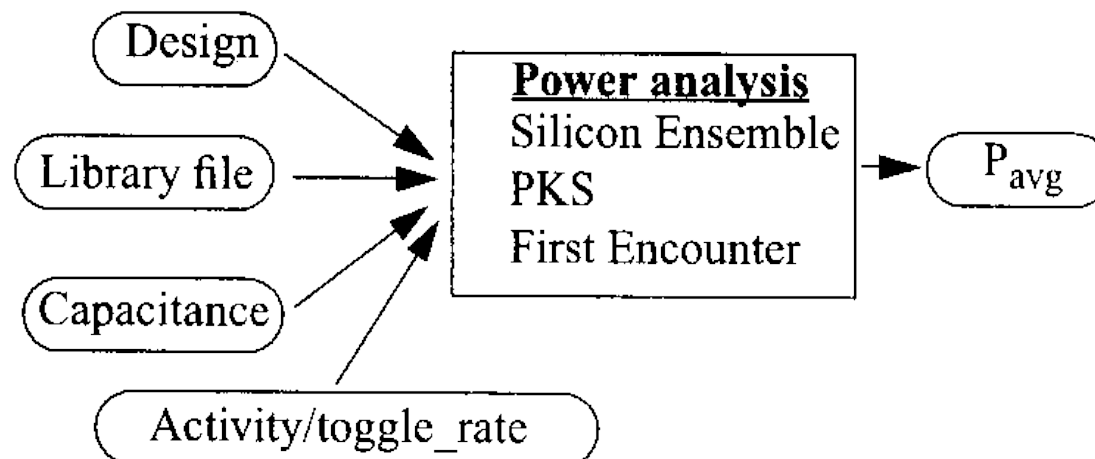


Calculating Power Consumption

You can manually calculate the power consumption of a block by measuring the following types of power usage:

- Combinational logic power
- Clock network power
- MegaCell power

However, because manual computation is tedious, Cadence recommends using Silicon Ensemble[®], Physically Knowledgeable Synthesis (PKS), or First Encounter[®] software to calculate the average power consumption. The following figure provides an overview of the power analysis process.



- **Silicon Ensemble** includes a power analyzer. After you run the power analyzer, a log file named `pa.log` is generated and you can get the average power consumption for the block. See the *Power Analyzer User Guide* for more information on how to use the power analyzer.

- **PKS (version 5.0)** includes a set of options that you can specify after you load the libraries, netlist, and clocking constraints. Issue the following commands:

```
pks_shell > set_switching_activity  
pks_shell > report_power >> power.log
```

The `power.log` file contains the block power value.

- **First Encounter** includes power analysis report files (`vdd.report` and `gnd.report`), which contain the block power values. See the *First Encounter User Guide* for more information.

Rough Power Estimation

You can get a rough power estimation by using the following procedure:

1. Calculate the *area ratio* between a block (or core) area (from DEF) and a NAND2 gate area (from LEF).
2. Use the energy or current number from the timing library for the NAND2 gate, because this is the middle point of a typical library.
3. Multiply the power number by the area ratio, and then multiply that value by the *area utilization* (normally around 50% to 60%). The result of that calculation is a very rough estimate of the power consumption for the block (or core).

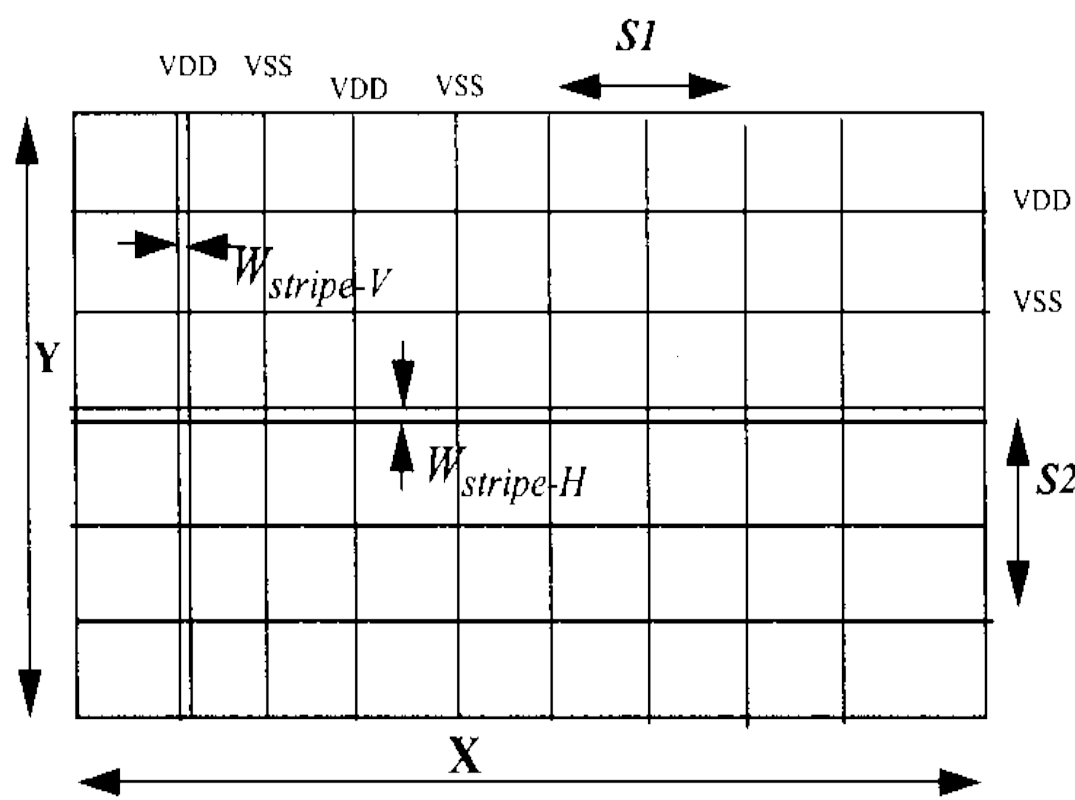
$$\text{Ratio}_{\text{area}} = (\text{Area}_{\text{block}}) \div (\text{Area}_{\text{NAND2}})$$

$$\text{Power}_{\text{block(rough_estimate)}} = \text{Power}_{\text{NAND2}} \times \text{Ratio}_{\text{area}} \times \text{Area Utilization}$$

Check this calculated power result against power consumption values generated by Silicon Ensemble-PKS or First Encounter to see if they are reasonably close. If the values are significantly different, make sure that power numbers are available in all libraries (especially the hard macros) when running Silicon Ensemble-PKS or First Encounter power analysis.

Estimating Power Stripe Width

Use the power consumption value to start designing a power grid that can distribute power over the entire design. Typically, power grids are designed as shown in the following figure.



