ANALYSIS OF POWER REQUIREMENTS INSIDE OF NMOS INTEGRATED CIRCUITS

Jeffrey Wilson

A thesis submitted to the faculty
of the Oregon Graduate Center
in partial fulfillment of the
requirements for the degree of
Master of Science
in
Computer Science.

March 1986

The thesis "Analysis of Power Requirements Inside NMOS Integrated Circuits" by Jeffrey Wilson has been examined and approved by the following examining committee:

Alan C. (Kit) Bradley, Thesis Research Advisor
Adjunct Assistant Professor
Dept. of Computer Science and Engineering

Richard B. Kieburtz
Professor and Chairman
Dept. of Computer Science and Engineering

Daniel Hammerstrom
Associate Professor
Dept. of Computer Science and Engineering

Shreekant S. Thakker
Assistant Professor
Dept. of Computer Science and Engineering

Abstract

Analysis of Power Requirements
Inside of NMOS Integrated Circuits

Jeffrey Wilson

Oregon Graduate Center, 1986

Supervising Professor: Alan C. Bradley

Software has been developed to analyze the power requirements of NMOS integrated circuits. Power usage is calculated for the entire chip. Current flow through each metal segment of VDD and GND lines is also calculated. The program, Pwranal, takes CIF format files as input and analyzes DC power requirements in the IC. Power estimates are worst case numbers. Power requirements may be less than the estimate but will not be more. Heuristics based on circuit topology are used to generate a more refined estimate of power needs. Initial values of nodes can be specified to provide an even more refined worst case power estimate. Current density is calculated and warning messages are displayed when it exceeds safe values. Maximum voltage drop in the VDD and GND lines is also calculated. An output summary is sent to the terminal. An optional CIF format output file can also be generated that contains detailed information about power distribution within the circuit.

Acknowledgements

The author would like to thank his thesis supervisor, Alan C. (Kit) Bradley, for suggesting the topic and for allowing the author a free hand in carrying it out.

The author would also like to thank his employer, Tektronix Inc., for providing financial support for this project.

Table of Contents

List of Figures

1. Introduction

Power is supplied to transistors on an IC using lines of metal. The minimum acceptable size of the power lines is determined by the property of electromigration. Metal in a power line will physically move if the current density in the metal line is too high. Electromigration can eventually cause the metal line to break, which causes the IC to fail. In current NMOS technology, metal traces must be a few microns in cross sectional area for each milliamp of current that they carry.

The current carrying capacity does not scale linearly as IC dimensions get smaller. As dimensions, including metal line sizes, are scaled down, the current needs of a particular IC remain constant. The current required per unit area increases linearly (Mead and Conway, 1980). Therefore, current flux through metal lines increases more than linearly with the scaling down of geometry sizes, since narrower metal lines must now carry more current. Therefore, future IC's with smaller geometries will face even more severe problems when sizing power lines than is faced today.

A software package, Pwranal, has been developed to analyze current flux for each line of Vdd and GND in a CIF format layout. The package is written in the "C" language for use on VAX computers running the UNIX operating system. It interfaces with the UC Berkeley CAD package. Pwranal will allow automated estimates of DC current flow in each individual metal line.

This thesis contains seven chapters. This chapter provides an introduction to the problem. The second examines related work. The third examines Pwranal from a user's perspective. The fourth chapter examines the internal structure of Pwranal. Chapter five discusses in depth some key algorithms used in Pwranal. Chapter six summarizes the results achieved by this project. Finally, chapter seven points out some possible directions for future investigations.

1.1. Electromigration

Electromigration is the mass transport of metal atoms in an IC. It is caused by momentum exchange between conducting electrons and the metal atoms. Metal moves toward the higher voltage potential. The process becomes more severe as current density increases, as temperature increases and as circuit dimensions get smaller (Glaser, 1982).

It is estimated that 26% of IC failures are caused by metalization problems (ibid). Failures caused by electromigration are especially serious since they generally only happen after an IC has been in service for some time. Many other causes of IC failures will be detected during component testing.

Guidelines for acceptable current densities in aluminum conductors range from 0.1 milliamps per square micron to 2.0 milliamps per square micron. The precise allowable density is still a subject of research (Black 1982). Aluminum is the most common metal used for IC metal lines.

Electromigration is caused mainly by DC current. Short current pulses cause less electromigration. NMOS circuits, unlike CMOS circuits, draw DC current when they are not switching. Therefore, electromigration occurs in NMOS circuits as long as power is on, even if the circuit is idle.

VLSI devices are now operating near their current density limit (Mead and Conway, 1980). Therefore, electromigration may be a serious limiting factor in attempts to develop NMOS circuits with smaller geometries.

1.2. Power Consumption in NMOS Transistors

Different representations of the same circuit are shown in figures 1.1 through 1.4.

Figure 1.1 is a logical representation of a simple circuit containing a

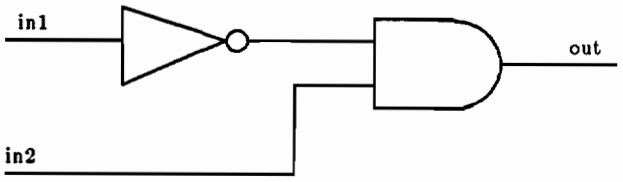

Fig. 1.1 Circuit 1 (logic representation)

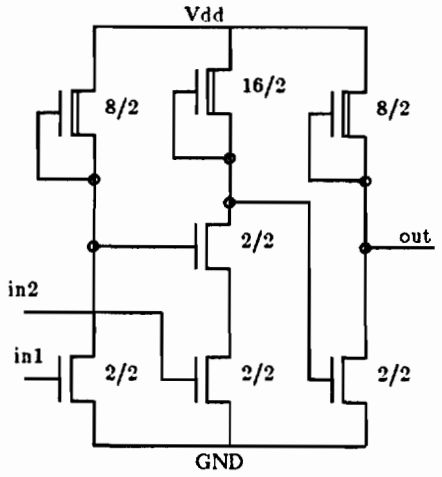

Fig. 1.2 Circuit 1 (MOS representation)

inverter and an AND gate. Figure 1.2 shows a NMOS implementation of the same circuit. The numbers next to each transistor in the form "a/b" denote the length (a) and width (b) of the transistor channel region in the layout. The length and width are specified as a ratio with no absolute scale. If the gate input is at logic ZERO, the channel resistance (resistance between source and drain) is infinite, i.e. an open circuit. Throughout this thesis, capital letters will be used to denote the logic values ZERO and ONE. If the gate input is at logic 1, the channel resistance is ChanRes*(a/b), where a and b are the channel dimensions and ChanRes is the channel resistivity. Channel resistivity is in units of ohms per square, which is a process dependent parameter.

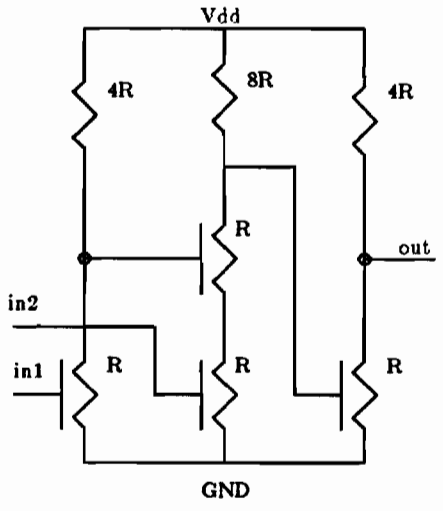

Fig. 1.3 Circuit 1 (switch representation)

Figure 1.3 shows the same circuit represented as switched resistors. Since depletion transistors are always ON (conducting current), they are shown as resistors. Enhancement transistors are shown as switches. These transistors will be resistors if the gate is 1 (transistor is ON) or open circuits if the gate is 0 (transistor is OFF). Resistivity is in units of ohms per square. Resistance is calculated by multiplying the number of squares by the channel sheet resistivity (ChanRes).

Figure 1.4 shows a CIF plot of a layout that implements the circuit of figures 1.1 through 1.3. The shape of the channels correspond to the ratios of figures 1.2 and 1.3.

Resistance of channels and other IC materials is based on the geometry that the current must flow through and the resistivity (ohms per square) of the material. If the rectangular channel is considered to be a structure through which current flows in a particular direction, its resistance is the number of "squares" that the current must flow through times the resistivity of the material. For example, in leftmost depletion transistor in figure 1.4, the current must flow through a vertical area of 4 units long by 1 wide, for a total of 4 squares. The total resistance of this channel is 4 times the channel

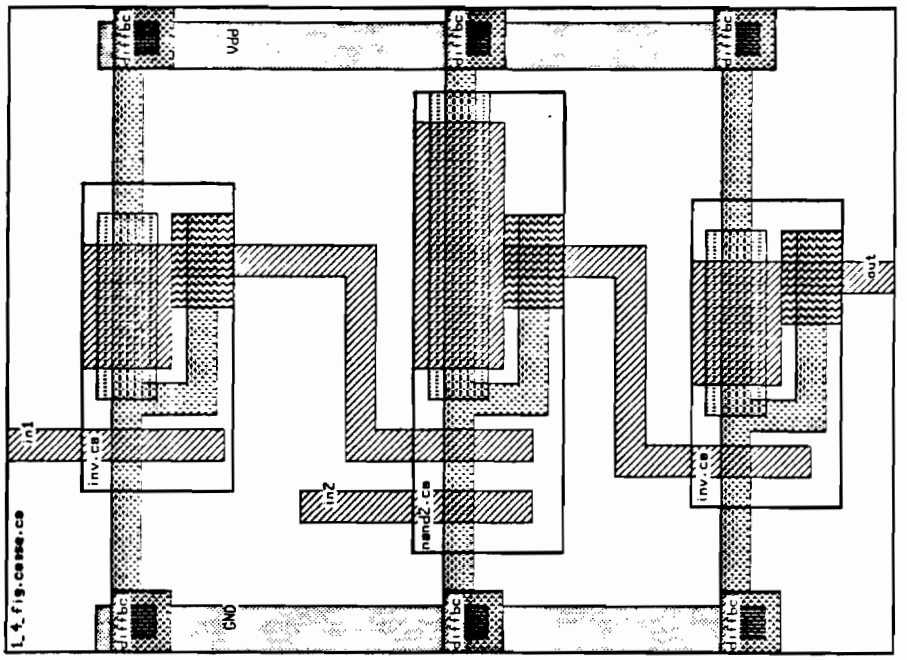

Fig. 1.4 Circuit 1 (CIF representation)

resistivity of the material. Channel resistances in current IC technologies are typically a few tens of kiloohms.

