res:drive@ - Nets with no driving transistors will normally not be extracted. This option allows the designer to specify from where in the net the signal is driven. This is primarily useful when extracting subcells, where the transistors driving a given signal may be located in a different cell.

4.2.4. Technology File Changes

Certain changes must be made in the extract section of the technology file to support resistance extraction. These include the **fetresist** and **contact** lines, plus a small change to the fet line. Full details can be found in Magic Maintainer's Manual #2. The only thing to note is that, contrary to the documentation, the **gccap** and **gscap** parts of the fet line MUST be set; the resistance extractor uses them to calculate RC time constants for the circuit.

5. Extraction Details and Limitations

This section explores in greater depth what gets extracted by Magic, as well as the limitations of the circuit extractor. A detailed explanation of the format of the **.ext** files output by Magic may be found in the manual page *ext*(5). "Magic Maintainer's Manual #2: The Technology File" describes how extraction parameters are specified for the extractor.

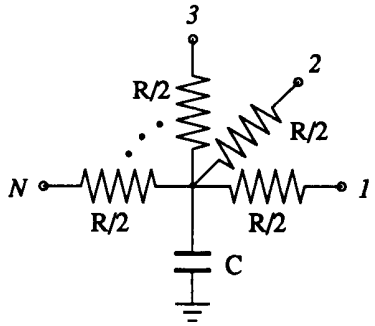

Figure 1. Each node extracted by Magic has a lumped resistance *R* and a lumped capacitance *C* to the substrate. These lumped values can be interpreted as in the diagram above, in which each device connected to the node is attached to one of the points *1*, *2*, ..., *N*.

5.1. Nodes

Magic approximates the pieces of interconnect between transistors as "nodes". A node is like an equipotential region, but also includes a lumped resistance and capacitance to substrate. Figure 1 shows how these lumped values are intended to be interpreted by the analysis programs that use the extracted circuit.

Each node in an extracted circuit has a name, which is either one of the labels attached to the geometry in the node if any exist, or automatically generated by the extractor. These latter names are always of the form *p_x_y#*, where *p*, *x*, and *y* are integers, *e.g.*, **3_104_17#**. If a label ending in the character "**!**" is attached to a node, the node is considered to be a "global". Post-processing programs such as *ext2sim*(1) will check to ensure that nodes in different cells that are labelled with the same global name are electrically connected.

Nodes may have attributes attached to them as well as names. Node attributes are labels ending in the special character "@", and provide a mechanism for passing information to analysis programs such as *crystal*(1). The man page *ext*(5) provides additional information about node attributes.

5.2. Resistance

Magic extracts a lumped resistance for each node, rather than a point-to-point resistance between each pair of devices connected to that node. The result is that all such point-to-point resistances are approximated by the worst-case resistance between any two points in that node.

By default, node resistances are approximated rather than computed exactly. For a node comprised entirely of a single type of material, Magic will compute the node's total perimeter and area. It then solves a quadratic equation to find the width and height of a simple rectangle with this same perimeter and area, and approximates the resistance of the node as the resistance of this "equivalent" rectangle. The resistance is always taken in the longer dimension of the rectangle. When a node contains more than a single type of material, Magic computes an equivalent rectangle for each type, and then sums the resistances as though the rectangles were laid end-to-end.

This approximation for resistance does not take into account any branching, so it can be significantly in error for nodes that have side branches. Figure 2 gives an example. For global signal trees such as clocks or power, Magic's estimate of resistance will likely be several times higher than the actual resistance between two points.

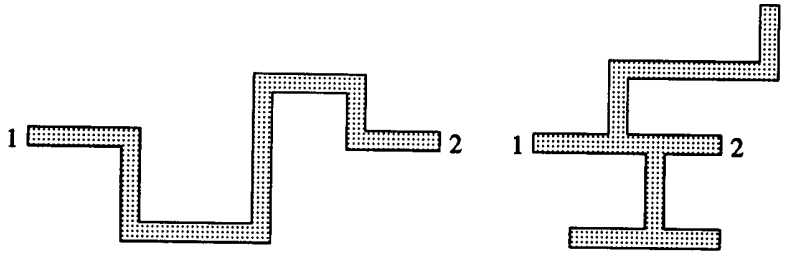

Figure 2. Magic approximates the resistance of a node by assuming that it is a simple wire. The length and width of the wire are estimated from the node's perimeter and area. (a) For non-branching nodes, this approximation is a good one. (b) The computed resistance for this node is the same as for (a) because the side branches are counted, yet the actual resistance between points 1 and 2 is significantly less than in (a).

