QuickCap® Auxiliary Package User Guide and Technical Reference

Version O-2018.06, June 2018

SYNOPSYS®

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

About This User Guide viii
Customer Support x

1. Introduction

Performance 1-2
Error Versus Uncertainty 1-3
Distrust of Statistics 1-3
Trust of Computer Analyses 1-4

2. Getting Started

Standard Procedure 2-2
Program Flow 2-2
Results 2-3
Common Input-File Commands 2-3
Controlling the Uncertainty of a Sum 2-5
Composite Goals 2-6
Planar Dielectrics 2-8
Nonplanar Dielectrics 2-8
Summary 2-9
Critical Nets 2-9
Dial-in Accuracy 2-9
Analysis of IC Layouts 2-9

3. Working With Uncertainty

Definitions 3-2
Combining Independent Results 3-3

Weighted Sums 3-4
Confidence 3-6
Self-Consistent Results 3-8
Averaging Results 3-9
Statistical Characterization 3-10
Pitfalls 3-11
Meeting Design Goals 3-12

4. Extraction Procedure

Monte Carlo Analysis 4-2
Floating Random-Walk Procedure 4-4
QuickCap Implementation 4-7
Runtime 4-7
Canceling a Job 4-7
The Floating Random-Walk Kernel 4-7
Walk Rate 4-7
Convergence Rate (Per Walk) 4-8
Limitations Regarding Dielectrics and Floating Metal 4-8
Other Limitations 4-10

5. The Command Line

Command-Line Format 5-2
Convergence-Goal Options 5-2
Other Options 5-4
Environment Variables 5-13

6. The Input File

Control, Script, and Encryption Commands 6-2
Character-Based Decryption 6-3
Length Parameters 6-3
Windowing 6-4
Window-Related Commands 6-4
Flags 6-6
Netlist Customization 6-11
QuickCap Objects 6-11

Nets, Nodes, and Signals 6-12
Net, Node, and Signal Names 6-12
Nets 6-12
Nodes and Signals 6-13
Net- and Signal-Related Statements 6-14
Floating (Dummy) Metal 6-18
Dielectrics 6-18
Device Capacitance 6-21
Selecting Nets for Extraction 6-22
Layers and Depth 6-23
Layer Definition 6-23
Layer Properties 6-24
Setting Layers and Depth 6-33
Adjusting Depth 6-33

7. Symmetry

Periodic Geometric Structure 7-2
Periodic and Reflective Boundary Conditions 7-3

8. Output

The Log File 8-2
The Numeric Output File 8-2
Output Type 8-2
Node Names 8-3
Values 8-3
The Capacitance Netlist 8-3
Netlist Header Information 8-4
Capacitance Values 8-4
The Summary File 8-4
Runtime 8-5
Convergence Information 8-5
Parallel Operation 8-5
Table Output 8-6
Table File Header Information 8-6
Terminal 8-6

9. Graphics Preview Mode

Graphics Log Files 9-5
Graphics Record and Play Files 9-5
Mouse Input 9-5
Keyboard Input 9-6
View Menu 9-6
Using Cross Section in Difference Dimensions 9-8
Specifying the Center and Size of a Window 9-8
Show Menu 9-9
Colors Menu 9-9
Selecting a Color by Name 9-10
Markup Menu 9-10
Markup Text 9-11
Measurement Points 9-11
Nets Menu 9-11
Selecting a Net by Name 9-12
Layers Menu 9-12
Selecting a Layer by Name 9-13
Surface Menu 9-13
Help Menu 9-14

Glossary

Preface

This preface includes the following sections:

- About This User Guide
- Customer Support

About This User Guide

This user guide describes the QuickCap tool to extract accurate capacitance in complex 3D structures.

Audience

This user guide is for engineers who use the QuickCap and gds2cap tool to create complex systems on a chip. The readers of this user guide must be technically oriented and have some familiarity with QuickCap products.

Related Publications

For additional information about the QuickCap tool, see the documentation on the Synopsys SolvNet® online support site at the following address:

https://solvnet.synopsys.com/DocsOnWeb

You might also want to see the documentation for the following related Synopsys products:

- gds2cap
- QuickCap® Auxiliary Package
- StarRC™

Release Notes

Information about new features, changes, enhancements, known limitations, and resolved Synopsys Technical Action Requests (STARs) is available in the *QuickCap Release Notes* on the SolvNet site.

To see the *QuickCap Release Notes*,

1. Go to the SolvNet Download Center located at the following address:

   https://solvnet.synopsys.com/DownloadCenter
2. Select QuickCap, and then select a release in the list that appears.

Licensing

You can use QuickCap using a single node or a floating node license.

Single-Node License

If you are using QuickCap on a single node license, you can start the license manager (supplied by Synopsys). Set the appropriate FLEXlm:

```
lmgrd -c path
setenv LM_LICENSE_FILE=path
setenv QUICKCAP_LICENSE_FILE=path
```

Using **QUICKCAP_LICENSE_FILE** rather than **LM_LICENSE_FILE** ensures that you do not adversely affect any application licensed with FLEXlm by using a different license file.

For information on environment variable, see “-” on page 5-13

Floating-Node License

When you are using QuickCap on a floating-node license, set up the license in either of these ways:

- Make the license file available to all nodes in the network that need it by placing it or a copy on as many file systems as necessary, and set **LM_LICENSE_FILE** or **QUICKCAP_LICENSE_FILE** appropriately; or
- Set either **LM_LICENSE_FILE** or **QUICKCAP_LICENSE_FILE** to [***port***]@***host***, where ***port*** and ***host*** are from the SERVER line in the license file. The ***port*** value need not be specified if the SERVER line uses a default port.

Customer Support

Customer support is available through SolvNet online customer support and through contacting the Synopsys Technical Support Center.

Accessing SolvNet

SolvNet includes a knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet also gives you access to a wide range of Synopsys online services including software downloads, documentation, and technical support.

To access SolvNet, go to the following address:

https://solvnet.synopsys.com

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.

If you need help using SolvNet, click HELP in the top-right menu bar.

Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

- Open a support case to your local support center online by signing in to SolvNet at https://solvnet.synopsys.com, clicking Support, and then clicking “Open A Support Case.”
- Send an e-mail message to your local support center.
  - E-mail support_center@synopsys.com from within North America.
  - Find other local support center e-mail addresses at https://www.synopsys.com/support/global-support-centers.html
  - Telephone your local support center.
  - Call (800) 245-8005 from within North America.
  - Find other local support center telephone numbers at https://www.synopsys.com/support/global-support-centers.html

1. Introduction

As ICs become more complex, and the width, height, and spacing are scaled down, parasitic capacitance associated with their multilevel interconnects becomes a dominant limitation on the speed at which these circuits can operate. Unfortunately, 1D and 2D analyses of parasitic capacitance often fail to take into account the true 3D distribution of fringing electrostatic fields. These fringing fields become generally more important as interconnect wire width and spacing decrease.

The QuickCap tool is the fastest and most reliable program for accurate capacitance extraction in complex 3D structures. It handles non-Manhattan shapes, multiple dielectrics, and large circuits. QuickCap is in close agreement—in both 2D and 3D and for simple and complex problems—with exact analytical solutions; with finite-difference, finite-element, and boundary-integral benchmark solutions; and with experimentally measured results for a number of test structures.

QuickCap supports advanced technology modeling (optical proximity correction (OPC), chemical-mechanical polishing (CMP), and trapezoidal wires); which includes netlist generation and reduction; and parallel execution. It also provides a 3D graphics viewer.

This user guide describes QuickCap, and uses the name QuickCap generically.

Performance

Standard methods that are deterministic and depend on fine meshing of the problem geometry are subject to spatial-discretization errors, matrix-inversion rounding errors, and prohibitive memory requirements. QuickCap uses a Monte Carlo technique known as the *floating random-walk method* and solves the exact governing partial differential equation (Laplace’s equation) to a theoretical accuracy limited by the numerical precision of the computer. In addition, QuickCap employs virtually exact boundary conditions (within the size of an interatomic spacing). Sources of error that exist for deterministic methods do not exist for QuickCap. Instead of deterministic results of an approximated problem, QuickCap extracts statistical estimates (with associated confidence factors) of the exact problem.

QuickCap has performance tradeoffs among accuracy, problem complexity, memory requirements, and speed that are markedly different from the performance tradeoffs of deterministic codes:

- Speed of convergence depends only weakly on the number of elements in the circuit or the area of the circuit.

  The capacitance of a 2-mm clock net can be found about as quickly as the capacitance of a 10-µm interconnect.

- Memory consumption of the extractor is approximately proportional to the memory required to store the geometry alone.

  For example, a microprocessor arithmetic-logic unit requiring 2 MB to store just the geometry of the circuit required about 10 MB to extract the capacitance of critical nets.

- The absolute uncertainty in capacitance is always known.

  Reported capacitance values are within two standard deviations with 95 percent confidence, and within three standard deviations with 99.7 percent confidence. Deterministic codes, on the other hand, base their confidence on indirect indicators, such as symmetry of the capacitance matrix, but not directly on the capacitance values themselves.

- The convergence rate is known, and QuickCap can predict when it converges to a specified accuracy.

  As with many statistical approaches, twice the accuracy requires four times the runtime. What is the relationship between runtime and accuracy for deterministic codes? There are various results. Even codes that apparently have better convergence behavior, cannot converge *faster* than QuickCap for complex structures, whereas the simple structures contain boundary effects so severe that such a program might converge faster, but to the wrong answer.

- QuickCap has no “hidden failure mode,” in which a grossly wrong answer is reported with any confidence.

Indicators used by deterministic codes, however, can only show that a solution to an approximated problem (one that is discretized) is self-consistent or that the solutions to a problem approximated two ways are similar. Such indicators can fail to detect bad results.

Error Versus Uncertainty

Each capacitance value printed by QuickCap is a statistical average and is accompanied by an uncertainty that indicates how the average relates to the value that you would get if statistics were not “in the way.” Acceptance of the QuickCap statistical approach is often hampered by a distrust of statistics combined with a trust of reproducible computer analyses. Consider the following results:

```
QuickCap result:         1.55fF, ±2% uncertainty
Deterministic result:    1.3912903fF
```

Confronted with these results, most users accept the answer from the deterministic code because there is no perceived error. Furthermore, if they run the same deterministic program again they get the same answer. Running QuickCap again generates a result that is statistically equivalent, even though it might not be exactly the same. In fact, the new result can be averaged with the old result to yield a more accurate capacitance value. The issues of *distrust of statistics* and *trust of computer analyses* are briefly addressed in the following sections.

Distrust of Statistics

Statistics are generally distrusted because of:

- Lack of knowledge of the rules for applying statistics
- Frequent misuse of these rules by others, often through ignorance
- The abundance of poor statistics

Chapter 3, “Working With Uncertainty” is a tutorial for basic understanding of statistics. That chapter shows, for example, that the previous QuickCap result does not conclude outright that the deterministic answer is wrong; rather, the result concludes there is only one chance in 150 that the correct capacitance value is nearer to the deterministic result than to the QuickCap estimate.

Uncertainties provided by QuickCap are reliable because the experimental results are mathematical and are not affected by the myriad of factors that typically affect statistical studies.

Trust of Computer Analyses

Computer results are subject to many errors. Some are rooted in defects that can be fixed when found. Other sources of errors are not as easily fixed, such as idealizing and simplifying geometry to make a problem tractable. These approximations can often be minor and have little significance for capacitance problems with known analytic solutions. Unfortunately, these “known problems” tend to be small and unrealistically simple.

The effect of approximations on realistic problems of moderate-to-large complexity can be found by comparison with other independent programs and with experimentally measured data from fabricated test structures. QuickCap results regularly agree with measured data for large problems, but they sometimes disagree, occasionally radically, with deterministic results. Because the QuickCap approximations (1Å spatial resolution, for example) do not vary with problem size; a significant discrepancy is probably due to a deficiency in the deterministic codes arising from a low mesh density (leading to discretization error), tight boundaries (leading to boundary errors), or approximations necessary to accommodate the memory requirements of the deterministic method.

2. Getting Started

QuickCap must be licensed to run on your computer. For specific information about licensing, see "Licensing" on page 1-ix.

This chapter describes commonly used features of QuickCap. Convergence rates generally depend on the platform used. Furthermore, because QuickCap is nondeterministic, its results vary. To test for significant differences between results, apply a statistical comparison as described in "Self-Consistent Results" on page 3-8.

⚠

QuickCap produces files that are named based on the name of the input file. Existing files with these names are overwritten. The suffixes used for QuickCap output-file names are log, .summary, .spice, and .tab.

If QuickCap is run with no arguments, it prints a brief description of itself, the calling format, and a one-line description for each possible option.

Standard Procedure

This section explains standard QuickCap procedures.

Program Flow

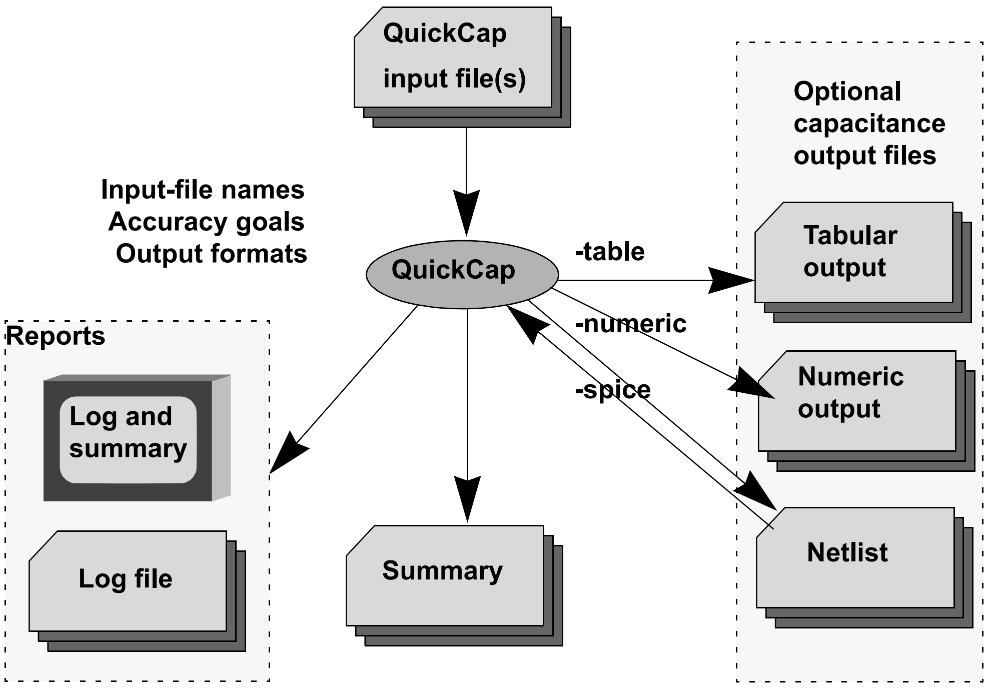
The syntax for running QuickCap is as follows:

```
quickcap [options] [file_list]
```

The options affect how long QuickCap runs and what it outputs, but do *not* affect which statistics it keeps.

The QuickCap input files can be prepared by hand or generated by gds2cap, a tool that converts 2D layout data to 3D QuickCap decks. QuickCap is normally run with an option list and the name of a file defining the problem. Specifying multiple files is equivalent to concatenating them into one input file. Figure 2-1 shows the relationship of the QuickCap input files to the files it outputs.

Figure 2-1: QuickCap Files



Specify input-file names, accuracy goals, and output formats on the command line.

QuickCap processes 3D QuickCap decks and can generate output in various formats. QuickCap can also incrementally update an existing netlist, such as one generated by the gds2cap **-spice** option.

Results

Each capacitance value QuickCap generates has an associated *uncertainty* (statistical error), a calculated value of the standard deviation of the result, which it outputs to the terminal, to the summary file, to the netlist (as a comment), and to the numeric file. The uncertainty indicates not only how the QuickCap estimate varies between independent runs, but also how the estimate is related to the value that you get by averaging the results of an infinite number of QuickCap runs.

For a value with a large *relative* statistical error, QuickCap prints a *likely* upper limit, two standard deviations above the average.

For example, instead of:

```
    1aF +/- 2aF (200%)
```

QuickCap might print:

```
    <~5aF
```

The true capacitance is smaller than 5 aF with a 95 percent confidence level. This can be valuable information. For example, if this result represents a coupling capacitance on a 1 fF net, you know that the crosstalk due to this interaction is negligible, <~0.5% (1 aF=0.001 fF). See Chapter 3, “Working With Uncertainty” for more information about interpreting the statistical nature of QuickCap results.

Common Input-File Commands

Of the many input-file commands, seven are commonly used in hand-generated QuickCap decks. The most common geometric primitive is a box.

- net *netName* [*object*]
- extract *netList*
- groundplane [at] *z*
- dielectric *eps* [*object*]
- eps eps up to *z*

Chapter 6, “The Input File” describes all input-file commands recognized by QuickCap. The order of the **extract**, **groundplane**, and **net** statements does not generally matter. In the absence of layer-based precedence, the order of **net** statements can make a difference for nets that share a common outer surface, such as a *p*-type source/diffusion region in an *n*-type well. The relative order of the **dielectric** statements is important when different dielectrics overlap: precedence is given to the first one specified. The precedence of dielectrics can be reversed through the **dielectricPrecedence** statement.

Any **eps up to** statements must appear in order of increasing *z*, but can be anywhere in the file.

The **net** statement declares the name of a net and can include a geometric object, such as a box or a sphere, on the same line. Different **net** statements refer to the same net if the names are the same.

Net names, by default, are case dependent. For example, Anode and aNode refer to different nets. You can override this feature with the **caseFolding** statement.

A box in one net can be bounded by a box in another net; in which case the inner box takes precedence on any coincidental surfaces. Otherwise, in the absence of any layer-based precedence, the first net defined takes precedence.

The **extract** statement lists the nets in the input file that are to be extracted. Net names that are in an **extract** statement but not defined by a **net** statement are ignored. If there is no **extract** statement in the file, all nets other than ground are extracted. Nets can also be automatically selected for extraction based on auxiliary data and special extract declarations.

Only one **groundplane** statement is allowed. It defines the height ***z*** at which there is a groundplane. The groundplane extends down from this height. Whenever the upper surface of a box in a net coincides with the groundplane, the box takes precedence.

The **dielectric** statement specifies a relative dielectric constant and can include a geometric object, such as a box, on the same line. For dielectrics that overlap, the first one takes precedence.

The **eps up to** statement is used to enter planar dielectrics. These must be entered from the bottom up. Nonplanar dielectrics *always* take precedence over planar dielectrics.

You can define a list of boxes or other geometric objects, not more than one per line, using a left parenthesis at the beginning of a line or even on the line of the **net** or **dielectric** command. Terminate the list with a right parenthesis. The objects belong to the most recently declared net or dielectric. Often, a **net** or **dielectric** command either includes a single object (parentheses optional) or is followed immediately by a list of objects enclosed by parentheses (these are not optional).

A box can be simply represented by six values (the coordinates of two opposite corners). The following are examples of simple nets and dielectrics:

```
net cube 1 1 1 2 2 2

net cube (1 1 1 2 2 2 )

net cube
( 1 1 1 2 2 2 )

net L
( 0 0 0   1 20 1
  0 0 0   10 1 1 )

net L (
  0 0 0   1 20 1
  0 0 0   10 1 1 )

dielectric 3.9; SiO2
( -1000 -1000 0 1000 1000 10 )
```

Text after a semicolon is a comment, as shown in the previous **dielectric** statement.

Controlling the Uncertainty of a Sum

This section illustrates how to control the uncertainty of a circuit analysis.

Because QuickCap employs a statistical method, the various capacitance estimates are scattered around the exact values—some are overestimates and some are underestimates. In any sum (weighted or not), there is a statistical cancelation of uncertainty (see "Weighted Sums" on page 3-4). Ultimately, you want to find the circuit behavior with a certain level of confidence. This goal is different from finding capacitance values to a certain level of uncertainty.

QuickCap does not calculate the time constant. This assumes that the effective resistances are correct. To be on the safe side, overestimate these resistances if you cannot estimate them well.

Consider some cases based on the previous values.

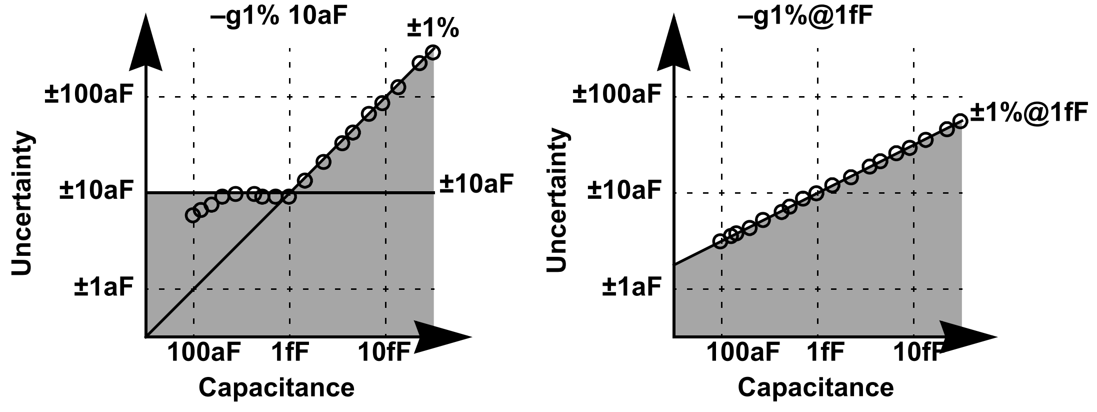
- If the design goal is to achieve a delay time of better than 5 ns and the delay-time analysis tool gives a value of 1 ns ± 0.02 ns (±2%), the relative uncertainty (±2 percent) is higher than the ±1 percent goal (from **-g 1%@4ns**). You have determined that you are well within the design goal.
- If the delay-time analysis tool gives a value of 16 ns ± 0.08 ns (±0.5 percent), the absolute uncertainty is higher than ±0.04 ns (1% of 4 ns). You have determined that you have failed to meet the design goal.
- If four nets in a critical path each contribute 1ns±0.02 ns, the total delay time is 4 ns±0.04 ns (see "Weighted Sums" on page 3-4). A goal of 1%@4 ns holds not only for each net, but for any combination of the nets.