Determining the widths of vertical stripes ($W_{stripe-V}$) and horizontal stripes ($W_{stripe-H}$) and the width of horizontal spacing ($S1$) and vertical spacing ($S2$) between two VDDs is challenging, but you can use the following guidelines as a starting point.

Guidelines for Determining $W_{\text{stripe-V}}$

$W_{\text{stripe-V}}$ should be a multiple of the vertical routing pitch (for *Metal2* in the example below) but less than maximum value of the spacing rule. This value is approximately four times the width of the minimum NAND2 gate value. The reason for using this value is to ensure that the thickness of the metal will not obstruct signal routing channels.

In the following LEF example, NAND2XL has a width of 1.98; $4 \times 1.98 = 7.92 \mu\text{m}$, which is less than the maximum value of the spacing range ($9.999 \mu\text{m}$).

```
LAYER METAL2
    PITCH .66 ;
    WIDTH .28 ;
    SPACING .6 RANGE 10 100000 ;
    SPACING .28 RANGE 0 9.999 ;
    DIRECTION VERTICAL ;
    RESISTANCE RPERSQ .101 ;
    CAPACITANCE CPERSQDIST .7E-04 ;
END METAL2
```

```

MACRO NAND2XL
    CLASS CORE ;
    SIZE 1.98 BY 5.04 ;
    . . . . .
END NAND2XL

```

Guidelines for Determining $W_{\text{stripe-H}}$

Important

It is best to use the highest level of *Metal* for power routing, because higher layers have the least resistance and consequently the least voltage drop.

$W_{\text{stripe-H}}$ should be a multiple of standard cell heights (preferably one to two times).

For the TSMC 6 layer process, assume that *Metal2*, *Metal4*, and *Metal6* are in the vertical routing direction and that *Metal1*, *Metal3*, and *Metal5* are in the horizontal routing direction.

- Use $W_{\text{stripe-V}}$ with *Metal4* or *Metal6* for block-level vertical stripes.
- Use $W_{\text{stripe-V}}$ with *Metal6* for chip-level vertical stripes.
- Use $W_{\text{stripe-H}}$ with *Metal5* for both block-level and chip-level horizontal stripes.

In order to have stacked vias from *Metal4* (or *Metal6*) all the way to standard cell power rails *Metal1*, specify the following variables with the Silicon Ensemble FOLLOWPIN command:

```

SET VAR SROUTE.ALLOWOVERLAPINSTACKVIA    TRUE ;
SET VAR SROUTE.STACKVIASATCROSSOVER      TRUE ;
SET VAR SROUTE.VIA.MERGEONSAMELAYERS     TRUE ;

```

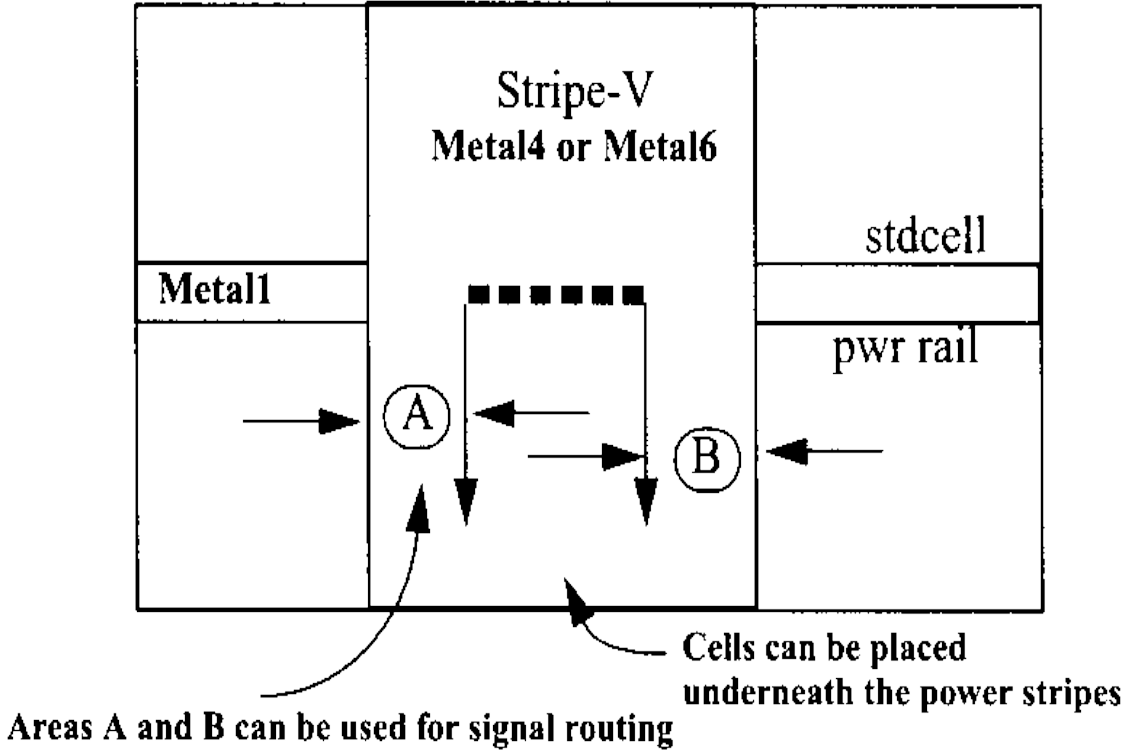
For congested designs, specify the following variable with the Silicon Ensemble FOLLOWPIN command:

```

SET VAR SROUTE.MAX.VIAARRAYROWS 6 ;

```

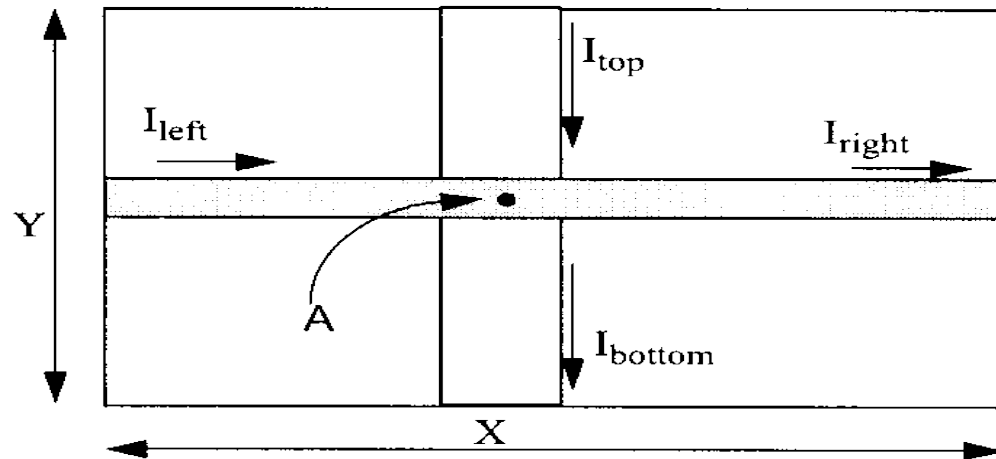
This provides more area for signal routing under the power stripes, as shown in the following figure.