2. Other Approaches to the Problem

IC designers are faced with two conflicting requirements. One is the need for small chip area. The other is the need to design reliable circuits that avoid problems with electromigration. Solving the electromigration problem involves making power distribution busses wide enough. This takes up significant portions of the chip area.

There are two ways to approach the problem. One is to develop and use analysis tools such as Pwranal to characterize the circuit and determine if the power busses have adequate width. The other is to improve IC fabrication techniques so that circuits have fewer problems with electromigration. These two approaches can complement each other.

2.1. Other Approaches Using Circuit Analysis

The only tool currently available for analyzing power requirements is powest, part of the Berkeley CAD package. It provides a number for total power requirements of the circuit. However, it doesn't provide any information on power distribution within the circuit. It also doesn't take into account either circuit topology or expected usage of the circuit.

Pwranal provides the following advantages over powest:

1. Power needs are determined for each metal segment of the power busses, in addition to the total power needs of the IC.
2. Circuit topology is taken into account when calculating power needs. In particular, complementary outputs are detected. Power requirements of complementary output configurations are adjusted since only one output in such a configuration can be drawing power at a particular time.
3. Initial values can be specified for circuit nodes. This allows for a more refined estimate of power needs, especially for circuits with

a regular structure. Pwranal will not include transistors that are known from initial node values to be OFF in power calculations. Since electromigration is less of a problem for short current bursts than with steady DC current, the ignoring of transistors that are off most of the time is justified.

2.2. Approaches to Electromigration Involving IC Fabrication

Several methods have been proposed to limit electromigration problems by improving IC fabrication techniques. One author (Mead and Conway 1980) feels that efforts to make metal lines thicker will be made. Thicker metal lines increase the area that current can flow through, thereby limiting electromigration problems.

Another possibility is to use a metal other than aluminum for power busses that can handle a higher current flux without having problems with electromigration. Tungsten is a possibility. However, tungsten also has a higher resistivity than aluminum. This will cause greater voltage drop along power busses made of tungsten than is currently the case for aluminum power busses. This voltage drop will cause a new set of problems with noise margins.

Another proposal (Song and Glasser 1984) involves a new method of distributing power in the IC. They propose using several layers of metalization. The bottom level would be used for signal interconnection. Upper layers would be used for power distribution. These upper layers contain increasingly wide and thick metal lines, each distributing power to the layer below. This method has the advantage of providing a more even distribution of power to all areas of the circuit. Song and Glasser provide calculations outlining tradeoffs between the number of layers, power distribution and noise margin. Good results are achieved with only a few extra layers. On the negative side, this method requires more layers and therefore a more complicated fabrication process.

Some future processes may work with VDD set to lower than 5 volts. This reduces electromigration problems somewhat, since lower voltages imply less

power dissipation and less current flux. VDD can only be scaled down to a limited extent, since noise margins decrease as VDD is lowered. Most projections have such scalings of voltages being small compared to anticipated feature size scaling. Furthermore, voltages scaled down linearly reduce current flux (and thereby electromigration problems) linearly, while scaling down of geometry by a factor of X increases current flux by a factor larger than X. It appears that the scaling down of VDD only mitigates the electromigration problem slightly, and is not a solution to the problem.

Such process related approaches can work in tandem with analysis tools such as Pwranal. Pwranal can characterize the power needs of a circuit and determine when a circuit is pushing the limits of metalization technology. Improvements in process technology can extend what those technology limits are.

3. Pwranal - A User's View

This chapter examines the functionality of Pwranal from a user's view. The internals of Pwranal are discussed in the next two chapters. The first section provides an outline of what the program does at an abstract level. The next section examines how circuit structures with complementary outputs are used to reduce total power estimates. Section 3 outlines how initialization can provide a better fit when it is known a priori what value a node will be. Assumptions made by the program are outlined in section 4. Section 5 discusses some limitations of the program. Finally, the inputs accepted by Pwranal and the outputs generated by Pwranal are described in sections 6 and 7.

3.1. What the Program Does

Pwranal analyzes in detail the power requirements of an NMOS IC. Input is a layout in CIF format. This format is described in (Mead and Conway 1980). Technology dependent parameters are controlled by invocation options. These parameters are channel resistivity, metal resistivity and permissible current flux through metal. The analysis of power requirements is done at the level of individual VDD and GND metal lines, thus providing more accuracy than a global analysis of the entire IC does by merely counting transistors. In addition, the program generates information about maximum voltage drop in internal power lines.

Pwranal generates a worst case estimate of power needs, not a best fit estimate. Circuit power requirements will be lower than the estimate Pwranal provides. Power usage will vary with the type of inputs applied to the running circuit. Pwranal provides a safe estimate of power consumption to use when designing an NMOS IC.

To summarize Pwranal's analysis procedure, the program constructs a internal representation of layout of the VDD and GND lines, called a *metal tree* hereafter. The metal tree structure corresponds to current flow in the

metal. Metal segments connected to the VDD and GND lines via a polysilicon or diffusion jumper are also included in the metal tree. Contact cuts corresponding to the intersection of transistor sources and drains to power lines are assigned to their appropriate position on the metal tree.

Starting at each contact cut, the transistor network is traversed to discover how much power is drawn from that point in the metal tree. All transistors are assumed to be ON (i.e. drawing power when the transistor gate is at logic ONE) unless it is known to be OFF (not drawing power when the transistor gate is at logic ZERO) because of initialization. The power drain of a particular transistor can also be reduced if the transistor is part of a complementary output structure. The metal tree is then traversed to discover both total power needs and the current that needs to be supplied by each VDD and GND metal segment. These needs are based on the power requirements of the contact cuts attached to the metal segment. Output is summarized and printed to the user's terminal. In addition, output may be generated in a graphical form using the CIF format.

There are two ways that Pwranal can reduce the estimated power needs of a transistor. One is if the transistor is part of a complementary output circuit configuration. The other is if the user enters node initialization values that force the transistor to be OFF.

3.2. Complementary Outputs

The program identifies complementary outputs when traversing the network of transistors. In NMOS circuits, only one portion of a complementary output configuration can be drawing power at one time. Such configurations arise often in NMOS circuits. Calculated power requirements are adjusted to reflect this. Superbuffer configurations that are in a complementary output configuration are handled properly in addition to transistors configured as standard logic gates.

The simplest circuit with a complementary output configuration is the chain of two inverters shown in figure 3.1. For any input, one of the outputs

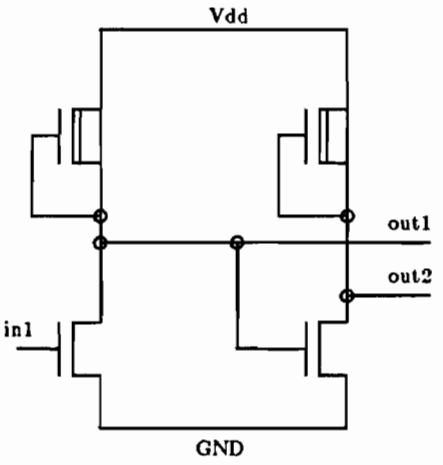

Fig. 3.1 Example complementary output configuration.

(out1 or out2) will be a ONE while the other will be ZERO. Output out1 will be ONE if in1 is ZERO, while out2 will be ONE if in1 is ONE. One of the two inverters will always be drawing no power, since the input to the gate of the enhancement transistor is ZERO. Total power needs for the two inverters can be reduced 50% from the power that would be drained by the uncoupled inverters.

3.3. Initialization

The user can specify the initial values of nodes in the circuit. The program uses this information when calculating power needs, since an OFF NMOS transistor draws no power. Nodes with initial values specified are set to those values. Initial values are then propagated throughout the circuit. Other node values are set to ZERO or ONE if it can be concluded that the node value is defined by the specified node initialization. Transistors that are OFF, based on initialization information, are marked so that this information can be used when power needs are calculated.

This is useful in deriving a more accurate, and lower, estimate of power needs when it is known that particular nodes will usually not be driven. Memory enable lines are an example of a type of node that is usually at one value.

In practice, providing initial values to key nodes can significantly reduce estimated power. Doing so is often justified if it is known that a node will be almost always be at one state, since electromigration is most caused primarily by DC power. Electromigration is a less serious problem for short bursts of AC current. If a transistor is known to be off most of the time during circuit operation, it can be ignored for the purposes of electromigration calculations.

3.4. Assumptions Made By Pwranal

Pwranal assumes that VDD is 5 volts. Pwranal also assumes that only one layer of metal is used for power and ground routing.

The circuit is modeled as a network of switched resistors. Transistors that are ON are modeled as a resistor with value dependent on the channel sheet resistivity and the length and width of the transistor. The sheet resistivity is set by an invocation option that defaults to 10K ohms/square. Transistors that are known to be OFF are modeled as open circuits.

Pwranal only calculates DC power consumption. For NMOS circuits, this is the dominant mode of power consumption. Such an assumption is not valid for CMOS circuits. The possibilities for handling CMOS circuits are discussed in the final section.

All transistors are assumed to be ON (i.e. a resistor) unless the program is able to determine that the transistor is OFF. The only way that this can happen is if a node is initialized by the user to a value that forces the transistor to be OFF. In addition, the power requirements of a transistor may be reduced by as much as 50% if it is part of a complementary output configuration.

Pwranal provides a worst case estimate of power consumption, not a "best

fit" estimate. The results obtained from Pwranal will be, if anything, pessimistic and therefore lead to conservative designs. This provides a safe number to use when considering the possibility of electromigration and when characterizing power consumption.