Because you have not yet learned how to instruct QuickCap what weights to use, for now, specify a goal on the *unweighted* sum of capacitances for the nets defined in misc. Enter the following command:

```
>quickcap -g 1%@1fF misc
```

This goal is stricter than the goal, **-g 1% 10aF**, used earlier. For comparison of the required uncertainties for these two goal specifications, See Figure 2-2 on page 2-6.

Figure 2-2: Converged Results (Shaded Areas) for -g 1% 10aF and -g 1%@1fF (QuickCap Data)



Composite Goals

This section illustrates how to specify a range of acceptable relative goals: a high-accuracy relative goal for large results and a less strict relative goal for small results.

A goal of the form **-g *pct1* *@value..pct2*** is a composite goal that specifies two relative goals, ***pct1*** and ***pct2***. This is an extension of the form **-g *pct@value***.

If ***pct1*** is less than ***pct2***, all results above ***value*** have a relative goal of ***pct1*** and the relative goal of smaller results are relaxed to a limit of ***pct2***. If ***pct1*** is larger than ***pct2***, all results below ***value*** have a relative goal of ***pct1*** and the relative goal of larger results is tightened to a limit of ***pct2***. For intermediate values, the goal is **-g *pct1@value***.

Enter the following command:

```
>quickcap -g 1%@1fF..0.3% misc
```

The goals used here give results that can be shown on the same scale as used in Figures 2 and 3, and are considerably more accurate than needed for analysis of realistic layouts. A more reasonable specification is **-g 1%@1fF..10%**.

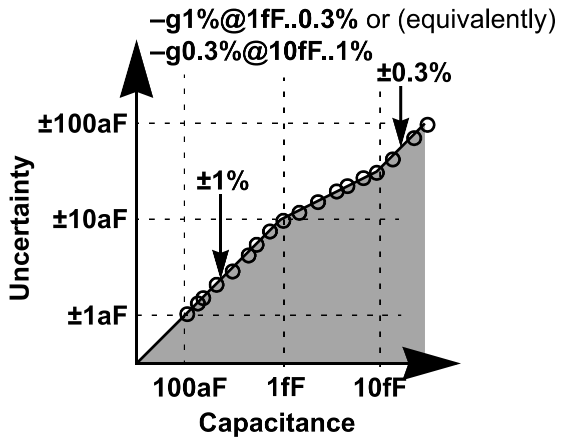
The values are similar to the following:

```
misc:  0' 2" elapsed,  0' 2" CPU
       a:   0.1005fF +/-  0.98aF (0.98%)
       b:   0.1261fF +/-   1.3aF (0.99%)
       c:   0.1501fF +/-   1.5aF (0.99%)
       d:    0.199fF +/-   1.9aF (0.97%)
       e:    0.293fF +/-   2.9aF (0.98%)
       f:   0.4263fF +/-   4.3aF (1%)
       g:   0.5263fF +/-   5.2aF (0.98%)
       h:   0.7523fF +/-   7.4aF (0.99%)
       i:    1.006fF +/-   9.9aF (0.98%)
       j:    1.437fF +/-    12aF (0.82%)
       k:    2.327fF +/-    15aF (0.65%)
       l:    3.698fF +/-    19aF (0.51%)
       m:    4.701fF +/-    21aF (0.45%)
       n:    6.911fF +/-    26aF (0.38%)
       o:    9.345fF +/-    30aF (0.32%)
       p:    13.72fF +/-    40aF (0.29%)
       q:    22.71fF +/-    68aF (0.3%)
       r:    31.41fF +/-    90aF (0.29%)

   GOAL ACHIEVED:  +/-1%@1fF..0.3%
```

Figure 2-3 shows the required uncertainties for this goal specification.

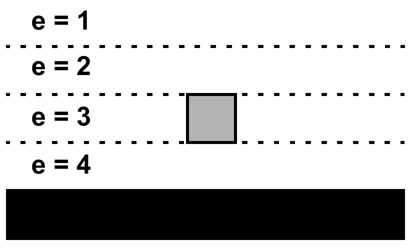
Figure 2-3: Converged Results (Shaded Region) for -g 1%@1fF..0.3% (QuickCap Data)



One breakpoint is ±***pct1*** at ***value*** (±1% at 1 fF in Figure 2-3). The other breakpoint is ±***pct2*** at ***value*** $\times$ (***pct1***/***pct2***)^2 (±0.3% at 10 fF in Figure 2-3).

Planar Dielectrics

This section illustrates the **eps up to** statement.



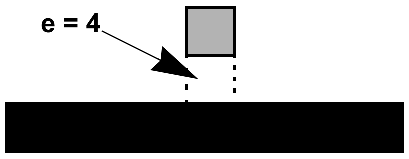
The **eps up to** statement is useful for entering planar dielectrics. When interfaces between different dielectrics are planar or nearly planar, use this statement. QuickCap converges faster for planar than for nonplanar dielectrics because QuickCap can perform some initial calculations on the planar dielectrics, which it then uses to improve speed. Create the following file called planar:

```
groundplane at z=0
net c 1 1 1 2 2 2
eps 4 up to 1
eps 3 up to 2
eps 2 up to 3
```

The capacitance found for the simpler problem (uniform dielectric of value 1) was around 95 aF. If the net is embedded in a thick dielectric with a relative dielectric constant of 4 (the same as the dielectric constant of the first dielectric layer in planar), you can expect four times the capacitance of the simple problem, or about 0.38 fF. If the net is embedded in a thick dielectric with a relative dielectric constant of 3 (the same as the dielectric constant of the second dielectric layer in planar), you can expect a capacitance of around 0.28 fF.

Nonplanar Dielectrics

This section illustrates the **dielectric** statement.



A nonplanar dielectric is declared similar to a net except that instead of specifying the name of a net, you specify the value of the relative dielectric. Create the following file called nonplanar:

```
groundplane at z=0
net c 1 1 1 2 2 2
dielectric 4 1 1 0 2 2 1
```

In this structure, only the region directly under the cube has a relative dielectric constant of 4. An estimate of the capacitance based on an earlier result is 0.12 fF.

You expect this capacitance to resemble a 3 µm-by-3 µm capacitor with no fringing, but with a 1 µm cube underneath wherein the dielectric constant is changed from 1 to 4. This result must be approximately εx3 µm/1 µm larger than the 94 aF you previously obtained, or 0.12 fF.

Summary

The exercises in this section introduce some of the basic features of QuickCap.

Critical Nets

By default, QuickCap extracts all nets.

Use the **extract** declaration to specify critical nets to be extracted.

Dial-in Accuracy

With the ability to set convergence goals comes the responsibility to set *reasonable* convergence goals. *Adequate* accuracy is not necessarily apparent to the novice user. In general, if QuickCap requires a long time to converge (time per extracted net), the goal might be much stricter than is actually needed. Accuracy goals might be over specified when extracting all capacitance elements in a critical cell (including negligible ones), generating matrix results, analyzing RC distributed results, or characterizing errors of other methods.

Analysis of IC Layouts

The QuickCap input deck can be configured to represent many situations well beyond those described here. The most common use of QuickCap, however, is to analyze IC layouts that are represented as 2D data; in which case you need not pay much attention to the contents of the QuickCap decks. Instead, use the gds2cap tool to convert 2D layout data to a 3D QuickCap deck. Most of the initial work involves creating a gds2cap technology file for this conversion process. The gds2cap documentation contains extensive information about the technology file.

After you analyze IC layouts, you still need to select nets to be extracted, decide whether individual coupling-capacitance values need to be reported (and which ones), and pick convergence goals.

3. Working With Uncertainty

A basic understanding of the uncertainty values that QuickCap provides helps you to use QuickCap efficiently. The ultimate goal is to find the circuit response to a given degree of accuracy, *not* to find capacitance values to a given degree of accuracy. This chapter contains information about how to use the QuickCap statistical estimates of capacitance.

Although this chapter is not an instruction guide for interpreting design goals and safety margins, it is important that any percentages used represent meaningful numbers that can be used in a manner consistent with the uncertainty values that QuickCap provides. For example, does a ±8 percent variation mean that the effects on capacitance are *always*, *usually*, or *normally* less than ±8 percent? This question is discussed in “Pitfalls” on page 3-11, after the concepts of statistical independence and confidence levels are explained.

Definitions

A statistical result is represented here by a pair of values: an estimate and its corresponding *uncertainty* (sometimes called a standard error), which is a calculation of one standard deviation of the estimate. The distinction between uncertainty and standard deviation is made to maintain a mathematically rigorous approach, but this distinction can generally be ignored for QuickCap results. A *standard deviation* describes the variation of a number of results that have been collected; whereas, an uncertainty is a calculation or prediction of that standard deviation without gathering multiple results. The uncertainty reported by QuickCap is compared with the standard deviation of many runs, and the values agree.

The following two results are equivalent:

25 fF ±2 fF

and

25 fF ±8%

The first shows the absolute uncertainty; the second shows the relative uncertainty. As shown in "Confidence" on page 3-6, from this result you can predict with 68 percent confidence that the capacitance (the result of averaging together an infinite number of similarly created estimates) is between 23 fF and 27 fF. The range 23 fF to 27 fF is a *confidence interval,* and 68 percent is its *confidence level.* You can predict with 95 percent confidence that the capacitance is between 21 fF and 29 fF, and with 99.7 percent confidence that the result is between 19 fF and 31 fF. To avoid confusion, values representing statistical errors are generally prefixed with ±. For example, ±70 percent is a statistical error; whereas, 68 percent is a confidence level.

Discrepancy between the result that QuickCap reports after infinite runtime and the true capacitance value is called *bias* and is due to approximations to the problem (1Å resolution, for example), truncation error, nonideal random-number generation, and so on. Bias is the statistical analogue of *error* in a deterministic calculation and can only be detected when it is larger than the uncertainty. Because the QuickCap bias on problems with analytic solutions is so small, it is quite difficult to detect. Thus, the QuickCap statistical error (described by the uncertainty) constitutes the only significant source of error. Deterministic capacitance extractors, on the other hand, can have significant numerical errors.

The rules given in this chapter apply to *independent* statistical estimates. They can be applied to results of different QuickCap runs, but only if the runs are independent. The values in a capacitance matrix are not truly independent of one another, but the dependence is weak except where the elements are symmetric or where QuickCap has applied symmetries of the problem as indicated in a symmetry file (see Chapter 7, "Symmetry").

It is important to distinguish between the uncertainty of a statistical estimate and an error. An error is caused by bugs, and computational limitations such as nonideal random-number generation or truncation errors, or any approximation that might be made to the problem being solved.

Deterministic methods have no statistical uncertainty. QuickCap has both statistical uncertainty and error (or bias); although, in cases where the uncertainty is larger than ±0.001 percent, any bias is difficult to detect.

Combining Independent Results

If you need a 5 percent result and that result is related to the sum of statistical estimates, the individual estimates need not be 5 percent estimates.

Consider two independent statistical results:

$$\text{Result A: } a \pm \sigma_a$$

and

$$\text{Result B: } b \pm \sigma_b$$

where *a* and *b* are the estimates, and σ_a and σ_b are the corresponding standard deviations, or uncertainties. Keep in mind that these are absolute values (such as ±6 fF), not relative values (such as ±10 percent). The sum is given by:

$$(a+b) \pm \sqrt{\sigma_a^2 + \sigma_b^2}$$

The difference is given by:

$$(a-b) \pm \sqrt{\sigma_a^2 + \sigma_b^2}$$

When adding or subtracting independent statistical results, the estimates add or subtract and, whether adding or subtracting, the uncertainty of the result is the *square root of the sum of the squares* of the uncertainties.

Example

Find the formula for the uncertainty of the sum of *n* values with the same uncertainty, σ.

Solution: If $n = 2$, the uncertainty is $\sqrt{\sigma^2 + \sigma^2} = \sqrt{2}\sigma$. If $n = 3$, the uncertainty is $\sqrt{2\sigma^2 + \sigma^2} = \sqrt{3}\sigma$. As a general function of *n*, the uncertainty is $\sqrt{n}\sigma$.

Example

Find the uncertainty of the total capacitance of two nets that have capacitances of 60 fF±6 fF (±10 percent) and 80 fF±8 fF (±10 percent).

Solution: The total capacitance is 140 fF±10 fF (or ±7.1 percent). (Use *absolute* uncertainties in the previous formulas.)

Example

A critical path consists of 16 interconnects, each of which introduces a delay that is about 0.5 ns with an uncertainty of ±0.1 ns (due to the uncertainty of the capacitance value). Find the total delay and its uncertainty.

Solution: assuming these time delays are just added, the total time delay is about 8 ns with an uncertainty of ±0.4 ns (using the results of the first example). Notice that the relative uncertainty of each of the 16 time delays is ±20 percent, but the relative uncertainty of the sum is ±5 percent.

This last example demonstrates that it is important to keep in mind that the results of a circuit analysis require a certain level of confidence, and that the level of confidence with respect to individual capacitance values is only secondary. QuickCap can find capacitance values in such a way that a weighted sum meets a specified uncertainty goal. This is a useful feature if you want to perform a timing analysis, for example, because it provides a mechanism for placing confidence levels on that analysis.

Weighted Sums

A time constant that is a function of several capacitors can have a lower uncertainty than you might expect given the uncertainty of the individual capacitance values.

Find the weighted sum of independent statistical results by multiplying each of the statistical results and absolute (not relative) uncertainties by a weight and then adding them together, as shown in "Combining Independent Results" on page 3-3. This is important with respect to interpreting the uncertainty of a delay time, for example, due to a series of interconnects. The delay time can be written as a sum of time constants, where each time constant is a capacitance weighted by an effective resistance:

$$t_d = R_1C_1 + R_2C_2 + R_3C_3 + R_4C_4 + \ldots$$

The *R*'s are effective resistance values that can be estimated (from parasitic net resistance, driver capabilities, characteristics of the logic family, and so on) well enough to establish the uncertainty of the time delay. It is not necessary to have the weights that give the same result as a delay-analysis tool. You are not trying to find the delay time. You are only trying to estimate its uncertainty. To be on the safe side, overestimate any effective resistance values given to QuickCap.

QuickCap can input these effective resistance values, and you can specify an extraction goal based on the weighted sum.

Example: Find the delay time of a critical path consisting of four interconnects with the following effective resistance values (estimated) and capacitance values (extracted)

$R_1 = 1\text{k}\Omega$	C_1 = 10fF± 5fF (±50%)
$R_2 = 1\text{k}\Omega$	C_1 = 10fF± 5fF (±50%
$R_3 = 5\text{k}\Omega$	C_3 =100fF±10fF (±10%)
$R_4 = 5\text{k}\Omega$	C_4 =200fF±10fF (±5%)

The individual time constants are

R_1C_1	=	10ps± 5ps (±50%)
R_2C_2	=	20ps± 8ps (±40%)
R_3C_3	=	500ps±50ps (±10%)
R_4C_4	=	1000ps±50ps (±5%)

Solution: Their sum is

t_d = 1530ps±71ps (±4.6%)

These are similar to the uncertainties that QuickCap generates for an accuracy goal, for example, of **5%@1.5ns**. If ±5 percent uncertainty for t_d is acceptable, the uncertainties for these capacitance values are also acceptable. If you need ±5 percent accuracy, however, and decide to demand ±5 percent uncertainty on all capacitance values, the extraction time becomes much larger. The time required to find C_3 increases by a factor of 4, the time required to find C_2 increases by a factor of 64, and the time required to find C_1 increases by a factor of 100. As is usual for Monte Carlo approaches, the uncertainty improves as the square root of runtime. Closely examine other codes before deciding whether this is a strength or a weakness.

The previous calculation is what QuickCap estimates the uncertainty on the timing analysis is and is based on estimates of the effective resistance values. Examine, numerically, the error introduced when the effective resistance values estimated might be high by up to 50 percent. Decreasing all resistance values by 50 percent scales the answers and does not affect the relative uncertainty. To analyze the worst-case relative uncertainty, decrease R_4 by 50 percent. The new time constants and their sum are as follows:

R_1C_1	=	10ps± 5ps (±50%)
R_2C_2	=	20ps± 8ps (±40%)
R_3C_3	=	500ps±50ps (±10%)
R_4C_4	=	500ps±25ps (±5%)

td = 1030ps±57ps (±5.5%)

The relative uncertainty on the timing analysis is slightly higher than the QuickCap estimate. The absolute uncertainty is smaller (because the effective resistance values were overestimated).

Confidence

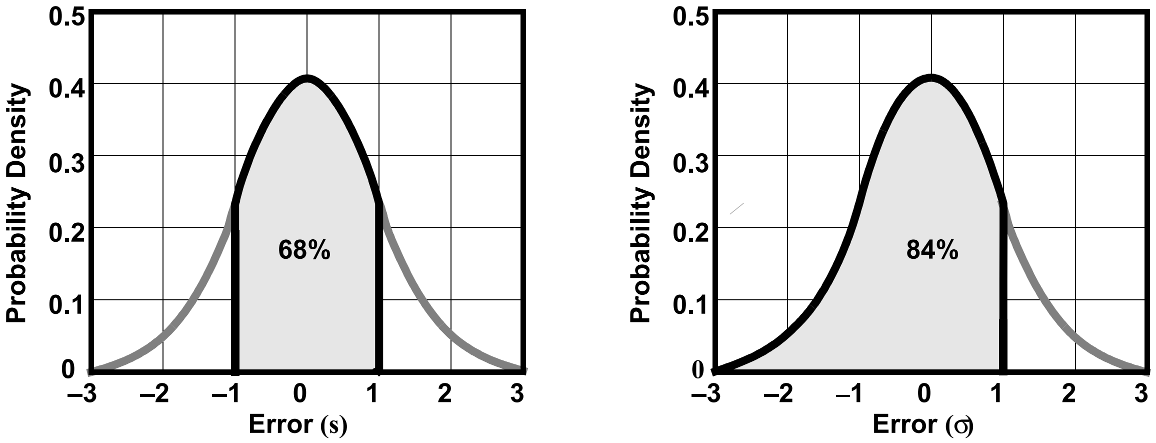
This section explains how to compare a statistical result with a deterministic one.

Establishing a confidence level requires you to make assumptions about the probability distribution of the estimate. However, when the estimate is a result of averaging many other estimates together, as QuickCap does, the probability distribution is very close to a classic *normal distribution* (also called a *bell curve*), which is fully defined by an average and a standard deviation. There is generally no more than a relatively small amount of bias in the QuickCap statistical estimate due to "error," and the QuickCap answer converges to the true capacitance as extraction time is increased. So, relationships between the true capacitance and the QuickCap estimate can be quantified as done in this section. If the uncertainty is small (less than about ±0.001 percent), the accuracy of QuickCap is limited by its bias.

A bell-curve distribution is assumed *only* to allow for quantifying confidence levels. The mathematical operations used elsewhere in this chapter do not require that any assumptions be made about the details of the probability distribution, except that it has an average and a standard deviation.

The graphs in Figure 3-1 show the confidence that a normally distributed statistical result is within σ of the correct answer (68 percent) and the confidence that it is not an overestimate by more than σ (84 percent). The σ denotes the uncertainty, or standard error.

Figure 3-1: Confidence Level



The confidence level is the probability that the error is within a confidence interval (shaded).

The following table shows the confidence levels for various confidence intervals. The σ denotes the uncertainty reported by QuickCap. These confidence levels can be less accurate for small capacitance values that have large relative uncertainties (larger than ±20 percent or so) when the capacitance estimates are the result of only a few significant walks. The percentages and the fractions are rounded.

Confidence Interval	**Confidence Level**
Not more than 1σ below the true capacitance:	84% (5/6)
Not more than 2σ below the true capacitance:	98% (49/50)
Not more than 3σ below the true capacitance:	99.87% (749/750)
Not more than 4σ below the true capacitance:	99.997% (1–1/30,000)
Not more than 5σ below the true capacitance:	99.99997% (1–1/3,000,000)
Within **1σ** of the true capacitance:	68% (2/3)
Within **2σ** of the true capacitance:	95% (19/20)
Within **3σ** of the true capacitance:	99.7% (349/350)
Within 4σ of the true capacitance:	99.994% (1–1/15,000)
Within 5σ of the true capacitance:	99.99994% (1–1/1,500,000)

These probabilities are based on the assumption that the QuickCap error (due to bugs, approximations, and so on.) is somewhat less than the reported uncertainty. Above uncertainties of about ±0.01 percent, no such error is detected. The 1σ, 2σ, and 3σ values in bold, commonly referenced, are worth memorizing.

Example: A statistical result is 90 fF ±10 fF and a corresponding deterministic result is 110 fF. What are the chances that the statistical result is the better estimate?

Solution: The statistical estimate is closer to the correct answer with a confidence level of 84 percent, the same as the confidence that the statistical estimate is not more than 1σ below the true capacitance.

Example: The QuickCap result is 90.00 fF±0.05 fF and the analytical result is 90.10 fF. Find the QuickCap bias.

Solution: The QuickCap error (or bias) is the difference, or 0.1 fF±0.05 fF. The confidence that you have found the bias is the same as the confidence that the QuickCap estimate is not more than 2σ low, or about 98 percent; and the confidence that QuickCap has a bias in between 0.05 fF and 0.15 fF is about 68 percent.

Self-Consistent Results

This section shows how to compare two statistical estimates of the same value.

To compare two statistical estimates of the same value, take the difference of the results. The difference of two independent *but equivalent* statistical estimates is a statistical estimate of zero. Whenever QuickCap combines statistical results before printing it also calculates the difference and tallies the ratio between (1) the statistical estimate of “zero” and (2) the uncertainty of that estimate. This result is called sigma. For result A, $a \pm \sigma_a$ and result B, $b \pm \sigma_b$, sigma is given by:

$$\frac{(a - b)}{\sqrt{\sigma_a^2 + \sigma_b^2}}$$

The average value of sigma is about 0, and the standard deviation is about 1.

Example: Are two estimates of the same value, 100±8 and 90±6, consistent?

Solution: The difference is 10±10 (the sigma is 1) and is a fair estimate of zero. The estimates are consistent.

