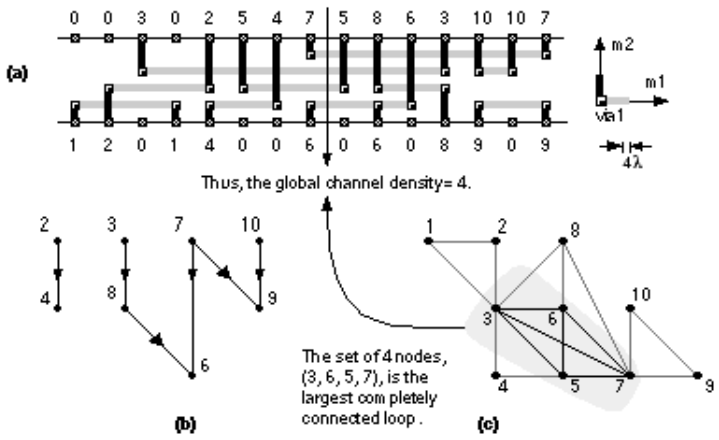


FIGURE 17.16 Routing graphs. (a) Channel with a global density of 4. (b) The vertical constraint graph. If two nets occupy the same column, the net at the top of the channel imposes a vertical constraint on the net at the bottom. For example, net 2 imposes a vertical constraint on net 4. Thus the interconnect for net 4 must use a track above net 2. (c) Horizontal-constraint graph. If the segments of two nets overlap, they are connected in the horizontal-constraint graph. This graph determines the global channel density.

We can also define a horizontal constraint and a corresponding horizontal-constraint graph . If the trunk for net 1 overlaps the trunk of net 2, then we say there is a horizontal constraint between net 1 and net 2. Unlike a vertical constraint, a horizontal constraint has no direction. Figure 17.16 (c) shows an example of a horizontal constraint graph and shows a group of 4 terminals (numbered 3, 5, 6, and 7) that must all overlap. Since this is the largest such group, the global channel density is 4.

If there are no vertical constraints at all in a channel, we can guarantee that the LEA will find the minimum number of routing tracks. The addition of vertical constraints transforms the restricted routing problem into an NP-complete problem. There is also an arrangement of vertical constraints that none of the algorithms based on the LEA can cope with. In Figure 17.17 (a) net 1 is above net 2 in the first column of the channel. Thus net 1 imposes a vertical constraint on net 2. Net 2 is above net 1 in the last column of the channel. Then net 2 also imposes a vertical constraint on net 1. It is impossible to route this arrangement using two routing layers with the restriction of using only one trunk for each net. If we construct the vertical-constraint graph for this situation, shown in Figure 17.17 (b), there is a loop or cycle between nets 1 and 2. If there is any such vertical-constraint cycle (or cyclic constraint) between two or more nets, the LEA will fail. A dogleg router removes the restriction that each net can use only one track or trunk. Figure 17.17 (c) shows how adding a dogleg permits a channel with a cyclic constraint to be routed.

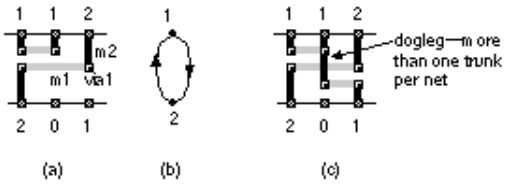


FIGURE 17.17 The addition of a dogleg, an extra trunk, in the wiring of a net can resolve cyclic vertical constraints.

The channel-routing algorithms we have described so far do not allow interconnects on one layer to run on top of other interconnects on a different layer. These algorithms allow interconnects to cross at right angles to each other on different layers, but not to overlap . When we remove the restriction that horizontal and vertical routing must use different layers, the density of a channel is no longer the lower bound for the number of tracks required. For two routing layers the ultimate lower bound becomes half of the channel density. The practical reasoning for restricting overlap is the parasitic overlap capacitance between signal interconnects. As the dimensions of the metal interconnect are reduced, the capacitance between adjacent interconnects on the same layer (coupling capacitance) is comparable to the capacitance of interconnects that overlap on different layers (overlap capacitance). Thus, allowing a short overlap between interconnects on different layers may not be as bad as allowing two interconnects to run adjacent to each other for a long distance on the same layer. Some routers allow you to specify that two interconnects must not run adjacent to each other for more than a specified length.

The channel height is fixed for channeled gate arrays; it is variable in discrete steps for channelless gate arrays; it is continuously variable for cell-based ASICs. However, for all these types of ASICs, the channel wiring is fully customized and so may be compacted or compressed after a channel router has completed the interconnect. The use of channel-routing compaction for a two-layer channel can reduce the channel height by 15 percent to 20 percent [Cheng et al., 1992].

Modern channel routers are capable of routing a channel at or near the theoretical minimum density. We can thus consider channel routing a solved problem. Most of the difficulty in detailed routing now comes from the need to route more than two layers and to route arbitrary shaped regions. These problems are best handled by area routers.

17.2.6 Area-Routing Algorithms

There are many algorithms used for the detailed routing of general-shaped areas (see the paper by Ohtsuki in [Ohtsuki, 1986]). Many of these were originally developed for PCB wiring. The first group we shall cover and the earliest to be used historically are the grid-expansion or maze-running algorithms. A second group of methods, which are more efficient, are the line-search algorithms.

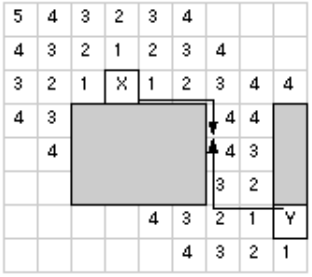


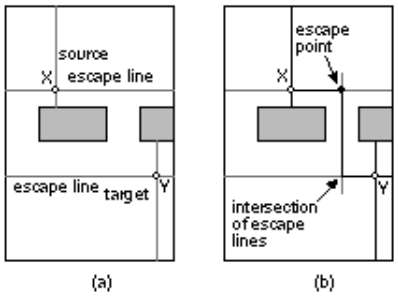
FIGURE 17.18 The Lee maze-running algorithm. The algorithm finds a path from source (X) to target (Y) by emitting a wave from both the source and the target at the same time. Successive outward moves are marked in each bin. Once the target is reached, the path is found by backtracking (if there is a choice of bins with equal labeled values, we choose the bin that avoids changing direction). (The original form of the Lee algorithm uses a single wave.)

Figure 17.18 illustrates the Lee maze-running algorithm . The goal is to find a path from X to Y-i.e., from the start (or source) to the finish (or target)-avoiding any obstacles. The algorithm is often called wave propagation because it sends out waves, which spread out like those created by dropping a stone into a pond.

Algorithms that use lines rather than waves to search for connections are more efficient than algorithms based on the Lee algorithm. Figure 17.19 illustrates the Hightower algorithm -a line-search algorithm (or line-probe algorithm):

1. Extend lines from both the source and target toward each other.
2. When an extended line, known as an escape line , meets an obstacle, choose a point on the escape line from which to project another escape line at right angles to the old one. This point is the escape point .
3. Place an escape point on the line so that the next escape line just misses the edge of the obstacle. Escape lines emanating from the source and target intersect to form the path.

FIGURE 17.19 Hightower area-routing algorithm. (a) Escape lines are constructed from source (X) and target (Y) toward each other until they hit obstacles. (b) An escape point is found on the escape line so that the next escape line perpendicular to the original misses the next obstacle. The path is complete when escape lines from source and target meet.



The Hightower algorithm is faster and requires less memory than methods based on the Lee algorithm.

17.2.7 Multilevel Routing

Using two-layer routing , if the logic cells do not contain any m2, it is possible to complete some routing in m2 using over-the-cell (OTC) routing. Sometimes poly is used for short connections in the channel in a two-level metal technology; this is known as 2.5-layer routing . Using a third level of metal in three-layer routing , there is a choice of approaches. Reserved-layer routing restricts all the interconnect on each layer to flow in one direction in a given routing area (for example, in a channel, either parallel or perpendicular to the channel spine). Unreserved-layer routing moves in both horizontal and vertical directions on a given layer. Most routers use reserved routing. Reserved three-level metal routing offers another choice: Either use m1 and m3 for horizontal routing (parallel to the channel spine), with m2 for vertical routing (HVH routing) or use VHV routing . Since the logic cell interconnect usually blocks most of the area on the m1 layer, HVH routing is normally used. It is also important to consider the pitch of the layers when routing in the same direction on two different layers. Using HVH routing it is preferable for the m3 pitch to be a simple multiple of the m1 pitch (ideally they are the same). Some processes have more than three levels of metal. Sometimes the upper one or two metal layers have a coarser pitch than the lower layers and are used in multilevel routing for power and clock lines rather than for signal interconnect.

Figure 17.20 shows an example of three-layer channel routing. The logic cells are 64 l high, the m1 routing pitch is 8 l, and the m2 and m3 routing pitch is 16 l . The channel in Figure 17.20 is the same as

the channel using two-layer metal shown in Figure 17.13 , but using three-level metal reduces the channel height from 40 l (= 5 ¥ 8 l) to 16 l . Submicron processes try to use the same metal pitch on all metal layers. This makes routing easier but processing more difficult.