Transistors that have a path to VDD but not to GND are assumed to have an external connection to GND. This allows Pwranal to guarantee a worst case estimate of power needs. Initialization can be used to set these transistors inactive. Calculated power drain from such transistors can be considerable and can cause inaccuracies.

Pwranal does not count pass transistors in power calculations. These generally cause trivial power usage. A pass transistor is defined as one that has all paths from its source or drain to either VDD or GND passing through the gate of another transistor.

3.5. Limitations of Pwranal

There are some limitations in the types of circuits that Pwranal can effectively analyze. In addition, Pwranal makes approximations for circuits containing cycles in metal lines and in the transistor network. These limitations are:

1. Pwranal will analyze only NMOS circuits. It will not handle CMOS circuit.
2. Pads are not included in power calculations. Pads generally contain transistors with irregular geometries. These transistors behave like analog circuits and can't be extracted with an extractor. Current is usually supplied to these pads with large metal bands around the perimeter of the circuit. The current requirements of pads must be analyzed manually or with tools other than Pwranal. Pwranal only analyzes the digital portion of the integrated circuit.

3. Multiple power pins are not handled. If more than one VDD pin or GND pin is present, Pwranal processes the circuit as if there were only one. The result is that Pwranal will give an overly conservative view of power distribution. Pwranal will assume that all power is sourced (VDD) or sunk (GND) through one pin, instead of several. The calculated current flow through some metal segments will be too high.
4. Cycles in metal power lines are broken by the program to form an acyclic structure. An example of this is shown in figure 3.2. This approximation will cause calculated current flow in some metal segments to be too high.
5. Cycles in transistor networks are also broken by deleting one transistor from the circuit. An example of cycles is shown in figure 3.3. The program chooses an arbitrary transistor to delete. This is similar to approximation made by the RNL simulator (Termin 1982).

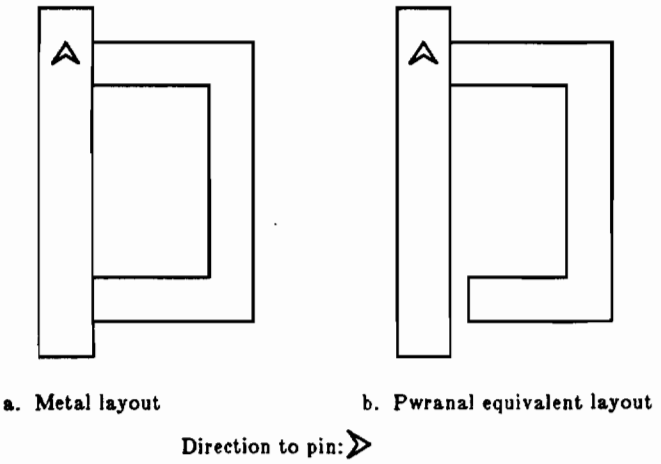

Fig. 3.2 Pwranal approximation for cycles in metal.

None of these approximations should have an appreciable impact on estimates calculated by Pwranal.

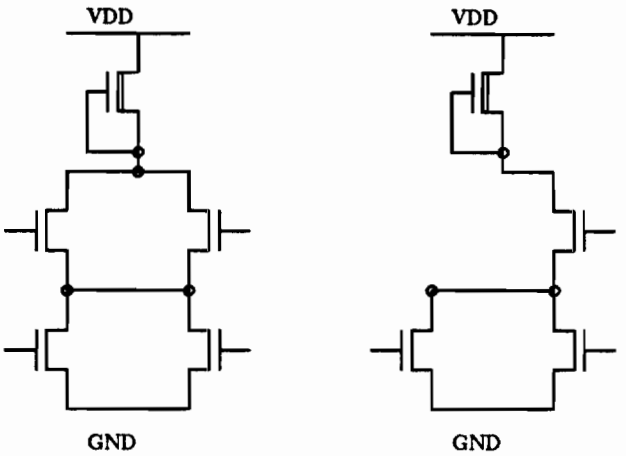

Fig. 3.3 Pwranal approximation for cycles in transistor network.

3.6. Inputs to Pwranal

Pwranal accepts the following inputs:

1. CIF description of the circuit (basename.cif)
2. Invocation options (optional). Option "-i filename" denotes an file containing initial node values. Option "-c" will generate a CIF representation of power busses; at various points the power drained will be displayed as well as the summary of power drain. This output will be placed in file "basename.pwr.cif". Invocation options are also used to specify values for channel resistivity ("-r" number), metal resistivity ("-m" number) and maximum allowable current flux through metal ("-f" number). These parameters are technology dependent. Defaults are used for parameters that are not explicitly specified.
3. Extracted view of the circuit (optional). This is contained in a file basename.pext and is generated by Pwranal from the CIF layout if it is not present. If the file is present from a previous run it is

reused.

Four labels in the CIF file are used by Pwranal. They must be present. These are VDD, GND, VDD_pin# and GND_pin#. Case of the labels is insignificant. For example, VDD and Vdd are both accepted.

Labels VDD_pin# and GND_pin# denote where the power pins are located. They will be at the root of the metal tree. The suffix "#" denotes to Pwranal that the label is a local label, not a global label. This avoids conflict with the labels VDD and GND, which will be electrically connected to VDD_pin and GND_pin.

The ".pext" file is generated by Pwranal from the input ".cif" file. The file contains transistors extracted from the layout. This file differs from a normal extraction in that the physical location of contact cuts is carried in the extraction.

The initialization file contains initial values of nodes. An example of such a file is below.

```
#
# Example initialization records
#
# Node node1 is initialized to logic 0
# Node enable is initialized to logic 1
#
node1 = 0
enable = 1
```

Initial values are denoted by entering the node name, followed by an optional "=", followed by the initial value. The value must be either 0 or 1. Each initialization must be on its own line. As many initializations as desired can be placed in the file. Comments can be included in the file by starting a line with the character "#".

3.7. Outputs of Pwranal

Pwranal provides two outputs. A summary of results is sent to the user's terminal. An optional graphical output showing distribution of power supplied on various points of the metal lines is also available using the "-c" option.

An example of a summary output of Pwranal is in figure 3.4. Line 1 contains the title and name of the input file. Lines 2 through 5 contain the technology dependent parameters that are used in Pwranal. The default values may be changed by setting an invocation option. Lines 5 and 6 denote how many polysilicon and diffusion jumpers connect VDD and GND metal.

Lines 7, 8 and 9 contain information about circuit initialization. If there is no "-i" option present line 7 will contain the words "none specified" and lines 8 and 9 will not be present. If the "-i" option is specified, line 7 contains the name of the file containing initial values. Line 8 contains the number of initial values in the file. Line 9 contains the total number of nodes initialized. This count includes both the nodes initialized directly from the file and also nodes initialized by propagating initial values throughout the circuit. Line 10 contains the number of complementary output structures that were identified and adjusted for power consumption.

Lines 11 through 14 display information about circuit power consumption calculated by Pwranal. Each line displays information about both VDD and GND. Line 11 contains the total estimated power usage. This number includes reductions in calculated power usage due to initializations and complementary output configurations. Line 12 contains the portion of line 11 that results from transistors without a path from their source and drain between both VDD and GND. These transistors are assumed to have the source or drain farthest from the power line connected to the other power line. Line 11 minus line 12 in the VDD column will equal line 11 minus line 12 in the GND column.

Lines 13 and 14 display information about raw power consumption. These lines contain information similar to that in lines 11 and 12 except that power consumption are calculated differently. Here, no reduction in power consumption is made for either initialization or complementary outputs. All transistors

1.	Pwranal Summary -- Cells/LLSCsubpwr.cif		
2.	Channel resistance (ohms/square)	10000.000	
3.	Metal resistance (ohms/square)	0.030	
4.	Max metal curr flux (mA/micron)	1.000	
5.	Vdd Poly/diffusion jumpers	0	
6.	Gnd Poly/diffusion jumpers	0	
7.	Initial values source file	LLSCsubpwr.init	
8.	Nodes with initial values	4	
9.	Total initialized nodes	11	
10.	Complementary output structures	25	
		Vdd	Gnd
11.	Total estimated power needs	24.136mW	18.011mW
12.	pullup/pulldown portion	6.125mW	0.000mW
13.	Raw DC power, no reductions	31.723mW	23.098mW
14.	pullup/pulldown portion	8.625mW	0.000mW
15.	Max metal current flux (based on estimated power needs)	0.702mA/micron	0.548mA/micron
16.	Location of max current flux		
17.	(absolute)	(43600, 0)	(56400, 12000)
18.	(normalized [0.0-1.0])	(0.462, 0.000)	(0.598, 0.363)
19.	Max. voltage drop (from pin)	0.017v	0.008v

Fig. 3.4 Sample pwranal output.

are assumed to be ON.

Line 15 contains the maximum current flux measured in the circuit, in milliamps per micron, for both the VDD and GND lines. This is calculated using

initializations and complementary output reductions; i.e. using the same assumptions as line 11. The numbers on line 15 must be compared with the maximum allowable flux (line 4) to see if the circuit passes. Pwranal will provide a warning at the last line of the summary if both the VDD and GND portions of line 15 are not less than line 4.

Lines 16 through 18 contain information about the location on the power busses where the maximum current flux occurs. Location is provided in two coordinates. Line 17 contains the CIF coordinates, based on the input CIF file. Line 18 contains the same point in coordinates normalized to the segment [0.0, 1.0], with the bottom and left edges of the circuit at 0.0 and the top and right edges of the circuit normalized to 1.0.

Finally, line 19 contains the maximum voltage drop in the circuit. This is a function of both the topology of the metal, metal resistance and the current needs of the circuit. Once again, power reductions from initializations and complementary outputs are included when calculating this number.

In addition, a graphical output showing power drain at various points in the power bus can be generated using the "-c" option. The "-c" option causes a CIF file to be generated with the VDD and GND lines and labels denoting power drain. This output is placed in the file "basename.pwr.cif", where basename.cif contains the input circuit. Each label denotes the power, in milliwatts, that the subtree of metal with root at that point must supply to the circuit. Direction to the power pin is also displayed. A typical label is:

Pwr_E=1.2mW

This label denotes that at the point on the metal line where it is located, 1.2mW of power is required in the subtree has that point as a root. In additions, the suffix "E" after the "Pwr" in the label denotes that the pin can be reached by tracing the metal line starting at that point in an east (E) direction. The suffix will always have a value of "N" (north), "S" (south), "E" (east) or "W" (west).