Example: Are two estimates of the same value, 100±4 and 90±3, consistent?

Solution: The difference is 10±5 (the sigma is 2), which is a dubious estimate of zero. These are not consistent.

Averaging Results

This section shows that averaging independent statistical estimates improves the uncertainty.

If two independent statistical results are estimates of the same capacitor value, any weighted average is also a valid estimate of that value. Perform a weighted average by multiplying the first estimate by *T* (some value between 0 and 1), multiplying the second estimate by (1−*T*), and then adding the results. Note that when multiplying an estimate by a constant, its absolute uncertainty is multiplied by the same constant.

For two independent estimates of the same value ($a \pm \sigma_a$ and $b \pm \sigma_b$), the weighted average is:

$$(T)a + (1 - T)b \pm \sqrt{[(T)\sigma_a]^2 + [(1 - T)\sigma_b]^2}$$

For a “straight” (equally weighted) average ($T = 1/2$), the weighted average is:

$$\frac{a + b}{2} \pm \frac{\sqrt{\sigma_a{}^2 + \sigma_b{}^2}}{2}$$

Example: Find the average of the results from two different runs, 10±0.5 and 11±0.5.

Solution: The average is 10.5±0.35. Averaging results improves the uncertainty.

Example: Find the average of the results from two different runs, 10±0.5 and 11±1.1.

Solution: The average is 10.5±0.6. In this case, averaging independent results does *not* improve the uncertainty. 10±0.5 is a better estimate than 10.5±0.6.

The second example demonstrates that a straight average does not always improve the uncertainty. However, if you want to perform some additional math, because any weighted average works, you can find the weight *T* that minimizes the uncertainty. This optimum value, T_o, is given by:

$$T_o = \frac{\sigma_b^2}{\sigma_a^2 + \sigma_b^2}$$

The optimum weighted average is:

$$\frac{(\sigma_b^2)a + (\sigma_a^2)b}{\sigma_a^2 + \sigma_b^2} \pm \frac{\sigma_a \sigma_b}{\sqrt{\sigma_a^2 + \sigma_b^2}}$$

Example: Find the optimum weighted average for the results from two different runs, 10±0.5 and 11±0.5.

Solution: The optimum weight is T_o=0.5. The weighted average is 10.5±0.35.

Example: Find the optimum weighted average for the results from two different runs, 10±0.5 and 11±1.1.

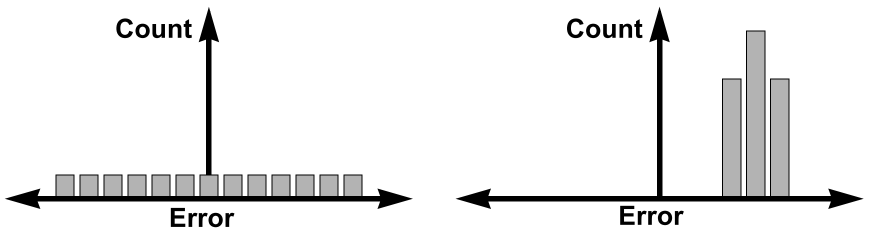
Solution: The optimum weight is T_o=0.83. The weighted average is 10.17±0.46. The uncertainty has improved.

Statistical Characterization

You can derive the root-mean-square (RMS) error of a population of estimates from one source using estimates from a second source when the standard error of the estimates from the second source is known. In particular, you can derive the standard error of layout parameter extraction (LPE) results or the results of another deterministic method by a calculation involving QuickCap results. This RMS error is a statistical characterization of a deterministic method. The statistics actually arise from the various layout “environments” that contribute to the capacitance of each net.

Like the statistical error that QuickCap reports, the RMS error represents one standard deviation of error. Unlike the statistical error that QuickCap reports, however, the distribution of error of a deterministic method cannot be fully characterized by such a value because it is not necessarily a bell curve. For example, the error distributions shown in Figure 3-2 have the same RMS error.

Figure 3-2: Two Error Distributions With the Same RMS Error



The following is the formula for the uncorrected normalized RMS error for the *n* data points:

$$\sigma_{XQ} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{C_{X,\ i} - C_{Q,\ i}}{C_{Q,\ i}}\right)^2}$$

where $C_{X,i}$ refers to a value supplied by another extractor and $C_{Q,i}$ refers to the corresponding QuickCap result. This is a *normalized* value because each error is normalized to the capacitance and an *uncorrected* value because it includes the statistical error of the QuickCap results. The *corrected* RMS normalized error is given by:

$$\sigma_X = \sqrt{\sigma_{XQ}^2 - \sigma_Q^2}$$

where σ_Q is the relative uncertainty of the QuickCap results.

These equations are approximate, but are reasonably accurate when the QuickCap statistical error is better than ±10 percent *and* somewhat better than the uncorrected RMS normalized error. For example, if σ_{XQ} is ±7 percent and σ_Q is ±5 percent, σX is ±5 percent. Naturally, the more data points collected, the better the results.

Pitfalls

A common mistake when dealing with statistics consists of combining or comparing statistical results that are not independent results. The less independent the results are, the more error that is introduced by combining or comparing them. This section gives some examples of this mistake.

Example: You have results from three different runs (10±0.5, 11±0.5, and 11±1.1 from "Averaging Results" on page 3-9, perhaps). You cannot combine the weighted average of the first and second with the weighted average of the first and third because these weighted averages are not independent. You can, however, combine the weighted average of the first and second with the third, or you can combine the first with the weighted average of the second and third.

Example: Neither the rules for combining statistically independent results nor the rules for establishing confidence can be applied to deterministic results. This is because sources of error, such as truncation or spatial discretization, can lead to errors that are correlated from one run to the next or even from one deterministic program to another. If enough studies are done so that the behavior of the error is characterized, it might be possible for a deterministic program to produce an indicator of confidence that can be used in a meaningful way. Averaging results from different programs, however, only guarantees that the error of the average is no worse than the error of the worst calculation.

Meeting Design Goals

A design goal might be specified as simply as “I need a delay time of less than 7 ns.” To compensate for errors introduced by any circuit-analysis approximations, such as treating capacitors between different nets as if they are attached to ground, you can add a margin of safety by restating the design goal. This new design goal might be, “I need a *calculated* delay time of less than 5 ns.” To compensate for variations in the fabrication process, worst-case fabrication parameters might be considered. In any case, you can constrain a design goal to minimum or maximum allowable calculated values.

The uncertainty that QuickCap provides means that no answer can be found to 100 percent confidence. You adjust for this effect by requesting 3σ values (99.7 percent confidence), 4σ values (99.994 percent confidence), or whatever provides a reasonable confidence level. For example, if “I need a calculated delay time of less than 5 ns” and the results of circuit analysis give 4 ns and are based on the QuickCap goal of **5%@5ns**, the uncertainty of the delay time due to the QuickCap statistical nature is less than 0.2 ns. Because you have a safety margin of 1 ns (five standard deviations), the confidence level that the fabricated circuit performs within specifications is about 99.99997 percent, assuming all other factors (design, fabrication, and circuit simulation) have a 100 percent confidence level. Even though the QuickCap uncertainty is ±5 percent, you are 99.99997 percent sure that the design goal has been met. A ±5 percent uncertainty in capacitance value does not imply a 95 percent confidence level.

Be careful of statements like “I want a ±10 percent result and I have a ±8 percent variation due to the fabrication process, so I need a ±2 percent capacitance calculation.” Recognize that the “±8 percent variation due to the fabrication process” might be a worst-case result that is ensured only by discarding any fabrication runs with more than ±8 percent variation. You need to compensate for this by using worst-case fabrication parameters (the highest dielectric constant, thinnest dielectrics, worst allowable mask alignment, and so forth.). Further, recognize that although the design might be within the design goal by only 2 percent, this imposes a ±2 percent bounds on the uncertainty of the circuit analysis, not necessarily a ±2 percent bounds on the uncertainty of the capacitance extraction of any capacitor.

With deterministic code, all you can generally do is add a generous margin of safety to the calculation, tightening the design goal considerably. Because QuickCap replaces the unknown error of deterministic codes with a known uncertainty, you can relax the design goal if the analyses give results that are within the design goal with an acceptable confidence level.

4. Extraction Procedure

QuickCap uses a Monte Carlo technique known as the *floating random-walk method*, which has speed and accuracy advantages over the classic fixed-grid random-walk method. Random-walk methods are, in general, fairly robust. Neither the number of geometric objects nor the dimensionality has a severe effect on the speed, accuracy, or required memory. In capacitance extraction, for example, the strength of the random-walk method is much more apparent now that capacitance extraction is often required for complex geometries, and now that 2D calculations are no longer sufficient.

QuickCap has limitations extracting capacitance in certain classes of problems, which are described in this chapter. Difficulty with realistic structures almost invariably stems from a poorly specified program parameter. Other difficulties can stem from analysis of unrealistic structures, such as dielectric constants of 10,000 or a capacitance between two 1μm cubes that are 100μm apart.

To explain the origin of these limitations, this chapter also outlines the steps in the extraction procedure.

Monte Carlo Analysis

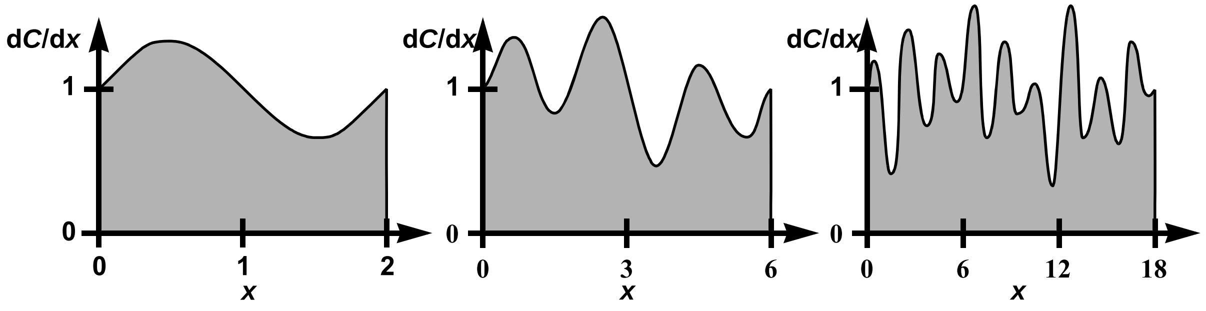
Monte Carlo approaches combine the predictable (deterministic equations) with the unpredictable (randomness). Monte Carlo analysis is useful for systems in which randomness or chaos causes any deterministic prediction to be useless. A less obvious, but equally valuable, advantage is that Monte Carlo analysis can be used to evaluate multidimensional integrals.

One application of the Monte Carlo technique involves ion implantation, where you might be interested in predicting the final position of a high-energy ion penetrating a solid. In this case, you simulate many ions in turn, randomizing the entry point of each one. The trajectory calculations, though deterministic, are quite chaotic—the number of interactions involved makes the final position sensitive to the last decimal place, rendering any deterministic result highly sensitive to the implementation. The final result, an implant profile, can be useful in IC technologies that are sensitive to details of the profile.

Another application involves calculation of the voltage at a point using a fixed-grid random-walk method. The governing equation, $\nabla^2 V = 0$, basically states that the voltage V at any point is the average of V at adjacent points. For a Monte Carlo estimate of V, use the voltage at an adjacent point, selected at random. If you do not know that voltage, use the Monte Carlo estimate of the voltage at the new point, similarly, by selecting at random a point adjacent to the new point. This “hop” process repeats until, eventually, the voltage is known, because the selected point is eventually on an electrode, where you have assigned a voltage. Thus, a random walk yields a single Monte Carlo estimate of the voltage at the initial point. This estimate can be refined by averaging the voltage over many walks. Like many deterministic approaches, this approach requires spatial discretization, which introduces error. Distinctly, however, this random-walk approach finds the voltage at a point without requiring that the voltage be calculated everywhere and without requiring memory to retain those voltages. The *floating* random-walk method, as used by QuickCap, is a variation of the fixed-grid random-walk method that does not require spatial discretization.

The floating random-walk method is related to Monte Carlo evaluation of integrals. Figure 4-1 shows examples of 1D integrals of the capacitance per unit length, $\mathrm{d}C/\mathrm{d}x = f(x)$, over the domains $0<x<2$, $0<x<6$, and $0<x<18$. These plots represent the capacitance per unit length of a 2μm-long wire with one crosswire, a 6μm-long wire with three crosswires, and an 18μm-long wire with nine crosswires. The total capacitance value is shown by the shaded area—approximately 2, 6, and 18 in the three integrals shown here.

Figure 4-1: Capacitance Represented as a 1D Integral, the Area Under a Curve



Using a Monte Carlo approach, you average a number of random samples of the function. Multiplying by the domain size (length, here) gives the capacitance. This is similar to the procedure you might use to estimate the area—estimating the average height and multiplying by the width.

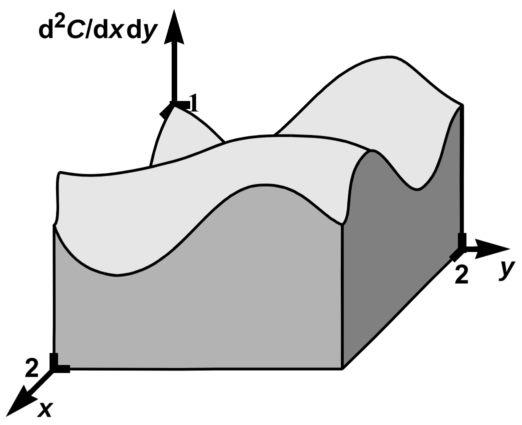
The following numerical example uses 10 Monte Carlo samples, which could have been collected from any of the functions shown:

- For each of $n = 10$ random values in the domain, values of dC/dx = f(x) are calculated: 1.39, 1.58, 1.14, 0.65, 0.75, 1.10, 1.42, 0.61, 1.57, 0.85.
- The average of these values is a_1 = 1.11.
- The average of the square of these values is a_2 = 1.35.
- The standard deviation is $\sigma = \sqrt{a_2 - a_1^2} = 0.35$.
- The uncertainty (standard error) is $\sigma / (\sqrt{n}) = 0.11$.
- Multiply by the domain size to get an estimate of the area: 2.22±0.22 (±10 percent), 6.66±0.67 (±10 percent), 19.98±1.98 (±10 percent).

Monte Carlo results are not sensitive to the noise spectrum of the function, nor are the results sensitive to the domain size. If these data points are collected over the domain $0<x<200$, the result is 222±22 (±10 percent). The domain size is only used as the multiplication factor in the final step. Because of this, the time required by QuickCap to extract the capacitance of a large net in a complex circuit is not much longer than the time required by QuickCap to extract the capacitance of a small net in a simple circuit.

You can readily apply Monte Carlo integration to higher dimensions. For a 2D integral, the domain size is an area. Because the average height of the function depicted in Figure 4-2 is approximately 1, the integral ($0<x<2$, $0<y<2$) is approximately 4. Monte Carlo integration is insensitive to dimensionality. The preceding numerical example could represent results from a 1D integral, a 2D integral, or a 1000D integral. Because of this, the 3D implementation of QuickCap is not much slower than an equivalent 2D implementation.

Figure 4-2: Capacitance Represented as a 2D Integral, the Volume Under a Surface



Floating Random-Walk Procedure

The mathematical basis of the floating random-walk procedure for capacitance extraction is based on formulating capacitance as a nested integral that contains no approximations. Capacitance C_{ab} between electrodes a and b is the ratio between the charge Q_a on electrode a and the voltage V_b on electrode b when all other electrodes have a voltage of zero.

The charge is found by finding the flux of the electric field E (scaled by the dielectric permitivity ε) through a surface. Mathematically:

$$Q_a = \int \int \varepsilon E(\boldsymbol{r}_0) \mathrm{d}A_0$$

The domain of the integral is a surface enclosing net a. This integration surface is arbitrary, but must enclose the electrode. Part of the QuickCap initialization time involves generating the integration surface for each net to be extracted.

$E(r_0)$ is the electric field at a point on the surface, consistent with the boundary conditions (the voltages on the electrodes). dA_0 is the incremental area on the surface. For a Monte Carlo estimate of this integral, select a random point r_0 on the surface, evaluate $E(r_0)$, and scale by ε and by the area of the surface.

$E(r_0)$, which must now be found, can be written as:

$$E(\boldsymbol{r}_0) = \int\int F_e(\boldsymbol{r}_1 - \boldsymbol{r}_0)V(\boldsymbol{r}_1)\mathrm{d}A_1$$

The domain of this integral is the surface of the largest electrode-free cube you can place around r_0. $F_e(r_1\text{-}r_0)$ is an analytic function. $V(r_1)$ is the voltage at a point on this surface (not the surface associated with the previous integral). $\mathrm{d}A_1$ is the incremental area associated with this surface. For a Monte Carlo estimate of this integral, select a random point r_1 on this surface, evaluate $V(r_1)$, and scale by $F_e(r_1\text{-}r_0)$ and by the surface area of the cube.

Find $V(r_1)$ using the following formula for $V(r_k)$ evaluated at $k = 1$:

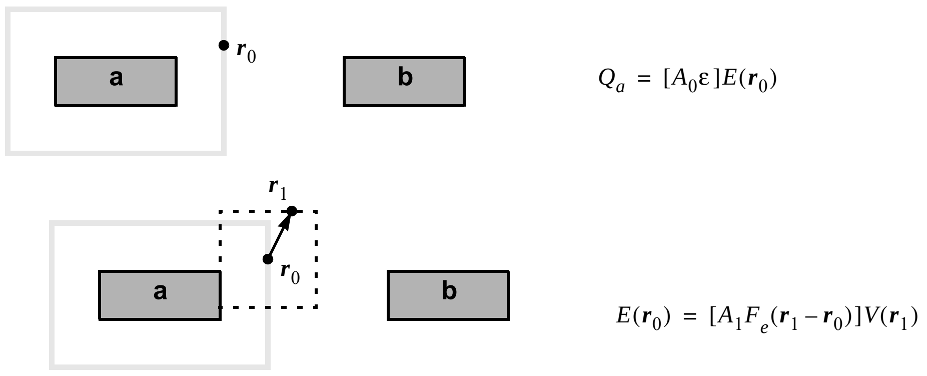
$$V(\boldsymbol{r}_k) = \int\int F_v(\boldsymbol{r}_{k+1} - \boldsymbol{r}_k)V(\boldsymbol{r}_{k+1})\mathrm{d}A_{k+1}$$

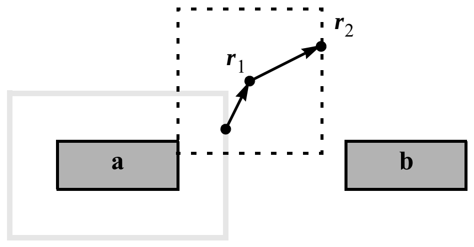
This domain is the surface of the largest electrode-free cube you can place around r_k. $F_v(r_{k+1}\text{-}r_k)$ is an analytic function. $V(r_{k+1})$ is the voltage at a point on this surface. $\mathrm{d}A_{k+1}$ is the incremental area associated with this surface. For a Monte Carlo estimate of this integral, select a random point r_{k+1} on this surface, evaluate $V(r_{k+1})$, and scale by $F_v(r_{k+1}\text{-}r_k)$, another analytical formula and by the surface area of the cube. For any point r_{k+1} on an electrode, $V(r_{k+1})$ is the voltage of that electrode. For any point that is not on an electrode, this integral is used recursively for successive iterations.

Evaluation of this nested-integral formulation for capacitance by using a Monte Carlo estimator is equivalent to a floating random walk consisting of a series of points: r_0, r_1, r_2, and so on. The size of each hop is limited only by the nearest electrode. Figure 4-3 shows one floating random walk, equivalent to the Monte Carlo evaluation of a tenth-order integral (each step is the evaluation of a second-order integral—a surface integral).

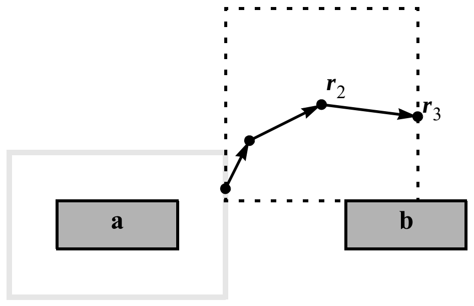
This nested-integral approach yields an exact formula for charge on net a as a function of the voltages on all electrodes in the system. The coefficients of the various electrode voltages are the capacitance terms in the column of the capacitance matrix associated with net a.

Figure 4-3: Sample Floating Random Walk

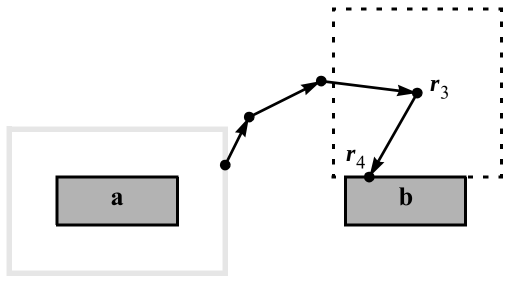

a
b
r_0
$Q_a = [A_0\varepsilon]E(\boldsymbol{r}_0)$
a
b
r_1
r_0
$E(\boldsymbol{r}_0) = [A_1F_e(\boldsymbol{r}_1 - \boldsymbol{r}_0)]V(\boldsymbol{r}_1)$



$$V(\mathbf{r}_1) = [A_2 F_v(\mathbf{r}_2 - \mathbf{r}_1)]V(\mathbf{r}_2)$$



$$V(\mathbf{r}_2) = [A_3 F_v(\mathbf{r}_3 - \mathbf{r}_2)]V(\mathbf{r}_3)$$



$$V(\mathbf{r}_3) = [A_4 F_v(\mathbf{r}_4 - \mathbf{r}_3)]V_b$$

In the random walk pictured here, the charge on net a is estimated as follows:

$$Q_a = V_a \times 0 + V_b \times [A_0 \varepsilon][A_1 F_e(\mathbf{r}_1 - \mathbf{r}_0)][A_2 F_v(\mathbf{r}_2 - \mathbf{r}_1)][A_3 F_v(\mathbf{r}_3 - \mathbf{r}_2)][A_4 F_v(\mathbf{r}_4 - \mathbf{r}_3)]$$

The estimated capacitances from this single Monte Carlo sample of the exact capacitance integral are:

$$C_{aa} = 0\,, \qquad C_{ab} = [A_0 \varepsilon][A_1 F_e(\mathbf{r}_1 - \mathbf{r}_0)][A_2 F_v(\mathbf{r}_2 - \mathbf{r}_1)][A_3 F_v(\mathbf{r}_3 - \mathbf{r}_2)][A_4 F_v(\mathbf{r}_4 - \mathbf{r}_3)]$$

These values converge to the correct capacitance values as other Monte Carlo samples are averaged in.