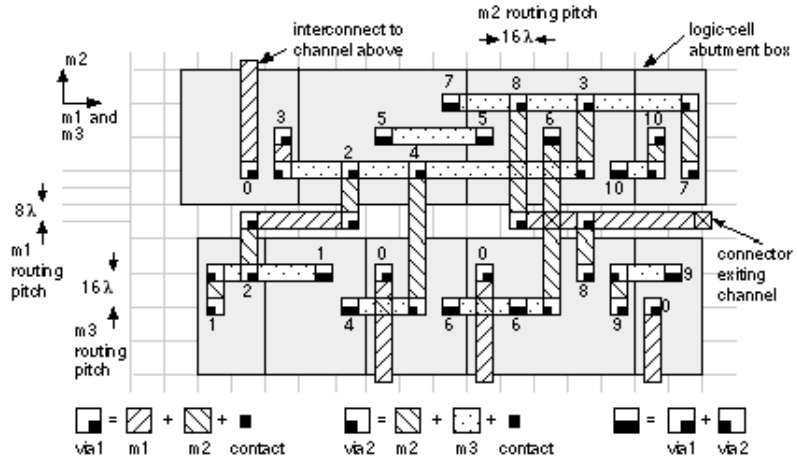


FIGURE 17.20 Three-level channel routing. In this diagram the m2 and m3 routing pitch is set to twice the m1 routing pitch. Routing density can be increased further if all the routing pitches can be made equal-a difficult process challenge.

With three or more levels of metal routing it is possible to reduce the channel height in a row-based ASIC to zero. All of the interconnect is then completed over the cell. If all of the channels are eliminated, the core area (logic cells plus routing) is determined solely by the logic-cell area. The point at which this happens depends on not only the number of metal layers and channel density, but also the routing resources (the blockages and feedthroughs) in the logic cell. This the cell porosity . Designing porous cells that help to minimize routing area is an art. For example, it is quite common to be able to produce a smaller chip using larger logic cells if the larger cells have more routing resources.

17.2.8 Timing-Driven Detailed Routing

In detailed routing the global router has already set the path the interconnect will follow. At this point little can be done to improve timing except to reduce the number of vias, alter the interconnect width to optimize delay, and minimize overlap capacitance. The gains here are relatively small, but for very long branching nets even small gains may be important. For high-frequency clock nets it may be important to shape and chamfer (round) the interconnect to match impedances at branches and control reflections at corners.

17.2.9 Final Routing Steps

If the algorithms to estimate congestion in the floorplanning tool accurately perfectly reflected the algorithms used by the global router and detailed router, routing completion should be guaranteed. Often, however, the detailed router will not be able to completely route all the nets. These problematical nets are known as unroutes . Routers handle this situation in one of two ways. The first method leaves the problematical nets unconnected. The second method completes all interconnects anyway but with

some design-rule violations (the problematical nets may be shorted to other nets, for example). Some tools flag these problems as a warning (in fact there can be no more serious error).

If there are many unroutes the designer needs to discover the reason and return to the floorplanner and change channel sizes (for a cell-based ASIC) or increase the base-array size (for a gate array). Returning to the global router and changing bin sizes or adjusting the algorithms may also help. In drastic cases it may be necessary to change the floorplan. If just a handful of difficult nets remain to be routed, some tools allow the designer to perform hand edits using a rip-up and reroute router (sometimes this is done automatically by the detailed router as a last phase in the routing procedure anyway). This capability also permits engineering change orders (ECO)-corresponding to the little yellow wires on a PCB. One of the last steps in routing is via removal -the detailed router looks to see if it can eliminate any vias (which can contribute a significant amount to the interconnect resistance) by changing layers or making other modifications to the completed routing. Routing compaction can then be performed as the final step.

17.3 Special Routing

The routing of nets that require special attention, clock and power nets for example, is normally done before detailed routing of signal nets. The architecture and structure of these nets is performed as part of floorplanning, but the sizing and topology of these nets is finalized as part of the routing step.

17.3.1 Clock Routing

Gate arrays normally use a clock spine (a regular grid), eliminating the need for special routing (see Section 16.1.6, "Clock Planning"). The clock distribution grid is designed at the same time as the gate-array base to ensure a minimum clock skew and minimum clock latency-given power dissipation and clock buffer area limitations. Cell-based ASICs may use either a clock spine, a clock tree, or a hybrid approach. Figure 17.21 shows how a clock router may minimize clock skew in a clock spine by making the path lengths, and thus net delays, to every leaf node equal-using jogs in the interconnect paths if necessary. More sophisticated clock routers perform clock-tree synthesis (automatically choosing the depth and structure of the clock tree) and clock-buffer insertion (equalizing the delay to the leaf nodes by balancing interconnect delays and buffer delays).

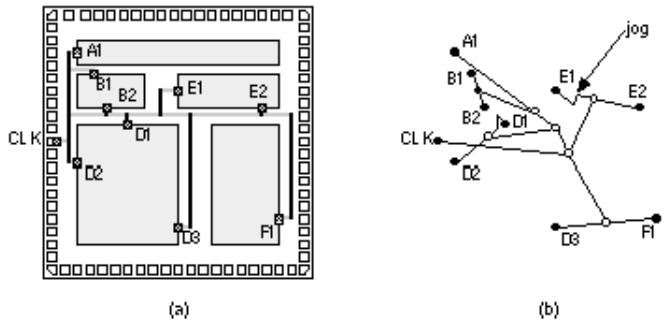


FIGURE 17.21 Clock routing. (a) A clock network for the cell-based ASIC from Figure 16.11. (b) Equalizing the interconnect segments between CLK and all destinations (by including jogs if necessary) minimizes clock skew.

The clock tree may contain multiply-driven nodes (more than one active element driving a net). The net delay models that we have used break down in this case and we may have to extract the clock network and perform circuit simulation, followed by back-annotation of the clock delays to the netlist (for circuit extraction, see Section 17.4) and the bus currents to the clock router. The sizes of the clock buses depend on the current they must carry. The limits are set by reliability issues to be discussed next.

Clock skew induced by hot-electron wearout was mentioned in Section 16.1.6, "Clock Planning." Another factor contributing to unpredictable clock skew is changes in clock-buffer delays with variations in power-supply voltage due to data-dependent activity. This activity-induced clock skew can easily be larger than the skew achievable using a clock router. For example, there is little point in using software capable of reducing clock skew to less than 100 ps if, due to fluctuations in power-supply voltage when part of the chip becomes active, the clock-network delays change by 200 ps.

The power buses supplying the buffers driving the clock spine carry direct current (unidirectional current or DC), but the clock spine itself carries alternating current (bidirectional current or AC). The difference between electromigration failure rates due to AC and DC leads to different rules for sizing clock buses. As we explained in Section 16.1.6, "Clock Planning," the fastest way to drive a large load in CMOS is to taper successive stages by approximately e ª 3. This is not necessarily the smallest-area or lowest-power approach, however [Veendrick, 1984].

17.3.2 Power Routing

Each of the power buses has to be sized according to the current it will carry. Too much current in a power bus can lead to a failure through a mechanism known as electromigration [Young and Christou, 1994]. The required power-bus widths can be estimated automatically from library information, from a separate power simulation tool, or by entering the power-bus widths to the routing software by hand. Many routers use a default power-bus width so that it is quite easy to complete routing of an ASIC without even knowing about this problem.

For a direct current (DC) the mean time to failure (MTTF) due to electromigration is experimentally found to obey the following equation:

$$MTTF = A J^{-2} \exp - E / k T , \quad (17.9)$$

where J is the current density; E is approximately 0.5 eV; k , Boltzmann’s constant, is 8.62 ¥ 10^{-5} eVK^{-1} ; and T is absolute temperature in kelvins.

There are a number of different approaches to model the effect of an AC component. A typical expression is

$$MTTF = \frac{A J^{-2} \exp - E / k T}{J | J | + k_{AC/DC} | J |^{2}} , \quad (17.10)$$

where J is the average of J(t) , and | J | is the average of | J |. The constant $k_{AC/DC}$ relates the relative effects of AC and DC and is typically between 0.01 and 0.0001. Electromigration problems become serious with a MTTF of less than 10^5 hours (approximately 10 years) for current densities (DC) greater than 0.5 GAm $^{-2}$ at temperatures above 150 °C.

Table 17.1 lists example metallization reliability rules -limits for the current you can pass through a metal layer, contact, or via-for the typical 0.5 m m three-level metal CMOS process, G5. The limit of 1 mA of current per square micron of metal cross section is a good rule-of-thumb to follow for current density in aluminum-based interconnect.

Some CMOS processes also have maximum metal-width rules (or fat-metal rules). This is because stress (especially at the corners of the die, which occurs during die attach -mounting the die on the chip carrier) can cause large metal areas to lift. A solution to this problem is to place slots in the wide metal lines. These rules are dependent on the ASIC vendor's level of experience.