With thousands of vertical stripes, the space under these stripes (as shown in areas A and B) provides a significant source of space for signal routing. Use the highest level of *Metal* for power routing to free up the maximum amount of space for signal routing.

Calculating Power Grids

If you know the average power consumption value (P_{avg}) from Silicon Ensemble-PKS or First Encounter, and the horizontal and vertical stripe widths ($W_{stripe-V}$ and $W_{stripe-H}$), you can calculate the total current (I_{total}) going through the midpoint of the block, shown as point A in the following figure



$$I_{total} = P_{avg} \div VDD$$

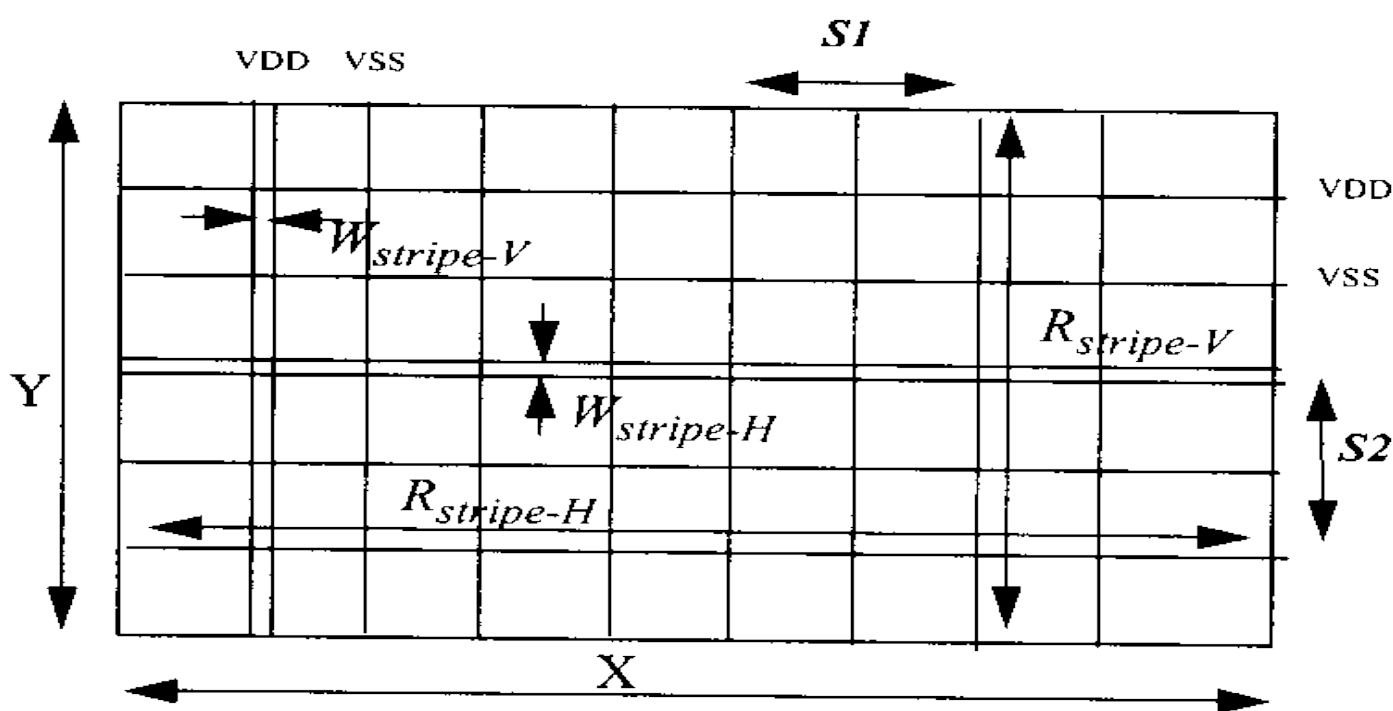
$$I_{top} = I_{bottom} = I_{total} \times \frac{X}{(X + Y)}$$

$$I_{left} = I_{right} = I_{total} \times \frac{Y}{(X + Y)}$$

With a 5% voltage drop at (A), the effective resistance is:

$$R_{effective}(vertical@A) = 0.05 \times (VDD \div 2) \div I_{top}$$

$$R_{effective}(horizontal@A) = 0.05 \times (VDD \div 2) \div I_{left}$$



$$R_{\text{stripe-V}} = R1_{\square} \times \frac{Y}{W_{\text{stripe-V}}}$$

($R1_{\square}$ is sheet resistance in ohm/square for vertical stripes)

$$R_{\text{stripe-H}} = R2_{\square} \times \frac{X}{W_{\text{stripe-H}}}$$

($R2_{\square}$ is sheet resistance in ohm/square for horizontal stripes)

(N = number of vertical stripes)

$$N = \frac{R_{\text{stripe-V}}}{R_{\text{effective-V}}}$$

(M = number of horizontal stripes)

$$M = \frac{R_{\text{stripe-H}}}{R_{\text{effective-H}}}$$

$$S1 = X/N$$

$$S2 = Y/M$$

"First Encounter auto_power.pl Mesh Generation Perl Script" on page 23 automates this formula.

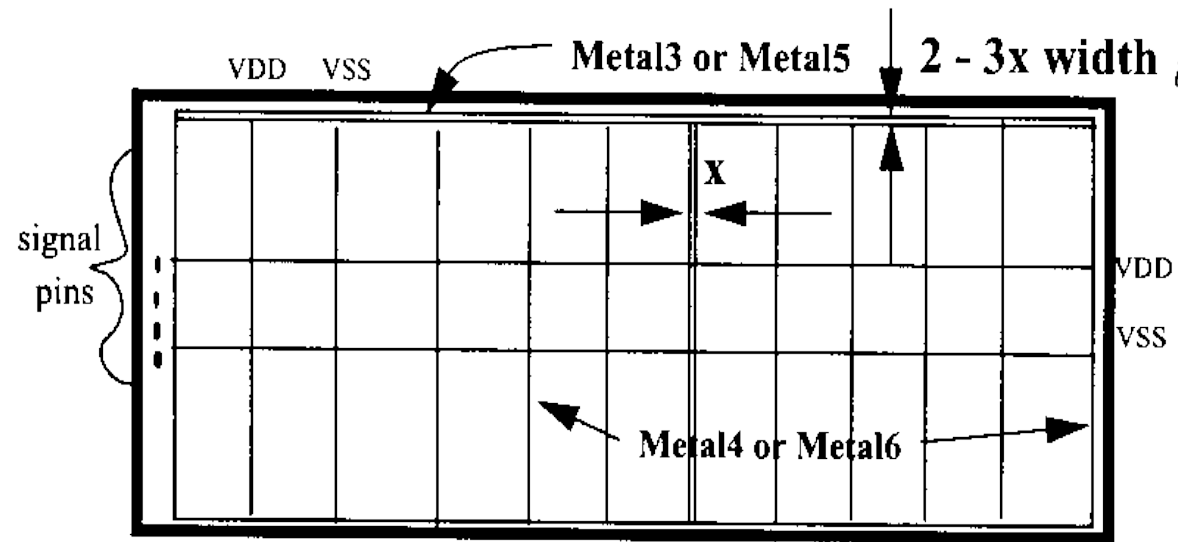
- For the TSMC 0.18 μm technology, S1 and S2 are in the range of 100 to 200 μm (assuming that $W_{\text{stripe-V}}$ is approximately four times the width of NAND2 and $W_{\text{stripe-H}}$ is the height of one standard cell).
- For the TSMC 0.13 μm technology, S1 and S2 are in the range of 50 to 100 μm (assuming that $W_{\text{stripe-V}}$ is approximately four times the width of NAND2 and $W_{\text{stripe-H}}$ is the height of two standard cells).