QuickCap Implementation

You can normally run QuickCap with an option list and the name of a file defining the problem. QuickCap first confirms that the license file is valid, starts recording elapsed time, and then processes the command-line input.

Generation of the integration surface is important and several QuickCap commands are related to it. *Manhattan* refers to an object with each face in the *xy, xz,* or *yz* plane. QuickCap approximates non-Manhattan dielectrics as a superposition of Manhattan dielectrics. QuickCap also uses a Manhattan fill to generate an integration surface around a non-Manhattan net.

Runtime

Runtime is expressed in terms of elapsed time, but it does not include time during which QuickCap awaits user input.

Canceling a Job

If you want to cancel a QuickCap run, press Control-c if QuickCap is running in the foreground, or enter **kill -INT *PID*** if QuickCap is running in the background.

The Floating Random-Walk Kernel

The floating random-walk kernel gathers random-walk statistics, but it does not interpret the statistics. That function is performed when the summary is computed. A single floating random-walk consists of a series of *hops* (steps). Each hop is from the center of a cube that does not contain conductors onto a point on the surface of that cube that is randomly selected according to probability rules dictated by the exact solution (limited by the numerical precision of the computer) of the governing Laplace equation. Unlike a fixed random-walk method, which occurs on a fixed *walk grid*, there is no discretization error due to a walk grid. Also, when the nearest conductor is far away, a floating random-walk can take an appropriately large step. The walk continues until it reaches a conductor.

Walk Rate

The time required to perform a walk is the time required to perform a hop multiplied by the number of hops in the walk. The time required to perform a hop is not a strong function of the number of geometric primitives (boxes, spheres, and so on) in the problem, but it might increase by 30 percent when a *large* fraction of the problem is non-Manhattan. The number of hops is most strongly affected by the dielectrics, because dielectric interfaces limit the size of the hop independent of the proximity of a conductor on which to terminate. Also, floating (dummy) metal can slow convergence when the float distance is small. In this case, floating metal can trap a hop as would a high-valued dielectric.

Convergence Rate (Per Walk)

A column of the capacitance matrix is calculated from the walks that originate on the surface enclosing a single net. The estimate of capacitance to any net in the column evolves from the walks that touch that net. The contribution to capacitance from a single walk depends primarily on the first hop. Specifically, it is inversely related to the size of the first hop. When the first hop of a walk is small, convergence is slowed. If the first hop is small because of a nearby conductor, most walks end on that conductor quickly, significantly compensating for this effect. When the first hop is small because of a nearby dielectric interface or because of nearby floating metal, however, there is no such compensation.

Limitations Regarding Dielectrics and Floating Metal

Because the time required to calculate a walk is most sensitive to dielectrics and because small initial hops are often caused by nearby dielectric interfaces, most limitations of QuickCap are related to dielectrics. The ability to handle floating metal is included in QuickCap 4.0 and later with the **float** declaration. Floating metal behaves like dielectric material with a large dielectric value.

For information about dielectrics, see “Dielectrics” on page 6-18. For information about floating metal, see “Floating (Dummy) Metal” on page 6-18.

The following are characteristics of a problem that might create difficulties:

- Large dielectric constants

  Dielectrics with large dielectric constants might trap a walk. This is especially true if no conductors touch the dielectric. In such cases, the convergence rate becomes slow because each walk requires a large number of hops. The convergence rate per walk might be the same as for a similar problem without the large dielectric constant, but the number of walks per second and the convergence rate per elapsed second are lower.

- Floating metal

  Floating metal is similar to a dielectric with a large dielectric constant. Use QuickCap test runs to determine whether this value yields sufficiently accurate results.

- Many small dielectric regions

  Structures with many small dielectric regions can significantly increase the number of hops per walk. This is true even if the relative dielectric constants are small. In general, thin dielectric layers can be averaged together. You can find the effect of such averaging by running some test problems with and without the averaged layers.

- Non-Manhattan dielectrics

  QuickCap approximates non-Manhattan dielectrics by a superposition of Manhattan boxes. When a walk enters this filled region, the advantages of a floating random-walk routine are compromised and the walk can become relatively slow.

- Non-Manhattan floating metal

  QuickCap approximates non-Manhattan floating metal by a superposition of Manhattan boxes, also influenced by the fill size.

- Dielectric resolution

  Any dielectric corner increases the number of hops per walk for walks that approach the corner. If this is set too small, the amount of time that a walk spends near the corner of a dielectric box can increase significantly. A reasonable value is about 1/100th of the *scale of the problem* (the approximate spacing between the nearest nets). If the dielectric structure is planar (no corners), the dielectric resolution can be the same as the spatial resolution, which defaults to 1Å. This parameter needs to be smaller than half the size of the thinnest or narrowest dielectric in the problem.

- Minimum initial jump

  The minimum initial jump is a length parameter that places a lower limit on the first hop length. When the first hop (which originates on the integration surface) is nearer a dielectric interface than this distance, the walk proceeds as if it started on the dielectric interface. This effectively blurs the position at which QuickCap calculates the E-field near intersections of the integration surface and a dielectric interface. If this value is too small, convergence is slowed even though the number of hops per second is not affected. If this value is too large, errors related to the position of the dielectric interface can no longer be negligible. Because the integration surface is generally not near any conductors, the electric field on this surface does not change rapidly with position, and values approximately 1/10th of the scale of the problem are reasonable. To experimentally check on reasonable ranges for this parameter, try several different values with a sample problem.

- Distant nonplanar dielectric

  A distant nonplanar dielectric (far from the nearest conductor) slows walks that happen to approach, because the hop length is limited by the distance to the dielectric. *Planar* dielectrics, defined by the **eps … up to** statement (see page 6-20), do not pose such a problem.

Other Limitations

Limitations other than those regarding dielectrics and floating metals are generally related to extremes (huge problems or tiny capacitance values) or to structures with a capacitance that is very sensitive to fabrication variation. The following are characteristics of a problem that might present difficulties.

- Diffusion regions outside of device regions

  Diffusion regions passed to QuickCap might touch or intersect the groundplane. In this case, because virtually every net in a realistic layout includes diffusion regions, every net touches or intersects ground. Unless you specify **proximityCheck off**, as QuickCap generates integration surfaces, it lists extracted nets that touch or intersect the groundplane or any other net if the proximity point (a point halfway between the objects) is outside of all device regions (**deviceRegion**).

- Surface or net conflict

  If the **spacing** parameter of a **layer** declaration is too large (see page 6-23), a surface or net conflict can arise during extraction if you are using **snapSurfaceToLayers on** in conjunction with **proximityCheck off** to minimize startup time. The **spacing** parameter indicates the Manhattan distance between objects. If 45° wires are allowed, the Manhattan spacing needs to be about a factor of two below the design rule spacing.

5. The Command Line

This chapter describes the QuickCap command-line format, and how you can view description of a command and its available options. It also discusses runtime and convergence-goal options. This chapter also describes common command-line options with examples. Finally, this chapter discusses environment variables and their effect on QuickCap operation.

Command-Line Format

The following is the format for running QuickCap on a system:

quickcap [**-license**[**s**]] [**-v**[**ersion**]] [***options***] [***file_list***]

Without any arguments, QuickCap prints a short description and a list of recognized command-line options.

The **-license** option causes QuickCap to print the license information. The **-version** option causes QuickCap to print the full path name of the executable and the build date, and then exit.

Specifying more than one file is equivalent to concatenating them.

Convergence-Goal Options

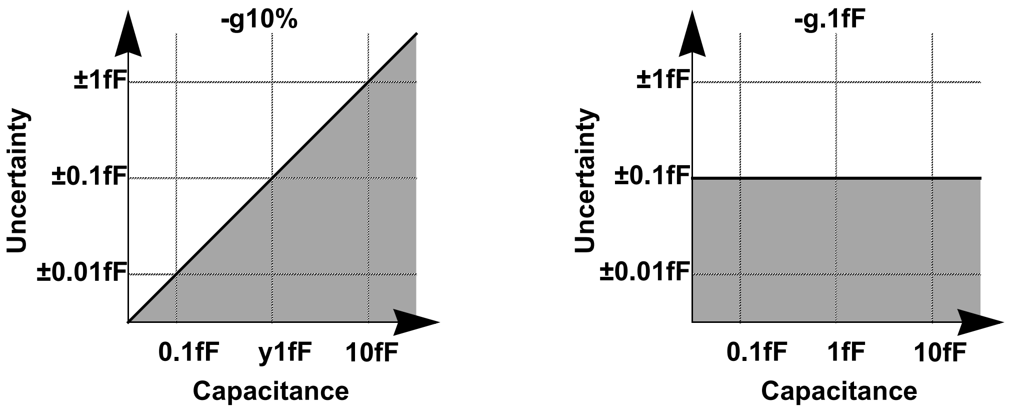
Recognized convergence goals fall into two categories: *value* goals and *sum* goals. When you specify both types of goals, the category that is less strict is used, which can vary from result to result.

Convergence goals use percentages and units of capacitance. The valid limits for capacitance are **F**, **mF**, **uF**, **nF**, **pF**, **fF**, and **aF**.

Value goals are relative or absolute (absolute capacitance or absolute RC). When you specify both kinds of absolute goals (capacitance and RC), the stricter one is used, which depends on R for each net. When you specify both a relative goal and an absolute goal, a result is considered converged when either goal is met. (See Figure 5-1.)

Sum goals are based on the sum of arbitrary capacitance, RC, or crosstalk values. When you specify multiple sum goals, the most strict goal is used.

Figure 5-1: Value Goals Showing Converged Results (Shaded) for Relative and Absolute Capacitance Goals



For distributed RC calculations, goals are applied to the combined capacitance values of nodes on a signal, even when the **-nodes** option is used to print values for separate nodes. This is because any circuit simulation includes statistical-error reduction that is almost the same as you would get by adding the node capacitances.

Note: For a signal containing multiple nodes, goals apply to the sum of capacitance across the nodes.

-d[**iag**] ***pct***
-g[**oal**] ***pct***

Sets a relative goal. Examples of percentages are **10** or **2.5%** (the percent symbol is optional). The **-g** goals apply to all matrix elements. Except for off-diagonal matrix elements, the goal is based on the total capacitance, including any device capacitance.

-d[**iag**] ***capacitance***
-g[**oal**] ***capacitance***
-d[**iag**] ***pct@capacitance***
-g[**oal**] ***pct@capacitance***
-d[**iag**] ***pct1@value..pct2***
-g[**oal**] ***pct1@value..pct2***

Sets a *composite* goal, which is an extension of the associated sum goal (***pct@value***) with minimum and maximum relative goals. One breakpoint is ±***pct1*** at ***value***; the other is ±***pct2*** at ***value***$\times$ (***pct1***/***pct2***)^2. Within the range ***pct1*** to ***pct2***, the convergence goal is based on a sum (**-g** ***pct1@value***). Outside this range, a relative convergence goal is used: ***pct1*** on one side and ***pct2*** from the other.

When you specify multiple sum goals, all goals must be met for a value to be considered converged. For multiple composite specifications, the minimum percentage goal must be identical. Similarly, a relative goal (***pct***), if specified, must also match the minimum percentage goal.

Other Options

This section describes commonly used command-line options in QuickCap.

-0

Ignores any state file. Unless the **-stateFile** option is specified, any existing state file is overwritten. QuickCap reads files named on the command line.

When you do not specify **-0**, if there is a state file corresponding to the file list, the state file is used. QuickCap prints a warning if any of the QuickCap decks are modified since the state file was created.

-asymmetric

Generates asymmetric results. Normally, QuickCap averages together the symmetric capacitance elements it extracts. The **-asymmetric** option bypasses this step, providing independent results, for example, for the capacitance from A to B compared to the capacitance from B to A. This is useful for checking the amount of asymmetry introduced by device regions.

-fit[**Data**] [**#*netDataLabels***]

Finds coefficients to fit net data to report QuickCap results for the total capacitance on each net. The **-fit** option is ignored if no net data exists. The **-fit** option minimizes the RMS value of the *absolute* error. With **-v**, a histogram is also provided. When you invoke **-matrix** and you do not specify **-xFit** nor **-%xFit**, the fit is performed for all matrix elements as well as for the self-capacitance. You can specify the **-fit** option multiple times.

The **-fit** option can include a list of **netData** labels over which to perform the fit. Such a list is indicated by an initial number sign (#) followed by a comma-delimited list of labels (no embedded spaces). QuickCap ignores the fit *unless* each label is specified in a **netData** declaration in the QuickCap input deck. For example, the **netData** declaration defines labels *L*, *W*, and *A*. Consider the following command line.

```
quickcap -fit -fit #L,W -fit #A ...
```

This generates three parameter fits to the total capacitance: a fit of all net parameters, a fit of *L* and *W*, and a fit of *A*.

The fitting procedure invoked by the **-fit**, **-xFit**, **-%fit**, or **-%xFit** option incorporates software subroutines from the second edition of *Numerical Recipes in C: The Art of Scientific Computing,* by W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, published by Cambridge University Press, 1992. The software is used with the permission of Numerical Recipes Software.

-log[**File**]
Does not create or modify any log file. Use the **-log** option to reduce the number of files output. The log file is described in “The Log File” on page 8-2.

-LSF *n*
This parallel capability is available in the QuickCap product. The **-LSF *n*** option specifies the number of hosts for parallel operation of QuickCap using LSF capability. QuickCap invokes the **-LSF bsub** option to spawn remote programs. The **bsub** argument can be customized by setting the environment variable **LSF_STR**. For example, to run on specific platform and queue, use:

```
setenv LSF_STR \
    '/tools/lsf/6.0/linux2.4-glibc2.2-x86/bin/bsub -N -q longq -R \
     select[type==LINUX72]'
```

The **-LSF** option supports Sun Grid Engine when the **LSF_STR** environment variable includes the string *qsub*.

-matrix [***int***]
Prints the full capacitance matrix. The optional integer specifies the number of columns to print within the page width. If you do not specify ***int***, the output is a simple list or four columns, whichever generates fewer lines of output. For **-v2** or higher, the data is printed as a list, whether you specify ***int*** or not, so that additional information related to each element can be printed. The **-matrix** option affects the netlist and numeric output file, if any.

If you do not specify **-matrix**, QuickCap prints the Miller capacitance for each net. The *Miller capacitance* of a net is the sum of the capacitances to other nets, each weighted by a *Miller factor,* usually 1. You can specify values other than 1 using a gain file. If all factors are 1 (all gain terms are 0), this is the same as the self-capacitance or the total capacitance.

The matrix consists of one column for each extracted net, and a row for the following:

- Each extracted net
- Each net mapped to **self** (unless you specify **-groundMisc)**
- Mapped capacitance (if you specify **-map** *and* if there is mapped capacitance)
- Ground
- Spurious capacitance (if you specify **-spurious** *and* if there is spurious capacitance)

If you specify **-groundMisc**, capacitance to unextracted or unmapped nets is added to the capacitance to ground. Unless you specify **-spurious**, any capacitance to the spurious net is added to the capacitance to ground. A row of the capacitance matrix is not printed if all elements to be printed are zero.

Table 1 lists examples that illustrate the use of the **-spurious** and **-groundMisc** options with the **-matrix** command-line option. In this case, there is a ground and three nets a, b, and c. Nets a and b are extracted, and c is mapped to **self**. By default, any spurious capacitance is added to the ground capacitance, and capacitance to c (the only unextracted net here) is printed.

Note: Without **-matrix**, only C_{aa} and C_{bb} output appears. Without **-spurious**, spurious capacitance is added to ground capacitance. With **-groundMisc,** miscellaneous capacitance is added to ground capacitance.

Table 1: Examples of Matrix Output

-matrix			**-matrix -groundMisc**		
	a	**b**		**a**	**b**
a	C_{aa}	C_{ba}	**a**	C_{aa}	C_{ba}
b	C_{ab}	C_{bb}	**b**	C_{ab}	C_{bb}
c	C_{ac}	C_{bc}			
ground	$C_{a0}+C_{as}$	$C_{b0}+C_{bs}$	**ground**	$C_{a0}+C_{ac}+C_{as}$	$C_{b0}+C_{bc}+C_{bs}$
-matrix -spurious			**-matrix -spurious -groundMisc**		
	a	**b**		**a**	**b**
a	C_{aa}	C_{ba}	**a**	C_{aa}	C_{ba}
b	C_{ab}	C_{bb}	**b**	C_{ab}	C_{bb}
c	C_{ac}	C_{bc}			
ground	C_{a0}	C_{b0}	**ground**	$C_{a0}+C_{ac}$	$C_{b0}+C_{bc}$
(spurious)	C_{as}	C_{bs}	**(spurious)**	C_{as}	C_{bs}

-MPP ***n***

The **-MPP** option specifies the number of parallel processes to be spawned. Without the **-MPP** option, QuickCap runs on one processor.

-nodes

Prints the capacitance to each node on a signal, rather than the capacitance to a signal. Similar to the netlist output file (**-spice**), the flags **-nodes** influence which values are output. When **-nodes** is *not* specified on the command line, signal capacitance rather than node capacitance is output. Unlike the netlist file, capacitance values and uncertainties are *not* used to filter which values appear in the numeric file. All values are output.

-nodePostfix on|off

Specifies whether QuickCap applies a numerical correction to node coupling capacitance. The default value is determined by the **nodePostfix** input-file flag and has a default value of **on**. For more information about the numerical correction, see the **nodePostfix** input-file flag description on page 6-10.

-num[**eric**]

Prints a numeric output file when QuickCap terminates. For information about the numeric output file, see “The Numeric Output File” on page 8-2.

-p

Inhibits printing to the standard output. The summary and log files are updated as usual. The option is useful for executing QuickCap in the background.

-partition [**width**[,**margin**[,**fringe**]]] (Defaults: **20**,**5**,**1**)

Extracts capacitance using partitioned QuickCap runs. The **width** value specifies the approximate size of a partition (in microns). QuickCap decreases the partition size independently in x and y so that all partitions are the same size. The **margin** value specifies the amount of geometry included from the neighboring partition to limit stitching error. The **fringe** value specifies the boundary added to the periphery of the layout to ensure that the integration surfaces are included.

QuickCap first reads from the input files looking for the bounds for the layout bounds, which gds2cap includes as parameters x0, y0, x1, and y1. QuickCap then generates a partition directory named ***root***.partitions. An alternate name can be specified by setting the **QUICKCAP_PARTITION_DIR** environment variable.

QuickCap generates a .state file and schedules the independent partition runs. Without the **-LSF** option, the partition runs occur sequentially on the same host. With **-LSF *nProcs***, the independent partition runs are run in parallel, up to ***nProcs*** at one time. Specifying **-LSF** without ***nProcs*** (allowed for **-partition** runs) schedules all partition runs. The number of simultaneous processes running might be limited by the LSF system or by licenses. As for other parallel QuickCap methods (**-MPP**, for example), the independent runs use QUICKCAP_MCPU licenses if available. With **-LSF,** the parent QuickCap program displays the status of each partition on a map: unscheduled, scheduled, starting up, running, and completed. The parent QuickCap program also prints an estimate of the completion time.

Each independent partition run maintains its own .state file. An interrupted partition run can be continued, using previously saved data.

The partition runs use data from the *totalC* file, named ***root***.totalC, when available. When the totalC file exists, QuickCap uses each capacitance value listed as the effective total capacitance of the net to optimally converge on the partial (partitioned) capacitance values. If such a file does not exist, QuickCap performs an initial partition run with a relaxed accuracy goal specified using the **-totalC** option (see page 5-10). A subsequent partition run with the desired accuracy goal then converges efficiently.

-pct[**Xtalk**] ***pct***

Specifies a crosstalk fraction, below which an off-diagonal term is mapped to ground or to a fictitious net (if you specify **-map**). For nodes, mapping is determined by comparing the coupling and total capacitance of the entire signal. If you specify **statistics self** in the input file, **-pct** is ignored. Otherwise, **-pct** results in matrix output, invoking **-matrix** if necessary. The ***pct*** value can be a decimal value. A worst-case crosstalk fraction associated with each off-diagonal term is assumed—the ratio of the off-diagonal term to the smallest associated diagonal term. For example, C_{AB} is compared with the smallest of C_{AA} and C_{BB} if both terms exist (that is, if both nets A and B are extracted). The **-pct** option requires full matrix statistics.

-r[**elax**]

Relaxes the severity based on the following fatal conditions:

A. Reading a zero-width polygon, a dielectric, an ignore volume box that has zero volume, or an electrode box that describes a point or line segment.

B. Reading a very narrow polygon.

C. Finding no nets to extract. This condition already results in a normal exit in a windowed run.

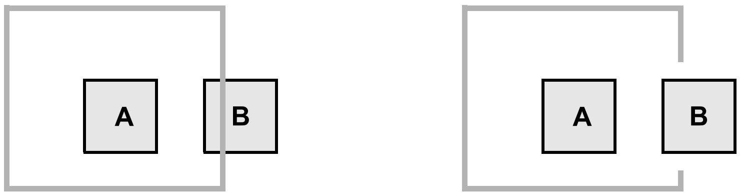
D. Starting a walk on an integration surface that is inside a net or within the **junction** distance (see page 6-16) from the net.

Without **-relax**, any of these conditions causes QuickCap to terminate. With **-relax**, QuickCap prints a warning message and continues execution.

The third case, an integration-surface or net conflict, can arise when QuickCap fails to clip the integration surface near a net. This can occur when non-Manhattan structures are poorly filled (either not filled or filled with a large **fill** size, when **proximityCheck off** is specified, or when QuickCap data includes hierarchical structures). The first two causes must be avoided. An integration-surface or net conflict due to hierarchy is inherent in hierarchical analysis and merits the **-relax** option.