To determine the power-bus widths we need to determine the bus currents. The largest problem is emulating the system's operating conditions. Input vectors to test the system are not necessarily representative of actual system operation. Clock-bus sizing depends strongly on the parameter $k_{AC/DC}$ in Eq. 17.10 , since the clock spine carries alternating current. (For the sources of power dissipation in CMOS, see Section 15.5, "Power Dissipation.")

Gate arrays normally use a regular power grid as part of the gate-array base. The gate-array logic cells contain two fixed-width power buses inside the cell, running horizontally on m1. The horizontal m1 power buses are then strapped in a vertical direction by m2 buses, which run vertically across the chip. The resistance of the power grid is extracted and simulated with SPICE during the base-array design to model the effects of IR drops under worst-case conditions.

TABLE 17.1 Metallization reliability rules for a typical 0.5 micron (l = 0.25 m m) CMOS process.

Layer/contact/via	Current limit 1	Metal thickness 2	Resistance 3
m1	1 mA m m $^{-1}$	7000 Å	95 m W /square
m2	1 mA m m $^{-1}$	7000 Å	95 m W /square
m3	2 mA m m $^{-1}$	12,000 Å	48 m W /square
0.8 m m square m1 contact to diffusion	0.7 mA		11 W
0.8 m m square m1 contact to poly	0.7 mA		16 W
0.8 m m square m1/m2 via (via1)	0.7 mA		3.6 W
0.8 m m square m2/m3 via (via2)	0.7 mA		3.6 W

Standard cells are constructed in a similar fashion to gate-array cells, with power buses running horizontally in m1 at the top and bottom of each cell. A row of standard cells uses end-cap cells that connect to the VDD and VSS power buses placed by the power router. Power routing of cell-based ASICs may include the option to include vertical m2 straps at a specified intervals. Alternatively the number of standard cells that can be placed in a row may be limited during placement. The power router forms an interdigitated comb structure, minimizing the number of times a VDD or VSS power bus needs to change layers. This is achieved by routing with a routing bias on preferred layers. For example, VDD

may be routed with a left-and-down bias on m1, with VSS routed using right-and-up bias on m2.

Three-level metal processes either use a m3 with a thickness and pitch that is comparable to m1 and m2 (which usually have approximately the same thickness and pitch) or they use metal that is much thicker (up to twice as thick as m1 and m2) with a coarser pitch (up to twice as wide as m1 and m2). The factor that determines the m3/4/5 properties is normally the sophistication of the fabrication process.

In a three-level metal process, power routing is similar to two-level metal ASICs. Power buses inside the logic cells are still normally run on m1. Using HVH routing it would be possible to run the power buses on m3 and drop vias all the way down to m1 when power is required in the cells. The problem with this approach is that it creates pillars of blockage across all three layers.

Using three or more layers of metal for routing, it is possible to eliminate some of the channels completely. In these cases we complete all the routing in m2 and m3 on top of the logic cells using connectors placed in the center of the cells on m1. If we can eliminate the channels between cell rows, we can flip rows about a horizontal axis and abut adjacent rows together (a technique known as flip and abut). If the power buses are at the top (VDD) and bottom (VSS) of the cells in m1 we can abut or overlap the power buses (joining VDD to VDD and VSS to VSS in alternate rows).

Power distribution schemes are also a function of process and packaging technology. Recall that flip-chip technology allows pads to be placed anywhere on a chip (see Section 16.1.5, "I/O and Power Planning," especially Figure 16.13d). Four-level metal and aggressive stacked-via rules allow I/O pad circuits to be placed in the core. The problems with this approach include placing the ESD and latch-up protection circuits required in the I/O pads (normally kept widely separated from core logic) adjacent to the logic cells in the core.

---

1. At 125 °C for unidirectional current. Limits for 110 °C are ¥ 1.5 higher. Limits for 85 °C are ¥ 3 higher. Current limits for bidirectional current are ¥ 1.5 higher than the unidirectional limits.

2. 10,000 Å (ten thousand angstroms) = 1 m m.

3. Worst case at 110 °C.

17.4 Circuit Extraction and DRC

After detailed routing is complete, the exact length and position of each interconnect for every net is known. Now the parasitic capacitance and resistance associated with each interconnect, via, and contact can be calculated. This data is generated by a circuit-extraction tool in one of the formats described next. It is important to extract the parasitic values that will be on the silicon wafer. The mask data or CIF widths and dimensions that are drawn in the logic cells are not necessarily the same as the final silicon dimensions. Normally mask dimensions are altered from drawn values to allow for process bias or other effects that occur during the transfer of the pattern from mask to silicon. Since this is a problem that is dealt with by the ASIC vendor and not the design software vendor, ASIC designers normally have to ask very carefully about the details of this problem.

Table 17.2 shows values for the parasitic capacitances for a typical 1 m m CMOS process. Notice that the fringing capacitance is greater than the parallel-plate (area) capacitance for all layers except poly. Next, we shall describe how the parasitic information is passed between tools.

17.4.1 SPF, RSPF, and DSPF

The standard parasitic format (SPF) (developed by Cadence [1990], now in the hands of OVI) describes interconnect delay and loading due to parasitic resistance and capacitance. There are three different forms of SPF: two of them (regular SPF and reduced SPF) contain the same information, but in different formats, and model the behavior of interconnect; the third form of SPF (detailed SPF) describes the actual parasitic resistance and capacitance components of a net. Figure 17.22 shows the different types of simplified models that regular and reduced SPF support. The load at the output of gate A is represented by one of three models: lumped-C, lumped-RC, or PI segment. The pin-to-pin delays are modeled by RC delays. You can represent the pin-to-pin interconnect delay by an ideal voltage source, V(A_1) in this case, driving an RC network attached to each input pin. The actual pin-to-pin delays may not be calculated this way, however.

TABLE 17.2 Parasitic capacitances for a typical 1 m m (l = 0.5 m m) three-level metal CMOS process. 1

Element	Area / fF m m $^{-2}$	Fringing / fF m m $^{-1}$
poly (over gate oxide) to substrate	1.73	NA 2
poly (over field oxide) to substrate	0.058	0.043
m1 to diffusion or poly	0.055	0.049
m1 to substrate	0.031	0.044
m2 to diffusion	0.019	0.038
m2 to substrate	0.015	0.035
m2 to poly	0.022	0.040
m2 to m1	0.035	0.046
m3 to diffusion	0.011	0.034
m3 to substrate	0.010	0.033
m3 to poly	0.012	0.034
m3 to m1	0.016	0.039
m3 to m2	0.035	0.049
n+ junction (at 0V bias)	0.36	NA
p+ junction (at 0V bias)	0.46	NA

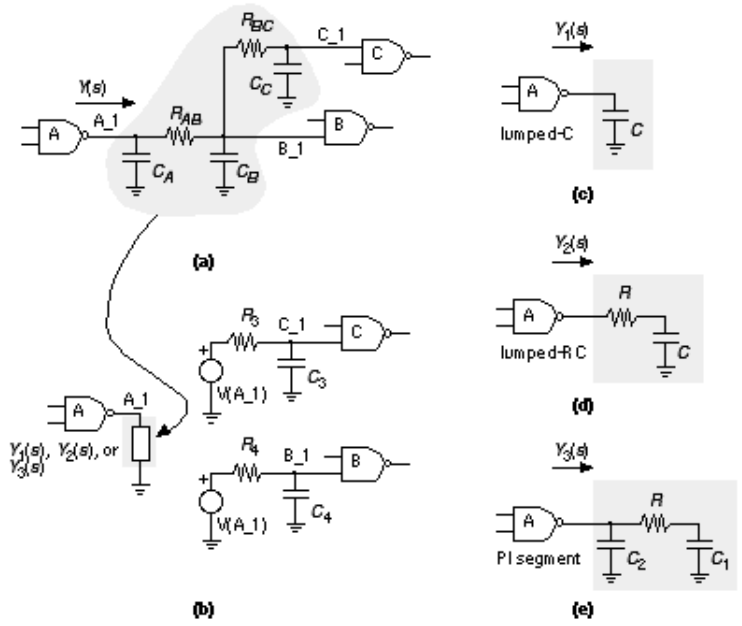


FIGURE 17.22 The regular and reduced standard parasitic format (SPF) models for interconnect. (a) An example of an interconnect network with fanout. The driving-point admittance of the interconnect network is Y (s). (b) The SPF model of the interconnect. (c) The lumped-capacitance interconnect model. (d) The lumped-RC interconnect model. (e) The PI segment interconnect model (notice the capacitor nearest the output node is labeled C_2 rather than C_1). The values of C , R , C_1 , and C_2 are calculated so that Y_1 (s), Y_2 (s), and Y_3 (s) are the first-, second-, and third-order Taylor-series approximations to Y (s).

The key features of regular and reduced SPF are as follows:

- The loading effect of a net as seen by the driving gate is represented by choosing one of three different RC networks: lumped-C, lumped-RC, or PI segment (selected when generating the SPF) [O'Brien and Savarino, 1989].
- The pin-to-pin delays of each path in the net are modeled by a simple RC delay (one for each path). This can be the Elmore constant for each path (see Section 17.1.2), but it need not be.