The approximated resistance also does not lend itself well to hierarchical adjustments, as does capacitance. To allow programs like **ext2sim** to incorporate hierarchical adjustments into a resistance approximation, each node in the **.ext** file also contains a perimeter and area for each "resistance class" that was defined in the technology file (see "Maintainer's Manual #2: The Technology File," and *ext*(5)). When flattening a circuit, **ext2sim** uses this information along with adjustments to perimeter and area to produce the value it actually uses for node resistance.

If you wish to disable the extraction of resistances and node perimeters and areas, use the command

:extract no resistance

which will cause all node resistances, perimeters, and areas in the **.ext** file to be zero. To re-enable extraction of resistance, use the command

:extract do resistance.

Sometimes it's important that resistances be computed more accurately than is possible using the lumped approximation above. Magic's **:extresist** command does this by computing explicit two-terminal resistors and modifying the circuit network to include them so it reflects more exactly the topology of the layout. See the section on **Advanced Extraction** for more details on explicit resistance extraction with **:extresist.**

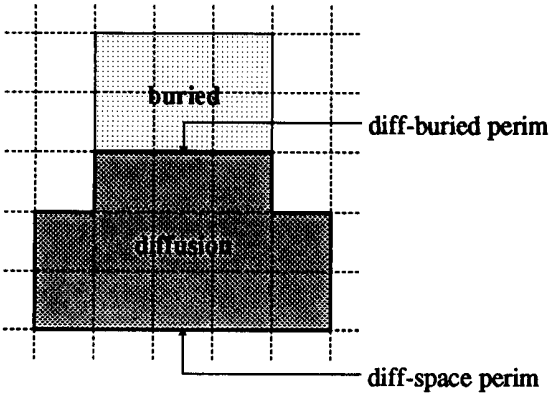

Figure 3. Each type of edge has capacitance to substrate per unit length. Here, the diffusion-space perimeter of 13 units has one value per unit length, and the diffusion-buried perimeter of 3 units another. In addition, each type of material has capacitance per unit area.

5.3. Capacitance

Capacitance to substrate comes from two different sources. Each type of material has a capacitance to substrate per unit area. Each type of edge (i.e, each pair of types) has a capacitance to substrate per unit length. See Figure 3. The computation of capacitance may be disabled with

:extract no capacitance

which causes all substrate capacitance values in the **.ext** file to be zero. It may be re-enabled with

:extract do capacitance.

Internodal capacitance comes from three sources, as shown in Figure 4. When materials of two different types overlap, the capacitance to substrate of the one on top (as determined by the technology) is replaced by an internodal capacitance to the one on the bottom. Its computation may be disabled with

:extract no coupling

which will also cause the extractor to run 30% to 50% faster. Extraction of coupling capacitances can be re-enabled with

:extract do coupling.

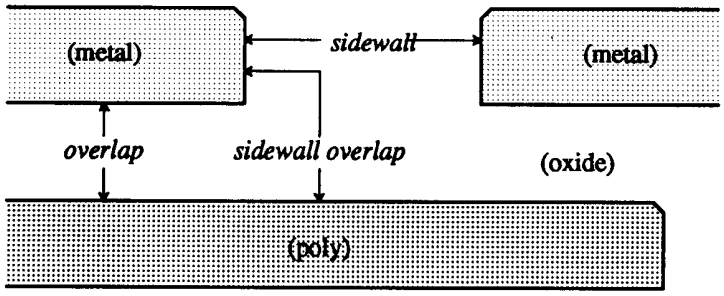

Figure 4. Magic extracts three kinds of internodal coupling capacitance. This figure is a cross-section (side view, not a top view) of a set of masks that shows all three kinds of capacitance. *Overlap* capacitance is parallel-plate capacitance between two different kinds of material when they overlap. *Sidewall* capacitance is parallel-plate capacitance between the vertical edges of two pieces of the same kind of material. *Sidewall overlap* capacitance is orthogonal-plate capacitance between the vertical edge of one piece of material and the horizontal surface of another piece of material that overlaps the first edge.

Whenever material from two subcells overlaps or abuts, the extractor computes adjustments to substrate capacitance, coupling capacitance, and node perimeter and area. Often, these adjustments make little difference to the type of analysis you are performing, as when you wish only to compare netlists. Even when running Crystal for timing analysis, the adjustments can make less than a 5% difference in the timing of critical paths in designs with only a small amount of inter-cell overlap. To disable the computation of these adjustments, use