In Figure 6, the integration surface around one net can intersect another net that is within **range** if **proximityCheck** is **off**. This condition can also arise for hierarchical objects. This normally causes QuickCap to cancel when a random walk begins within junction of net B. Specifying **-relax** causes such walks to be ignored, resulting in the effective integration surface as shown on the right in Figure 6.

Figure 6: Using the -relax Option That Effectively Clips the Integration Surface



[redacted] [redacted]

-seed <*int*>

Specifies the starting random seed. The option is used to provide reproducibility, when required.

-spice [***smallCaps***|***absLimit***|***relLimit***]]

Outputs a netlist containing the extracted capacitance values. This file is produced even if you interrupt QuickCap. For information on the capacitance netlist, you can specify a minimum capacitance (with units of capacitance), a maximum relative uncertainty (optionally with a **%**), or both. *Small capacitance values*—values less than the minimum capacitance or with a relative uncertainty greater than the maximum relative uncertainty—are filtered. Small coupling-capacitance terms are lumped to ground. Small self-capacitance terms are discarded. Small nodal terms (for nodes on a signal) are discarded.

The **-spice** command-line option also recognizes the keyword **discardSmallCaps**, **groundSmallCaps**, or **distributeSmallCaps** as an argument, taking precedence over any **netlistSmallCaps** statement in the input deck. Any or all arguments can be specified, separated by commas with no intervening space.

The netlist file can be customized through declarations in the input file or in the preference file.

-spiceFile *fileName*

Generates or updates the named netlist. A gds2cap-generated QuickCap deck includes the name of the netlist for QuickCap to update. The **-spiceFile** option allows QuickCap to generate or update a different netlist. This option does not change the name of the netlist stored in the state file. A later QuickCap run without the **-spiceFile** option updates the netlist named in the original QuickCap deck.

-spNum[**eric**]

Generates a numeric file, named ***root***.*sp.numeric*, after generating a netlist. The QuickCap tool adjusts capacitance values as necessary to avoid generating negative capacitances in the netlist. The **-spNumeric** option generates a numeric file with these adjusted values. The option causes QuickCap to generate or update a netlist even if the **-spice** option is not specified on the command line.

-spTab[**le**]

Generates a tab-delimited file, named ***root***.*sp.tab*, after generating a netlist. The QuickCap tool adjusts capacitance values as necessary to avoid generating negative capacitances in the netlist. The **-spTable** option generates a tab-delimited file with these adjusted values. The option causes QuickCap to generate or update a netlist even if the **-spice** option is not specified on the command line.

-summary

Does not generate summary information. If **-summary** is not specified, a summary file is generated. All coupling capacitances are reported. If **-summary** is not specified but **-asymmetric** is specified, asymmetric capacitances are reported in .summary file.

-tab[**le**]

Generates a tab-delimited output file named *root* .tab just before QuickCap terminates. The file consists of initial comments and a tab-delimited header line, followed by tab-delimited data. The output format is described in "Table Output" on page 8-6.

-totalC [***convergenceRelaxationFactor(s)***]

Generates a *totalC* file named ***root***.totalC, which lists the total capacitance of each net, including device (**Cd**) and device parasitic (**Cdp**) components. QuickCap uses the *totalC* file during a tiled run (**-tiled)** or partition run (**-partition**) to optimally converge on partial net capacitances.

Use **-totalC** on a tiled or partitioned run with a relaxed goal, **-g15%**. For example, a subsequent run with a tighter goal converges based on relatively accurate capacitance values. Without the totalC data, each tiled or partitioned run converges based on any gds2cap-generated **Cp** values, which are generally less accurate.

Note: Using inaccurate Cp values result in statistical uncertainties that are low or high, not resulting in inaccurate QuickCap results.

In tiled or partitioned runs, the achieved goal differs from the requested goal by a factor of the square root of the ratio of the estimated and actual total capacitance values.

In a partition run (**-partition**), the **-totalC** option accepts a comma-delimited list of one or more monotonically decreasing integers (*convergence-relaxation factors*), 100 to 1. QuickCap then runs with each relaxation factor to produce accurate *totalC* results. QuickCap always includes a final run with a relaxation factor of 1, whether or not it is specified. The totalC results are used in a partition run to optimize convergence.

Example

Consider a convergence goal **-d1%** with **-totalC 30,8,2** or equivalently **-totalC 30,8,2,1**. QuickCap first runs with a goal of **-d30%** and outputs totalC results. QuickCap then runs with a goal **-d8%**, using the previous totalC results for a moderate goal (QuickCap might over or under converge by about 30%, reaching goals of **-d5%** to **-d11%** on various nets.) Next, QuickCap run is done effectively with **-d2%**, using the latest totalC results. The final **-d1%** QuickCap run is efficient because it uses accurate totalC values.

-verbose [***int***]

Produces more output according to ***int***, the verbosity level. At any level of verbosity, the output includes the information of all lower levels. The following levels are valid:

-verbose

Causes QuickCap to print to the terminal the capacitance summary each time it regenerates a summary file.

-verbose 1

On an initial run, prints the size of the state file with the following information:

- Physical description

- Integration surface

- Statistics

Prints equivalences from the equivalence file, if any. Prints Miller relationships from the gain file, if any. Includes in the summary, the number of walks, the walk rate, additional decimals, the least-converged value, and how concentrated the walks are (1 = uniform). Includes a description of each graphics entry in the ***root***.xRec file (**-xRecord**) as a comment.

-verbose 2

On an initial run, prints any **node**, **net**, and **dielectric** commands. Prints capacitances in *long format*, one value per line.

-verbose 3

Prints the number of walks contributing to each capacitance value. A QuickCap partition run (**-partition**) generates a *child log file* in each partition directory with a suffix of **.child.log**. The child log file describes interactions between the (remote) partition run and the main QuicKCap run. The child log file can be used to track unexpected behavior of a partition run.

-verbose 4

Prints a checksum for each net. The checksum is an estimate of “zero” based on a column of the raw capacitance matrix. Typically, it has an uncertainty on the order of the estimate.

-verbose 5

On an initial run, prints information that could be useful for debugging.

-x

Enables the graphics preview mode. After QuickCap inputs the structure, it displays the top view of the structure in a graphics window. Use the mouse and various keys to change the view, color, net shown, and so on. For more information, see Chapter 9, “Graphics Preview Mode”.

-xRecord

Invokes graphics preview, like **-x**, but creates a record of the keyboard and mouse entries by the user, and starts recording in a separate file a log of graphics commands used to generate the drawing data. The entry-record file (containing keyboard and mouse entries) is named ***root***.xRec. The log file of graphics commands sent to the terminal is named ***root***.xLog.

By changing or copying the entry-record file from ***root***.xRec to ***root***.xPlay and running QuickCap with the **-x** or **-xRecord** option, QuickCap replays the original keyboard and mouse entries until the end of the file. Any user-generated keystroke or mouse click, however, ends the replay.

Graphics commands designed to be used in conjunction with this option allow introduction of short and medium pauses into the entry-record file ("**,**" and "**;**"), and allow the recording to be terminated ("**.**"). Also, the log file of graphics commands can be closed ("**)**"), and a new one can be started ("**(**"). For more information, see Chapter 9, "Graphics Preview Mode".

-

As the last option, allows the first file name in the file list to begin with a dash (**-**) or a digit.

Environment Variables

The environment variables listed in this section affect the operation of QuickCap.

setenv LSF_STR *string* (Default: **bsub**) **-LSF** *option*

Customizes the **bsub** argument invoked by **-LSF**. For example, to run on a specific platform and queue use the following syntax:

```
setenv LSF_STR \
     '/tools/lsf/6.0/linux2.4-glibc2.2-x86/bin/bsub -N -q longq -R /
select[type==LINUX72]'
```

The **-LSF** option supports Sun Grid Engine when the **LSF_STR** environment variable includes the string *qsub*.

setenv QUICKCAP_DISPLAY_HEIGHT *yPixels* *Graphics*
setenv QUICKCAP_DISPLAY_WIDTH *xPixels*

If defined, decreases the display size used for displaying graphics windows from the actual display size. The ***xPixels*** and ***yPixels*** values have no effect if they are greater than the actual display size.

setenv QUICKCAP_HEADER_SUFFIX *suffix* (Default: **.hdr**) *Header suffix*

Defines the string used to identify a header file. The ***suffix*** value must be a printable string beginning with ".". When the QuickCap deck as specified on the command line ends with ***suffix***, ***suffix*** is trimmed from the file name to create the root name. For example, an input deck named myFile.cap.hdr results in a root name of myFile.cap (by default).

setenv QUICKCAP_LICENSE_FILE *fullPath* *Licensing*

Specifies the license file to be used. This environment variable can be used in lieu of the FLEXlm variable **LM_LICENSE_FILE** if the QuickCap license file is different from the license file for other programs. See the licensing information in "Licensing" on page 1-ix.

setenv QUICKCAP_LICENSE_PREF restricted[**[*program*]**][**=*count***] *Licensing*
setenv QUICKCAP_LICENSE_PREF NXonly[**[*program*]**]
setenv QUICKCAP_LICENSE_PREF NXorMCPU[**[*program*]**] (Default)

Specifies licensing options. By default, any QuickCap child process spawned by the **-lsf** option first attempts to check out a QUICKCAP_MCPU license, and on failure, checks out a QUICKCAP_NX license. The **QUICKCAP_LICENSE_PREF** environment variable does not affect a QuickCap parent process or a QuickCap **-NOW** or **-MPP** process, which requires a QUICKCAP_NX license. **QUICKCAP_LICENSE_PREF** is also used by gds2cap and by tools in the auxiliary package such as cap2sigma and gds2tag.

restricted[**[*program*]**][**=*count***]

When **restricted** is specified with a positive ***count***, counts the number of licenses available to determine which to use. This might take some time depending on how many licenses are defined and on the configuration of the license servers. In previous releases, gds2cap, QuickCap (except for the parent process), or any tool in the auxiliary package counts licenses even when **restricted** is not specified.

When **restricted** is specified with no ***count***, programs that do not require a QUICKCAP_NX license use a QUICKCAP_MCPU license. Specifying **restricted** with a ***count*** of 0 is equivalent to **NXonly**.

NXonly[**[*program*]**]

Checks out only QUICKCAP_NX license. If you have only QUICKCAP_NX licenses, specifying **NXonly** reduces the effort gds2cap uses to check out a license.

NXorMCPU[**[*program*]**] (default)

If unable to check out a QUICKCAP_MCPU license, check out a QUICKCAP_NX license.

The **QUICKCAP_LICENSE_PREF** environment variable can contain multiple **restricted NXonly** and **NXorMCPU** specifications. A specification can be targeted for QuickCap, for gds2cap, or for a particular auxiliary program such as cap2sigma or gds2tag by including the program name in square brackets immediately after **restricted**, **NXonly**, or **NXorMCPU**. Any targeted restrictions should precede any general restriction (no ***program*** specification).

setenv QUICKCAP_LICENSE_TIMEOUT *minutes* *Licensing*

Specifies the number of minutes that QuickCap waits for an available license if none is available. If the value is negative (default), QuickCap waits indefinitely.

setenv QUICKCAP_VENDOR_CHAR_CRYPT *DLL(s)* *Vendor-based encryption*

Specifies full path names of DLLs for including character-based decryption. Multiple DLLs can be separated by commas or spaces. The DLLs for character-based decryption are described in "Character-Based Decryption" on page 6-3. QuickCap decrypts data in **#hidden** blocks.

setenv SNPSLMD_LICENSE_FILE *path* *Licensing*

Specifies the license file or path. If defined, **SNPSLMD_LICENSE_FILE** takes precedence over **QUICKCAP_LICENSE_FILE**. Users upgrading from an earlier version might require a new license file from Synopsys.

6. The Input File

The input file defines the structure of nets and dielectrics, indicates *critical* nets (for which capacitance needs to be extracted), and specifies QuickCap control parameters. This chapter describes the general format, ways of expressing length, input-file statements, and the geometric primitives.

QuickCap searches for a preference file, which, if found, is read before any input files specified on the command line are read.

The input file consists of a list of statements and comment lines. The header (initial comments in the first input file) is printed to the capacitance netlist (**-spice**) and to the numeric output file (**-numeric**). Each statement is on a single line, except for any list of geometric objects. You can delimit such a list using parentheses and can include an arbitrary number of lines. Most values can be given in the form of an *expression*, which can use user-defined parameters. A comment begins with a semicolon and ends at the end of the line. Blank lines are ignored, as are extra spaces.

Individual elements of a line, such as net names, must be less than 256 characters.

Control, Script, and Encryption Commands

Commands described in this section control the flow of the input file. Script statements (those beginning with #) can be placed independent of other input-file statements, even within lists of objects.

#hidden [**by** ***vendor(s)***]

Specifies that subsequent data is encrypted. The gds2cap (NX) tool generates a QuickCap deck with hidden sections (**#hidden**) to protect proprietary data. QuickCap hides data on any layer that is defined by a **layer** or **eps** command inside a **#hidden** block. To protect proprietary data, the graphics preview mode might not be available, or some features of the graphics preview mode might not be available, or some structures might not appear in the graphics preview mode. Also error messages might include question marks (?) denoting proprietary information. Error messages generated from within a hidden block are very general, also to protect proprietary data.

The **#hidden** script command allows a list of one or more vendors used to decrypt **#hidden** blocks. The location of the associated DLLs are defined by the **QUICKCAP_VENDOR_CHAR_CRYPT** environment variable (see page 5-15). Each vendor must match a vendor key supported by one of the DLLs. The DLLs for character-based decryption are described in “Character-Based Decryption” on page 6-3.

#include ***fileName***

Includes data from ***fileName***. When ***fileName*** ends with **%uid**, QuickCap expands **%uid** to a character string consisting of the host ID and the program ID, delimited by a period (.). When a run is continued, QuickCap checks modification times of input files on the original command line, not modification times of included files. Thus, an **#include** file that has been modified since the initial run does *not* trigger a warning message. At your discretion, **#include** statements can be generated by gds2cap. For more information, see the *gds2cap User Guide and Technical Reference*.

Character-Based Decryption

QuickCap applies vendor-supported character-based decryption to **#hidden** blocks when the **#hidden** command include a suffix listing one or more vendors. The suffix format is **by *vendor(s)***, where ***vendor(s)*** consists of one or more public vendor keys. When QuickCap needs to implement character-based decryption, it parses the **QUICKCAP_VENDOR_CHAR_CRYPT** environment variable for the locations of the DLLs. Then, QuickCap queries each DLL to find out which library supports each vendor key. Subsequent calls to the DLL support decryption. Related DLL encryption routines are used by gds2cap for encrypting **#qtfHide...#qtfEndHide** blocks, by qtfx for weak and strong encryption, and for **#qtfHide.. #qtfEndHide** blocks. DLL decryption routines are also used by qtfx, gds2cap, and gds2density when reading hidden data.

To create a DLL for character-based encryption and decryption, contact your local technical support representative from Synopsys.

Length Parameters

QuickCap length parameters define spatial resolution, critical distances with respect to generating integration surfaces, and maximum distances between objects for keeping separate statistics. Commands for setting the length parameters are described in this section.

For a parameter declaration that involves a single ***length*** (such as **scale** or **range**), the keyword can be followed immediately by a colon (:) or equal sign (=). For example, *scale 0.1um*, *scale=0.1um*, and *scale: 0.1 um* are all equivalent.

The following list shows the length parameters and their default values. Because many default values are linked to **scale**, setting **scale** to a value on the order of feature sizes and spacing in the circuit to be analyzed is generally sufficient.

QuickCap Length	**Default Value**
units	**um**
scale	**units** (or value of **defaultScale** in preference file)
trap[**ezoid**]**Res**[**olution**]	**scale**/100
depth	- (no depth)

units *lengthUnit* (Default: **um**)

Specifies the unit to apply to lengths for which there is no specified unit. Valid length units are **um**, **nm**, and **A**. If **units** is not specified, the default length unit is µm. The **units** statement can appear only before any length that has been referenced.

Windowing

QuickCap can apply various boundary conditions to its input data.

- **Windowed extraction:** The **xCell** and **yCell** commands with the **window** keyword, described on page 6-5, direct QuickCap to filter input geometry, accepting shapes only within a margin of a calculation window. To support various modes of hierarchical extraction (blackbox, graybox, and whitebox, for example), QuickCap recognizes several commands and options, *box-analysis modes*, that affect the reported capacitance values to objects outside the calculation window and inside the margin.
- **Unit-cell extraction:** The **xCell** and **yCell** commands with the optional **nis "n" correct?** keyword, described on page 6-4, evaluates a unit cell in which the geometry is periodic, but not the electric field or voltage. This mode can be useful for evaluating capacitance of a memory cell or other structure typically instantiated in a regular periodic array.
- **Periodic and reflective boundary conditions:** The **xCell** and **yCell** commands with the **periodicBoundary** or **reflectiveBoundary** keywords, described on page 6-5, applies periodic or reflective boundary conditions to the electric field and voltage. This mode can be useful for evaluating the source of error introduced by capacitance-extraction programs that use such boundary conditions.

The windowed extraction, unit-cell extraction, and periodic and reflective boundary conditions involve the **xCell** and **yCell** commands. QuickCap does not allow different modes to be used in multiple **xCell** commands, nor in multiple **yCell** commands.

Window-Related Commands

xCell ***xMin xMax*** **[periodicGeometry] [clip] [n [=]** ***n*****]**
yCell ***yMin yMax*** **[periodicGeometry] [clip] [n [=]** ***n*****]**

Specifies the size of a unit cell. All geometric data is treated as an infinite array in x and/or y, but no assumptions are made regarding electric field. The **clip** keyword causes QuickCap to clip subsequent input geometry to the cell bounds. Nets and signals that overlap their "replicated selves" are considered extended, as are nets and signals that are connected to themselves in a **cellPinX** or **cellPinY** command. You can specify the **xCell** and **yCell** values only one time each and they affect all structures, independent of the order in the input file. For additional information, see "Periodic Geometric Structure" on page 7-2.

Beyond the unit cell, QuickCap maintains, separate statistics *n* cells (the default is zero) in each direction. Any counterpart of a net that does not overlap its "original self" and is within ***n*** cells of the central one is assigned a name that is the name of the original followed by an *x* or *y* offset; for example, clock[+1*x*] or pin[+2*x*][-3*y*]. When the periodic structure includes signals and nodes, QuickCap also includes spurious nets and nodes named with a suffix of **[**x]**, **[**y]**, or both, for example, node[**x][+1y], or node[**x][**y].

Capacitances to cells more than *n* away are lumped to the spurious net. This *spurious* capacitance is normally lumped to ground but can be reported separately using the **-spurious** command-line option. QuickCap also maintains separate capacitance statistics involving nodes and objects outside the unit (*external node capacitance*). By default, QuickCap maps external node capacitance to the associated internal node. The **-external** command-line option reports such elements separately.

xCell ***xMin xMax*** **periodicBoundary**[**clip**]
yCell ***yMin yMax*** **periodicBoundary**[**clip**]
xCell ***xMin xMax*** **reflectiveBoundary**[**clip**]
yCell ***yMin yMax*** **reflectiveBoundary**[**clip**]

Specifies the size of a unit cell. A periodic or reflective boundary condition is applied to the electric field at the perimeter of the unit cell. The **clip** keyword causes QuickCap to clip subsequent input geometry to the cell bounds. You can specify **xCell** and **yCell** only one time and they affect all structures, independent of the order in the input file. Non-Manhattan structures must not cross the boundary. For additional information, see "Periodic and Reflective Boundary Conditions" on page 7-3.

Note: Periodic and reflective boundary conditions are features supported by QuickCap to investigate approximations imposed by other capacitance extractors that might routinely use such conditions. The general application of these boundary conditions to real problems is not recommended. Analyze structures that are geometrically periodic using **xCell** ***xMin xMax*** [**n** [**=**] ***n***] and **yCell** ***yMin yMax*** [**n** [**=**] ***n***].

xCell ***xMin xMax*** **window** [**clipMargin** [**=**] ***dx***]
yCell ***yMin yMax*** **window** [**clipMargin** [**=**] ***dy***]

Specifies the size of a calculation window. The capacitance is only calculated within the window. The geometry is clipped *only* if a **clipMargin** is specified. In this case, the geometry is clipped to a clip window that is larger than the calculation window by ***dx*** and ***dy***, which can be zero. Non-Manhattan objects that intersect the edge of the clip window are *not* clipped. The groundplane (if any) is not clipped. Capacitance to the groundplane outside of the clip window is attributed to the spurious net. This command should appear before any nets or dielectrics because it does not affect structures defined earlier in the input file. You can use multiple **xCell**…**window** (or **yCell**…**window**) commands; in which case, the **clipMargin** of the first such command is used and the final window is a range common to all such commands. A clip window can lead to an asymmetric capacitance results.

Note: Using a clip window is recommended when there are memory limitations. In this case, use a reasonable value for **clipMargin** (not zero).

Flags

QuickCap flags are used to enable or disable some features of QuickCap.

caseFolding[:] **off** (Default)
caseFolding[:] **on**

Indicates whether parameter names and net names are case dependent. This affects names of nets, structures, and parameters. Keywords are never case dependent.

critDeviceLimitedFraction[=] ***fraction*** (Default: **50%**)

Specifies the critical amount of device capacitance above which QuickCap reports in the capacitance summary that capacitance is device limited. When the device capacitance exceeds **critDeviceLimitedFraction** (50%, the default) of the total capacitance (device plus parasitic), QuickCap prints a note in the capacitance summary similar to the following.