Here is an example regular SPF file for just one net that uses the PI segment model shown in Figure 17.22 (e):

#Design Name : EXAMPLE1

#Date : 6 August 1995

#Time : 12:00:00

#Resistance Units : 1 ohms

#Capacitance Units : 1 pico farads

#Syntax :

#N <netName>

#C <capVal>

F <from CompName> <fromPinName>

GC <conductance>

|

REQ <res>

GRC <conductance>

T <toCompName> <toPinName> RC <rcConstant> A <value>

|

RPI <res>

C1 <cap>

C2 <cap>

GPI <conductance>

T <toCompName> <toPinName> RC <rcConstant> A <value>

TIMING.ADMITTANCE.MODEL = PI

TIMING.CAPACITANCE.MODEL = PP

N CLOCK

C 3.66

F ROOT Z

RPI 8.85

C1 2.49

C2 1.17

GPI = 0.0

T DF1 G RC 22.20

T DF2 G RC 13.05

This file describes the following:

- The preamble contains the file format.
- This representation uses the PI segment model (Figure 17.22 e).
- This net uses pin-to-pin timing.
- The driving gate of this net is ROOT and the output pin name is Z .
- The PI segment elements have values: C1 = 2.49 pF, C2 = 1.17 pF, RPI = 8.85 W . Notice the order of C1 and C2 in Figure 17.22 (e). The element GPI is not normally used in SPF files.
- The delay from output pin Z of ROOT to input pin G of DF1 is 22.20 ns.
- The delay from pin Z of ROOT to pin G of DF2 is 13.05 ns.

The reduced SPF (RSPF) contains the same information as regular SPF, but uses the SPICE format. Here is an example RSPF file that corresponds to the previous regular SPF example:

* Design Name : EXAMPLE1

* Date : 6 August 1995

* Time : 12:00:00

* Resistance Units : 1 ohms

* Capacitance Units : 1 pico farads

*| RSPF 1.0

*| DELIMITER "_"

.SUBCKT EXAMPLE1 OUT IN

*| GROUND_NET VSS

* TIMING.CAPACITANCE.MODEL = PP

*|NET CLOCK 3.66PF

*|DRIVER ROOT_Z ROOT Z

*|S (ROOT_Z_OUTP1 0.0 0.0)

R2 ROOT_Z ROOT_Z_OUTP1 8.85

C1 ROOT_Z_OUTP1 VSS 2.49PF

```
C2 ROOT_Z VSS 1.17PF

*|LOAD DF2_G DF1 G

*|S (DF1_G_INP1 0.0 0.0)

E1 DF1_G_INP1 VSS ROOT_Z VSS 1.0

R3 DF1_G_INP1 DF1_G 22.20

C3 DF1_G VSS 1.0PF

*|LOAD DF2_G DF2 G

*|S (DF2_G_INP1 0.0 0.0)

E2 DF2_G_INP1 VSS ROOT_Z VSS 1.0

R4 DF2_G_INP1 DF2_G 13.05

C4 DF2_G VSS 1.0PF

*Instance Section

XDF1 DF1_Q DF1_QN DF1_D DF1_G DF1_CD DF1_VDD DF1_VSS DFF3

XDF2 DF2_Q DF2_QN DF2_D DF2_G DF2_CD DF2_VDD DF2_VSS DFF3

XROOT ROOT_Z ROOT_A ROOT_VDD ROOT_VSS BUF

.ENDS

.END
```

This file has the following features:

- The PI segment elements (C1 , C2 , and R2) have the same values as the previous example.
- The pin-to-pin delays are modeled at each of the gate inputs with a capacitor of value 1 pF (C3 and C4 here) and a resistor (R3 and R4) adjusted to give the correct RC delay. Since the load on the output gate is modeled by the PI segment it does not matter what value of capacitance is chosen here.
- The RC elements at the gate inputs are driven by ideal voltage sources (E1 and E2) that are equal to the voltage at the output of the driving gate.

The detailed SPF (DSPF) shows the resistance and capacitance of each segment in a net, again in a SPICE format. There are no models or assumptions on calculating the net delays in this format. Here is an example DSPF file that describes the interconnect shown in Figure 17.23 (a):

```
.SUBCKT BUFFER OUT IN
* Net Section
*|GROUND_NET VSS
*|NET IN 3.8E-01PF
*|P (IN I 0.0 0.0 5.0)
*|I (INV1:A INV A I 0.0 10.0 5.0)
C1 IN VSS 1.1E-01PF
C2 INV1:A VSS 2.7E-01PF
R1 IN INV1:A 1.7E00
*|NET OUT 1.54E-01PF
*|S (OUT:1 30.0 10.0)
*|P (OUT O 0.0 30.0 0.0)
*|I (INV:OUT INV1 OUT O 0.0 20.0 10.0)
C3 INV1:OUT VSS 1.4E-01PF
C4 OUT:1 VSS 6.3E-03PF
C5 OUT VSS 7.7E-03PF
R2 INV1:OUT OUT:1 3.11E00
R3 OUT:1 OUT 3.03E00
*Instance Section
XINV1 INV:A INV1:OUT INV
.ENDS
```

The nonstandard SPICE statements in DSPF are comments that start with '*|' and have the following formats:

```
*|I(InstancePinName InstanceName PinName PinType PinCap X Y)
*|P(PinName PinType PinCap X Y)
```

*|NET NetName NetCap

*|S(SubNodeName X Y)

*|GROUND_NET NetName

Figure 17.23 (b) illustrates the meanings of the DSPF terms: InstancePinName , InstanceName , PinName , NetName , and SubNodeName . The PinType is I (for IN) or O (the letter ’O’, not zero, for OUT). The NetCap is the total capacitance on each net. Thus for net IN, the net capacitance is

0.38 pF = C1 + C2 = 0.11 pF + 0.27 pF.

This particular file does not use the pin capacitances, PinCap . Since the DSPF represents every interconnect segment, DSPF files can be very large in size (hundreds of megabytes).

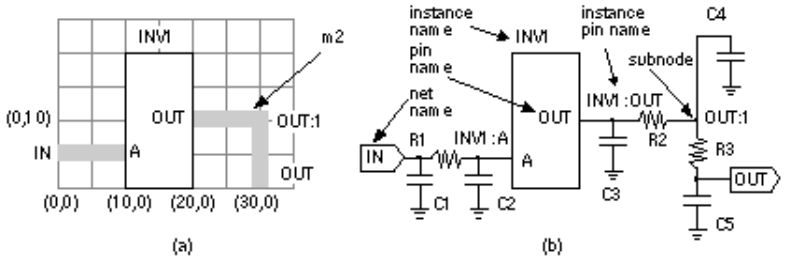


FIGURE 17.23 The detailed standard parasitic format (DSPF) for interconnect representation. (a) An example network with two m2 paths connected to a logic cell, INV1. The grid shows the coordinates. (b) The equivalent DSPF circuit corresponding to the DSPF file in the text.

17.4.2 Design Checks

ASIC designers perform two major checks before fabrication. The first check is a design-rule check (DRC) to ensure that nothing has gone wrong in the process of assembling the logic cells and routing. The DRC may be performed at two levels. Since the detailed router normally works with logic-cell phantoms, the first level of DRC is a phantom-level DRC , which checks for shorts, spacing violations, or other design-rule problems between logic cells. This is principally a check of the detailed router. If we have access to the real library-cell layouts (sometimes called hard layout), we can instantiate the phantom cells and perform a second-level DRC at the transistor level. This is principally a check of the correctness of the library cells. Normally the ASIC vendor will perform this check using its own software as a type of incoming inspection. The Cadence Dracula software is one de facto standard in this area, and you will often hear reference to a Dracula deck that consists of the Dracula code describing an ASIC vendor’s design rules. Sometimes ASIC vendors will give their Dracula decks to customers so that the customers can perform the DRCs themselves.

The other check is a layout versus schematic (LVS) check to ensure that what is about to be committed to silicon is what is really wanted. An electrical schematic is extracted from the physical layout and compared to the netlist. This closes a loop between the logical and physical design processes and ensures that both are the same. The LVS check is not as straightforward as it may sound, however.

The first problem with an LVS check is that the transistor-level netlist for a large ASIC forms an enormous graph. LVS software essentially has to match this graph against a reference graph that describes the design. Ensuring that every node corresponds exactly to a corresponding element in the schematic (or HDL code) is a very difficult task. The first step is normally to match certain key nodes (such as the power supplies, inputs, and outputs), but the process can very quickly become bogged down in the thousands of mismatch errors that are inevitably generated initially.

The second problem with an LVS check is creating a true reference. The starting point may be HDL code or a schematic. However, logic synthesis, test insertion, clock-tree synthesis, logical-to-physical pad mapping, and several other design steps each modify the netlist. The reference netlist may not be what we wish to fabricate. In this case designers increasingly resort to formal verification that extracts a Boolean description of the function of the layout and compare that to a known good HDL description.

17.4.3 Mask Preparation

Final preparation for the ASIC artwork includes the addition of a maskwork symbol (M inside a circle), copyright symbol (C inside a circle), and company logos on each mask layer. A bonding editor creates a bonding diagram that will show the connection of pads to the lead carrier as well as checking that there are no design-rule violations (bond wires that are too close to each other or that leave the chip at extreme angles). We also add the kerf (which contains alignment marks, mask identification, and other artifacts required in fabrication), the scribe lines (the area where the die will be separated from each other by a diamond saw), and any special hermetic edge-seal structures (usually metal).