Sample Power Mesh Structure Schemes

The type and spacing of stripes that you use depends on the shape of the block.

Long Horizontal Rectangular Shape

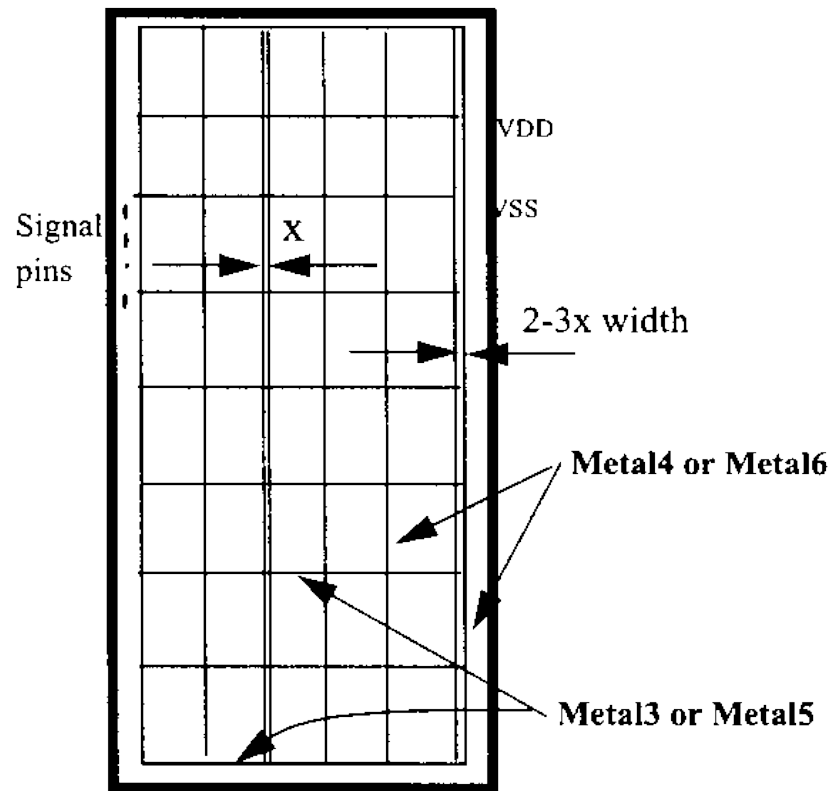
The following figure shows an example of a shape that requires many vertical stripes. Horizontal stripes are not necessary.



Horizontal rectangular shape power mesh

Long Vertical Rectangular Shape

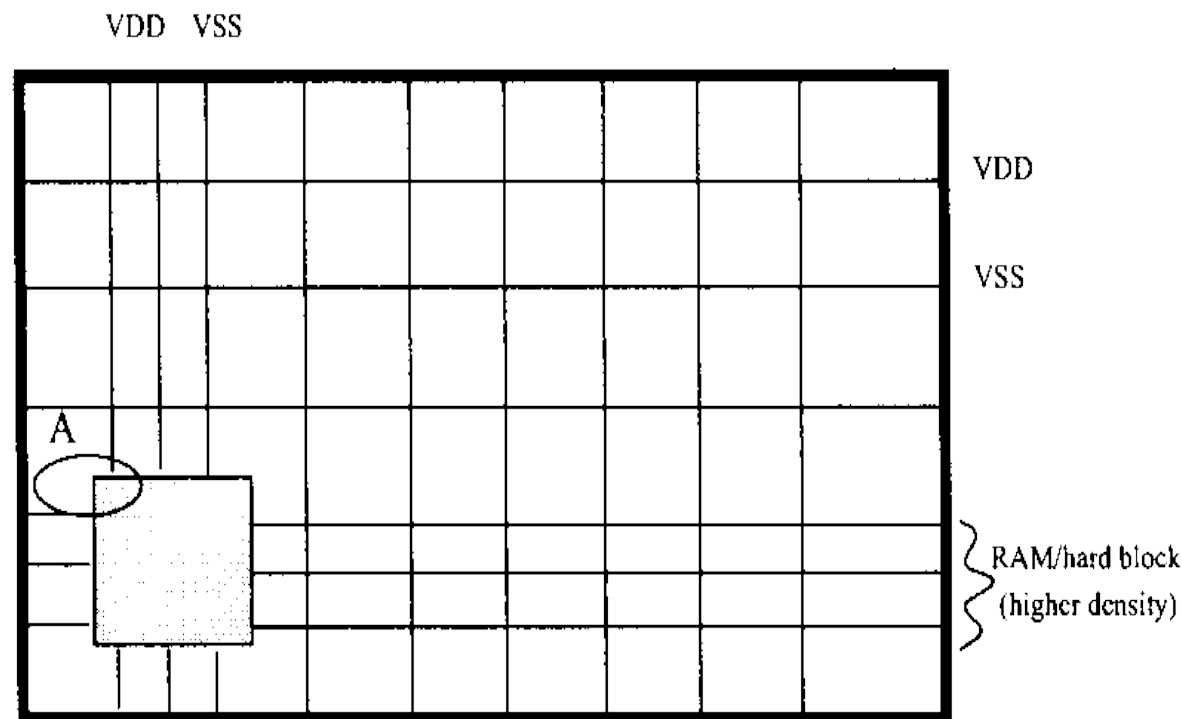
This shape requires many horizontal stripes, but also needs vertical stripes for power followpin connections. The tall skinny block shown in the following figure has limited routing resources for the vertical routing direction.