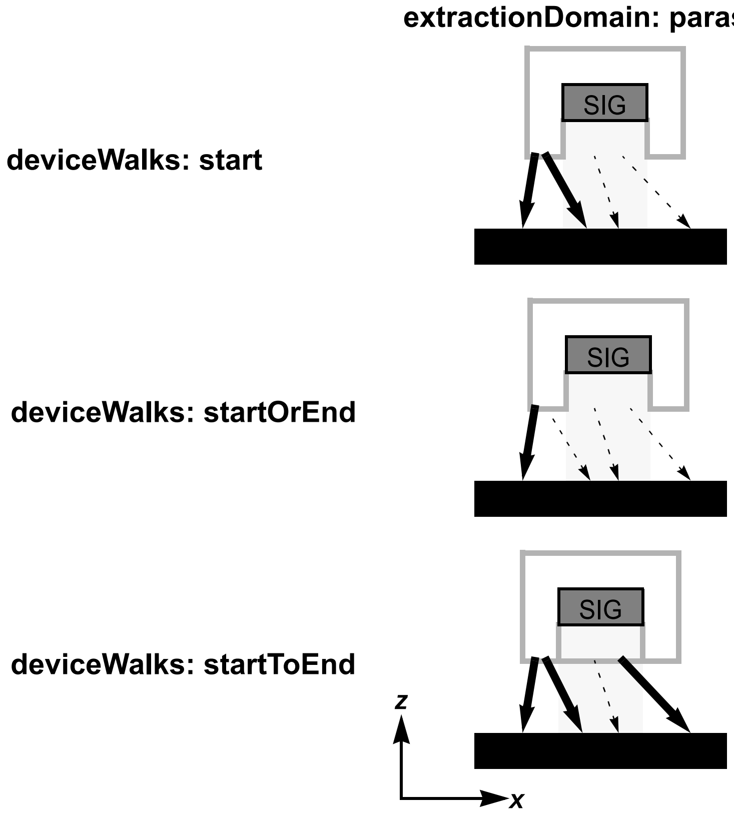
```
NOTE: critDeviceLimitedFraction 50% exceeded (62% on BIT[16])
```

QuickCap relaxes the convergence criteria to be consistent with the amount of device capacitance so that the uncertainty of simulated values such as timing and crosstalk is consistent with QuickCap's dial-in accuracy.

devices[:] ***surfaceBased***

Specifies whether QuickCap generates an integration surface that isolates a surface that is within a device region (**surfaceBased**). For surface-based devices, QuickCap generates an integration surface that approaches near the net (clipped to **junction**) along the edge of the device region. Surface-based devices affect capacitance only for combinations of **deviceWalks** and **extractionDomain** where some regions can have no starting walks as shown in Figure 6-1 on page 6-7. For surface-based devices, the integration surface isolates device surfaces (clipped to junction) for applicable combinations of **deviceWalks** and **extractionDomain** as shown in Figure 6-1 on page 6-7.

Figure 6-1: Surface-based walks contributing to capacitance (bold arrows) for all applicable combinations of deviceWalks and extractionDomain



deviceWalks[:] ***start*** | ***startOrEnd*** | ***startToEnd***

Specifies which walks QuickCap considers to be related to device capacitance.

For surface-based walks shown in Figure 6-1, the integration surface extends along the edge of the device region, clipped to **junction** from any net. Results from surface-based walks are not sensitive to **range**, but can vary slightly as **junction** is changed.

Device surfaces generally consist of all surfaces that are facing into a device region. The only exception to this rule is that a groundplane at the edge of a device region is not considered a device surface for **deviceWalks *startOrEnd***. When the device region extends from the groundplane to the bottom of the gate region, for example, QuickCap considers the bottom of the gate to be a device surface, but does not consider the groundplane to be in the device region for **deviceWalks *startOrEnd***. When the device region extends up to the top of the diffusion region, QuickCap treats the bottom of the diffusion region, but not the top, as a device region.

start

Device walks start within a device region, whether they terminate inside or outside of a device region. Parasitic (non-device) walks start outside of device regions. Results converge faster for the **deviceWalks: start** option than for other **deviceWalks** options walks, because QuickCap does not start any walks that it should ignore, and all walks started count towards capacitance. Also, QuickCap does not need to maintain the device regions after generating the clipped integration surfaces.

startOrEnd (default)

Device walks either start or end within a device region. Parasitic walks start and end outside of device regions. For this approach, a device region must extend *below* groundplane for that region of the groundplane to count as a device region.

startToEnd

Device walks are those that end within a device region, whether they started inside or outside of a device region. Parasitic walks start outside of device regions or end outside of device regions.

floatTuning[:] **deprecated**
floatTuning[:] **dielectricBased** (Default)
floatTuning[:] **planarBased**

Specifies the method for tuning floating-metal parameters. By default (**dielectricBased**), the tuning method considers both conformal and planar dielectrics. The **planarBased** method does not consider conformal dielectrics. Earlier releases of QuickCap use the **deprecated** method, similar to the **planarBased** method but misses some minor enhancements.

floatingMetal[:] **ground** | **float** | **ignore** (Default: **float**)

Indicates how to handle simple and complex floating metal structures above the groundplane. Use the **floatingSubstrate** flag to control modeling of floating metal structures that touch or intersect the groundplane. The **floatingMetal** flag provides an easy mechanism for determining how approximations to floating metal affect capacitance values. The **floatingMetal** flag can be defined anywhere in the input deck. If defined multiple times, QuickCap uses only the last value specified. The QuickCap tool prints a warning message in the summary file if floating metal is **ground** or **ignore**.

For **float**, QuickCap applies standard floating-metal analysis.

For **ground**, floating metal is treated as ground, generally resulting in larger total-capacitance values and smaller coupling-capacitance values than the default (**float**).

For **ignore**, QuickCap discards all floating metal structures.

floatingSubstrate[**:**] **float** | **ground** | **ignore** (Default: **ground**)

Indicates how to handle *floating substrate*: simple and complex floating metal structures that touch the groundplane. Use the **floatingMetal** flag to control modeling of floating metal structures above the groundplane. The **floatingSubstrate** flag can be defined anywhere in the input deck. If defined multiple times, QuickCap uses only the last value specified. The QuickCap tool prints a warning in the summary file if floating substrate is **float** or **ignore**.

For **float**, QuickCap applies standard floating-metal analysis, which is generally not accurate because the coupling from floating substrate to the groundplane is not correctly modeled.

For **ground** (the default), floating substrate is treated as ground, generally consistent with tight capacitive coupling to the groundplane.

For **ignore**, QuickCap discards all floating substrate structures.

layerBasedNodes

Converts all nets to signals with layer-based node names. For example, a **net Z** statement effectively becomes a collection of nodes such as **node Z “Z M1”**, **node Z “Z V12”**, and so on. This statement affects *all* nets whether defined before and after the **layerBasedNodes** statement. The **layerBasedNodes** statement can be useful for investigating layer-based components of capacitance results. It can be used, for example, to identify layer-based errors in rule-based capacitance extractors.

layerPrecedence[**:**] **explicit | implicit** (Default: **implicit**)

Specifies whether QuickCap infers a layer precedence for layers with the same precedence (or no precedence) that share a top or bottom surface. For **layerPrecedence implicit** (the default), QuickCap gives higher precedence to the thinner of two such layers. For **layerPrecedence explicit**, precedence of overlapping boxes with the same layer precedences is based on the order of declaration of the associated net or node. Objects associated with the first defined **net** or **node** declaration take precedence. When the same net or node is declared multiple times, precedence is according to the first declaration for that net or node. Layer precedence can be defined through the **precedence** property of the **layer** declaration (page 6-32).

minAdjustedTrapLayerFraction[**=**] ***fraction*** (Default: **80%**)

Specifies the minimum fraction of a trapezoidal shape (sloped sidewalls) expected to overlap an **adjustDepth** layer (in z). The interaction between sloped sidewalls and thickness variation is not uniform over the height of a trapezoidal shape. When the fraction of overlap between an **adjustDepth** layer and layer containing trapezoidal shapes is below **minAdjustedTrapLayerFraction**, QuickCap issues a warning.

netlistSmallCaps[:] **discard**|**ground**|**distribute** (Default: **distribute**)

Specifies handling of *small* capacitance values. Any negative capacitance value is considered small. A *small* capacitance value is determined by arguments of the **-spice** (page 5-9) command-line option. By default, a *small* capacitance value only includes negative values.

The **-spice** command-line option can define an absolute value, such as 0.1aF, and a relative uncertainty, such as 50%. In this case, a *small* capacitance value is either less than 0.1aF or has a relative uncertainty larger than 50 percent. For **discard**, small capacitance values are ignored, changing the total capacitance. For **ground** or **distribute**, small capacitance values are added to the ground cap.

The **netlistSmallCaps** flag does not affect capacitance in the state file. The **-spice** option takes precedence when it includes **discard**, **ground**, or **distribute** as an argument.

nodePostfix[:] **off | on** (Default: **on**)

Specifies whether QuickCap applies a numerical fix to node coupling capacitance. The **-nodePostfix** option, described on page 5-7, takes precedence over the **nodePostfix** flag.

When extracting the capacitance of multiple nets, any net that includes capacitance nodes is subject to a charge-partitioning error where some capacitance that should be assigned to one node might be assigned to a touching node on the same net. This might result in an asymmetry (when C_{AB} and C_{BA} are different). Using the **nodePostfix** method after extraction averages symmetric parts of each node-coupling pair, it compares the sum of node-based coupling caps (between two nets) to the value it would get if it averaged the pair of net-based coupling caps. QuickCap then uniformly scales the node-based coupling caps for consistent results.

statsMerge[:] **deprecatedMethod**|**generalMethod** *(*Default*:* **generalMethod)**

Specifies the internal method used to merge statistics. The **deprecated** method can introduce an error.

trackFloatingWalks[:] **off**|**on** (Default: **on**) *Modified flag*

Specifies how to consider walks that involve a floating structure. For **on** (the default), a walk that involves a floating structure contributes to a capacitance that can be normally ignored based on its ending point. This applies to a walk that can be ignored because it terminates in a device region or because it starts and ends on a pair of layers named in an **ignoreCoupling** statement.

The default behavior is a good practice. A floating net is only correctly modeled if all its capacitance components are taken into account.

trapMethod[**:**] **parmBased**
trapMethod[**:**] **deprecated**
trapMethod[**:**] **transitional**

Specifies the method for handling trapezoids (shapes with sloped sidewalls). The default method, **parmBased**, is layer based, using a fixed-bias or fixed-slope approach depending on whether a **layer** statement has properties **topBias**[**X**|**Y**] and **bottomBias**[**X**|**Y**], or **sideTan**[**X**|**Y**]. This method accounts for interactions between thickness variation (**adjustDepth** layers) and the sloped sidewall.

The **deprecated** and **transitional** methods, which do not correctly account for interactions between the sloped sidewall and thickness variation, allow earlier behavior to be replicated.

Netlist Customization

Use the following QuickCap declarations described in this section to customize the netlist generated by **-spice**.

netlistMidlineCommentChar[**:**] ***char*** (Default: *****)

Specifies the character for commenting the remainder of a line in the netlist file.

netlistName[**:**] ***fileName***[**:**] [***subcktName***]

Specifies the name of the netlist, independent of the root name of the QuickCap run. If **-spice** is specified, QuickCap performs an incremental update of the netlist file. If a subcircuit name is specified, QuickCap updates only the part of the netlist associated with the subcircuit—capacitance values outside the scope of the subcircuit are not modified. If no subcircuit name is specified, QuickCap updates only the part of the netlist that is not part of a subcircuit—capacitance values that are part of any subcircuit are unaffected.

Note: The **netlistName** statement cannot be declared in the preference file.

QuickCap Objects

QuickCap considers several types of objects: nets, nodes, and signals; hierarchical structures; floating metal; dielectrics; and device regions.

- Nets, nodes, and signals (see “Nets, Nodes, and Signals” on page 6-12) are electrodes that can accumulate charge. QuickCap searches for capacitance of nets, nodes, and signals. These are defined with **net**, **node**, **ground**, and **groundPlane** declarations.

- Floating metal (see “Floating (Dummy) Metal” on page 6-18) has *floating* voltage that is a function of voltages on nets, consistent with the constraint that no net charge accumulates on the floating metal.
- Dielectrics (see “Dielectrics” on page 6-18) are used to specify regions with different dielectric permitivities. QuickCap recognizes several kinds of dielectrics: background dielectric (**eps**), planar (**eps up to**) and nonplanar (**conformal, dielectric**, and **dielectricLayer**). QuickCap also recognizes directional effects (**scaleCapXY** and **dielectricXY**).
- Device regions (see “variableDi[electric] [uniformDielectricBox(es)]” on page 6-21) are regions for QuickCap to ignore with respect to the capacitance calculation. Fields in these regions are assumed to be associated with device capacitance and do not contribute to capacitance. Fields outside these regions do contribute to capacitance, even when affected by geometry within device regions.

Nets, Nodes, and Signals

All nonfloating conductors in the problem have names and are called *nets* or *nodes*. Each node is associated with a *signal*, and each signal consists of one or more nodes. Nodes and signals are used when the capacitance of a net must be represented as several lumped-element values.

When a conductor and a dielectric overlap, the conductor takes precedence. Conductors are replicated in *x* and/or *y* if there is an **xCell** and/or a **yCell** statement that is not of type **window**.

Net, Node, and Signal Names

Net, node, and signal names can include any characters except space, comma (,), and semicolon (;). These characters can also be included if the net name is delimited by a pair of single or double quotation marks (' or "). Names can be suffixed by a number of integer expressions and string literals. This can be useful in scripted decks for generating parameter-dependent names.

Nets

In addition to any user-defined nets, QuickCap has a special net, the ground net.

The *ground net* includes the groundplane (if a **groundplane** statement is specified), any structure defined by a **net ground** statement, and might include capacitance to *infinity*. The ground net is named *ground*. In the capacitance netlist and numeric output file, ground is referenced as a node named “0”.

groundplane [**at**] ***height***

Defines a groundplane that extends down from the height specified. An object in a net that is beneath the groundplane but has a surface coincident with the groundplane takes precedence. The **groundplane** statement can be specified only one time and cannot be used in a 2D *xy* environment.

net ***netName*** [***auxData***] [***primitive***[***s***]]

Defines a net named ***netName***. The ***netName*** argument cannot be used as a node or signal name in a **node** declaration. When a net is declared more than one time, any new auxiliary data is added to the old data. If the auxiliary data includes capacitance, the *effective* resistance is recalculated from the total delay time. Until another object declaration (or **extract**, **rename**, or **short** declaration) occurs, subsequent lists of geometric objects are part of the specified net.

Nodes and Signals

QuickCap allows a net to be broken into any number of nodes. In this case, the collection of nodes is called a *signal*. By default, QuickCap output to the summary file has capacitance values for nets and signals. If run with the **-nodes** command-line option (see page 5-7), QuickCap output has the capacitance values for nets and nodes.

node ***sigName nodeName*** [***auxData***] [***primitive***[***s***]]

Defines a node named ***nodeName*** on a signal named ***sigName***. Neither ***nodeName*** nor ***sigName*** can be referenced as the name of a net in a **net** statement. In some ways, the nodes on any given signal are treated collectively by QuickCap. Automatic selection is based on the signal capacitance and resistance rather than individual node capacitance. By default, QuickCap prints the total (signal) capacitance, even though QuickCap finds the capacitance associated with each node (ignoring any capacitance between nodes on a common signal). Use the command-line option **-nodes** to print capacitance to individual nodes. A node declared with the same node name on different signals results in an error; although, a node can have the same name as a signal. The auxiliary data associated with a signal is the sum of the data for each node. When a **node** is declared more than one time (with the same signal), the auxiliary data is accumulated. If the auxiliary data includes capacitance, the effective resistance is recalculated from the total delay time. Until another object declaration (or **extract**, **rename**, or **short** declaration) occurs, subsequent lists of geometric objects are part of the specified node.

nodeDelimiter ***delimiter*** (Default: **none**)

Prefixes any node name that does not begin with the signal name (in a **node** statement) with the signal name and ***delimiter***. For example, if ***delimiter*** is *&,* the statement node *sigName 3* generates a node named *sigName&3*.

Net- and Signal-Related Statements

bounds *netName x0 y0 x1 y1*

Defines a rectangle associated with the bounding-box description of a net or signal. The entire bounding-box description is the union of all rectangles associated with that net through the **bounds** declaration. The bounding-box description of a net is used to filter out objects in the input file when the **windowNets** declaration appears earlier in the input file (or in a preceding input file) and either **extractNetsWithBounds** is declared earlier or the net was earlier specified in an **extract** statement (see “Selecting Nets for Extraction” on page 6-22).

The gds2cap tool with the **-bounds** option generates a bounds file and an associated header file. The bounds file consists of **bounds** declarations for nets defined in the main QuickCap deck. These files, together, can be used for filtering the geometry to include objects that are only near extracted nets by editing the header file to ensure it names the nets to be extracted in **extract** statements and to ensure it has the correct **windowNets** declaration.

By passing the header file and the bounds file to the main QuickCap deck to process the main deck and to identify the bounding boxes of nets to be extracted and can keep just those objects that are within a critical distance (set by the **windowNets** statement) of nets to be extracted.

Cdp *net1 net2 value* [[,] ***x*** [,] ***y***]

Specifies a value for the *device parasitic capacitance*, **Cdp**, between nets or nodes. The device parasitic capacitance is the amount of capacitance that is outside of any device regions but included as part of the device capacitance. QuickCap subtracts device parasitic capacitance from capacitance data to report in the summary and in any numeric file (**-numeric**), capacitance table (**-table**), or netlist (**-spice**).

QuickCap uses the xy position, if specified, to include or exclude the **Cdp** value according to the calculation window (**xCell ... window**, **yCell ... window**). This avoids double counting **Cdp** values during tiled runs. The xy position data can be delimited by parentheses.

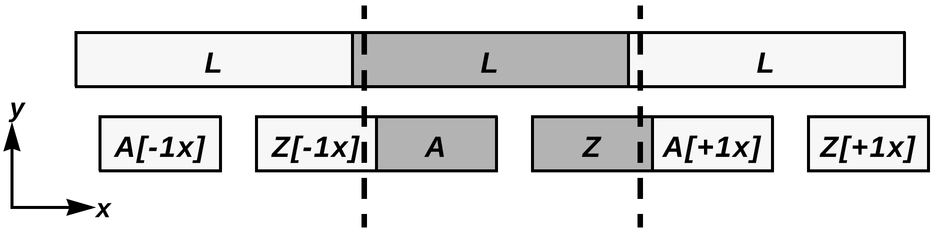
cellPinX *net1 net2*
cellPinY *net1 net2*

Specifies an ohmic junction between ***net1*** in the unit cell and ***net2*** in the unit cell offset by one unit cell in x (**cellPinX**) or in y (**cellPinY**). The ***net1*** and ***net2*** arguments can be net, node, or signal names.

When both nets are the same or are nodes on the same signal, QuickCap treats the net or signal as a line in x (or y). An example is shown in the following figure.

When the QuickCap deck includes any **cellPinX** or **cellPinY** declaration or when any **ohmic** statement includes a cell offset, QuickCap uses such specifications to determine whether a net is a line in the x- or y-direction. Without any **cellPinX** and **cellPinY** declarations and without any **ohmic** statements, QuickCap determines lines in the x- or y-direction by checking whether a net translated by one unit cell touches or overlaps itself.

In the following figure, a **cellPinX L L** statement establishes that L in each unit cell is the same. Whereas, a **cellPinX Z A** statement establishes an ohmic contact between Z and A[+1x], and between Z[-1x] and A. The cell boundaries are shown as dashed lines. The lightly shaded objects are virtual objects resulting from the periodicity of the unit cell (shaded).



map misc. to ground
map misc. to net *net*
map misc. to node *node*
map misc. to self
map misc. to spurious

Treats any net or signal that is not extracted as another. The default is **map misc. to ground**. This does not affect automatic selection of nets for extraction. The **map misc.** statement only affects nets declared later. If you want to maintain individual statistics to the nets that are not extracted, use **map misc. to self**. If ***net*** is the name of a node, it is replaced by the associated signal.

map *fromNet*|*fromResistor* [(***pct***)] **to ground**
map *fromNet*|*fromResistor* [(***pct***)] **to net *toNet***
map *fromNet*|*fromResistor* [(***pct***)] **to node to*fullNodeName***
map *fromNet*|*fromResistor* to self
map *fromNet*|*fromResistor* [(***pct***)] **to spurious**

Treats a ***net*** as **ground**, another net or node, itself, or **spurious** for purposes of keeping any statistics. The **map** statement can apportion the capacitance of ***fromNet*** or ***fromResistor*** to multiple nodes and nets, including *ground* and *spurious* (not applicable to **self**). When mapping to a net or node, the **map** statement requires the appropriate keyword **net** or **node**.

Each **map** statement specifies a net or a node on a named signal to map the capacitance. Mapping net or resistor capacitance is not compatible with the **-partition** command-line option.

The **map** statement can reference the name of a resistor. It also allows a mapped fraction to be specified in parentheses either as a value (0 to 1) or as a percentage (0% to 100%). For a given ***fromNet*** or ***fromResistor***, either *all* the associated **map** statements must include a fraction or *none* of the associated **map** statements can include a fraction. When all associated **map** statements include a map fraction, any unmapped fraction is mapped to ground. When no associated **map** statements include a map fraction, each object (the net or the resistor) is mapped to receive an equal fraction. For two **map** statements, for example, each "to" net or node receives 50 percent of the capacitance. Mapping capacitance from a net, a node, or a resistor is not compatible with a partition run (**-partition**). Mapping capacitance to a net or node is not compatible with the **statistics self** statement.

netData *label1* [[,] ***label2*** [[,] ***label3*** [**...**]]]

Defines labels to be associated with any auxiliary LPE data. Like net names, labels can include any characters except space, comma (,), and semicolon (;). These characters can also be included if the net name is delimited by a pair of quotation marks ("). More than one **netData** statement can be declared, in which case, the new labels are appended to the existing list of labels. All labels must be defined before any auxiliary LPE data is specified. For a **net** or **node** containing LPE data, the number of data entries must agree with the number of data labels declared in **netData** statements.

ohmic [**junction**] ***net1 net2*** [***cx cy***]

Ignores the capacitance of touching nets. This is useful when touching nets form an ohmic junction, which effectively shorts out any capacitance between the pair of nets. When the default junction type is **rectifying**, you must specify a pair of touching nets named as forming an ohmic junction to disregard the capacitance. Nodes on a common signal do not require an **ohmic junction** declaration.