The final output of the design process is normally a magnetic tape written in Caltech Intermediate Format (CIF , a public domain text format) or GDSII Stream (formerly also called Calma Stream, now Cadence Stream), which is a proprietary binary format. The tape is processed by the ASIC vendor or foundry (the fab) before being transferred to the mask shop .

If the layout contains drawn n -diffusion and p -diffusion regions, then the fab generates the active (thin-oxide), p -type implant, and n -type implant layers. The fab then runs another polygon-level DRC to check polygon spacing and overlap for all mask levels. A grace value (typically 0.01 m m) is included to prevent false errors stemming from rounding problems and so on. The fab will then adjust the mask dimensions for fabrication either by bloating (expanding), shrinking, and merging shapes in a procedure called sizing or mask tooling . The exact procedures are described in a tooling specification . A mask bias is an amount added to a drawn polygon to allow for a difference between the mask size and the feature as it will eventually appear in silicon. The most common adjustment is to the active mask to allow for the bird’s beak effect , which causes an active area to be several tenths of a micron smaller on silicon than on the mask.

The mask shop will use e-beam mask equipment to generate metal (usually chromium) on glass masks or reticles . The e-beam spot size determines the resolution of the mask-making equipment and is usually 0.05 m m or 0.025 m m (the smaller the spot size, the more expensive is the mask). The spot size is significant when we break the integer-lambda scaling rules in a deep-submicron process. For example, for a 0.35 m m process (l = 0.175 m m), a 1.5 l separation is 0.525 m m, which requires more expensive mask-making equipment with a 0.025 m m spot size. For critical layers (usually the polysilicon mask) the mask shop may use optical proximity correction (OPC), which adjusts the position of the mask edges to allow for light diffraction and reflection (the deep-UV light used for printing mask images on

the wafer has a wavelength comparable to the minimum feature sizes).

---

1. Fringing capacitances are per isolated line. Closely spaced lines will have reduced fringing capacitance and increased interline capacitance, with increased total capacitance.

2. NA = not applicable.

17.5 Summary

The completion of routing finishes the ASIC physical design process. Routing is a complicated problem best divided into two steps: global and detailed routing. Global routing plans the wiring by finding the channels to be used for each path. There are differences between global routing for different types of ASICs, but the algorithms to find the shortest path are similar. Two main approaches to global routing are: one net at a time, or all nets at once. With the inclusion of timing-driven routing objectives, the routing problem becomes much harder and requires understanding the differences between finding the shortest net and finding the net with the shortest delay. Different types of detail routing include channel routing and area-based or maze routing. Detailed routing with two layers of metal is a fairly well understood problem.

The most important points in this chapter are:

- Routing is divided into global and detailed routing.
- Routing algorithms should match the placement algorithms.
- Routing is not complete if there are unroutes.
- Clock and power nets are handled as special cases.
- Clock-net widths and power-bus widths must usually be set by hand.
- DRC and LVS checks are needed before a design is complete.

17.6 Problems

*=Difficult, **=Very difficult, *** = Extremely difficult

17.1 (Routing measures, 20 min.). Channel density is a useful measure, but with the availability of more than two layers of metal, area-based maze routers are becoming more common. Lyle Smith, in his 1983 Stanford Ph.D. thesis, defines the Manhattan area measure (MAM) as:

$$MAM = \text{area needed/area available.} \quad (17.11)$$

where you calculate the area needed by assuming routing on a single layer and ignore any interconnect overlaps. Calculate the MAM for Figure 17.14 . Once the MAM reaches 0.5, most two-layer routers have difficulty.

17.2 (*Benchmarking routers, 30 min.) Your design team needs a new router to complete your ASIC project. Your boss puts you in charge of benchmarking. She wants a list of the items you

will test, and a description of how you will test them.

17.3 (Timing-driven routing) (a) Calculate the delay from A to C in Figure 17.3 (b) if the wire between V_3 and V_4 is increased to 5 mm. (b) If you want to measure the delay to the 90 percent point, what is the skew in signal arrival time between inverters B and C? (c) If you use the Elmore constant to characterize the delay between inverter A and inverter C as an RC element, what is the delay (measured to the 50 percent trip point) if you replace the step function at the output of inverter A with a linear ramp with a fall time of 0.1 ns?

17.4 (Elmore delay, 30 min.) Recalculate t_{D4}, t_{D2}, and $t_{D4} - t_{D2}$ for the example in Section 17.1.2 neglecting the pull-down resistance R_{pd} and comment on your answers.

17.5 (Clock routing, 30 min.) Design a clock distribution system with minimum latency given the following specifications: The clocked elements are distributed randomly, but uniformly across the chip. The chip is 400 mil per side. There are 16,000 flip-flops to clock; each flip-flop clock input presents a load of 0.02 pF (one standard load). There are four different types of inverting buffer available (typical for a 0.5 m m process):

1X buffer: $T_D = 0.1 + 1.5\ C_L$ ns; 4X buffer: $T_D = 0.3 + 0.55\ C_L$ ns;

8X buffer: $T_D = 0.5 + 0.25\ C_L$ ns; 32X buffer: $T_D = 2 + 0.004\ C_L$ ns.

In these equations T_D is the buffer delay (assume rise and fall times are approximately equal) and C_L is the buffer load expressed in standard loads. Electromigration limits require a limit of 1 mA (DC) per micron metal width or 10 mA per micron for AC signals with no DC component. No metal bus may be wider than 100 m m. The m2 line capacitance is 0.015 f F m m^{-2} (area) and 0.035 f F m m^{-1} (fringing).

17.6 (Power and ground routing, 10 min.) Calculate the parallel-plate capacitance between a VDD power ring routed on m2 and an identical VSS ring routed on m1 directly underneath. The chip is 500 mil on a side; assume the power ring runs around the edge of the chip. The VDD and VSS bus are capable of carrying 0.5 A and are both 500 m m wide. Assume that m1 and m2 are separated by a SiO_2 dielectric 10,000 Å thick. This capacitance can actually be used for decoupling supplies.

17.7 (Overlap capacitance, 10 min.) Consider two interconnects, both of width W , separated by a layer of SiO_2 of thickness T, and that overlap for a distance L.

- a. What is the overlap capacitance, assuming there are no fringing effects?
- b. Calculate the overlap capacitance if W = 1 m m, T = 0.5 m m, for L = 1, 10, and 100 m m.
- c. Calculate the gate capacitance of an n -channel transistor with transistor size W / L = 2/1 (that is, W = 2 m m, L = 1 m m), with a gate oxide thickness of 200 Å (again assuming no fringing effects).
- d. Comment on your answers.

17.8 (Standard load, 10 min.) Calculate the size of a standard load for the 1 m m process with the parasitic capacitance values shown in Table 17.2 . Assume the n -channel and p -channel devices in a two-input NAND gate are all 10/1 with minimum length.

17.9 (Fringing capacitance, 45 min) You can calculate the capacitance per unit length (including fringing capacitance) of an interconnect with rectangular cross section (width W , thickness T , and a distance H above a ground plane) from the approximate formula (from [Barke, 1988]-the equation was originally proposed by van der Meijs and Fokkema):

$$C = e [(W / H) + 1.064 \div (W / H) + 1.06 \div (T / H) + 0.77] , \quad (17.12)$$

where $e = e_r e_0$ is the dielectric constant of the insulator surrounding the interconnect. The relative permittivity of a SiO_2 dielectric $e_r = 3.9$, and the permittivity of free space $e_0 = 3.45 ¥ 10^{-11}$ Fm^{-1} .

- a. Calculate C for W = T = H = 1 m m.
- b. Compare this value with the parallel-plate value (assuming no fringing capacitance).
- c. Assume that the interconnect cross-sectional area (i.e., WH) is kept constant as technology scales, in order to keep the resistance per unit length of the interconnect constant. Assume that the width scales as sW , the height as sH , and the thickness as T / s , where s is a scaling factor from one to 0.1. Use a spreadsheet to calculate values for different scaling factors, assuming that for s = 1: W = T = H = 1 m m.
- d. Plot your results (with C on the y -axis vs. s on the x -axis).

17.10 (Coupling capacitance, 30 min.) One of the reasons to follow quasi-ideal scaling for the physical dimensions of the interconnect is to try and reduce the parasitic area capacitance as we scale. (The other reason is to try and keep interconnect resistance constant.) Area capacitance scales as $1/ s$ by following ideal scaling rules, but scales as $1/ s^{1.5}$ by using quasi-ideal scaling. Using quasi-ideal scaling means reducing the widths and horizontal spacing of the interconnect by $1/ s$ and the height of the lines and their vertical separation from other layers by only $1/ s^{0.5}$. The effect is rather like turning the interconnects on their sides. As a result we must consider parasitic capacitances other than just the parallel-plate capacitance between two layers. The parasitic capacitance between neighboring interconnects is called coupling capacitance . Fringing capacitance results from the fact that the electric field lines spill out from the edges of a conductor. This means the total parasitic capacitance is greater than if we just considered the capacitance to be formed by two parallel plates.