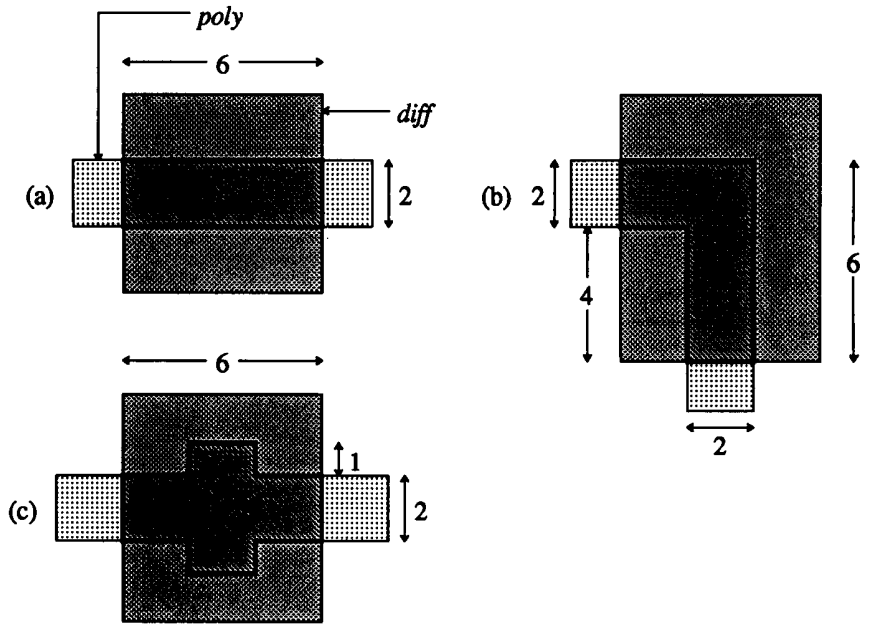

Figure 5.
(a) When transistors are rectangular, it is possible to compute L/W exactly. Here *gateperim* = 4, *srcperim* = 6, *drainperim* = 6, and $L/W = 2/6$. (b) The L/W of non-branching transistors can be approximated. Here *gateperim* = 4, *srcperim* = 6, *drainperim* = 10. By averaging *srcperim* and *drainperim* we get $L/W = 2/8$. (c) The L/W of branching transistors is not well approximated. Here *gateperim* = 16, *srcperim* = 2, *drainperim* = 2. Magic's estimate of L/W is 8/2, whereas in fact because of current spreading, W is effectively larger than 2 and L effectively smaller than 8, so L/W is overestimated.

:extract no adjustment

which will result in approximately 50% faster extraction. This speedup is not entirely additive with the speedup resulting from **:extract no coupling**. To re-enable computation of adjustments, use **:extract do adjustment**.

5.4. Transistors

Like the resistances of nodes, the lengths and widths of transistors are approximated. Magic computes the contribution to the total perimeter by each of the terminals of the transistor. See Figure 5. For rectangular transistors, this yields an exact L/W. For non-branching, non-rectangular transistors, it is still possible to approximate L/W fairly well, but substantial inaccuracies can be introduced if the channel of a transistor contains branches. Since most transistors are rectangular, however, Magic's approximation works well in practice.

In addition to having gate, source, and drain terminals, MOSFET transistors also have a substrate terminal. By default, this terminal is connected to a global node that depends on the transistor's type. For example, p-channel transistors might have a substrate terminal of **Vdd!**, while n-channel transistors would have one of **GND!**. However, when a transistor is surrounded by explicit "well" material (as defined in the technology file), Magic will override the default substrate terminal with the node to which the well

Type	*Loc*	*A P*	*Subs*	*Gate*	*Source*	*Drain*
fet nfet	59 1 60 2	8 12	GND!	Mid2 4 **N3**	Out 4 0	Vss#0 4 0
fet nfet	36 1 37 2	8 12	Float	Mid1 4 **N2**	Mid2 4 0	Vss#0 4 0
fet nfet	4 1 5 2	8 12	Vss#0	In 4 **N1**	Mid1 4 0	Vss#0 4 0
fet pfet	59 25 60 26	8 12	Vdd!	Mid2 4 **P3**	Vdd#0 4 0	Out 4 0
fet pfet	36 25 37 26	8 12	VBias	Mid1 4 **P2**	Vdd#0 4 0	Mid2 4 0
fet pfet	4 25 5 26	8 12	Vdd#0	In 4 **P1**	Vdd#0 4 0	Mid1 4 0

Table 1. The transistor section of **tut8l.ext**.

material is connected. This has several advantages: it allows simulation of analog circuits in which wells are biased to different potentials, and it provides a form of checking to ensure that wells in a CMOS process are explicitly tied to the appropriate DC voltage.

Transistor substrate nodes are discovered by the extractor only if the transistor and the overlapping well layer are in the same cell. If they appear in different cells, the transistor's substrate terminal will be set to the default for the type of transistor.

Load the cell **tut8l**, extract it, and look at the file **tut8l.ext**. Table 1 shows the lines for the six transistors in the file. You'll notice that the substrate terminals (the *Subs* column) for all transistors are different. Since each transistor in this design has a different gate attribute attached to it (shown in bold in the table, *e.g.*, **N1**, **P2**, etc), we'll use them in the following discussion.