Areas of metal with current flux that is too high (as determined by line 4 in the summary) are highlighted in the output CIF plot. The file generated with

the "-c" option can be displayed using other CAD tools such as "cifplot", "cif2ca" and "caesar".

4. Internal Structure of Pwranal

This chapter examines the internal architecture of Pwranal. The major modules of the program are described. These modules correspond to both the steps that are used to analyze circuits and the software modules that implement Pwranal. Some key algorithms and data structures are discussed in depth in the next chapter.

Extraction, the process of determining the transistors present in a layout, is described in the first section. Pwranal constructs a representation of the metal power lines from the layout. This is discussed in the section 2. In section 3, the process of building and representing transistors and nodes is outlined. The circuit is then initialized using initial node values provided by the user, if necessary. This is discussed in section 4. In section 5, the process of identifying complementary output structures is examined. Section 6 discusses the process of calculating power consumption from the internal representation of the circuit. Finally, section 7 outlines the process of outputting results.

4.1. Extraction

Pwranal calls a program named pwrmextra in a subshell to extract transistors from the CIF layout. The program pwrmextra is based on the program mextra that is part of the UC Berkeley CAD tool set. Mextra extracts transistors from a layout. It is usually used to convert a layout into a netlist that is then input into a simulator. Mextra was extended by the author to meet the needs of Pwranal.

Pwrmextra extracts transistors from a layout, like mextra. Two additional pieces of information are generated. Physical location of contact cuts is included with the output. Also, information about polysilicon and diffusion jumpers that connect VDD or GND metal lines together is generated. Other than this, pwrmextra it is functionally identical to mextra.

The output of this program is placed in the file basename.pext. There are

three sections to the output. First is a header record. The second section denotes the transistors and has one line per transistor. The third denotes polysilicon or diffusion jumpers between power busses; those are denoted in the file by a "j" in the first character of the line.

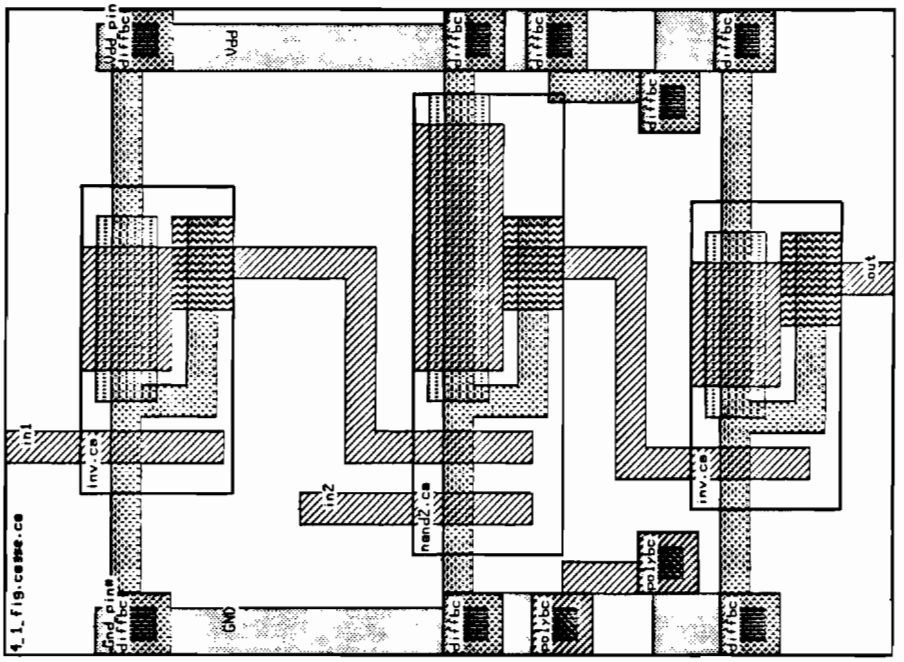

Fig. 4.1 CIF plot of basic circuit with jumper

Examples of a header record, seven transistor records and two jumper records are below. This was generated by Pwranal from the circuit in figure 4.1. This circuit is logically identical to the circuit in figure 1.4. As is sometimes done in large circuits, polysilicon and diffusion jumpers have been added to the power lines.

```
| units: 100    tech: nmos
d 9  VDD_6000_7800_22_-1_23 9 32 4
d 11  VDD_1600_7800_19_-1_23 11 16 4
d out  CUT_9600_7800_18_-1_20 out 16 4
e 11 9 8 4 4
e in1 11  GND_1600_200_4_-1_1 4 4
e 9 out  CUT_9700_200_6_-1_2 4 4
e in2 8  GND_6000_200_5_-1_1 4 4
j  GND_7000_200_-1_3_1  CUT_8400_1200_-1_3_2 null  0 0
j  VDD_7100_7800_21_-1_23  CUT_8600_7200_21_-1_20 null  0 0
```

The first character in the transistor section denotes the record type, and is either "e", for enhancement transistor or "d", for depletion transistor. The next three words denote the nodes that the gate, source, and drain, respectively, are connected to. Information in GND, VDD and contact cut nodes (denoted with a prefix of CUT above) have been expanded to contain more information than would be present in a normal extraction. Finally, the height and width of the transistors is present.

The GND, VDD and contact cut node labels carry information about the contact cut separated by "_". First is the X location of the contact cut (6000 for the first VDD label), Y location (7800 for the first VDD label), diffusion node number (-1 if the contact cut is in polysilicon), polysilicon node number (-1 if the contact cut is with diffusion), and finally the metal node number. The first VDD label has a diffusion node number of 22, a polysilicon node number of -1 (indicating that the contact cut isn't connected to polysilicon) and a metal node number of 23. This information has been added to the output of the extractor for Pwranal. In mextra, these labels would have names "vdd" and "gnd", with no additional information. The diffusion, polysilicon and metal node numbers are numbers that are generated internally by pwrmextra.

The two records starting with "j" denote the two power line jumpers in the layout. The "j" and the first two nodes are the only fields of the record that are used. The other fields are included to have a format consistent with the transistor records. The two nodes denote locations that are connected together. Prefix "CUT" is used for contact cuts that are not directly connected by metal

to the power pin label. Pwranal will use this information later to build an extension to the VDD and GND metal tree.

4.2. Construction of the Metal Tree

The file basename.cif is scanned by Pwranal, identifying all metal rectangles. These are used to generate an internal representation of the power bus topology. A library function from the UC Berkeley CAD tool set is used to parse the CIF input.

A tree-like internal representation of the metal rectangles is built from this information, with the location of labels "VDD_pin#" and "GND_pin#" at each root. This tree structure allows current flow to be represented and traced. Current flowing through each node equals the sum of current to all contact cuts connected to the metal node and the current requirements of the subtree with the metal node as a root. Information about contact cuts that are attached to the metal tree is added later after the ".pext" file is read in. See sections 5.1 and 5.2 for more information on the construction of the metal tree.

4.3. Construction of Transistor and Node Data Structures

The ".pext" file containing information about transistors is read to generate electrical information about the circuit. Data structures are constructed to represent each transistor and node. Each node data structure points to all transistors connected to it. Each transistor data structure points to the nodes to which its source, drain and gate is connected.

This step also resolves polysilicon and diffusion jumpers that extend power busses. Metal lines that are so extended, as denoted by ".pext" records starting with a "j", are connected to the metal tree at the appropriate point. The metal tree is then extended to contain metal rectangles connected to the power pins by polysilicon or diffusion jumpers. A transitive closure of jumpers is performed to ensure that all metal connected to VDD or GND using several

jumpers is included in the metal tree. For the purposes of future voltage drop calculations, it is assumed that the metal nodes connected by jumpers have no voltage drop across them.

Finally, contact cuts that are connected to VDD and GND are linked with the proper point on the VDD and GND metal tree. This allows the current needs of these transistors to be assigned to the proper point in the metal layout.

4.4. Circuit Initialization

It is assumed initially that the logic values of all nodes in the circuit are unknown and that all transistors are ON, i.e. conducting current. The user can specify a file containing initial node values using the "-i" option. The initial values in the file set node values to ZERO or ONE. Initial values are propagated to other nodes in the circuit that have initial values that can be deduced from initializations specified in the file.

Initialized node values reduce power estimates by allowing the program to conclude that some transistors are OFF, and drawing no power. This reduction can sometimes be considerable.

The initialization algorithm works as follows. All nodes are set to their specified value. Once this is done, each node value is propagated to other nodes in the circuit. This process is described in detail below. At a top level, propagation is done first in the forward direction by examining transistor gates connected to the initialized node and then propagating initial values to transistor sources and drains. Propagation is then done in the backward direction by examining transistor sources and drains connected to the initialized node and propagating values to the transistors gate.

It is possible to have different node initializations give conflicting values when the initializations are propagated. In such cases, the node keeps the value of the first initialization and the propagation of the second initialization stops. This makes initializations that are specified first in the "-i" file have precedence over those that occur later.

The following algorithm for forward propagation is done for each gate connected to the initial node. The transistor is initialized to ON or OFF, depending on the node value. The program then initializes the source and drain of the transistor, if possible, by testing for the following cases in the order specified:

1. If the transistor is ON and the transistor source or drain is connected to GND, the opposite source or drain terminal is initialized to logic ZERO.
2. If the transistor is OFF and there is a pullup transistor connected to a source or drain terminal and there is no other transistor connected to that node, the pullup node is initialized to logic ONE. Note that if there are more than one enhancement transistor connected to the node (as in a NOR gate), nothing definitive can be said about the pullup node.
3. If the transistor is ON and either the source or drain is initialized to logic ZERO or ONE, the opposite terminal is initialized to the same value.

If a node connected to the source or drain is initialized during propagation, that initial value is propagated as well.