The following equation is an approximate expression for the capacitance per unit length of an isolated conductor of width W and thickness T , separated by a distance H from a conducting plane, and surrounded by a medium of permittivity e [Sakurai and Tamaru, 1983]:

$$C_1 / e = 1.15 (W / H) + 2.80 (T / H)^{0.222}] . \quad (17.13)$$

This equation is of the form,

$$C_1 = C_a + C_b . \quad (17.14)$$

where C_a represents the contribution from two parallel plates and C_b is the fringing capacitance (for both edges). The following equation then takes into account the coupling capacitance to a neighbor conductor separated horizontally by a gap G between the edges of the conductors:

$$C_2 / e = C_1 / e + [0.03(W / H) + 0.83 (T / H) - 0.07 (T / H)^{0.222}](G / H)^{-1.34} . \quad (17.15)$$

This equation is of the form,

$$C_2 = C_1 + C_c . \quad (17.16)$$

where C_c is the coupling capacitance from the conductor to one neighbor. For a conductor having two neighbors (one on each side), the total capacitance will be

$$C_2 = C_1 + 2 C_c . \quad (17.17)$$

Table 17.3 shows the result of evaluating these equations for different values of T/H , W/H , and S/H for l = 0.5 m m.

- a. Calculate the corresponding values for l = 0.125 m m assuming quasi-ideal scaling.

TABLE 17.3 Calculated fringing capacitance (per unit length and normalized by permittivity) using quasi-ideal scaling and the Sakurai-Tamaru equations. Problem 17.10 completes this table.

Parameter	l = 0.5 m m	l = 0.125 m m
T (m m)	0.5	
W (m m)	1.5	
S (m m)	1.5	
H (m m)	0.5	
T / H	1	
W / H , S / H	3	
C_1 = Ca + Cb	6.25	
$C_2 = C_1$ + Cc	6.44	
$C_3 = C_1$ +2 Cc	6.63	
C_a = parallel plate	3.45	
C_b = fringe (two edges)	2.80	
C_c = coupling (one neighbor)	0.19	
C_c / C_a	6%	

C_a / C_3	52%
C_b / C_3	42%
C_c / C_3	3%

Table 17.4 shows the predicted fringing and coupling capacitance for a l = 0.5 m m process expressed in $pFcm^{-1}$.

TABLE 17.4 Predicted line capacitance including fringing and coupling capacitance ($pFcm^{-1}$) for l = 0.125 m m and using quasi-ideal scaling and the Sakurai equations. Problem 17.10 completes this table.

Parameter	l = 0.5 m m	l = 0.125 m m	Comment
$C_1 = C_a + C$	2.16		C_1 is capacitance of line to ground.
$C_2 = C_1 + C_c$	2.22		C_2 is capacitance including one neighbor.
$C_3 = C_1 + 2\,C_c$	2.29		C_3 is capacitance including two neighbors.
C_a = plate	1.19		C_a is parallel-plate capacitance.
C_b = fringe	0.97		C_b is fringe for both edges.
C_c = coupling	0.07		C_c is coupling to one neighbor only.

- b. Complete the corresponding values for l = 0.125 m m, again assuming quasi-ideal scaling.
- c. Comment on the difference between l = 0.5 m m and l = 0.125 m m.

17.11 (**Routing algorithms, 60 min.) "The Lee algorithm is guaranteed to find a path if it exists, but not necessarily the shortest path." Do you agree with this statement? Can you prove or disprove it?

"The Hightower algorithm is not guaranteed to find a path, even if one exists." Do you agree with this statement? Can you prove or disprove it? Hint: The problems occur not with routing any one net but with routing a sequence of nets.

17.12 (Constraint graphs, 10 min.) Draw the horizontal and vertical constraint graphs for the channel shown in Figure 17.13 (a). Explain how to handle the net that exits the channel and its pseudoterminal.

17.13 (**Electromigration, 60 min.) You just received the first prototype of your new ASIC. The first thing you do is measure the resistance between VDD and VSS and find they are shorted. Horrified, you find that you added your initials on m1 instead of m2 and shorted the supplies, next to the power pads. Your initials are only 10 m m wide, but about 200 m m high! Fortunately only the first capital "I" is actually shorting the supplies. The power-supply rails are approximately 100 m m wide at that point. A thought occurs to you-maybe you can electromigrate your initial away. You remember that electromigration obeys an equation of the form:

$$MTTF = A\,J^{-2} \exp - E / k\,T, \quad (17.18)$$

where MTTF is the mean time to failure, A is a constant, J is the current density, E is an activation energy, k is Boltzmann's constant, and T is absolute temperature. You also remember the rule that you can have about 1 mA of current for every l of metal width for a reasonable time to failure of more than 10 years. Since this chip is in 0.5 m m CMOS (l = 0.25 m m), you guess that the metal is about 0.5 m m thick, and the resistance is at least 50 m W /square.

- a. How much current do you estimate you need to make your initials fail so that you can test the chip before your boss gets back in a week's time?
- b. What else could you do to speed things up?
- c. How are you going to do this? (P.S. This sometimes actually works.)

17.14 (**Routing problems, 20 min.) We have finished the third iteration on the new game chip and are having yield problems in production. This is what we know:

1. We changed the routing on v3 by using an ECO mechanism in the detailed router from Shortem. We just ripped up a few nets and rerouted them without changing anything else.

2. The ASIC vendor, Meltem, is having yield problems due to long metal lines shorting-but only in one place. It looks as though they are the metal lines we changed in v3. Meltem blames the mask vendor-Smokem.

3. To save money we changed mask vendors after completing the prototype version v1, so that v2 and v3 uses the new mask vendor (Smokem). Smokem confirms there is a problem with the v3 mask-the lines we changed are shifted very slightly toward others and have a design rule violation. However, the v2 mask was virtually identical to v3 and there are no problems with that one, so Smokem blames the router from Shortem.

4. Shortem checks the CIF files for us, claims the mask data is correct, and they suggest we blame Meltem.

We do not care (yet) who is to blame, we just need the problem fixed. We need suggestions for the source of the problem (however crazy), some possible fixes, and some ideas to test them. Can you help?

17.15 (*Coupling capacitance, 30 min.) Suppose we have three interconnect lines running parallel to each other on a bus. Consider the following situations (VDD = 5 V, VSS = 0 V):

- a. The center line switches from VSS to VDD. The neighbor lines are at VSS.
- b. The center line switches from VSS to VDD. At the same time the neighbor lines switch from VDD to VSS.
- c. The center line switches from VSS to VDD. At the same time the neighbor lines also switch from VDD to VSS.

How do you define capacitance in these cases? In each case what is the effective capacitance from the center to the neighboring lines using your definition?

17.16 (**2LM and 3LM routing, 10 min.) How would you attempt to measure the difference in die

area obtained by using the same standard-cell library with two-level and three-level routing?

17.17 (***SPF, 60 min)

- a. Write a regular SPF file for the circuit shown in Figure 17.3 (b), using the lumped-C model and the Elmore constant for the pin-to-pin timings.
- b. Write the equivalent RSPF file.
- c. Write a DSPF file for the same circuit.
- d. Calculate the PI segment parameters for the circuit shown in Figure 17.3 (b). Hint: You may need to consult [O'Brien and Savarino, 1989] if you need help.

17.18 (***Standard-cell aspect ratio, 30 min.) How would you decide the optimum value for the logic cell height of a standard-cell library?

17.19 (Electromigration, 20 min.)

- a. What is the current density in a 1 m m wide wire that is 1 m m thick and carries a current of 1 mA?
- b. Using Eq. 17.9 , can you explain the temperature behavior of the parameters in Table 17.1 ?
- c. Using Eq. 17.10 , can you explain the dependence on current direction?

17.20 (***SPF parameters, 120 min.). Hint: You may need help from [O'Brien and Savarino, 1989] for this question.

- a. Find an expression for $Y(s)$, where $s = jw$, the driving-point admittance (the reciprocal of the driving-point impedance), for the interconnect network shown in Figure 17.22 (a), in terms of C_A , C_B , C_C , R_{AB} , and R_{BC} .
- b. Find the first three terms of the Taylor-series expansion for $Y(s)$.
- c. Derive expressions for $Y_1(s)$, $Y_2(s)$, and $Y_3(s)$ for the lumped-C, the lumped-RC, and the PI segment network models (Figure 17.22 b-d).
- d. Comparing your answers to parts b and c, derive the values of the parameters of the lumped-C, the lumped-RC, and the PI segment network models in terms of C_A , C_B , C_C , R_{AB} , and R_{BC} .