Use optional arguments ***cx*** and ***cy*** to reference ***net2*** in the unit cell, offset by ***cx*** cells in the x-direction and ***cy*** cells in the y-direction. The value of ***cx*** (or ***cy***) can only be nonzero when **xCell** (or **yCell**) defines a periodic structure (no **window**, **periodicBoundary**, or **reflectiveBoundary** modifier) and defines a value of **n** (number of unit cells to maintain statistics). In this case, ***cx*** (or ***cy***) must be between +/-***n***. A statement **ohmic *net1 net2 cx cy*** is equivalent to **ohmic *net2 net1 -cx -cy*** (reversing the order of the nets and changing the sign of the unit-cell offset).

QuickCap recognizes a special case when both nets are the same (or are nodes on the same signal) and ***cx*** is +/-1 and ***cy*** is zero (or ***cx*** is zero and ***cy*** is +/-1). QuickCap considers such a net to be a line in x (or y). This is equivalent to using the **cellPinX** (or **cellPinY**) statement.

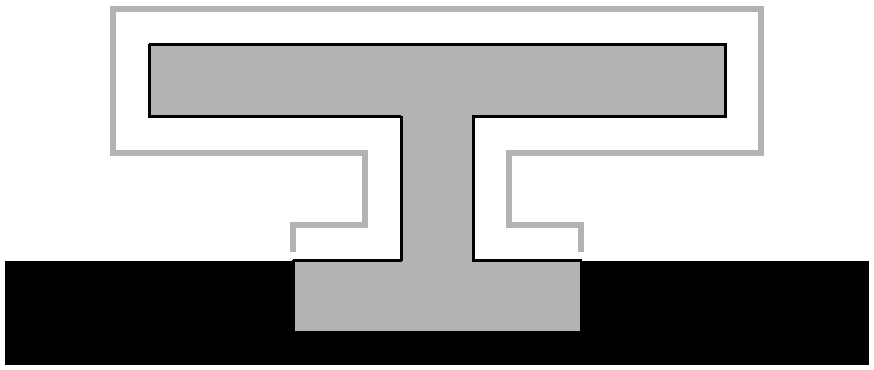
When any **ohmic** statement includes a cell offset or when the QuickCap deck includes any **cellPinX** or **cellPinY** declaration, QuickCap uses such specifications to determine whether a net is a line in the x- or y-direction. Without such **ohmic** statements and without any **cellPinX** and **cellPinY** declarations, QuickCap determines lines in the x- or y-direction by checking whether a net translated by one unit cell touches or overlaps itself.

pn [**junction**] ***netName netName***
diode [**junction**] ***netName netName***
rectifying [**junction**] ***netName netName***

Finds the capacitance of touching nets, including any fringing capacitance beyond the junction distance (see **junction** on page 6-16). When the default junction type is **ohmic**, you must specify a pair of touching nets named as forming a rectifying junction (or diode junction or pn junction) to extract the capacitance associated with the pair.

For the net shown in Figure 7, the parasitic capacitance to the groundplane can only be calculated if the net and the groundplane are separated by a nonohmic junction. In this case, the integration surface should extend no closer than the junction distance. QuickCap does not analyze junction capacitance. The junction capacitance, if required, must be calculated separately by another method.

Figure 7: Integration Surface (Light Line) Around a Net That Includes a Diffusion Run in the Groundplane



rename[**Net**] ***oldName newName***

Renames a net named ***oldName*** to ***newName***. If a net already exists with the new name, objects in the net ***oldName*** are transferred to the new net. Renaming a net that was specified in an **extract** statement causes QuickCap to print a comment to that effect. In such a case, the newly named net is extracted and the **extract** statement is still in effect for the original net name. The **rename** statements are generated by gds2cap when it is run with the **-tile** command-line option (this is different from the QuickCap **-tiled** command).

Until another **rename** declaration, an object declaration, or an **extract** or **floatNet** declaration occurs, subsequent lists of geometric objects are part of the newly named net.

short[**ToSignal**] ***fromNetOrSignal toNetOrSignal***
Joins nets or signals to a single signal.

When ***toNetOrSignal*** is a net, it becomes a node on a signal of the same name.

When ***fromNetOrSignal*** is a net, it becomes a node of the same name on ***toNetOrSignal***. When ***fromNetOrSignal*** is a signal, any associated pin metal and all associated nodes are transferred to the other signal.

Floating (Dummy) Metal

As with nets, floating (dummy) metal is replicated if there is an **xCell** or a **yCell** statement.

complexFloat
Similar to the functionality of the **net** statement, treats subsequent geometric elements as components of a single floating net. Until another object declaration (or **extract**, **rename**, or **short** declaration) occurs, subsequent lists of geometric objects are part of the same floating structure. Use the **floatingMetal** statement (see description on page 6-8) to ignore or ground floating structures above the groundplane. By default, QuickCap treats as ground any floating structures that touch or intersect the groundplane. This behavior may be modified by the **floatingSubstrate** statement (see description on page 6-9).

simpleFloat
Similar to the functionality of the **net** statement, treats subsequent geometric elements as isolated floating (dummy) metal. Until another object declaration (or **extract**, **rename**, or **short** declaration) occurs, subsequent lists of geometric objects define independent floating metal shapes. Use the **floatingMetal** statement (see description on page 6-8) to ignore or ground floating structures above the groundplane. This behavior may be modified by the **floatingSubstrate** statement described on page 6-9.

Dielectrics

A dielectric consists of a relative dielectric constant and a geometrical description. A planar dielectric is defined by the bounding *z* values. A nonplanar dielectric is defined by geometric primitives. As with nets, dielectrics are replicated if there is an **xCell** and/or a **yCell** statement.

conformalLayer ***name eps baseLayer properties***

Specifies a conformal dielectric layer associated with ***baseLayer***. Each object on ***baseLayer*** in the body of the QuickCap deck implicitly includes the associated conformal dielectric box.

A conformal layer takes precedence over subsequent **conformalLayer** statements, over any **dielectricLayer** boxes based on a subsequent **layer** declarations (**type=dielectric**), but not over **dielectric** boxes. For an example, see "variableDi[electric] [uniformDielectricBox(es)]" on page 6-21.

The dielectric value of the layer is ***eps***.

QuickCap recognizes the following **conformalLayer** properties.

down[**=**]***distance***

Specifies the distance from the bottom of **baseLayer** down to the bottom of the conformal dielectric. When neither **down** nor **z0** is specified, **down** defaults to **0**. You cannot specify both **down** and **z0**.

out[**X**|**Y**][**=**]***distance*** (Default: **0**)

Specifies the thickness of the dielectric layer on the sides of ***baseLayer***. Specifying **out** is equivalent to specifying both **outX** and **outY**. The **outX** and **outY** properties cannot be specified when **out** is defined.

ID[**=**]***count***

Specifies layer ID. QuickCap does not use the layer ID.

up[**=**]***distance***

Specifies the distance from the top of ***baseLayer*** down to the bottom of the conformal dielectric. When neither **up** nor **z1** is specified, **up** defaults to **0**. You cannot specify both **up** and **z1**.

z0[**=**]***height***

Specifies the height of the bottom of the dielectric layer. When neither **z0** nor **down** is specified, **down** defaults to **0**. You cannot specify both **z0** and **down**.

z1[**=**]***height***

Specifies the height of the top of the dielectric layer. When neither **z1** nor **up** is specified, **up** defaults to **0**. You cannot specify both **z1** and **up**.

dielectric ***eps*** [***primitive***[***s***]]

Defines a nonplanar dielectric with a relative dielectric constant of ***eps***. When planar and nonplanar dielectrics overlap, the nonplanar dielectrics take precedence. Until another object declaration (or **extract**, **rename**, or **short** declaration) occurs, subsequent lists of geometric objects are part of this dielectric. Dielectric boxed defined with dielectric take precedence over those defined with **conformalLayer** or **dielectricLayer**.

A dielectric with a value of zero is equivalent to **variableDielectric**. For this structure, all primitives must be uniform-dielectric boxes, described with **variableDielectric** on page 6-21

dielectricXY

Specifies a nonphysical dielectric value that overrides the nominal dielectric value. This effectively scales the lateral capacitance because parts of the integration surface are within the **dielectricXY** region. QuickCap adds a warning in the summary file when you define any **scaleCapXY** regions.

The format of a **dielectricXY** box in a subsequent box list can begin with the keyword **boxXY** and includes the dielectric values in x and y. If you define depth using a preceding **depth** or the **setLayer** statement, the associated box is defined by an xy rectangle.

[**boxXY**[:]] ***epsX***[,] ***epsY***[:] ***box***

dielectricLayer *layerName*

Specifies that the following shapes are associated with the layer of the same name, which in turn must include a dielectric value (**eps** layer property).

The precedence **dielectricLayer** boxes correspond to the location of the associated **layer *layerName*** statement. The boxes take precedence over **conformalLayer** statements defined after the layer statement, and over any dielectric-layer boxes based on a subsequent **layer** declarations, but not over **dielectric** boxes.

When the referenced layer has no defined value of **eps**, or a value**eps=0**, all primitives must be uniform-dielectric boxes, described with **variableDielectric** on page 6-21

eps[:|=] ***eps***

Defines the relative dielectric constant of the ambient to be ***eps***. You can specify the **eps** statement only one time. The default value is 1.

eps[:|=] ***eps* up to *height***

Defines a planar dielectric layer with a relative dielectric constant of ***eps***. By default, you must enter planar dielectrics from the bottom up. When planar and nonplanar dielectrics overlap, the nonplanar dielectrics take precedence. Examples of **eps … up to** are shown in "Floating (Dummy) Metal" on page 6-18.

scaleCapXY (3D)

Specifies a nonphysical scale applied to the nominal dielectric value. This effectively scales the lateral capacitance because parts of the integration surface are within the **scaleCapXY** region. QuickCap adds a warning in the summary file when you define any **scaleCapXY** regions.

The format of a **scaleCapX** box in a subsequent box list can begin with the keyword **boxXY** and includes the dielectric values in x and y. If you define depth using a preceding **depth** or the **setLayer** statement, the associated box is defined by an xy rectangle.

[**boxXY**][:] ***scaleX***[,] ***scaleY***[:] ***box***

variableDi[**electric**] [***uniformDielectricBox(es)***]

Defines a structure with uniform-dielectric boxes, each with its own dielectric value. The format of a **variableDielectric** box in a subsequent box list can begin with the keyword **boxXY** and includes the dielectric value. If you define depth through a preceding **depth** or **setLayer** statement, the associated box is defined by an xy rectangle.

[**boxEps**[**:**]] ***eps***[**:**] ***box***

Device Capacitance

Device capacitance is included as part of the device model and therefore should not be calculated. QuickCap includes the following three mechanisms to exclude device capacitance from its calculations:

- Device regions (**deviceRegion**) clip the integration surface, so that QuickCap calculates only some of the capacitance associated with a net.
- Layer-based capacitance mapping (**ignoreCoupling** statement) affects how QuickCap handles specific components of the capacitance from a net based on the layers involved.

Any parts of an integration surface that are within a device region (**deviceRegion**) are clipped. QuickCap capacitance results then exclude charge associated with electric fields in these regions. Device regions can lead to an asymmetric capacitance matrix. The **deviceRegion** declaration is described in this section.

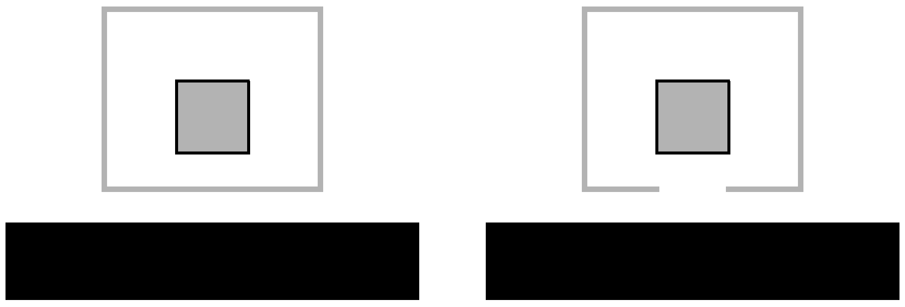
Use the **ignoreCoupling** statement (described in this section) for layer-based mechanisms to recognize capacitance components attributed to devices.

deviceRegion

Defines a volume for which capacitance is *not* to be extracted, possibly because the capacitance is already included as part of a netlist device model. This statement can appear any number of times. Figure 6-1 on page 6-22 shows cross sections of the *cube* problem and the effect of adding an **deviceRegion** under the cube. In this case, QuickCap does not extract the portion of capacitance that would be associated with a capacitance calculation under the cube. Until another object declaration (or **extract**, **rename**, or **short** declaration) occurs, subsequent lists of geometric objects are device regions.

QuickCap does not extract capacitance within a device region.

Figure 6-1: Effect of Adding a Device Region Under a Cube



Full integration surface left ($C/\varepsilon = 10.7 \pm 0.1$) and a partial integration surface right ($C/\varepsilon = 9.8 \pm 0.1$) created using a **deviceRegion** that is a 1µm cube, 1µm above the groundplane

ignoreCoupling ***list1***[,] ***list2***

Ignores the coupling capacitance between any layer in ***list1*** and any layer in ***list2***. Each list can be a single layer name, or multiple layer names enclosed in parentheses and optionally separated by commas (,). Neither list can include a layer with a **trapLayer** property. Layers with the **trapLayer** property inherit all coupling properties from the associated **trapLayer**.

The keyword **groundplane** in ***list1*** or ***list2*** references the groundplane.

Specific faces of a layer can be referenced by adding one or more of the following keywords after the layer name: **[all]** (default), **[bottom]**, **[side]** or **[sides]**, and **[top]**, and .

Use this declaration in conjunction with device regions to prevent QuickCap from extracting as parasitic capacitance those capacitance components included in the device model. For example, the following declaration ignores capacitance between POLY and RUN, POLY and CONT_DIFF, POLY and the groundplane, RUN and the groundplane, and CONT_DIFF and the groundplane.

```
ignoreCoupling (POLY,groundplane) (RUN,CONT_DIFF,groundplane)
```

Selecting Nets for Extraction

If no nets are selected for extraction, nets are extracted based on LPE capacitance values (**Cp** and **Cd**). If no nets are selected for extraction and no nets have a defined capacitance, however, all nets except those explicitly mapped are extracted.

extract ***netName1*** [[,] ***netName2*** [[,] ***netName3*** [[,] **...**]]]

Extracts the named nets or signals.

Until another object declaration (or **extract**, **rename**, or **short** declaration) occurs, subsequent lists of geometric objects are part of the last net named on the list, or part of the first node of the last signal named.

When gds2cap encounters different nets with the same name, it generates **rename** declarations in the QuickCap deck to distinguish the nets. For example, two nets with the name CRIT results in **rename** declarations changing CRIT to CRIT&n1 and CRIT&n2. If CRIT is named in an earlier **extract** declaration, QuickCap extracts both CRIT&n1 and CRIT&n2.

global ***net(s)***
notCapCrit[**icalNet**] ***net(s)***
Specifies that the named nets (or signals) are not to be extracted unless explicitly named in an **extract** statement.

Layers and Depth

The **layer** declaration (see “Layer Definition” on page 6-23 and “Layer Properties” on page 6-24) defines nominal layer depth (bottom and top z values) and other layer-dependent properties that are associated with geometric objects defined on the layer.

QuickCap also supports position-dependent layer thickness (thickness maps) through the **adjustDepth** declaration (see “Adjusting Depth” on page 6-33). An **adjustDepth** declaration scales all z values within a nominal depth, and increases or decreases higher z values, as consistent with the thicker or thinner layer.

Layer Definition

Layer information can be used by QuickCap for generating integration surfaces, for designating floating-metal parameters, and when reporting an error involving a geometric object (such as when an integration surface intersects a net). A layer, with any associated properties, is defined by the **layer** declaration. Also, a depth can be defined over which all *z* values are scaled, and above which all z values are decreased or decreased, according to values in a table as a function of position (x, y).

layer ***name z0 z1*** [***layerProperties***] (3D dimensionality)
Defines a layer. QuickCap associates an object with a layer through the **setLayer** declaration (see page 6-33) or (in 3D dimensionality) by matching minimum and maximum *z* values with ***z0*** and ***z1***. A layer name that includes parentheses must be enclosed in quotation marks.

Layer Properties

In addition to the layer depth, a layer definition can include any of several layer properties.

bottomBias[**X**|**Y**][**=**|**:**] ***distance***
topBias[**X**|**Y**][**=**|**:**] ***distance***

Specifies the bias of each edge for boxes on the layer. A positive value indicates an expansion. The **topBias** and **bottomBias** values cannot both be negative. Only boxes are affected. QuickCap does not support polygons or dielectrics with trapezoidal cross sections.

The **X** and **Y** suffixes can be used to specify trapezoidal edges independently, in x and y directions. Similar to **bottomBias** and **topBias**, **bottomBiasX** and **topBiasX** cannot both be negative, and **bottomBiasY** and **topBiasY** cannot both be negative.

capLayer[**=**|**:**] ***layerRef***

Treats the layer as if it were ***layerRef*** for the purpose of grounding or ignoring coupling capacitance based on the layer.

color[**=**|**:**] ***string***

Specifies the color used for plotting.

QuickCap recognizes many color formats and has an extended color palette. The simplest palette consists of red (**r**), green (**g**), blue (**b**), cyan (**c**), magenta (**m**), yellow (**y**), black (**d**), and hidden (**h**). The extended palette includes 216 web-safe colors and 140 colors with corresponding standard web names, some of which are in the web-safe palette. The RGB values associated with the web-safe colors have one of six values for the each of the red, green, and blue components: **0x00** (**0**), **0x33** (**51**), **0x66** (**103**), **0x99** (**154**), **0xCC** (**204**), or **0xFF**, (**255**). The 140 web names and their associated RGB values are shown in Table 6-3 on page 6-26.

You can reference colors by any of the following formats:

Basic color: **r**, **g**, **b**, **c**, **m**, **y**, **d**, or **h**

Specifies a basic or hidden color. Basic colors are red, green, blue, cyan, magenta, yellow, and dark (black). *Hidden* (**h**) has no associated color. Table 6-1 shows equivalent representations of each basic color.

Table 6-1:Equivalent Representations for Basic Colors

Character	Shaded	Base Six	Decimal	Hexadecimal
r	r5	500	(255,0,0)	0xFF0000
g	g5	050	(0,255,0)	0x00FF00
b	b5	005	(0,0,255)	0x0000FF
c	c5	055	(0,255,255)	0x00FFFF

Table 6-1:Equivalent Representations for Basic Colors ***(Continued)***

Character	Shaded	Base Six	Decimal	Hexadecimal
m	m5	505	(255,0,255)	0xFF00FF
y	y5	550	(255,255,0)	0xFFFF00
d	d5	000	(0,0,0)	0x000000

Shaded basic color: **r**, **g**, **b**, **c**, **m**, **y**, or **d** followed by a digit **1-9**

Specifies a shaded basic color. A shading of **5** is equivalent to the basic color. Lower values indicate light shades and larger values indicate darker shades. (However, for black (**d**), shading values **d5**-**d9** are equivalent.) Each shade is one of the 216 web-safe colors.

Three-digit base-six value: **000-555**

Selects one of the 216 web-safe colors. The three digits correspond to the red, green, and blue components, respectively. Table 6-2 shows the hexadecimal and decimal values corresponding to the base-six digits(**0**-**5**). For example, **135** (base six) is equivalent to **0x3399FF** (hexadecimal) or **(51,153,255)** (decimal).

Table 6-2:Base-Six Digits and Corresponding Decimal and Hexadecimal Values

Digit	Decimal	Hexadecimal
0	0	0x00
1	51	0x33
2	102	0x66
3	153	0x99
4	204	0xCC
5	255	0xFF

Decimal RGB representation: **(0,0,0)-(255,255,255)**

Selects the nearest color from among the 216 web-safe colors and the 140 recognized color names. Each of the three RGB values must be in the range 0-255. For example, **(0,128,0)** is equivalent to **0x008000** (hexadecimal) and **green** (color name defined in Table 6-3).

Six-digit hexadecimal RGB value: **0x000000-0xFFFFFF**

Selects the nearest color among the 216 web-safe colors and the 140 recognized color names. The first two digits specify the red component, the next two digits specify the green component, and the final two digits specify the blue component. For example, **0x008000** is equivalent to the **(0,128,0)** (decimal) and **green** (color name defined in Table 6-3).

Color name (Table 6-3)

Selects one of the 140 recognized color names. For example, **green** is equivalent to **0x008000** (hexadecimal) and **(0,128,0)** (decimal).