The following algorithm for backward propagation is then done for each transistor with a source or drain connected to the node being initialized. This algorithm initializes the node at the opposite terminal or gate of the transistor, if possible. The following cases are tested for each transistor with a source or drain connected to the node:

1. If the node is initialized to logic ZERO and the opposite terminal is GND and there is only one enhancement transistor between the node and GND, the gate of the transistor is initialized to logic ONE.
2. If the node is initialized to 1 and the node is pulled up to VDD and there is exactly one enhancement transistor connected to the

node and the opposite terminal of the enhancement transistor is connected to GND, the gate of the enhancement transistor is initialized to 0.

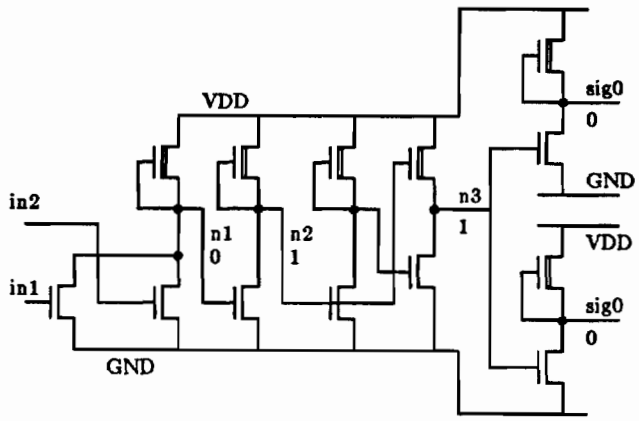

Fig. 4.2 Results of Initialization n2 = 1.

An example of initialization is in figure 4.2. Assume that n2 has been initialized to logic ONE. Then n3 could be initialized to ONE and both sig0 outputs could be initialized to ZERO. In addition, the initial value of n2 could be propagated backwards to set n1 to logic zero. However, nothing definitive could be said about the value of either in1 or in2. Either could be ONE or both could be ONE and still have n1 be ZERO.

4.5. Circuit Preparation for Power Requirements Calculation

Some analysis is now done on the circuit to give a lower, but still worst case, number on power requirements. This analysis is based on the circuit topology.

All inverter and superbuffer chains are identified. In such constructs, only

alternating inverters and superbuffers can have output ZERO, and therefore be drawing current, at one time.

Each transistor in the chain will be marked to have its power needs reduced by an appropriate factor between 1.0 and 0.5. This factor will be calculated by first measuring the power drain in the chain for the case of the input to the base of the chain set to ZERO or ONE. The factor is the ratio of the maximum power usage in these cases compared with the power usage that would occur if all transistors in the chain were ON simultaneously. If an inverter chain has had its values set by node initialization, this will not be done since the initialization process has taken care of power reduction. These calculations are discussed in depth in section 5.5.

4.6. Calculations of Power Needs and Voltage Drop

A depth first traversal of the VDD metal tree is done to determine the power needs of the circuit. The power needs of the current flowing from each contact cut at a given place in the tree is calculated. This is done by tracing the network of transistor sources and drains starting at the contact cut. Transistors that are ON for the purposes of Pwranal (i.e. those that have not been forced to OFF by initialization) are modeled as resistors and combined in serial and parallel combinations. Transistors that are OFF are modeled as open circuits. Multipliers to power requirements of transistors in complementary output configurations discussed in the last section are taken into account.

When the traversal reaches a GND node, the calculated power needs are stored at the appropriate place in both the VDD and GND metal trees. In a NOR gate configuration, the power need will be stored at the contact cut farthest from the GND pin. If a GND node is not reached, it is assumed that the circuit is connected to GND via an external connection. This will be the case with circuit constructs such as pullups that are used for precharging dynamic busses. The algorithm used to calculate current needs is discussed in depth in the section 5.6.

The metal tree is constructed such that all current is sourced (for VDD) and

sunk (for GND) at the root of the tree. For each node in the metal tree, total power requirements are the sum of power drawn by contact cuts associated with that metal node plus the power needs of all child branches of the metal tree.

Another pass through the metal tree is made to calculate voltage drop based on the resistance of the metal lines and the amount of current being drawn at each point.

4.7. Generation of Output

Pwranal then generates output information based on the previous power calculations. At this point, global circuit power needs are known in the root of the two metal trees. If a CIF plot of output is desired, the metal tree is traversed to generate the graphical shape of the tree and labels denoting power needs at points in the tree. This information is written to the file "basename.pwr.cif" in the CIF format.

5. Algorithms and Data Representation

This chapter examines in more depth some of the key algorithms and data representations used in Pwranal. The first three subsections examine the representation and construction of metal trees and methods of searching for metal rectangles, and methods for constructing metal trees. Representation of nodes and transistors is examined. Handling of complementary output configurations and transistor network traversal is then discussed.

5.1. Representation of Metal Trees

Pwranal uses a data structure called a "metal tree" to represent the power lines of a circuit. Two metal trees are constructed, one to represent VDD lines and another for GND lines. These also correspond to the direction of current flow in metal. The VDD pin or GND pin are each at the root of their respective metal trees. The location of these pins are specified by the user. Each node of the metal tree contains the following information:

1. Direction of current flow. One of east, west, north or south.
2. Coordinates of the metal rectangle that the metal node corresponds to.
3. Pointer to the parent metal node.
4. Pointer to the child metal node or nodes.
5. Pointer to a list of transistors that are connected via a contact cut to the metal node. Power may be drawn from these contact cuts.
6. Power drawn from the above contact cuts.
7. Power drawn by the contact cuts in all metal nodes in the subtree with a root of the current metal node.

Except for the last three items, all information is generated when the metal tree is constructed. The last three items are generated at later steps.

5.2. Construction of the VDD and GND Metal Trees

The metal trees are constructed from the input CIF file. There is one tree each for VDD and GND. Each tree consists of metal node data structures organized into an unbalanced tree format. Each metal node corresponds to a metal rectangle in the power bus and may have several of child nodes. The metal tree represents both the metal that carries current to the circuit and the flow of the current.

The first step is to read the CIF file and place the metal rectangles in memory. These rectangles in the CIF file are in no particular order and overlap in a random fashion. The rectangles are then organized into a data structure that allows for the rapid determination of intersecting rectangles. This data structure, a two dimensional tree of groups, is discussed in detail in the next subsection along with methods that find intersecting rectangles quickly. The group data structure allows the metal tree to be constructed quickly.

Labels VDD_pin# and GND_pin# are identified when the CIF input is read. These points represent the root of each metal tree. The root of the metal tree is now known and the metal rectangles are available. Each metal tree is then built. The rest of this section describes the building of one metal tree. Both VDD and GND trees are built in a similar manner.

First, rectangles intersecting the pin labels are found. It is a fatal error if no such rectangles exist. The rectangle is turned into a metal node and used as the root of the metal tree. An initial direction of current flow is set.

A top level view of tree construction follows. Construction is done recursively (depth first) starting with the root metal node until the leaf nodes of the tree have no more subtrees. Figures 5.1 and 5.2 illustrate some intermediate steps in this process.

The metal node (mn1 in figure 5.1) being expanded first has its associated metal rectangle (denoted here as the current rectangle) extended as far as possible along its full width in a direction parallel to the current flow. Extending

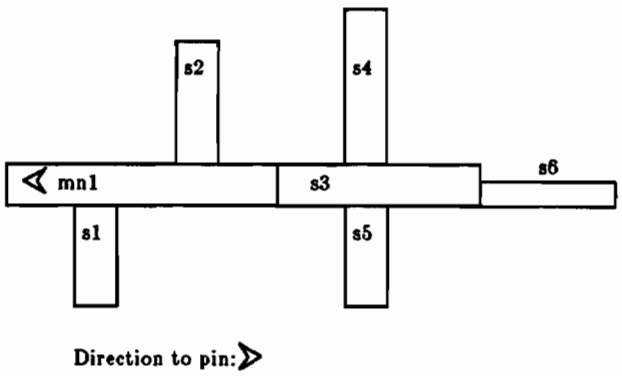

Fig. 5.1 Portion of a metal tree under construction.

rectangles are identified in the set of all metal rectangles using the search methods discussed in the following section; each rectangle may have an arbitrary amount of contact cuts attached to it. This step extends mn1 by metal segment s3. In general, this extension may have to deal with several rectangles that individually extend the metal node along part of its width or extend the rectangle along the both the parallel direction and the perpendicular one. Such rectangles are merged or split into two rectangles (with one returned to the set of available rectangles so that it can be included in the tree later) to make the tree correspond to current flow.

Next, all extensions of the current rectangle perpendicular to the current flow are found (s1, s2, s4 and s5). Finally, all metal rectangles are found that both extend the current rectangle in a direction parallel to the current flow and are narrower than the current rectangle. Metal segment s6 is found during this step. The metal rectangles are adjusted, if necessary, so that there is no overlap between them and the current metal node. The result of this is a list of rectangles that are adjacent to and not overlapping the current rectangle. This list, which can be in a random order, is sorted in the order that the extensions draw current from the current metal node. This gives the list (s1, s2, s4, s5 and s6). Segment s3 was merged into metal node mn1.

These metal segments (s1 through s6) are then marked as being included in

the metal tree. Marked segments will not be reconsidered for inclusion in the metal tree. This prevents the program from looping if metal cycles are present.

For each perpendicular extension, the following is done. The extension becomes its own metal node. In addition, the current metal node is split into two metal nodes at the point of intersection. In figure 5.2, mn2 is the new metal node resulting from the perpendicular extension while mn3 is the new metal node resulting from the splitting of mn1. The portion of the current metal node closest to the root (as determined by current flow) of the tree becomes the parent of both the perpendicular extension and the other portion of the current metal node. This parent metal node (mn1) supplies current to both mn2 and mn3. The new metal node (mn2) perpendicular to the original will be expanded to find child rectangles attached to it. The child metal node (mn3) that was split off from the original will now be the parent for the remainder of rectangles on the list of intersections (s2, s4, s5 and s6) to the original metal node. Metal node mn2 can then be expanded and the remaining extensions to mn3 processed.