If a library has standard cell pins on *Metal1*, then this tall block will have routing congestion on the vertical layers (such as *Metal3* and *Metal5*), because many vertical routing signals must go through *VIA32*, then *Metal2*, and finally *VIA21* before they are able to access the *std_cell Metal1* pins. This makes it very hard to get a high-utilization design that is routable for tall skinny blocks. For this reason, during block partitioning, you should try to use long horizontal rectangular shapes.

Low Frequency Power Mesh Structure with a Hard Macro

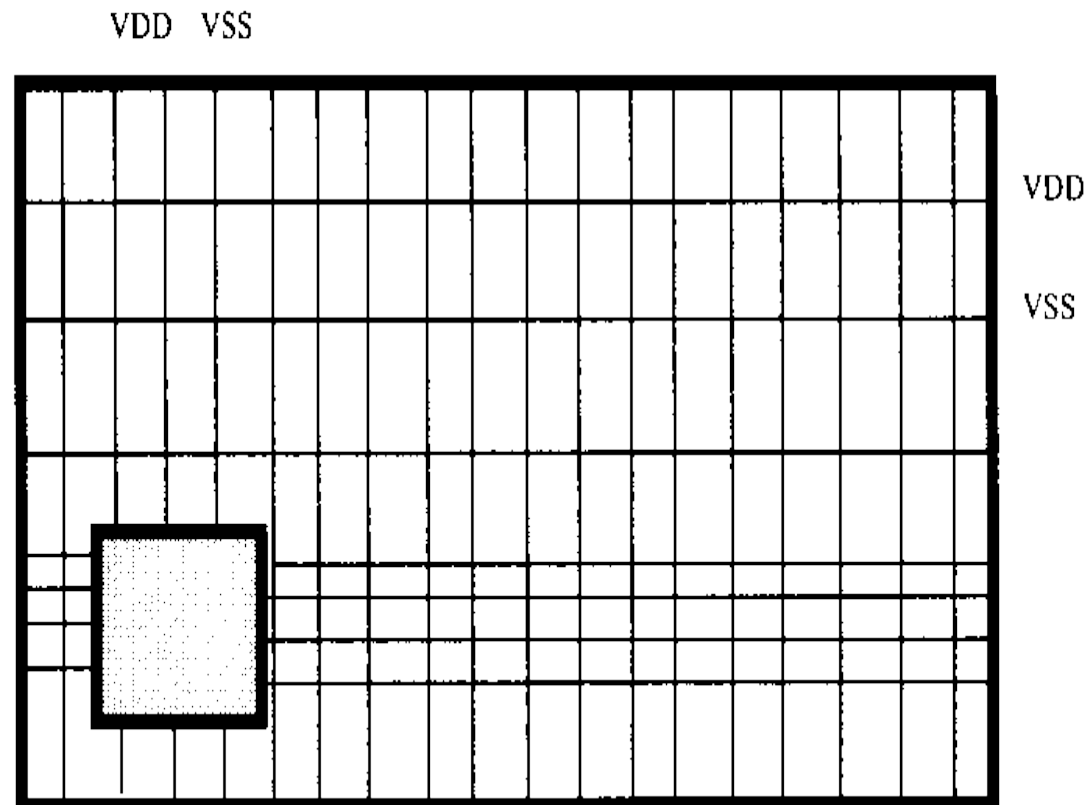
RAM and other hard blocks consume more power and therefore require a higher density power grid, as shown in the following figure.



Note: To avoid DRC violations (vias over vias), try not to connect power at a corner.

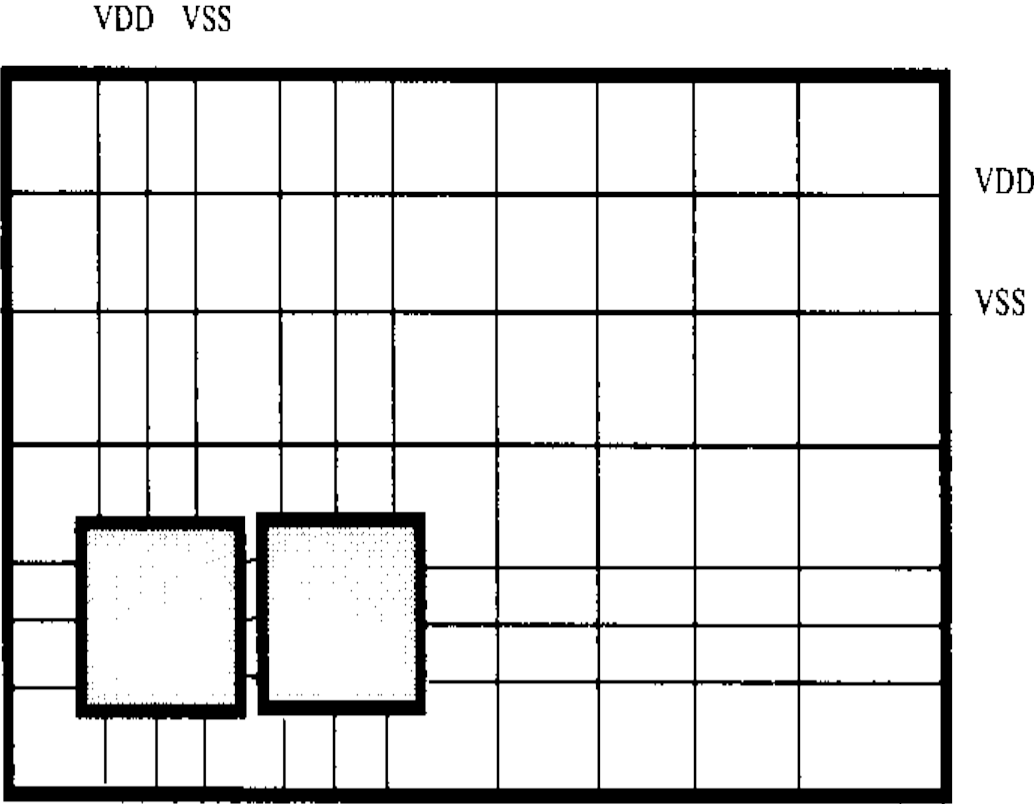
High Frequency Power Mesh Structures with Hard Macro

For high frequency designs, use denser power grids, as shown in the following figure.