The simplest two transistors are **N3** and **P3**, which don't appear in any explicitly drawn wells. The substrate terminals for these are **GND!** and **Vdd!** respectively, since that's what the technology file says is the default for the two types of transistors. **N1** and **P1** are standard transistors that lie in wells tied to the ground and power rails, labelled in this cell as **Vss#0** and **Vdd#0** respectively. (They're not labelled **GND!** and **Vdd!** so you'll see the difference between **N1** and **N3**). **P2** lies in a well that is tied to a different bias voltage, **VBias**, such as might occur in an analog design. Finally, **N2** is in a well that isn't tied to any wire. The substrate node appears as **Float** because that's the label that was attached to the well surrounding **N2**.

The ability to extract transistor substrate nodes allows you to perform a simple check for whether or not transistors are in properly connected (*e.g.*, grounded) wells. In a p-well CMOS process, for example, you might set the default substrate node for n-channel transistors to be some distinguished global node other than ground, *e.g.*, **NSubstrateNode!**. You could then extract the circuit, flatten it using *ext2spice* (1) (which preserves substrate nodes, unlike *ext2sim* (1) which ignores them), and look at the substrate node fields of all the n-channel transistors: if there were any whose substrate nodes weren't connected to **GND!**, then these transistors appear either outside of any explicit well (their substrate nodes will be the default of **NSubstrateNode**), or in a well that isn't tied to **GND!** with a substrate contact.

6. Extraction styles

Magic usually knows several different ways to extract a circuit from a given layout. Each of these ways is called a *style*. Different styles can be used to handle different fabrication facilities, which may differ in the parameters they have for parasitic

capacitance and resistance. For a scalable technology, such as the default **scmos**, there can be a different extraction style for each scale factor. The exact number and nature of the extraction styles is described in the technology file that Magic reads when it starts. At any given time, there is one current extraction style.

To print a list of the extraction styles available, type the command

:extract style.

The **scmos** technology has the styles **lambda=1.5**, **lambda=0.7**, and **oldlambda=0.5**. To change the extraction style to *style*, use the command

:extract style *style*

Each style has a specific scale factor between Magic units and physical units (*e.g.*, microns); you can't use a particular style with a different scale factor. To change the scalefactor, you'll have to edit the appropriate style in the **extract** section of the technology file. This process is described in "Magic Maintainer's Manual #2: The Technology File."

7. Flattening Extracted Circuits

Unfortunately, very few tools exist to take advantage of the *ext* (5) format files produced by Magic's extractor. To use these files for simulation or timing analysis, you will most likely need to convert them to a flattened format, such as *sim* (5) or *spice* (5).

There are several programs for flattening *ext* (5) files. *Ext2sim* (1) produces *sim* (5) files suitable for use with *crystal* (1), *esim* (1), or *rsim* (1). *Ext2spice* (1) is used to produce *spice* (5) files for use with the circuit-level simulator *spice* (1). Finally, *extcheck* (1) can be used to perform connectivity checking and will summarize the number of flattened nodes, transistors, capacitors, and resistors in a circuit. All of these programs make use of a library known as *extflat* (3), so the conventions for each and the checks they perform are virtually identical. The documentation for *extcheck* covers the options common to all of these programs.

To see how *ext2sim* works, load the cell **tut8n** and expand all the **tutm** subcells. Notice how the **GND!** bus is completely wired, but the **Vdd!** bus is in three disconnected pieces. Now extract everything with **:extract**, then exit Magic and run **ext2sim tut8n**. You'll see the following sort of output:

```
*** Global name Vdd! not fully connected:
One portion contains the names:
  left/Vdd!
The other portion contains the names:
  center/Vdd!
I'm merging the two pieces into a single node, but you
should be sure eventually to connect them in the layout.

*** Global name Vdd! not fully connected:
One portion contains the names:
  left/Vdd!
  center/Vdd!
The other portion contains the names:
  right/Vdd!
I'm merging the two pieces into a single node, but you
should be sure eventually to connect them in the layout.

Memory used: 56k
```

The warning messages are telling you that the global name **Vdd!** isn't completely wired in the layout. The flattener warns you, but goes ahead and connects the pieces together anyway to allow you to simulate the circuit as though it had been completely wired. The output of *ext2sim* will be three files: **tut8n.sim**, **tut8n.al**, and **tut8n.nodes**; see *ext2sim*(1) or *sim*(5) for more information on the contents of these files. "**Magic Tutorial #11: Using RSIM with Magic**" explains how to use the output of *ext2sim* with the switch-level simulator, *rsim*(1).