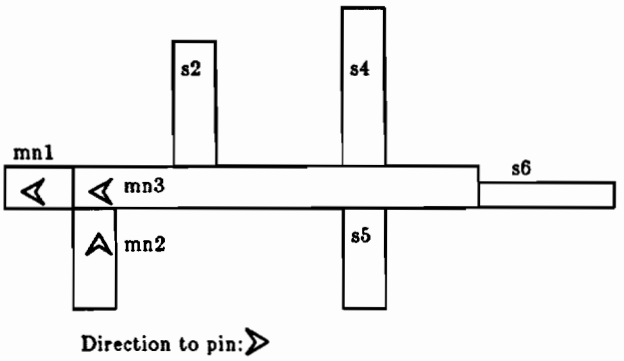

Fig. 5.2 Portion of a metal tree after metal node split.

Parallel extensions of the current rectangle (s6) become metal nodes with the current metal node as a parent. These rectangles are also expanded to find their children.

5.3. Searching For Metal Rectangles Using a Group Representation

While building the metal tree the set of metal rectangles in the circuit is searched to discover rectangles that intersect a rectangle of interest. Such a search must be done once for each metal node during the construction of the VDD and GND metal trees to find other rectangles that intersect the metal node to form subtrees.

An efficient search technique is critical to acceptable performance of the system. A brute force search through an array or linked list would require a linear search to find each intersection. However, even a moderately sized circuit may have tens of thousands of rectangles, making such a search method unacceptable.

The method used in Pwranal is an extended version of the McCreight algorithm (Ullman 1984 and McCreight 1980). First, a one dimensional group structure is outlined. Such a structure represents one dimensional segments. This structure is then generalized to the two dimensional case used by Pwranal to handle rectangles.

A group structure is a tree, with each node of the tree containing one group. Each group, denoted as G(a,b), spans the interval (a,b). The root of the tree spans the entire interval being represented, G(0,max). The lower bound is taken to be 0 without loss of generality. The number "max" must be a power of 2. Each group G(a,b) has two children, which are G(a,(a+b)/2) G((a+b)/2,b).

Every segment whose endpoints are within the range [0,max] is assigned to exactly one group. A segment [m,n] is in group G(a,b) if both of the following hold:

1. The segment is wholly within the group; i.e. m>=a and n<=b.
2. The segment spans the midpoint of the group; i.e. m<=(a+b)/2 and n>=(a+b)/2.

A portion of a group structure for the small interval G(0,32) is in Figure 5.3. Some of the lower level groups are deleted from the figure.

For example, the segment [0,20] is assigned to G(0,32), since it is wholly within the group and spans the midpoint of G(0,32). Segment [22,30] is assigned to G(16,32) and segment [22,23] is assigned to G(20,24).

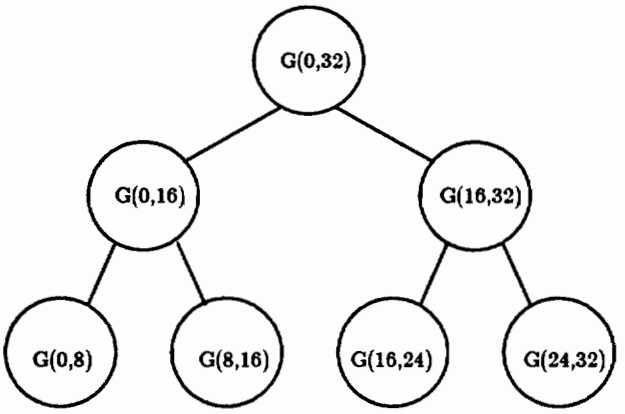

Fig. 5.3 One dimensional group tree for interval [0,32].

Once the group structure is constructed and segments are assigned to it, it can then be used to quickly determine which segments in a group structure that intersect a particular segment. This is done by traversing the tree of groups, starting at the root. At each node visited, it can be determined a priori whether there are possible intersections at that node and whether there are possible intersections on either the left branch or the right branch. Some subtrees will not have to be searched for intersections.

For example, say it is necessary to find all intersections of the interval [5,7] in the above example. The groups G(0,32), G(0,16), G(0,8) and G(4,8) must be searched. Groups G(16,32), G(8,16) and their associated subtrees need not be visited. These groups contain no segments that intersect [5,7] because of the way the tree is constructed.

The McCreight algorithm uses the above one dimensional group data structure to determine intersections of segments. It can also be used to identify the intersections of rectangles once rectangles are sorted in one dimension.

Pwranal uses a two dimensional group structure to search for intersections of random rectangles and points. This also allows for the dynamic reconfiguring of the set of rectangles available for search. These features are required to preserve current flow information in the metal trees, to handle CIF rectangles that abut and overlap in a random fashion, and to handle polysilicon and diffusion jumpers.

In Pwranal, the above structure is extended to two dimensions. There is a group structure similar to the one above representing the X dimensions. Each group in the X dimension tree (X-group) references the root of another tree of groups that handle the Y-dimension. Each group in the Y tree (Y-group) then points to rectangles that are within the group. Rectangles are referenced by Y-groups, while X-groups refer only to the root of Y-group trees. There is one Y-group tree for each node in the X-group tree.

Each rectangle R with edges xmin, xmax, ymin and ymax is a member of a group pair Gx and Gy. It is pointed to by Gy. Group Gy is in the tree of Y-groups referenced by Gx. A rectangle R is placed in groups Gx and Gy as follows:

1. R is a member of the X-group Gx as determined by xmin and xmax.
2. R is a member of a Y-group Gy that is in the tree of Y-groups pointed to by Gx. The particular Y-group is determined by ymin and ymax.

This two dimensional group structure is constructed from the input CIF file. An example two dimensional group structure showing only one Y-group tree is shown in figure 5.4.

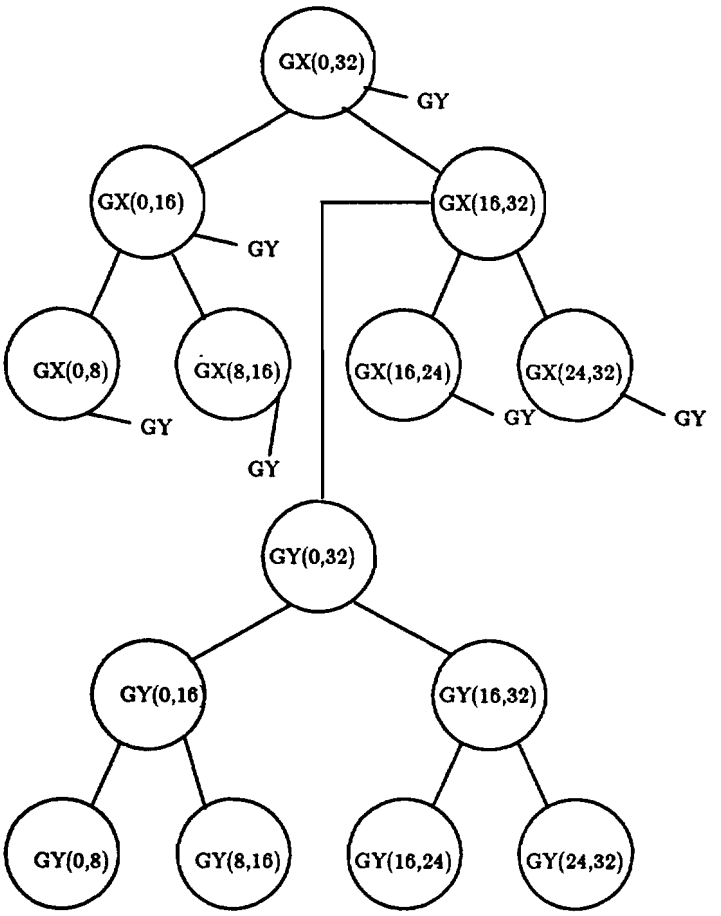

Fig. 5.4 Portion of a multi-dimensional group tree for area [0,32] x [0,32] with one expanded Y-tree shown.

Searches for all intersections of a particular rectangle R work as follows. All Gx's that contain a possible intersection are determined by searching the X-group tree using xmin and xmax. For each such Gx, its local group structure is traversed for Gy's that contain possible intersections using ymin and ymax. Such intersections are reported. The cost in time is at most two group structure searches per reported intersection.

Two steps are done to keep memory requirements down. First, Gx's are

only created if they contain Gy's with rectangles. Likewise, Gy's are created only after a rectangle that is part of the group is found. This results in an unbalanced tree of smaller size than an equivalent balanced tree of groups. Second, the resolution is set to a level above 1 unit. At the lowest level of the group tree, all rectangles are members of the leaf node even if they don't straddle the midpoint. Leaf nodes are searched linearly in such groups, similar to that used to handle collisions in a hash table. These steps keep the group tree to a reasonable size.

5.4. Nodes and Transistors

Data structures for each electrical node and transistor in the circuit are generated from the input ".pext" file.

The node data structures can be accessed from a node name via a hash table. Each node data structure contains the following information:

1. Name of the node.
2. Type of the node (VDD, GND or other)
3. Initial value of the node (ZERO, ONE or unknown)
4. Pointer to the metal node that the electrical node is connected to. Used only if node is at VDD or GND.
5. Pointers to all transistors connected to the node.
6. Information about circuit constructs such as inverter chains that this node is a part of.

Each transistor data structures contain the following information.

1. Transistor type (Enhancement, depletion or pullup)
2. State. (ON, OFF, or unknown). Unknown values are considered to be ON when doing power calculations.

3. Pointers to the nodes that terminals of the transistors are connected to.
4. Geometric information (length and width)
5. Multiplier to the current needs of the transistor. This is between 0.5 and 1.0. It is generated when the transistor is identified as being part of a complementary output circuit construct. Its initial value is 1.0.

These data structures are constructed from the input ".pext" file. They are traversed to analyze the circuit's power needs.

5.5. Complementary Output Configurations

Complementary output configurations result from chains of inverters and superbuffers. Only some of the elements in the chain can be drawing power at one time for any input. The worst case power needs for such configurations are lower. This calculation maintains the conservatism of Pwranal's estimates. They are deterministic in that the power needs of the configuration are guaranteed to be less than or equal to the calculated value for all possible input signals.