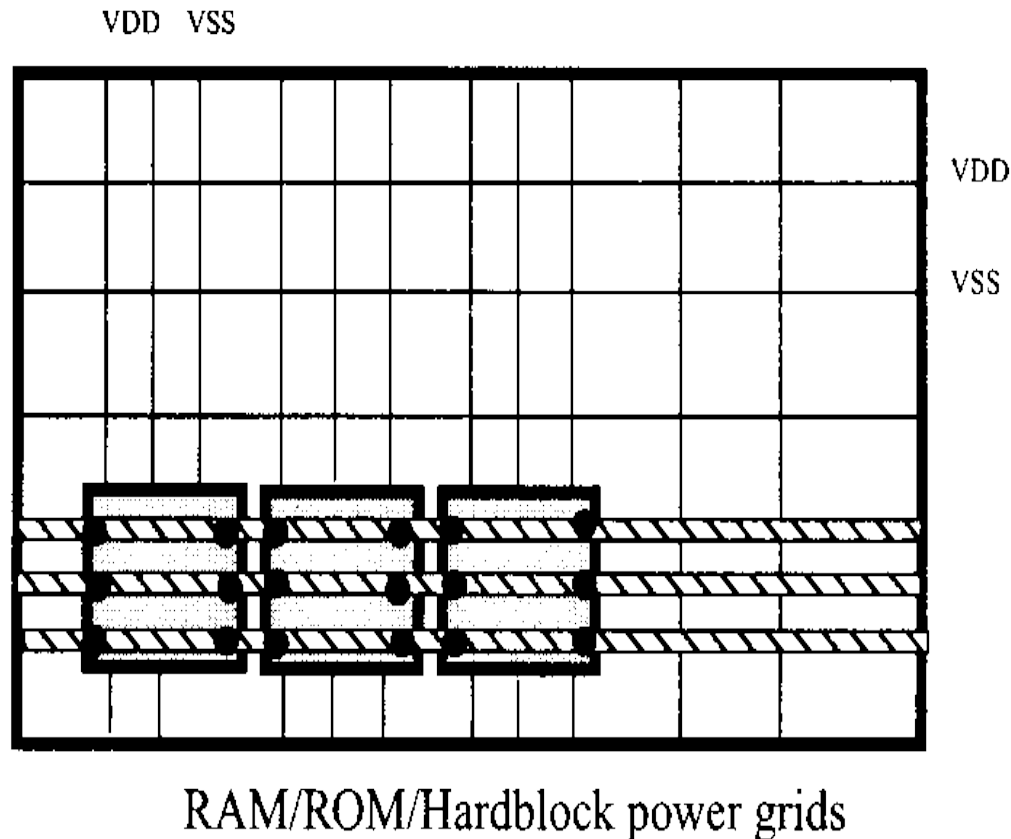
Multiple Hard Block Power Mesh Structures

Create power structures between hard blocks so they can share power, as shown in the following figure.



Multiple Hard Block Power Mesh Structures

For multiple hard blocks placed side by side, power grids should contain thick metals on top of these structures (using *Metal5*) to supply current to blocks in the middle, as shown in the following figure.



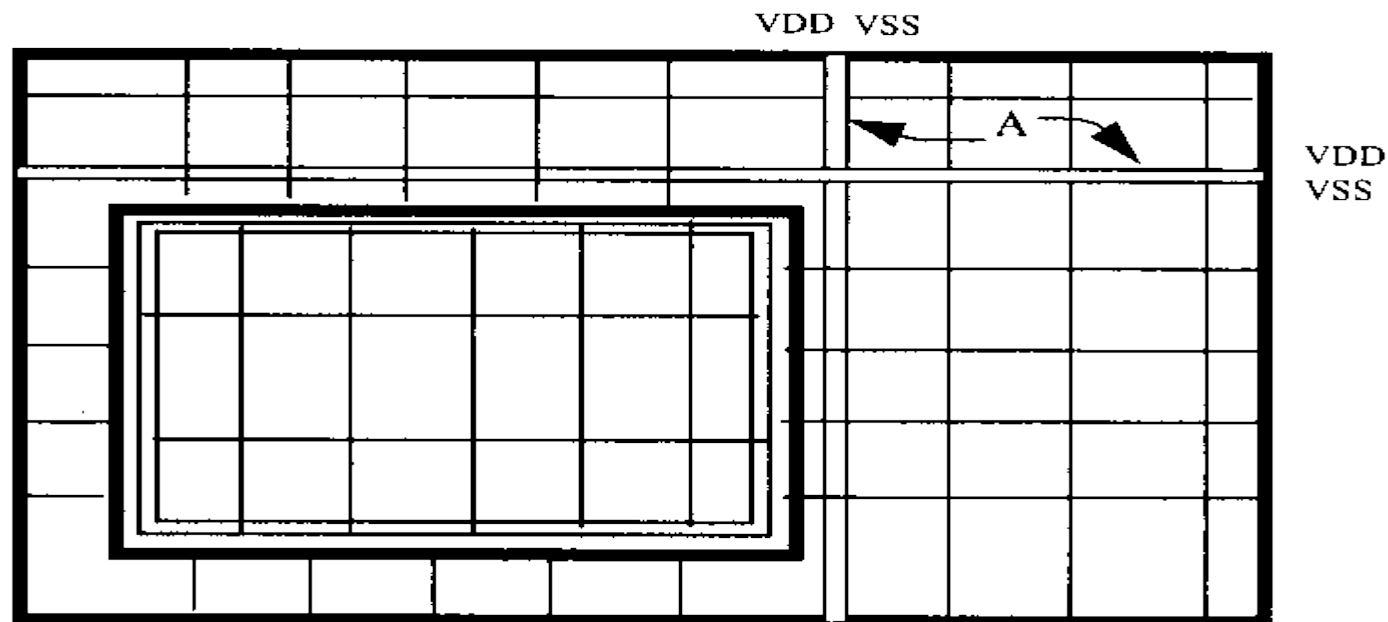
Hierarchical Power Design

The recommended method for designing a hierarchical power structure depends on whether the design is bottom up or top down.