17.21 (**Distributed-delay routing, 120 min. [Kahng and Robins, 1995]) The Elmore constant is one measure of net delay,

$$t_{Di} = \sum_k R_{ki} C_k . \quad (17.19)$$

The distributed delay , defined as follows, is another measure of delay in a network:

$$t_p = \sum_k R_{kk} C_k . \quad (17.20)$$

We can write this equation in terms of network components as follows:

$$t_p = \sum_{\text{node } k} (R_0 L_{kn} + R_d)(C_3 + C_n). \quad (17.21)$$

In this equation there are two types of capacitors: those due to the interconnect, C_0, and those due to the gate loads at each sink, C_n. R_d is the driving resistance of the driving gate (the pull-up or pull-down resistance); R_0 is the resistance of a one-grid-long piece of interconnect; and C_0 is the capacitance of a one-grid-long piece of interconnect. Thus,

$$C_k = C_0 + C_n \text{ and } R_{kn} = R_0 L_{kn} + R_d, \quad (17.22)$$

since every path to ground must pass through R_d. L_{kn} is the path length (in routing-grid units) between a node k and one of the n sink nodes.

With these definitions we can expand Eq. 17.21 to the following:

$$t_p = \sum_{\text{node } k} C_0 R_0 L_{kn} + \sum_{\text{node } k} C_n R_0 L_{kn} + R_d C_0 + R_d C_n. \quad (17.23)$$

Figure 17.24 shows examples of three different types of trees. The MRST minimizes the rectilinear path length. The shortest-path tree (SPT) minimizes the sum of path lengths to all sinks. The quadratic minimum Steiner tree (QMST) minimizes the sum of path lengths to all nodes (every grid-point on the tree).

- a. Find the measures for the MRST, SPT, and QMST for each of the three different tree types shown in Figure 17.24 .
- b. Explain how to apply these trees to Eq. 17.23 .
- c. Compare Eqs. 17.19 and 17.20 for the purposes of timing-driven routing.

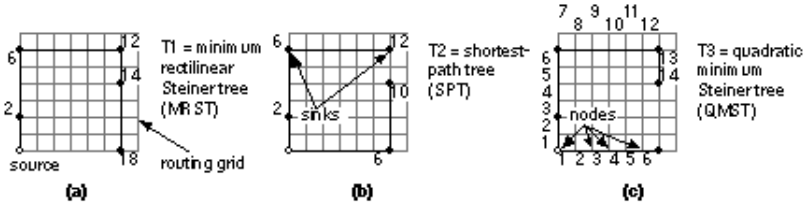


FIGURE 17.24 Examples of trees for timing-driven layout. (a) The MRST. (b) The shortest-path tree (SPT). (c) The quadratic minimum Steiner tree (QMST). (Problem 17.21)

17.22 (**Elmore delay, 120 min.) Figure 17.25 shows an RC tree. The m th moment of the impulse response for node i in an RC tree network with n nodes is

$$m_1(i) = \sum_{k=1}^{n} R_{ki} C_k ,$$

$$m_{n+1}(i) = (m+1) \sum_{k=1}^{n} R_{ki} C_k m_m(k) . \quad (17.24)$$

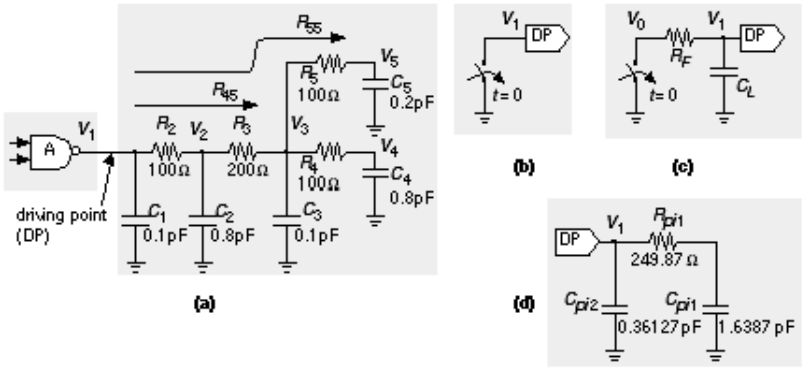


FIGURE 17.25 Standard parasitic format (SPF) (Problem 17.22). (a) An RC interconnect tree driven by a NAND gate. (b) The NAND gate modeled by an ideal switch. (c) The NAND gate modeled with a pull-down resistance, R_F, and output capacitance, C_L. (d) The PI segment model for the RC tree (the order of C_{pi1} -last-and C_{pi2} is correct).

The Elmore constant is the first moment of the impulse response. We calculate the weighted-capacitance values in Eq. 17.24 as follows:

$$k_0 = \sum_{k=1}^{n} C_k ,$$

$$k_m = \frac{1}{m!} \sum_{k=1}^{n} C_k m_m(k) . \quad (17.25)$$

We derive the PI segment parameters used in SPF from the k_i as follows:

$$C = k_2^2 / k_1 ; R = k_2^2 / k_3 ; C = k_1 - C .$$

$C_{pi\,1} = \kappa_1^2 / \kappa_2$; $R_{pi\,1} = \kappa_2^2 / \kappa_1^3$; $C_{pi\,2} = \kappa_0 - C_{pi\,1}$.

- a. Calculate Elmore's constant for the RC tree in Figure 17.25 (a).
- b. Derive the PI segment model shown in Figure 17.25 (d).
- c. What is the difference between using the model of Figure 17.25 (b) and the model of Figure 17.25 (c) for the NAND gate?

17.7 Bibliography

The IEEE Transactions on Computer-Aided Design (TK7874.I327, ISSN 0278-0070) contains papers and tutorials on routing (with an emphasis on algorithms). The Proceedings of the ACM/IEEE Design Automation Conference (DAC, TA174.D46a, ISSN 0146-7123, catalogued under various titles) and the Proceedings of the IEEE International Conference on Computer-Aided Design (ICCAD, TK7874.I3235a, ISSN 1063-6757 and 1092-3152) document the two conferences at which new ideas on routing are often presented.

The edited book by Preas and Lorenzetti [1988] is the best place to learn more about routing. Books by Sarrafzadeh and Wong [1996] and Sait and Youssef [1995] are more recent introductions to physical design including routing. The book Hu and Kuh [1983] edited for IEEE Press contains early papers on routing, including an introductory paper with many references. Ohtsuki's [1986] edited book on layout contains tutorials on routing, including reviews of channel routing by Burstein and area routing by Ohtsuki; a more recent edited book by Zobrist [1994] also contains papers on routing. A good introduction to routing is Joobanni's thesis published as a book [1986]. New routing techniques are becoming important for FPGAs, a recent paper describes some of these [Roy, 1993]. Books by Lengauer [1990] and Sherwani [1993] describe algorithms for both the global and detailed routing problems. A book by Sherwani et al. [1995] covers two-level and three-level routing. Kahng and Robins [1995] cover timing-driven detailed routing in their book; Sapatnekar and Kang [1993] cover timing-driven physical design in general. The book by Pillage et al. [1994] includes a chapter on bounding and asymptotic approximations that are related to the models used in SPF. Nakhla and Zhang [1994] and Goel [1994] cover modeling of interconnect. The IEEE Press book edited by Friedman [1995] covers clock distribution. Routing is often performed in parallel on several machines; books by Banerjee [1994] and Ravikumar [1996] describe parallel algorithms for physical design. Taylor and Russell [1992] review knowledge-based physical design in an edited book.

Najm's review paper covers power estimation [1994]. Books by Shenai [1991] and Murarka [1993] cover all aspects of metallization. To learn more about the causes of electromigration in particular, see D'Heurle's [1971] classic paper and a paper by Black [1969]. The edited book by Gildenblat and Schwartz [1991] covers metallization reliability. A tutorial paper by Young and Christou [1994] reviews current theories of the causes of electromigration. To learn more about masks and microlithography in VLSI, see the handbook by Glendinning and Helbert

17.8 References

Page numbers in brackets after a reference indicate its location in the chapter body.

Banerjee, P. 1994. Parallel Algorithms for VLSI Computer-Aided Design Applications. Englewood Cliffs, NJ: Prentice-Hall, 699 p. ISBN 0130158356. TK7874.75.B36. [reference location]

Barke, E. 1988. "Line-to-ground capacitance calculation for VLSI: A comparison." IEEE Transactions on Computer-Aided Design, Vol. 7, no. 2, pp. 295-298. Compares various equations for line to ground capacitance and finds the van der Meijs and Fokkema equation the most accurate. [reference location]

Black, J. R. 1969. "Electromigration failure modes in aluminum metallization for semiconductor devices." Proceedings of the IEEE, Vol. 57, no. 9, pp. 1587-1594. Describes mechanism and theory of electromigration. Two failure modes are discussed: dissolution of silicon into aluminum, and condensation of aluminum vacancies to form voids. Electromigration failures in aluminum become important (less than 10 year lifetime) at current densities greater than 50 kA/sq.cm and temperatures greater than 150 °C. [reference location]

Cadence. 1990. "Gate Ensemble User Guide." Product Release 2.0. Describes gate-array place-and-route software. The algorithms for timing-driven placement are described in A. H. Chao, E. M. Nequist, and T. D. Vuong, "Direct solution of performance constraints during placement," in Proceedings of the IEEE Custom Integrated Circuits Conference, 1990. The delay models for timing analysis are described in "Modeling the driving-point characteristic of resistive interconnect for accurate delay estimation," in P. R. O'Brien and T. L. Savarino, in Proceedings of the International Conference on Computer-Aided Design, 1989. [reference location]

Cheng, C.-K., et al. 1992. "Geometric compaction on channel routing." IEEE Transactions on Computer-Aided Design, Vol. 11, no. 1, pp. 115-127. [reference location]

Chowdhury, S., and J. S. Barkatullah. 1988. "Current estimation in MOS IC logic circuits." In Proceedings of the International Conference on Computer-Aided Design. Compares estimates for transient current flow for CMOS logic gates. Algebraic models give results close to SPICE simulations. The rest of the paper discusses the calculation of static current flow for nMOS logic gates. A model for static current for CMOS gates is developed in terms of the nMOS models.