Complementary output configurations are detected from information in the VDD metal tree. The set of VDD contact cuts in the VDD metal tree is traversed. Action is taken for each contact cut that is at the pullup transistor of an inverter or superbuffer that has not already been marked as part of an inverter chain.

First, the inverter or superbuffer is traced backwards to find the base of the chain, if any. If there is a chain formed by two or more inverters or superbuffers, then the base of the inverter of the chain will be either drawing power (input at ONE if it is an inverter) or not drawing power (input at ZERO for an inverter). Other elements of the chain will have their values forced by the value of the base element. The power used by the chain is calculated for

the case of the base being at logic ONE (pwr1) and logic ZERO (pwr0). A multiplier is then calculated by taking max(pwr1, pwr0) / (pwr0 + pwr1). This represents the maximum power that can be drawn from the chain, as a ratio to the total power that could be drawn from the elements of the chain if all elements were disconnected and drawing power simultaneously. This number will be between 0.5 and 1.0.

The multiplier is stored with each transistor in the chain. It is used in the next step when calculating path conductances of the transistor network.

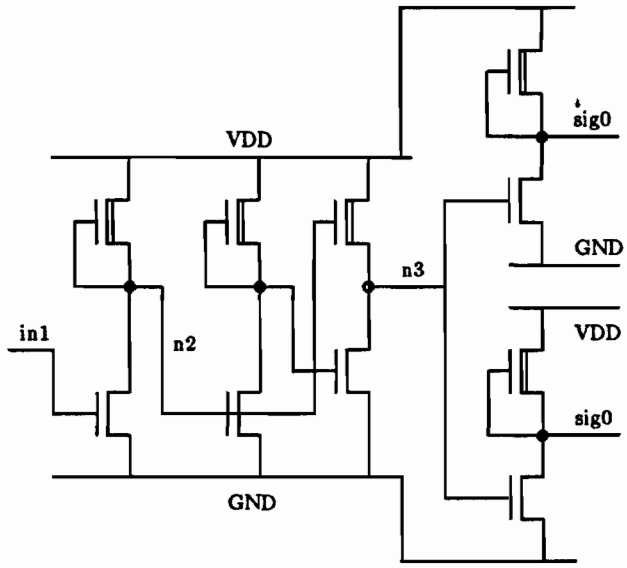

Fig. 5.5 Complementary outputs configuration. Power reduction factor is 0.6.

In figure 5.5, assume that node n3 was encountered during the traversal of VDD contact cuts. Furthermore, assume that the ratios of all diffusion transistors is 8/2 and the ratio of all enhancement transistors is 2/2, and that the channel resistivity is R.

The base of the chain is first found, this is the inverter with input in1. The power needs of the configuration would then be calculated with in1 set to ZERO

and ONE. A value of ZERO at n1 implies that n2 is ONE and n3 is ONE. For this case, the first stage of the superbuffer and the two inverters with output sig0 are the only ones drawing power. This gives the total resistance between VDD and GND as three resistors with value 5R in parallel, or 1.67R. Since power is proportional to 1/R, the power needs for this case (pwr0) is 0.6c, where c is a proportionality constant.

In the other case, a value of ONE at in1 implies that n2 is ZERO and n3 is ZERO. The first inverter with output n2 and the second stage of the superbuffer with output n3 are drawing power. There are only two resistors of value 5R in parallel between VDD and GND, giving a net resistance of 2.5R and a value of 0.4c for pwr1.

The multiplier applied to all transistors in the chain. In this example, the multiplier is max(0.4c, 0.6c) / (0.4c + 0.6c), or 0.6.

5.6. Traversal of the Transistor Network to Calculate Power Needs

The power needs of each contact cut in the VDD metal tree are calculated. Power needs for points in the GND metal tree are calculated at the same time. The VDD metal tree is traversed in a bottom up fashion. At each metal node, the current needs of each contact cut attached to the metal segment is calculated. The total current flowing through a metal node is the sum of the current requirements of the contact cuts at the metal node and the current requirements of the metal node subtree with the metal node as a root.

Current needs at each contact cut are determined by calculating the conductance (reciprocal of resistance) from the contact cut to GND. Power and current flow can then be calculated directly from conductance. This is done by traversing the network of transistors starting at the VDD contact cut.

Conductance of an individual transistor is 0 (open circuit) if the transistor is OFF. If the transistor is not initialized to OFF, the conductance g is:

$$g = 1 / (\text{ChanRes} * (\text{length} / \text{width}) * \text{co\_mult})$$

The term co_mult is a multiplier between 0.5 and 1.0 that adjusts for the effects of complementary outputs. The value of co_mult is calculated according to the steps outlined in the previous section.

The circuit is modeled as a resistor network when calculating conductance. Resistors (MOS transistors across the source and drain terminals) are combined in series and parallel to determine net conductance between a particular VDD contact cut and GND.

The top level algorithm for determining the conductance between a VDD contact cut and GND is shown below in pseudocode. This algorithm is executed once for each VDD contact cut in the VDD metal tree.

Three primitive functions are used. Function series_conductance() returns the net conductance of two conductances in series. Function opposite_node() returns the node connected to the transistor source or drain that is not equal to the node in the argument. Finally, conductance() returns the conductance of a transistor calculated by the above formula. The marking of nodes in tcond_to_gnd() prevents infinite loops on transistor cycles. It also results in the approximation discussed in section 3.5. All transistors and nodes are unmarked before cond_to_gnd() is called for the first time.

```
/* Calculate conductance from a node to GND */
real cond_to_gnd(n)
node  n;
{
    real cond_sum;
    cond_sum = 0;
    foreach unmarked transistor t with source or drain
          connected to node n do {
       cond_sum = cond_sum + tcond_to_gnd(n, t);
    }
    return(cond_sum);
}
```

```
/* Calculate conductance from a node through a */
/* transistor to GND */
real tcond_to_gnd(n, t)
node  n;
transistor  t;
{
    real opp_cond;
    node opp_node;
    mark n as a visited node;
    mark t as a visited transistor;
    opp_node = opposite_node(n, t);
    if (opp_node is GND)  return conductance(t);
    if (opp_node is marked as being visited)
        return(0);
    opp_cond = cond_to_gnd(opp_node);
    return(series_conductance(conductance(t), opp_cond));
}
```

Routine cond_to_gnd() is called with the VDD contact cut as an argument. Note that each VDD and GND contact cut is treated as if it were a distinct node in Pwranal even though all the VDD contact cuts and all the GND contact cuts are electrically connected. The resulting conductance is stored with the VDD metal node and the appropriate GND metal node. There may be 0, 1 or several GND contact cuts that correspond to a particular VDD contact cut. A VDD contact cut with no corresponding GND contact cuts usually correspond to a transistor pulling up an off-chip signal. One corresponding GND contact cut can be the result of a inverter or nand gate configuration. N corresponding contact cuts can correspond to a NOR gate configuration. In the latter case, the conductance is treated as if it was sinking current at the contact cut furthest away from the GND pin.

As an example, the power needs of the circuit in figure 5.6 would be calculated by calling cond_to_gnd() with the VDD contact cut node as an argument. This routine would call tcond_to_gnd() with the depletion transistor and the contact cut as arguments. Routine tcond_to_gnd would call cond_to_gnd() with node n1 as an argument. The path would be traced until both contact cuts at

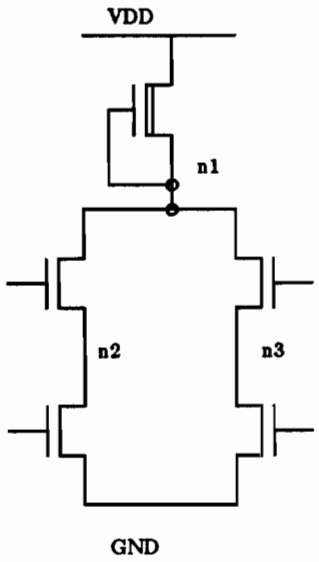

Fig. 5.6 Example circuit.

GND were found.

6. Summary of Results

Software has been developed by the author to analyze the power requirements of NMOS integrated circuits. Unlike previous methods, Pwranal takes circuit topology and expected circuit operating conditions into account to generate more accurate power estimates. Pwranal also analyzes power requirements at the level of individual VDD and GND lines in addition to the power requirements of the entire circuit.

6.1. Verification of Pwranal

Eight large circuits have been analyzed by Pwranal. These include the circuits generated by the design projects of the 1983 VLSI Design class at the Oregon Graduate Center. These circuits contain a rich variety of circuit constructs and range from a few hundred to over 6000 transistors. Some outputs from these Pwranal runs are displayed in Appendix B. These eight circuits are:

Project	Transistors
C2P2: Cube connected parallel processing system	3819
Gstk: G machine stack	3281
HAGWSA: Hash address generator	1813
LERUS: Least recently used addr translator	442
LLSC: Link list search chip	2487
PSE: Packet switching element	2120
SQRT: Square root calculator	6110
TCC: Timing controller chip	1937

Fig. 6.1. Summary of projects used as Pwranal input

Several of the project teams tested their chips and verified their functionality. Two of these project teams also measured the power consumption of their

chips. These measurements are of total power drain, including pads. These measurements can be compared to the power consumption predicted by Pwranal added to the expected dissipation of the pads. Pwranal does not calculate power requirements for pads, as discussed in section 3.5. Output pads require 3.0 mA of current while tri-state pads require 3.5 mA according to MOSIS, the fabrication facility used. The predicted and measured power consumptions are:

Expected and Measured Results

Project	Pads	Pwranal	Predicted Power	Measured Power
TCC	230mW	104mW	334mW	320mW
PSE	450mW	107mW	557mW	450-600mW

Fig. 6.2. Expected and measured power consumption

The TCC (timing controller chip) project is a subsystem of a digital signal processor that contains 1937 transistors. TCC contains 10 input, 6 output and 8 tri-state pins. The PSE project is a portion of a packet switching system. It contains 2120 transistors, 20 input pins, 16 output pins and 12 tri-state pins.

6.2. Pwranal in the IC Design Cycle

Pwranal is useful in the final stages of an IC design. It can be used to characterize power usage in detail. It can be used to determine if current flux through metal will cause electromigration problems. It can also be used to size metal lines so that they require no more IC area than necessary. Currently, metal line width are established using approximate rules of thumb.