Bottom Up

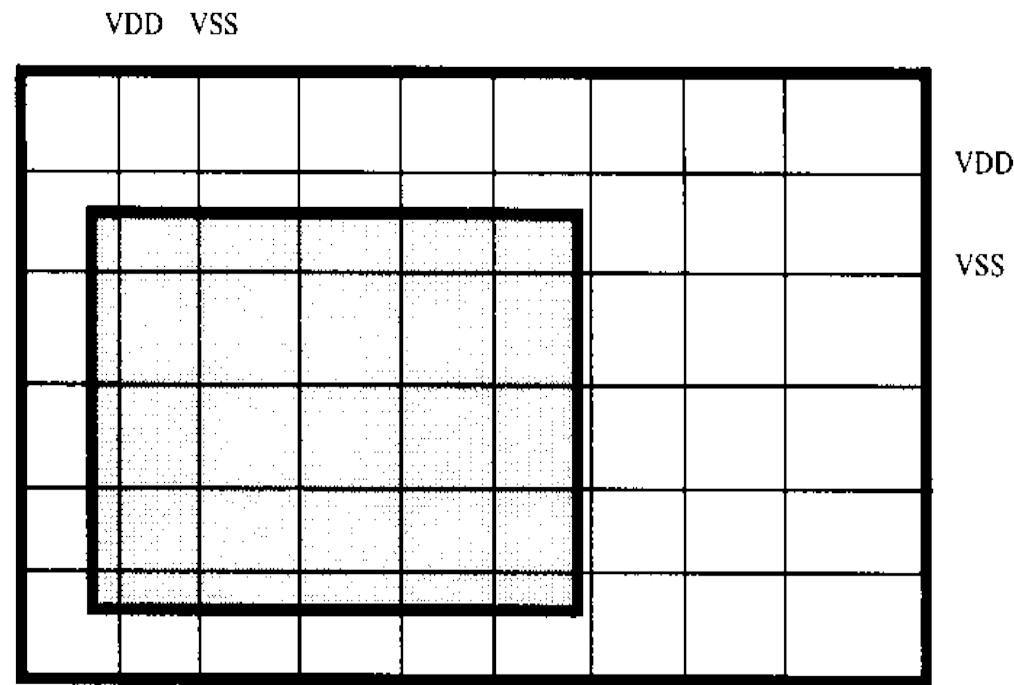
For bottom up designs, Cadence recommends creating a *ring* around the block so that the chip-level power stripes can be connected to the block ring. Another reason for creating the block ring is to allow users to move (or adjust) the hard block anywhere, while allowing the power router from the chip level to make the connection easily. Block ring width is typically two to three times the width of the block-level stripes.

Note: Some high frequency blocks or large hard blocks might require extra-thick metal outside the block (indicated by A in the figure below) to avoid current density problems or to prevent voltage drop.



Top Down Cookie Cutter Power Mesh (Using the First Encounter Partition Function)

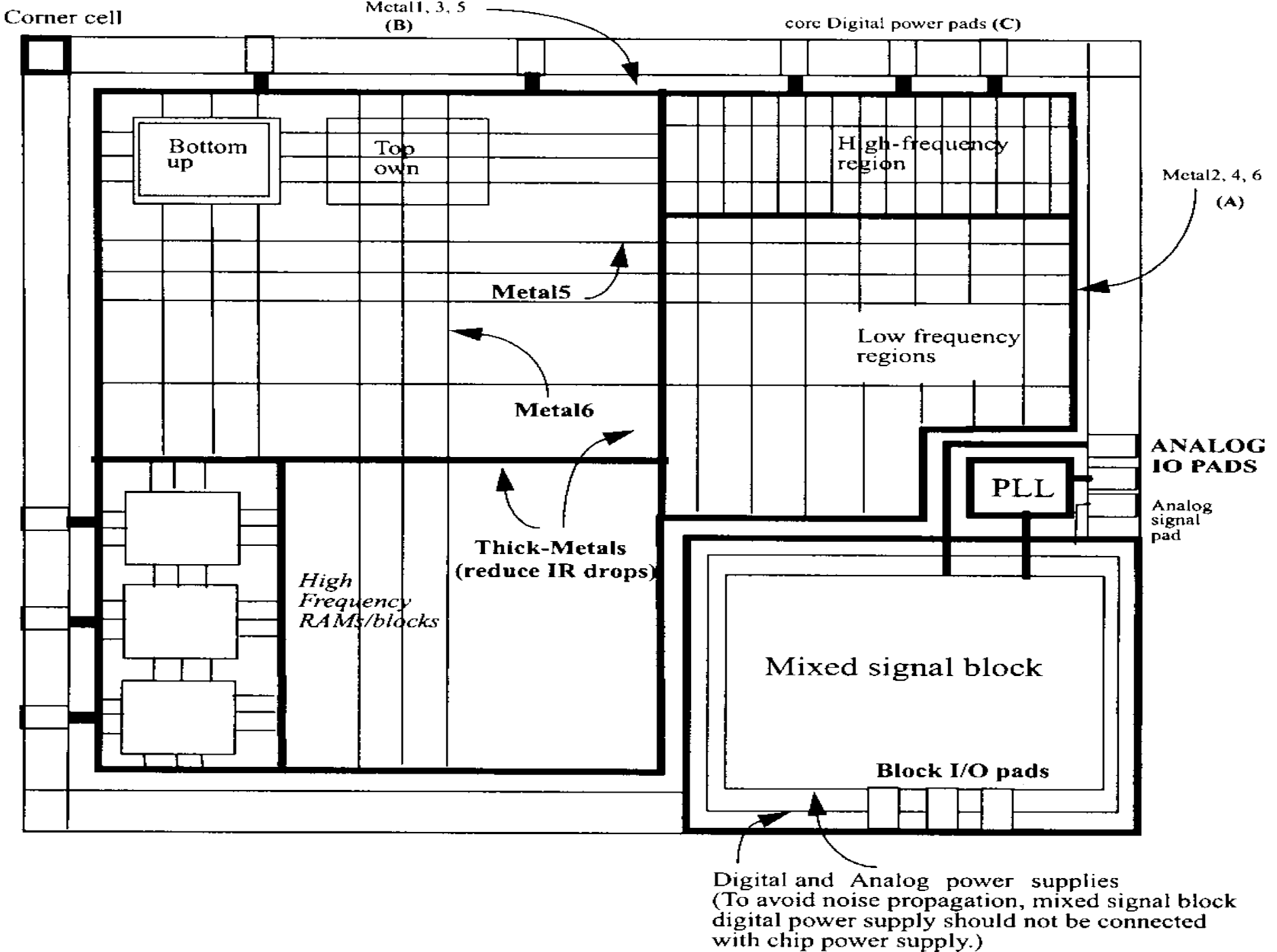
Top down power can be cut from First Encounter during partitioning, as shown in the following figure.



The advantage is that the power lines are automatically aligned. A drawback is that the block cannot be moved or adjusted to avoid internal or external routing congestion.

Full Chip Assembly

For some irregular power mesh structures (such as chips containing a mixture of high frequency, low frequency, and analog connections), you might need to use the Silicon Ensemble commands `ADD WIRE` and `DELETE WIRE` interactively to get the desired results. The following figure shows an example of power routing for a chip with components that have different power requirements.



Important

After power routing, you can save the design into a `golden_DEF` file that can be used repeatedly with Wroute to output GDSII for a tapeout.

- You can have a power ring of *Metal6* at location A. After adding I/O pad and stripe connections, you can use the `ADD WIRE Silicon Ensemble` command to add *Metal2* and *Metal4* overlapped with *Metal6* to deliver more current to the core area.
- You can have a power ring of *Metal5* at location B. After adding I/O pad and stripe connections, you can use the `ADD WIRE Silicon Ensemble` command to add *Metal1* and *Metal3* overlapped with *Metal5* to deliver more current to the core area.
- Core power pads, shown in location C, supply power for the core. High-frequency areas require more power pads.

