$$V_B = L\,\frac{dI}{dt} = 3\text{ nH} \cdot \frac{50\text{ mA}}{3\text{ ns}} = 50\text{ mV}$$

This means the "ground" of the circuit actually jumps 50 mV when the circuit needs this current. This noise or ground bounce generated on the ground conductor can feed into other circuits on the chip and cause problems. What's worse is when ten circuits on chip pull this current through the same conductor at the same time.

Ground bounce can be reduced by increasing the width, and thus decreasing the inductance, of the conductors supplying power to a circuit (the wider the metal the better). Also, increasing the capacitance between the conductor supplying current and the conductor returning current helps reduce ground bounce. The simplest method of increasing the capacitance is to lay the conductors out side by side. An on-chip decoupling capacitor between the circuit *VDD* and ground will help as well.

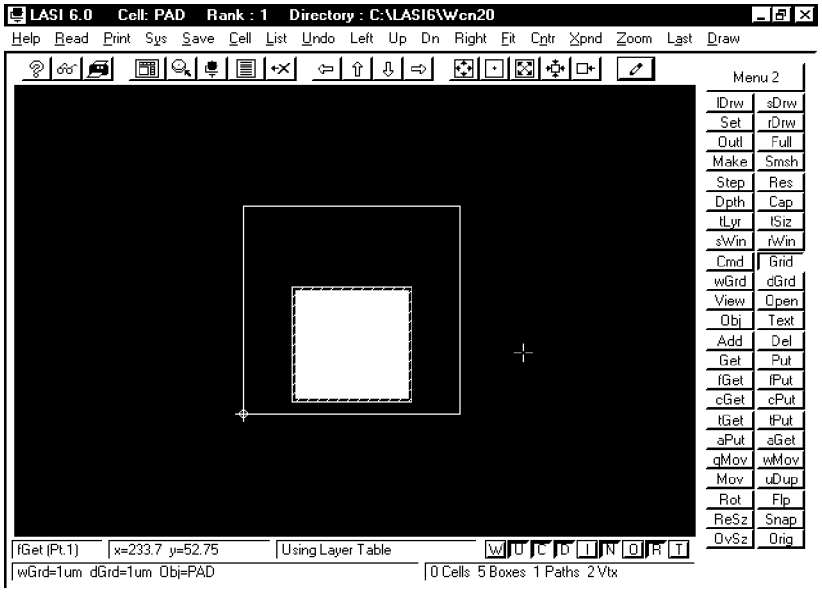
In the previous analysis, we neglected the resistance of the metal wires. In most practical situations, the inductance of the wire is negligible. The ground bounce is dominated by the resistive drop across the wire. Calculating the ground bounce due to the resistive component in Fig. 3.16 begins by calculating the resistance of a wire. The sheet resistance of metal1 is 0.06 Ω/square. The resistance of the wire is 200 Ω. When $I = 0$, the potential at point B is 0. However, when $I = 5$ mA, the potential at point B is 1 V! Again increasing the width of the conductors, and thus decreasing the resistance, helps to reduce the ground bounce.

3.4 Layout Using Cell Hierarchy

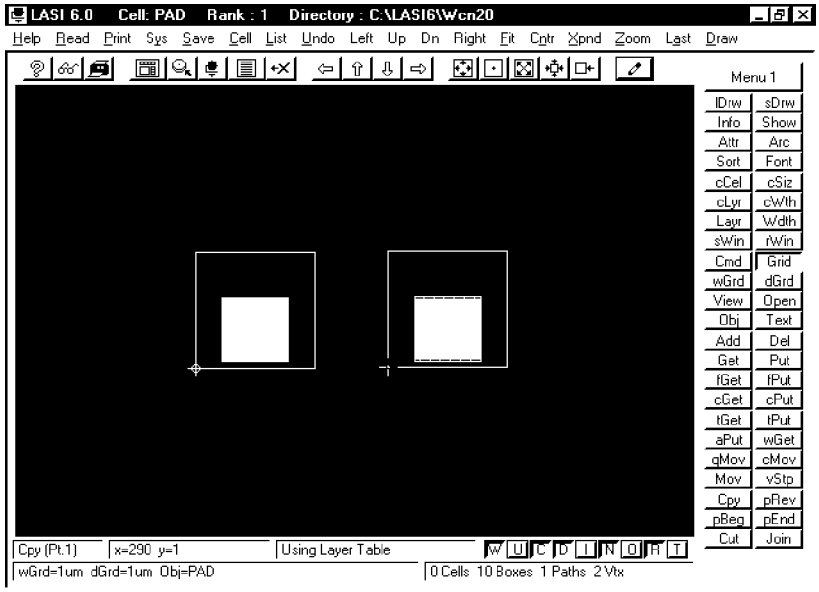
Do not lay out a chip unless the material in this section is understood!