Pwranal could also be useful in characterizing complex components used as building blocks in IC's. These include such things as macrocells and modules generated by silicon compiler.

7. Possible Directions for Future Research

Currently, Pwranal only handles NMOS circuits. It is based on NMOS circuits draining power when in a DC state. CMOS circuits are coming into wider use. They consume less power than NMOS circuits. Furthermore, CMOS circuits do not consume DC power, but instead consume power when switching. Electromigration is less of a problem for CMOS circuits. However, it is still useful to characterize their power consumption.

Power drain in CMOS circuits is roughly dependent on frequency of operation and not on DC logic levels. Pwranal models NMOS circuits as a network of switched resistors. Different paradigms must be used for analyzing CMOS power needs.

One possible approach to CMOS circuits is to combine some future version of Pwranal with a logic or switch level simulator. Logic simulators are currently used to gather statistical information for statistical fault analysis (Jain and Agrawal 1984). For use with Pwranal, the logic simulator would gather statistical information about how many times each node changed state and would save the duration (in time) of the simulation. The future version of the Pwranal program would analyze circuit topology to build a metal tree and to assign transistors to their proper place in the tree, as is done now. The information on transistor switching from the simulator would be added to generate an estimate of power requirements of the circuit and on current requirements on each metal line. One difference between this and the current Pwranal is that the results would be dependent on a set of input vectors. The current Pwranal would also have to be extended to handle several layers of metal, since many CMOS processes use more than one metal layer.

References

1. Black, J. R.; *Thin Film Electromigration Workshop*; Reliability Physics 1982 Proceedings; IEEE; New York; 1982.
2. Glaser, Arthur B. and Subak-Sharpe, Gerald E.; *Integrated Circuit Engineering*; Addison Wesley Publishing Co.; Reading, Ma.; 1977.
3. Jain, Sunil K. and Agrawal, Vishwani D; *Stafan: An Alternative to Fault Simulation*; Proceedings of the 21st Design Automation Conference; Alberqueque, NM; 1984.
4. Mead, Carver and Conway, Lynn; *Introduction to VLSI Systems*; Addison Wesley Publishing Co.; Reading, Ma.; 1980.
5. McCreight, Edward M.; *Efficient Algorithms for Enumerating Intersecting Rectangles and Intervals*; CSL-80-9; Xerox PARC; Palo Alto, Ca.
6. Song, William S. and Glasser, Lance A.; *Power Distribution Techniques for VLSI Circuits*; 1984 Conference on Advanced Research in VLSI, MIT; Jan. 1984.
7. Termin, Chris; *User's Guide to NET, PRESIM and RNL/NL (ver. 3.0)*; MIT Laboratory for Computer Science; Cambridge, Ma.; 1982
8. Ullman, Jeffrey D.; *Computational Aspects of VLSI*; Computer Science Press; Rockville, Md.; 1984.

Appendix A - Pwranal Man Page

NAME

pwranal – Analyze the power needs of a NMOS circuit

SYNOPSIS

pwranal [–c] [–**f value**] [–**r value**] [–**m value**] [**-fast**] [–**i filename**] basename

DESCRIPTION

Pwranal takes as input two files: basename.cif and basename.pext. The ".pext" file consists of extracted transistors with the physical location of Vdd and Gnd contact cuts preserved. If the "-fast" option is not present basename.pext created from the file basename.cif. If the "-fast" option is present, the existing basename.pext file is used.

The input ".cif" file must have local labels "Vdd_pin#" and "Gnd_pin#" at the point where the circuit sources/sinks current. These points are used by pwranal as a root point when constructing power bus topology. The labels may be in lower or upper case. A fatal error is generated if these labels don't exist.

Output is to standard out. The summary includes technology-dependant parameters in effect, along with the effects of initialization with the "-i" option, if present. It also includes a summary of power usage and maximum current flux through Vdd and Gnd line. There are separate summaries for power needs as traced through Vdd, Gnd, and power needs with and without heuristic reductions. Pullups and pulldowns may make calculated Vdd and Gnd power requirements in the circuit where it occurs (in absolute CIF coordinates and normalized [0.0 to 1.0] coordinates is also shown. The maximum voltage drop from the power pin to the leaf of the metal lines is shown for both Vdd and Gnd. A warning is printed if the maximum current flux through a power bus is exceeded.

Recognized invocation options are:

–c Causes a CIF format output to be written in the file basename.pwr.cif. This file is in the ".cif" format. It contains the power lines and labels generated by pwranal indicating power requirements on various points on the power busses. Labels are of the form Pwr_DIR=XXXXXmW, where DIR is the direction of current flow at the point and is one of the compass points (north (N), south (S), east (E) or west (W)). Component XXXXX is the power requirements of subsidiary metal lines at that point, in milliwatts. Current paths with only one Vdd source contact cut and several Gnd sinking contact cuts (such as NOR gates) only generate labels at one of the sinking (Gnd) contact cuts corresponding to the total power requirements of the gate.

–f value

Changes the default maximum current flux (mA per micron) allowed in the design rules. The default is 1.0 mA per micron. If the "-c" option is used, regions of metal which are exceed the maximum current flux will be highlighted in the CIF output.

–r value

Changes the default channel resistence (transistor on) of NMOS transistors. The default is 10K ohms per square.

–m value

Changes the default resistence of metal. The default is 0.03 ohms per metal square.

–**fast** Causes the current basename.pext file to be used as input. If the "-fast" option is not present, the ".pext" file is extracted from the file "basename.cif".

–i filename

The file is read to initialize nodes to 0 or 1. The file contains one initialization per line in the following syntax:

node_name [=] (0|1)

Comments may be included in the file by beginning the line with a ""#" character. The node initial values are propagated through the circuit and used to initialize transistors to ON or OFF. Transistors which are initialized to OFF are assumed to draw no current. This is useful for initializing portions of the circuit not expected to contribute to DC power.

The program analyzes the circuit's power line (Vdd and Gnd) topology and the location of contact cuts. It then calculates the DC power requirements at each Vdd and Gnd line.

Heuristics are applied to complementary outputs resulting from inverter chains and superbuffer configurations to obtain more accurate maximum power requirements.

Pwranal assumes that all transistors are ON unless they are known to be OFF by node initializations using the "-i" option. The power drain of a group of transistors may be reduced if they are part of a complementary output configuration.

Pullup transistors without a path to Gnd are treated as if they had a direct path to Gnd outside of the circuit. Accordingly, they can generate larger power requirements than if this were not the case.

The calculated power represents the worst case requirements (not the average power drain) of the circuit, given the specified node initial values, if any.

DIAGNOSTICS

Error messages are written to standard error and should be self explanatory.

AUTHOR

Jeffrey Wilson (OGC).

Appendix B - Sample Runs of Pwranal

This appendix contains sample results from Pwranal. The input circuits are projects generated by the 1983 VLSI design class at Oregon Graduate Center. These circuits contain a few hundred to several thousand transistors each.

Pwranal Summary -- pGstk.cif

Channel resistance (ohms/square)	17000.000
Metal resistance (ohms/square)	0.030
Max metal curr flux (mA/micron)	1.000
Vdd Poly/diffusion jumpers	0
Gnd Poly/diffusion jumpers	0
Initial values source file	(none)

	Vdd	Gnd
Total estimated power needs	199.809mW	148.747mW
pullup/pulldown portion	51.062mW	0.000mW
Raw DC power, no reductions	203.840mW	152.778mW
pullup/pulldown portion	51.062mW	0.000mW
Max metal current flux (based on estimated power needs)	1.571mA/micron	1.281mA/micron
Location of max current flux (absolute)	(101600, 337200)	(330600, 310600)
(normalized [0.0-1.0])	(0.250, 0.799)	(0.812, 0.736)
Max. voltage drop (from pin)	0.050v	0.037v

Warning: maximum current flux exceeded; check power busses

Pwranal Summary -- pPSE.cif

Channel resistance (ohms/square)	17000.000
Metal resistance (ohms/square)	0.030
Max metal curr flux (mA/micron)	1.000
Vdd Poly/diffusion jumpers	0
Gnd Poly/diffusion jumpers	0
Initial values source file	(none)

	Vdd	Gnd
Total estimated power needs	107.274mW	83.240mW
pullup/pulldown portion	24.034mW	0.000mW
Raw DC power, no reductions	127.920mW	103.887mW
pullup/pulldown portion	24.034mW	0.000mW
Max metal current flux (based on estimated power needs)	0.658mA/micron	0.677mA/micron
Location of max current flux		
(absolute)	(403000, 239800)	(51600, 110000)
(normalized [0.0-1.0])	(0.849, 0.543)	(0.109, 0.249)
Max. voltage drop (from pin)	0.078v	0.039v

Pwranal Summary -- pTCC.cif

Channel resistance (ohms/square)	17000.000
Metal resistance (ohms/square)	0.030
Max metal curr flux (mA/micron)	1.000
Vdd Poly/diffusion jumpers	0
Gnd Poly/diffusion jumpers	0
Initial values source file	(none)

	Vdd	Gnd
Total estimated power needs	104.091mW	46.195mW
pullup/pulldown portion	57.896mW	0.000mW
Raw DC power, no reductions	113.125mW	55.229mW
pullup/pulldown portion	57.896mW	0.000mW
Max metal current flux (based on estimated power needs)	0.867mA/micron	1.688mA/micron
Location of max current flux (absolute)	(216200, 75800)	(348200, 364200)
(normalized [0.0-1.0])	(0.398, 0.152)	(0.641, 0.732)
Max. voltage drop (from pin)	0.071v	0.042v

Warning: maximum current flux exceeded; check power busses

Biographical Note

The author was born on November 1, 1958 in Portland, Oregon. He attended Cleveland High School in Portland and graduated in June, 1976. He then attended the Massachusetts Institute of Technology, receiving a Bachelor of Science degree in Electrical Engineering in June, 1980.

In 1982 he began part time studies at the Oregon Graduate Center while working at Tektronix Inc. He is currently employed as a senior engineer at Integrated Measurement Systems in Beaverton.