D'Heurle, F. M. 1971. "Electromigration and failure in electronics: an introduction." Proceedings of the IEEE, Vol. 59, no. 10, pp. 1409-1417. Describes the theory behind electromigration in bulk and thin-film metals. Includes some experimental results and reviews work by others. Describes the beneficial effects of adding copper to aluminum metallization. [reference location]

Friedman, E. G. (Ed.). 1995. Clock Distribution Networks in VLSI Circuits and Systems. New York: IEEE Press, ISBN 0780310586. TK7874.75.C58. [reference location]

Gildenblat, G. S., and G. P. Schwartz (Eds.). 1991. Metallization: Performance and Reliability Issues for VLSI and ULSI. Bellingham, WA: SPIE, the International Society for Optical Engineering, 159 p. ISBN 0819407275. TK7874.M437. [reference location]

Glendinning, W. B., and J. N. Helbert, (Eds.). 1991. Handbook of VLSI Microlithography : Principles, Technology, and Applications. Park Ridge, NJ: Noyes Publications, 649 p. ISBN 0815512813. TK7874.H3494. [reference location]

Goel, A. K. 1994. High Speed VLSI Interconnections: Modeling, Analysis, and Simulation. New York: Wiley-Interscience, 622 p. ISBN 0471571229. TK7874.7.G63. 21 pages of references. [reference location]

Hashimoto, A., and J. Stevens. 1971. "Wire routing by optimal channel assignment within large apertures." In Proceedings of the 8th Design Automation Workshop, pp. 155-169. [reference location]

Hu, T. C., and E. S. Kuh (Eds.). 1983. VLSI Circuit Layout: Theory and Design. New York: IEEE Press. ISBN 0879421932. TK7874 .V5573. Contains 26 papers divided into six chapters; Part 1: Overview (a paper written for this book with 167 references on layout and routing); Part II: General; Part III: Wireability, Partitioning and Placement; Part IV: Routing; Part V: Layout Systems; Part VI: Module Generation. [reference location]

Joobbani, R. 1986. An Artificial Intelligence Approach to VLSI Routing. Hingham, MA: Kluwer. ISBN 0-89838-205-X. TK7874.J663. Ph.D thesis on the development and testing of an intelligent router including an overview of the detailed routing problem and the Lee and "greedy" algorithms. [reference location]

Kahng, A. B., and G. Robins. 1995. On Optimal Interconnections for VLSI. Norwell, MA: Kluwer. ISBN 0-7923-9483-6. TK7874.75.K34. Extensive reference work on timing-driven detailed routing. [pp. 953, 956]

Lengauer, T. 1990. Combinatorial Algorithms for Integrated Circuit Layout. Chichester, England: Wiley. ISBN 0-471-92838-0. TK7874.L36. Background: Introduction to circuit layout; Optimization problems; Graph algorithms; Operations research and statistics. Combinatorial layout problems: The layout problem; Circuit partitioning; Placement, assignment, and floorplanning; Global routing and area routing; Detailed routing; Compaction. 484 references. [reference location]

Nakhla, M. S., and Q. J. Zhang (Eds.). 1994. Modeling and Simulation of High Speed VLSI Interconnects. Boston: Kluwer, 106 p. ISBN 0792394410. TK7874.75.M64. [reference location]

Murarka, S. P. 1993. Metallization: Theory and Practice for VLSI and ULSI. Stoneham, MA: Butterworth-Heinemann, 250 p. ISBN 0-7506-9001-1. TK7874.M868. Includes chapters on metal properties; crystal structure; electrical and mechanical properties; diffusion and reaction in thin metallic films; deposition method and techniques; pattern definition; packaging applications; reliability. [reference location]

Najm, F. N. 1994. "A survey of power estimation techniques in VLSI circuits." IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 2, no. 4, pp. 446-55. 43 references. [reference location]

O’Brien, P. R., and T. L. Savarino. 1989. "Modeling the driving-point characteristic of resistive interconnect for accurate delay estimation." In Proceedings of the International Conference on Computer-Aided Design, pp. 512-515. Describes SPF PI segment model. [p reference location , 974].

Ohtsuki, T. (Ed.). 1986. Layout Design and Verification. New York: Elsevier Science, ISBN 0444878947. TK7874.L318. Includes nine papers on CAD tools and algorithms: "Layout strategy, standardisation, and CAD tools," Ueda, Kasai, and Sudo; "Layout compaction," Mylynski and Sung; "Layout verification," Yoshida; "Partitioning, assignment and placement," Goto and Matsuda; "Computational complexity of layout problems," Shing and Hu; "Computational and geometry algorithms," Asano, Sato, and Ohtsuki; an excellent survey and tutorial paper by M. Burstein - "Channel routing;" "Maze-running and line-search algorithms," a good, easily readable paper on detailed routing by Ohtsuki; and a more mathematical paper, "Global routing," by Kuh and Marek-Sadowska. [pp. 932, 957]

Pillage, L., et al. 1994. Electronic Circuit and System Simulation Methods. New York: McGraw-Hill, 392 p. ISBN 0-07-050169-6. TK7874.P52. [reference location]

Preas, B. T., and M. J. Lorenzetti. 1988. Physical Design Automation of VLSI Systems. Menlo Park, CA: Benjamin-Cummings, 510 p. ISBN 0805304129. TK7874.P47. Chapters on: physical design automation; interconnection analysis, logic partitioning; placement, assignment and floorplanning; routing; symbolic layout and compaction; module generation and silicon compilation; layout analysis and verification; knowledge-based physical design automation; combinatatorial complexity of layout problems. [reference location]

Ravikumar, C. P. 1996. Parallel Methods for VLSI Layout Design. Norwood, NJ: Ablex, 195 p. ISBN 0893918288. TK7874.R39. [reference location]

Roy, K. 1993. "A bounded search algorithm for segmented channel routing for FPGA’s and associated channel architecture issues." IEEE Transactions on Computer-Aided Design, Vol. 12, no. 11, pp. 1695-1704. [reference location].

Sait, S. M., and H. Youssef. 1995. VLSI Physical Design Automation, Theory and Practice. New York: IEEE Press/McGraw-Hill copublication, 426 p. ISBN 0-07-707742-3. TK7874.75.S24. Covers floorplanning, placement, and routing. [reference location]

Sakurai, T., and K. Tamaru. 1983. "Simple formulas for two- and three-dimensional capacitances." IEEE Transactions on Electron Devices . Vol. 30, no. 2. [reference location]

Sapatnekar, S. S., and S.-M. Kang. 1993. Design Automation for Timing-Driven Layout Synthesis. Boston: Kluwer, 269 p. ISBN 0792392817. TK7871.99.M44.S37. 19 pages of references. [reference location]

Sarrafzadeh, M., and C. K. Wong. 1996. An Introduction to VLSI Physical Design. New York: McGraw-Hill, 334 p. ISBN 0070571945. TK7874.75.S27. 17 pages of references. [reference location]

Shenai, K. (Ed.). 1991. VLSI Metallization: Physics and Technologies. Boston: Artech

House, 529 p. ISBN 0890065012. TK7872.C68.V58. [reference location]

Sherwani, N. A. 1993. Algorithms for VLSI Physical Design Automation. 2nd ed. Norwell, MA: Kluwer, 538 p. ISBN 0-7923-9294-9. TK874.S455. See also the first edition. [reference location]

Sherwani, N. A., et al. 1995. Routing in the Third Dimension: From VLSI Chips to MCMs. New York: IEEE Press. ISBN 0-7803-1089-6. TK7874.75.R68. Reviews two-layer and multilayer routing algorithms. Contains chapters on: graphs and basic algorithms; channel routing; routing models; routing algorithms for two- and three-layer processes and MCMs. [reference location]

Taylor, G., and G. Russell. (Eds.). 1992. Algorithmic and Knowledge Based CAD for VLSI. London: P. Peregrinus, 273 p. ISBN 086341267X. TK7874.A416. [reference location]

Veendrick, H. J. M. 1984. "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits." IEEE Journal of Solid-State Circuits, Vol. 19, no. 4, pp. 468-473. [reference location]

Young, D., and A. Christou. 1994. "Failure mechanism models for electromigration." IEEE Transactions on Reliability, Vol. 43, no. 2, pp. 186-192. A tutorial on electromigration and its relation to microstructure. [p reference location , 957]

Zobrist, G. W. (Ed.). 1994. Routing, Placement, and Partitioning. Norwood, NJ: Ablex, 293 p. ISBN 0893917842. TK7874.R677. [reference location