Using cell hierarchy, that is, ranking and nesting of cells, is important to keep the size of the design file (the final GDS file sent to the fab-house) from getting too large. Consider the pad cell shown in Fig. 3.5. Let's assume this cell is called PAD and has a rank of 1. We can generate the layout of Fig. 3.6 in two ways: the right way and the wrong way. Let's consider the wrong way first. Working within the pad cell shown in Fig. 3.17a, use the copy command, **Cpy**, to copy the pad layout Fig. 3.17b, until the number of pads needed is copied into the cell. This is the wrong way to do layout. Each time the cell is copied, boxes or polygons are generated at the new location in the cell, resulting in a significant increase in the size of the design file. In addition, a change in the pad, for example, changing the size of metal2 or adding objects to the cell, must be made to each of the copied pads. This can result in a waste of time. The **Cpy** command should be used as little as possible when laying out a chip.

The correct procedure for laying out the padframe of Fig. 3.6 begins by making a cell called PADFRAME with a rank of 2 using the **Cell** command button. Once in the padframe cell, the object is changed to PAD, the cell we want to use 40 times in the padframe cell using the **Obj** command button. Notice that the PAD cell has a lower ranking than the PADFRAME cell. Figure 3.18 shows use of the **Add** command to add the PAD cell to the PADFRAME cell. Each time the PAD cell is added to the



(a)



(b)

Figure 3.17 How not to lay out a padframe, (a) basic pad cell with a rank of 1 and (b) using the Cpy command to lay out the padframe of Fig. 3.6.

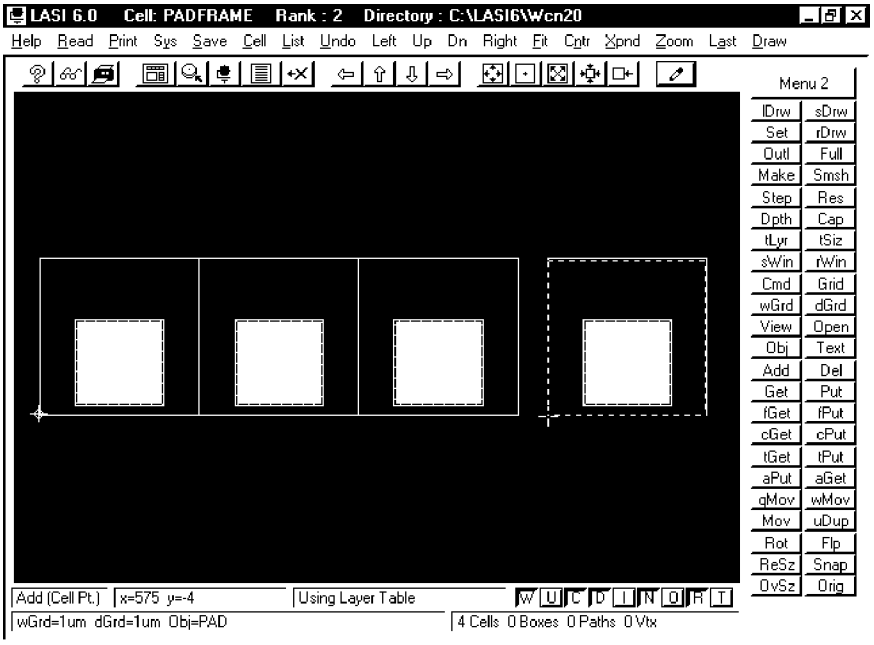


Figure 3.18 Correct method of laying out the PADFRAME cell.