- The required number of pads depends on the power consumption inside the core. You can calculate the number of pads you should use based on core power consumption and VDD/VSS power supply. For example, if the core consumes 1500 mW (P), and each VDD/VSS can supply 50 mA (based on a 3.3 V power supply), the following equations show the number of pads to use:

$$P = V \times I$$

$$1500 \text{ mW} = 3.3 \times I$$

$$I_{\text{total-core}} = P \div V$$

$$I_{\text{total-core}} = 1500 \div 3.3 = 455 \text{ mA}$$

Because you need 455 mA, and each pad supplies 50 mA, you can easily calculate the total number of I/O VDD/VSS pads you need:

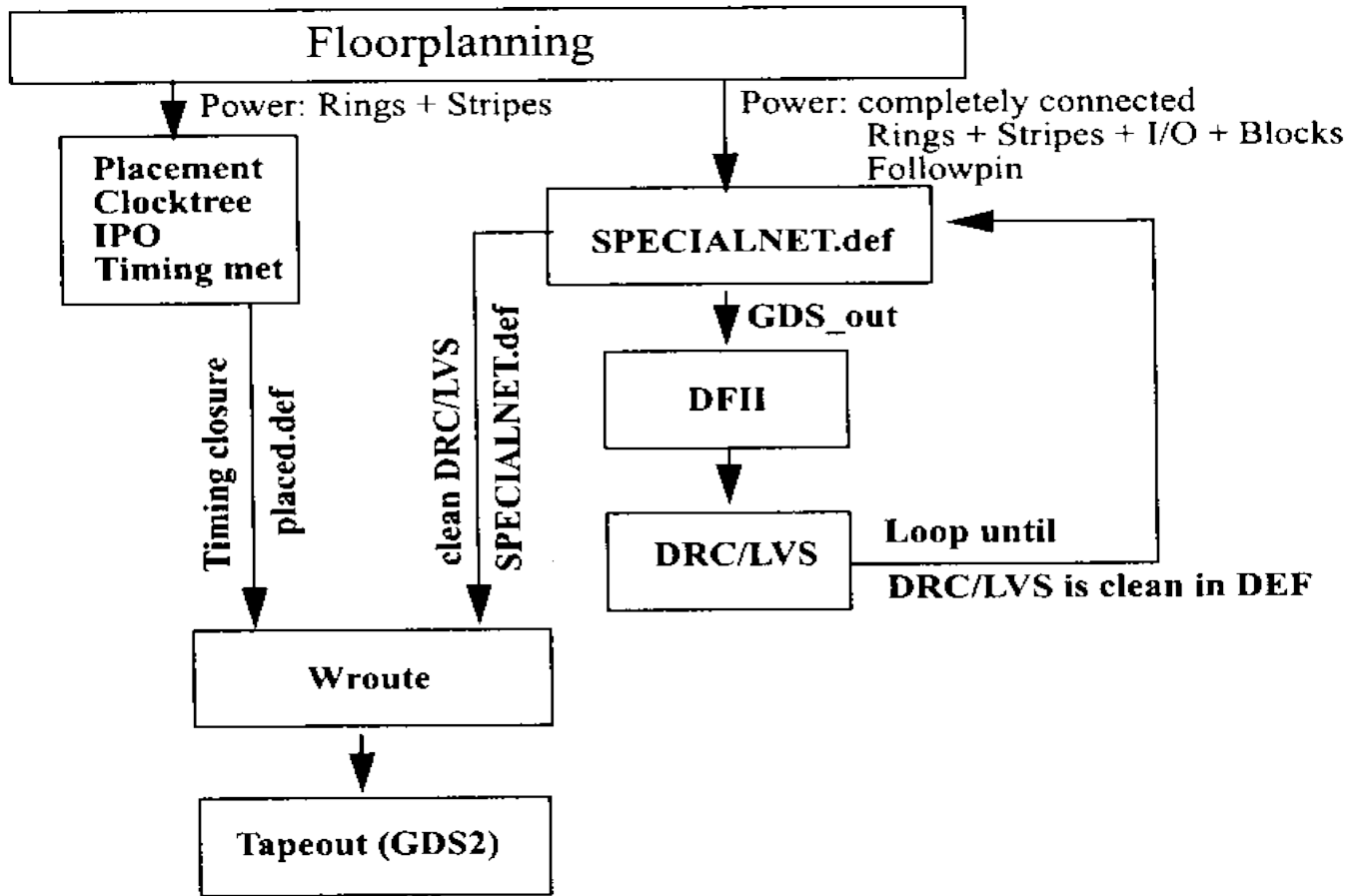
$$455 \text{ mA} / 50 \text{ mA} = 9.1 \approx 10$$

In this example, you need 10 VDD and 10 VSS I/O pads to be placed evenly around the chip periphery, as indicated in location C in the preceding figure.

To reduce internal ground bounce, you can add 50% more VSS I/O pads. In this example, you would add five more VSS I/O pads. Place these close to the high-frequency area.

Full Chip Power DRC/LVS Methodology

Power routing causes approximately 90% of the DRC and LVS block-level and chip-level violations. If a chip's power mesh is built without DRC/LVS violations, then you can assume that 90% of the entire chip is DRC/LVS clean. Therefore, when you build the power mesh at the chip level, if you want to run DRC/LVS on power alone, you can strip out all cells *except* power and ground. The following figure shows the flow for running DRC/LVS on power alone.



Use the following steps to run DRC/LVS on power alone:

1. Write out a DEF file with power I/O pad cells and SPECIALNETS sections only.
2. Read the DEF file into Silicon Ensemble to generate a chip-level power-only GDSII file.
3. Read the GDSII file into DFII, merge with the I/O DFII libraries, and write out a new GDSII file with a name such as `chip_pwr.gds`.
4. Run DRC and correct the violations in the Silicon Ensemble and DFII databases.
5. Create a Verilog[®] module file that contains only power I/O pads, as shown in the following example:

```
module chip_pwr_pads ( VDD, VSS );  
    inout  VDD, VSS;  
    PVDD1DGZ CVDD3 ( .VDD(VDD) );
```



```
PVDD2DGZ CVDD11 ( .VDD(VDD) );  
PVDD2DGZ RVDD2 ( );  
PVSS2DGZ RVSS6 ( );  
PVSS1DGZ CVSS11 ( .VSS(VSS) );  
PVSS4P AVSS18 ( );  
PVSS2P AVSS_331 ( );  
.....  
endmodule
```

6. Use the `chip_pwr_pads` Verilog netlist, created in the previous step, for LVS runs, and then correct the LVS violations in the Silicon Ensemble database.
7. Use DRC/LVS to correct the Silicon Ensemble database and write out a DEF file named `chip_pwr_clean.def`. This DEF file will be used for Wroute and for the final GDSII tapeout.