PADFRAME cell a single specifying line, used to reference the added cell and the location, is added to the design file. The result is a significantly smaller design file. Also, if a change is made to the PAD cell, the change is automatically propagated through the hierarchy to the other higher ranking cells using the PAD cell. Figure 3.19 shows the cells of Fig. 3.18 drawn in outlines using the **Outl** command. This can speed up the redraw time.

Cell hierarchy can be extended to every aspect of layout. For example, we know that the via size, used in the CN20 process, is exactly 2 µm by 2 µm square. Instead of drawing a via this size each time we need to make a connection between metal1 and metal2, we can lay out a cell called VIA with a rank of 1 that is a box on the via layer measuring 2 µm by 2 µm square. Now each time we need to make a connection between metal1 and metal2 we simply use the via cell as an object (time is saved since the size of the via is fixed in the VIA cell). The same idea can be used for the active and poly contacts that will be discussed in the next chapter.

When designing VLSI digital circuits, it is common to use minimum-size MOSFETs. A minimum-size MOSFET can be laid out in a cell called NMIN with a rank of one. Each time a MOSFET is needed in a circuit, the NMIN cell can be used. This idea can and should also be used in analog VLSI circuits. Note that an added cell cannot be edited from a higher ranking cell. For example, to edit the PAD cell of Fig. 3.18 we would first have to leave the PADFRAME cell and return to the PAD cell.

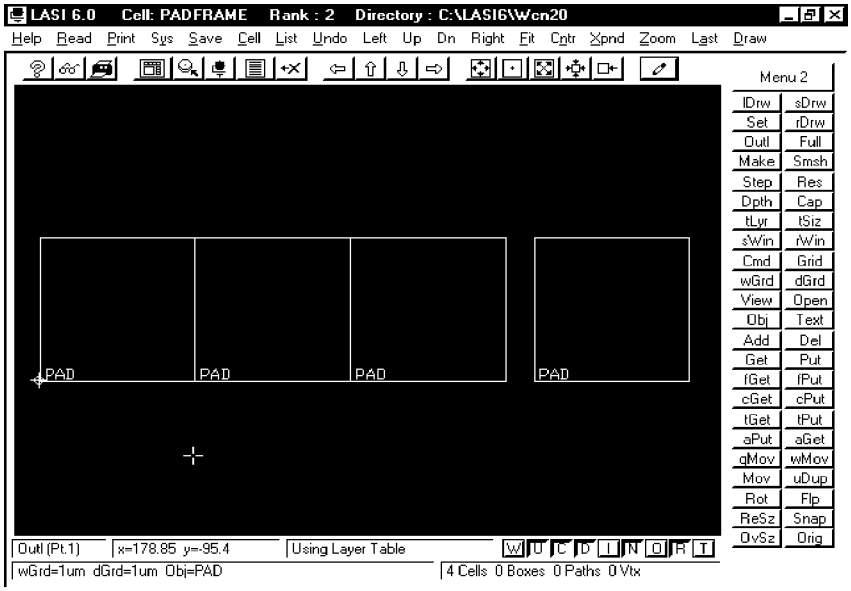


Figure 3.19 Drawing cells in the outline mode can speed up redraw time.

REFERENCES

[1] W. Tanner, *MOSIS User Manual*, Release 4.0, August 1994.

[2] H. W. Johnson and M. Graham, *High Speed Digital Design: A Handbook of Black Magic*, Prentice-Hall Publishing Company, 1993. ISBN 0-13-395724-1.

PROBLEMS

3.1 Lay out and DRC the pad given in Fig. 3.5. Also lay out the padframe shown in Fig. 3.6. The pad cell should be a rank of 1. The padframe should have a rank of 2. Use unique names for the cells. Keep these cells backed up. We will add ESD protection in the next chapter. This is basically the beginnings of the chip you will submit to MOSIS.

3.2 Sketch the cross-sectional view of the layout shown in Fig. P3.2.

3.3 Sketch the cross-sectional view across the center of the pad in Fig. 3.5 without the via layer present. Does the outline layer, layer 58, have any fabrication significance? Why would the outline layer be used?

3.4 What would be the result of submitting a chip without the OVGL layer drawn in the pad cell?

3.5 Explain how to label the pin numbers on the padframe cell of Problem 1 using the **Text** command. Use a Text size of at least 20 µm.