Table 6-3:Standard Color Names Recognized by QuickCap and Decimal Equivalents

Color Name	**Decimal**
AliceBlue	(240,248,255)
AntiqueWhite	(250,235,215)
Aqua	(0,255,255)
Aquamarine	(127,255,212)
Azure	(240,255,255)
Beige	(245,245,220)
Bisque	(255,228,196)
Black	(0,0,0)
BlanchedAlmond	(255,235,205)
Blue	(0,0,255)
BlueViolet	(138,43,226)
Brown	(165,42,42)
BurlyWood	(222,184,135)
CadetBlue	(95,158,160)
Chartreuse	(127,255,0)
Chocolate	(210,105,30)
Coral	(255,127,80)
CornflowerBlue	(100,149,237)
Cornsilk	(255,248,220)

Table 6-3:Standard Color Names Recognized by QuickCap and Decimal Equivalents ***(Continued)***

Color Name	Decimal
Crimson	(237,164,61)
Cyan	(0,255,255)
DarkBlue	(0,0,139)
DarkCyan	(0,139,139)
DarkGoldenRod	(184,134,11)
DarkGray	(169,169,169)
DarkGreen	(0,100,0)
DarkKhaki	(189,183,107)
DarkMagenta	(139,0,139)
DarkOliveGreen	(85,107,47)
DarkOrange	(255,140,0)
DarkOrchid	(153,50,204)
DarkRed	(139,0,0)
DarkSalmon	(233,150,122)
DarkSeaGreen	(143,188,143)
DarkSlateBlue	(72,61,139)
DarkSlateGray	(47,79,79)
DarkTurquoise	(0,206,209)
DarkViolet	(148,0,211)
DeepPink	(255,20,147)
DeepSkyBlue	(0,191,255)
DimGray	(105,105,105)
DodgerBlue	(30,144,255)
FireBrick	(178,34,34)
FloralWhite	(255,250,240)
ForestGreen	(34,139,34)

Table 6-3:Standard Color Names Recognized by QuickCap and Decimal Equivalents *(Continued)*

Color Name	**Decimal**
Fuchsia	(255,0,255)
Gainsboro	(220,220,220)
GhostWhite	(248,248,255)
Gold	(255,215,0)
GoldenRod	(218,165,32)
Gray	(128,128,128)
Green	(0,128,0)
GreenYellow	(173,255,47)
HoneyDew	(240,255,240)
HotPink	(255,105,180)
IndianRed	(205,92,92)
Indigo	(75,0,130)
Ivory	(255,255,240)
Khaki	(240,230,140)
Lavender	(230,230,250)
LavenderBlush	(255,240,245)
LawnGreen	(124,252,0)
LemonChiffon	(255,250,205)
LightBlue	(173,216,230)
LightCoral	(240,128,128)
LightCyan	(224,255,255)
LightGoldenRodYellow	(250,250,210)
LightGrey	(211,211,211)
LightGreen	(144,238,144)
LightPink	(255,182,193)
LightSalmon	(255,160,122)

Table 6-3:Standard Color Names Recognized by QuickCap and Decimal Equivalents *(Continued)*

Color Name	Decimal
LightSeaGreen	(32,178,170)
LightSkyBlue	(135,206,250)
LightSlateGray	(119,136,153)
LightSteelBlue	(176,196,222)
LightYellow	(255,255,224)
Lime	(0,255,0)
LimeGreen	(50,205,50)
Linen	(250,240,230)
Magenta	(255,0,255)
Maroon	(128,0,0)
MediumAquaMarine	(102,205,170)
MediumBlue	(0,0,205)
MediumOrchid	(186,85,211)
MediumPurple	(147,112,219)
MediumSeaGreen	(60,179,113)
MediumSlateBlue	(123,104,238)
MediumSpringGreen	(0,250,154)
MediumTurquoise	(72,209,204)
MediumVioletRed	(199,21,133)
MidnightBlue	(25,25,112)
MintCream	(245,255,250)
MistyRose	(255,228,225)
Moccasin	(255,228,181)
NavajoWhite	(255,222,173)
Navy	(0,0,128)
OldLace	(253,245,230)

Table 6-3:Standard Color Names Recognized by QuickCap and Decimal Equivalents *(Continued)*

Color Name	Decimal
Olive	(128,128,0)
OliveDrab	(107,142,35)
Orange	(255,165,0)
OrangeRed	(255,69,0)
Orchid	(218,112,214)
PaleGoldenRod	(238,232,170)
PaleGreen	(152,251,152)
PaleTurquoise	(175,238,238)
PaleVioletRed	(219,112,147)
PapayaWhip	(255,239,213)
PeachPuff	(255,218,185)
Peru	(205,133,63)
Pink	(255,192,203)
Plum	(221,160,221)
PowderBlue	(176,224,230)
Purple	(128,0,128)
Red	(255,0,0)
RosyBrown	(188,143,143)
RoyalBlue	(65,105,225)
SaddleBrown	(139,69,19)
Salmon	(250,128,114)
SandyBrown	(244,164,96)
SeaGreen	(46,139,87)
SeaShell	(255,245,238)
Sienna	(160,82,45)
Silver	(192,192,192)

Table 6-3:Standard Color Names Recognized by QuickCap and Decimal Equivalents *(Continued)*

Color Name	Decimal
SkyBlue	(135,206,235)
SlateBlue	(106,90,205)
SlateGray	(112,128,144)
Snow	(255,250,250)
SpringGreen	(0,255,127)
SteelBlue	(70,130,180)
Tan	(210,180,140)
Teal	(0,128,128)
Thistle	(216,191,216)
Tomato	(255,99,71)
Turquoise	(64,224,208)
Violet	(238,130,238)
Wheat	(245,222,179)
White	(255,255,255)
WhiteSmoke	(245,245,245)
Yellow	(255,255,0)
YellowGreen	(154,205,50)

implicitDepth (Default)
explicitDepth

Indicates whether the depth specified in the **layer** declaration is to be automatically applied to objects on the layer. For the default, **implicitDepth**, objects after a corresponding **setLayer** declaration must be defined without z values. The gds2cap tool generates QuickCap **layer** declarations with **explicitDepth** for gds2cap layers that include the **adjustTop**, **adjustBottom**, or **adjustHeight** properties.

eps [=] ***value***

Defines the dielectric value associated with this layer. Any **dielectricLayer** statement representing a layer uses the dielectric value and the precedence of the layer.

ID[**=**|**:**] ***1-32767***
Defines an ID that can be used for reference in lieu of the layer name. This is useful for streamlining layer references by **setLayer** declarations. If not specified, **ID** defaults to the next available ID value (starting at 1) when the layer is defined.

nSublayers [**=**] ***count***
Defines the *effective* number of sublayers associated with a trapezoidal layer (containing **topBias**[**X**|**Y**] or **bottomBias**[**X**|**Y**]). The integration surface associated with such a layer includes ***count*** steps.

pattern[**=**|**:**] ***pattern***
Defines the plotting pattern associated with this layer. QuickCap recognizes 16-digit hexadecimal values (**0x** followed by 16 hexadecimal digits, **0**–**9** and **a**–**f**, or **A**–**F**) and the keywords **cross**, **light**, **light/**, **light**, **medium**, **medium/**, **medium**, and **dark**.

plotLayer[**=**|**:**] ***layerRef*** *Conductor layer*
Specifies that plotting parameters used by QuickCap are to be based on another layer. This is useful, for example, when representing a nonplanar interconnect layer as multiple planar layers: each planar layer must have the same plot parameters.

precedence[**=**|**:**]***1-32767***
Sets the precedence value of a layer. The ***precedenceValue*** must be a positive integer. Layers with higher precedence values take higher precedence. The gds2cap tool generates the **precedence** property in response to precedence specifications in the technology file. For layers sharing a top or bottom surface and with the same precedence (or no precedence) using **layerPrecedence implicit** (the default), QuickCap gives higher precedence to the thinner layer. To change this default behavior, specify **layerPrecedence explicit** (see **layerPrecedence** on page 6-9).

spacing[**=**|**:**] ***distance***
The **spacing** value must be the smallest distance to other nets at the same height.

width [**=**] ***distance***
Specifies the minimum line width.

trap[**ezoid**]**Layer**[**=**|**:**] ***layerID***
Specifies the main layer associated with a trapezoid sub-layer. This allows modeling of nonlinear edges by dividing a layer into several sub-layers, each with its own top and bottom bias. Such sub-layers can each reference a single trapezoid layer that links the layers together for graphics preview mode.

type[**=**|**:**] ***description***
The **type** statement can be specified but is not used by QuickCap. Recognized layer types are **device**, **dielectric**, **float**, **ground**, **interconnect**, **junction**, **label**, **miscellaneous**, **resistor**, and **via**. Any other layer type triggers a warning, but does not affect the QuickCap run.

Setting Layers and Depth

Statements for setting the layer (**setLayer**) can be declared inside or outside a list of primitives in a 3D environment.

setLayer ***layerRef***
: Sets the layer and (in a 3D environment) the depth of subsequent geometric primitives. ***layerRef*** can be a layer name or its ID. The **setLayer** statement can be embedded in a list of geometric primitives.

Adjusting Depth

The **adjustDepth** declaration allows the depth within a nominal range (z) to be varied as a function of position. The depth variation is defined in a table file (a map). Any z value within the nominal range (specified in the **adjustDepth** declaration) is scaled, and z values above the nominal range are increased or decreased as necessary.

For example, if the depth is 1µm to 1.5 µm and the table specifies at some point a value of -0.2 (**delta**, **rel**), 0.8 (**total**, **rel**), -0.1e-6 (**delta**, **abs**) or 0.4e-6 (**total**, **abs**), then QuickCap scales nominal *z* values in the range 1µm to 1.5µm using the formula 1µm + (z – 1µm) * 0.8, and decreases by 0.1µm nominal *z* values larger than 1.5µm. An **adjustDepth** statement has the following format.

The use of **adjustDepth** tables can significantly slow the convergence rate of QuickCap. QuickCap implements a *uniform* **adjustDepth** table by adjusting all affected z values, resulting in more efficient convergence for uniform tables.

QuickCap adjusts the stack uniformly to account for any **adjustDepth** layers that are constant or nearly constant.

adjustDepth ***fileName z0 z1*** [***properties***]
: Uses an xy table (map), ***fileName***, to apply position-dependent variations in the stack due to a varying layer thickness (nominally ***z0*** to ***z1***). The **adjustDepth** statements must be defined from the bottom up, and the depths cannot overlap. Distinct **adjustDepth** statements can reference the same file. QuickCap expects the file to be the same table format as generated by gds2density: A table with index lines defining *x* and *y* values, each as an array (minimum value, maximum value, and number of values). QuickCap does *not* recognize a list-format table, as generated by gds2density with the **-pt** or **-rect** option. QuickCap reads the variation map and modifies the stack (as a function of position) by scaling the nominal *z* values within the range ***z0*** to ***z1***, and appropriately increasing or decreasing any nominal *z* values larger than ***z1***. For any nominal z value, QuickCap applies the effects of all underlying **adjustDepth** statement. See Figure 6-2 for an example. The **adjustDepth** statement can include the following properties.

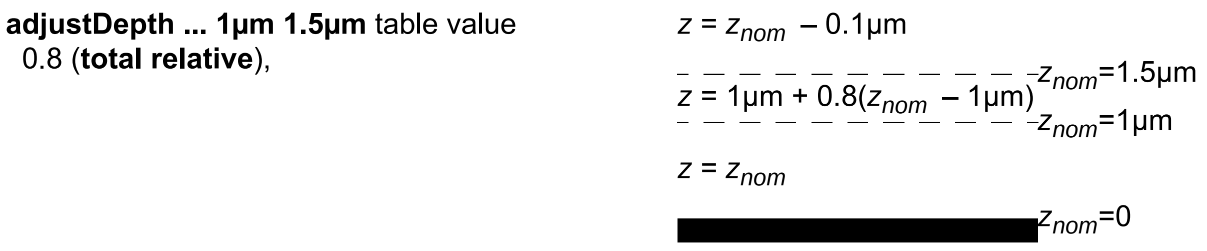
total: Specifies values in the table defined by the **adjustDepth** file is the total. A total value is equivalent to an incremental value plus 1 (for **relative** values).

rel[**ative**]: Specifies values in the table defined by the **adjustDepth** file are relative. A relative value is equivalent to the absolute value divided by the nominal depth (***z1*** – ***z0***).

QuickCap expects ***fileName*** to be a file in the same table format as generated by gds2density: A table with index lines defining *x* and *y* values, each as an array (minimum value, maximum value, and number of values). Keywords that normally appear in a table formatted by gds2cap are optional.

Note: The QuickCap implementation of **adjustDepth** statements is designed for a stack that has a negligible variation from between line and any nearby features that affect capacitance.

Figure 6-2: How adjustDepth Affects Depth


adjustDepth ... 1µm 1.5µm table value 0.8 (**total relative**),
$z = z_{nom} - 0.1\mu m$
$z_{nom}=1.5\mu m$
$z = 1\mu m + 0.8(z_{nom} - 1\mu m)$
$z_{nom}=1\mu m$
$z = z_{nom}$
$z_{nom}=0$

7. Symmetry

The symmetries of a problem are reflected in the QuickCap results only in a statistical sense. Two capacitance values that should be the same because of such symmetry are only approximately equal. If you specify the reflection or rotation symmetries of the problem, however, QuickCap takes them into account when correlating its statistics.

For example, if C_{ab} (10 fF±0.5 fF) and C_{cd} (11 fF±1.1 fF) are equivalent due to symmetry, QuickCap, if informed of the symmetry relationships, uses an optimally weighted average, (10.17 fF±0.46 fF) for both results.

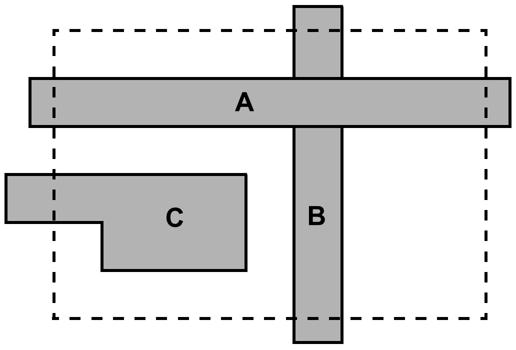
Translational symmetry (periodicity) is handled by the **xCell** and **yCell** input-file statements (see page 6-4). These statements can be used to reflect periodicity of the geometry (where nets in adjacent cells are unique) or of the electric field (where the signals on the nets must also be periodic).

Periodic Geometric Structure

QuickCap can analyze a periodic structure without approximating the E-field as periodic. For periodic structures with a small unit cell or for periodic structures far above the groundplane, such an approximation can introduce considerable error.

Using the **xCell** and **yCell** statements (see page 6-4) allows QuickCap to mathematically represent the periodicity of a structure. In the replicated structure, nets are assigned names based on which cell they are in and whether they overlap their analogous representations in adjacent cells. In this section, a net that is in the unit cell is referred to as a *local net*. A net that is in a replicated cell is referred to as an *external net*. The Figure 7-1 on page 7-2 depicts how **xCell** and **yCell** statements determine the unit-cell boundaries.

Figure 7-1: Unit Cell (Dashed Box) of a Periodic Structure



In this figure, Net A is a line, extended in *x*, because it overlaps its replicated self in that direction. Similarly, net B is a line, extended in *y*. Net C is local to the cell. Even though it crosses the boundary, it does not overlap itself when replicated. External nets associated with the local net A are A[+1*y*], A[–1*y*], A[+2*y*], and so on. External nets associated with the local net B are B[+1*x*], B[–1*x*], B[+2*x*], and so on. External nets associated with the local net C are C[+1*x*], C[–1*x*], C[+1*y*], C[+1*x*][+1*y*], C[–1*x*][+1*y*], and so on. Note that it is not necessary that a local net be entirely within the unit cell. In this case, a part of C is in the cell to the left of the unit cell and, consequently, part of C[+1*x*] is in the unit cell even though it is associated with the cell to the right of the unit cell.

The number of nets for which separate statistics are kept is controlled by the third argument (optional) in the **xCell** and **yCell** statements. For example:

```
xCell 0 3 n=1
yCell 0 2 n=2
```

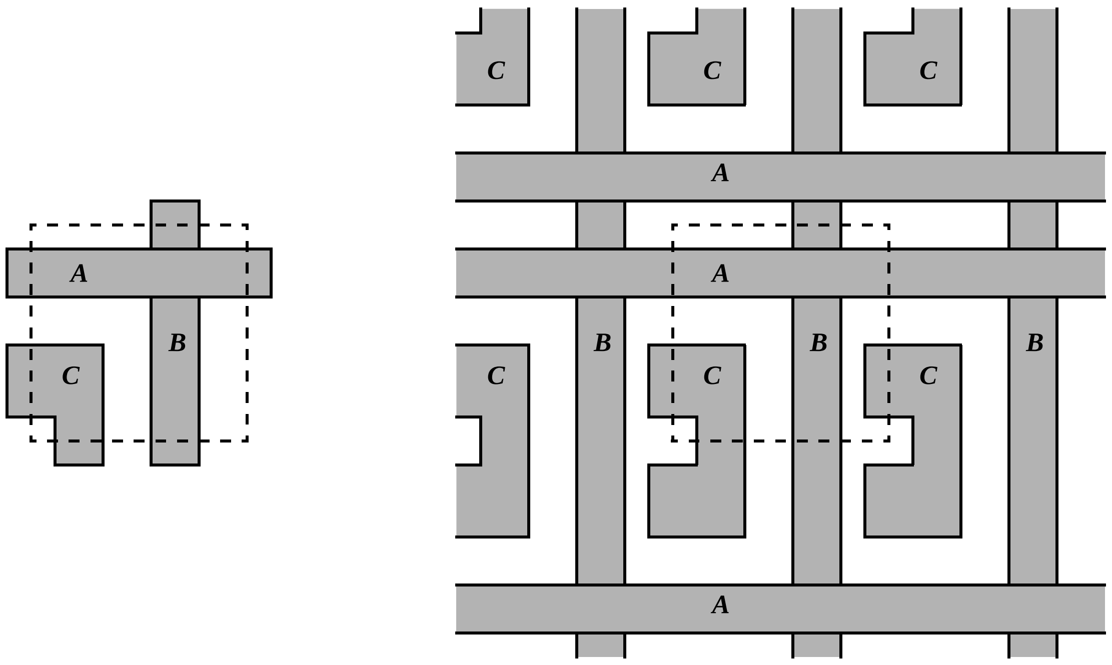
These statements keep statistics for all cells within a 3-by-5 block (one cell in *each* direction from center in *x*, and two in *y*). The geometric periodicity extends to *infinity*, even though separate statistics are not maintained for distant cells—the capacitances to external nets in cells outside the defined block are lumped to the spurious net.

Periodic and Reflective Boundary Conditions

QuickCap can apply periodic or reflective boundary conditions to the *E*-field when the **xCell** or **yCell** statements is used in conjunction with the **periodicBoundary** or **reflectiveBoundary** keyword (see page 6-5). This feature is useful, for example, for comparing QuickCap against a capacitance extractor that uses such approximations. An application of both conditions is shown in Figure 7-2 on page 7-3.

A periodic boundary condition is equivalent to replicating the structure *and* the net names into a periodic array. A reflective boundary condition is equivalent to replicating the structure *and* the net names into a periodic array, mirroring the structure in every second cell. Periodic and reflective boundary conditions are features supported by QuickCap to investigate approximations imposed by other capacitance extractors that might routinely use such conditions. The general application of these boundary conditions to "real" problems is not recommended.

Figure 7-2: Part of Effective Structure (Right) When Unit Cell (Left) is Periodic in x and Reflective in y



8. Output

This chapter describes various types of output.

The log file, generated unless **-log** is specified on the command line, contains information output to the terminal related to file input, start and stop times; the QuickCap command line used; and any warnings or errors.

You can specify **-spice** on the command line to produce a capacitance netlist. It contains a netlist of the capacitance results.

The summary file lists the most recent results. You can view this file if the program is running in the background or after the program has terminated. In the latter case, the summary file shows the results of the run.

You can produce a tab-delimited output file by specifying **-table** on the command line. The tab-delimited output file is in a form that can be easily imported to spreadsheet programs.

The Log File

The log file contains a summary of the input files, information related to the problem, and a list of start and stop times. The log file is not created or modified if **-log** (see page 5-5) is specified on the command line.

The log file is named according to the input files—the input-file name with a .log suffix. For example, if the input file is alu or alu.hdr, the log file name is alu.log. Or, if there are two input files, hbt.header and alu, the log file name is hbt.header.alu.log.

Each time you start QuickCap, the QuickCap version, start time, and command line are printed to the log file. Each time QuickCap terminates either normally or due to a detected error, the stop time is printed. Any warning or error messages are also printed to the log file.

If QuickCap terminates when running with **-p -log** specified (no output to the terminal and no log file), it generates an error message on standard output although the **-p** command-line option is specified.

The Numeric Output File

If you specify **-numeric** on the command line, QuickCap outputs a numeric output file just before it terminates. The output consists of header information, a list of node names, and a list of capacitance results with uncertainties.

The numeric output file is named according to the input files—the input-file name with a .numeric suffix. For example, if the input file is alu or alu.hdr, the numeric output file name is alu.numeric. Or, if there are two input files, hbt.header and alu, the numeric output file name is hbt.header.alu.numeric.

Output Type

After the header, an optional line contains a single keyword describes the output.

asymmetric
Indicates values are asymmetric. It can include, for example, capacitance from A to B and capacitance from B to A as separate entries. QuickCap generates asymmetric results in response to the **-asymmetric** option (page 5-4).

symmetric (Default)
Indicates values are symmetric capacitance from A to B and capacitance from B to A.

Node Names

After the header, a line containing two integer values indicates the number of node names and the maximum number of characters in a name:

```
numNodes maxChars
```

Each of the subsequent ***numNodes*** lines contains the name of a node:

```
nodeName
```

The first such line is the name of node 1, the second is the name of node 2, and so on.

Values

After the list of node names, the remainder of the file consists of capacitance declarations of the form:

```
nodeA nodeB capacitance uncertainty
```

The ***nodeA*** and ***nodeB*** values are integers corresponding to the node names by their *rank*. The number 1 refers to the first node named, 2 refers to the second node named, and so on. A 0 refers to ground.

The ***capacitance*** and ***uncertainty*** values are floating-point numbers corresponding to the capacitance and uncertainty (standard error) and are in units of farads (F).

Unlike the capacitance netlist, the numeric output file does *not* include any comments related to the capacitance values.

The Capacitance Netlist

If you specify **-spice** on the command line, QuickCap outputs a capacitance netlist just before it terminates. The output consists of header information and a list of extracted capacitances with uncertainties.

Unless **netlistName** is declared in an input deck, the capacitance netlist is named according to the input files—the input-file name with a .spice suffix. For example, if the input file is alu or alu.hdr, the capacitance netlist name is alu.spice. Or, if there are two input files, hbt.header and alu, the capacitance netlist name is hbt.header.alu.spice.

The flag **-nodes** influences which values are output.

Netlist Header Information

The header information in the capacitance netlist contains information created by QuickCap and any header information from the input files that QuickCap used. Lines in the header begin with an asterisk.

The first line indicates the QuickCap version number and the time the capacitance netlist was created. QuickCap then prints to the capacitance netlist the name of each input file used, followed by any header information from the QuickCap input file. The header information in a QuickCap input file consists of any comments before the first command.

Capacitance Values

Small coupling-capacitance terms are lumped with capacitance to ground. *Small values* are those that are negative or less than an optional capacitance argument to the **-spice** command-line flag. If **-spice** includes a percentage argument, *small values* include values with an uncertainty larger than the specified percentage.

For cell analysis (***xCell*** with ***periodicGeometry***, or ***yCell*** with ***periodicGeometry***), QuickCap comments (non-spurious) capacitance to objects outside the unit cell by prefixing the line with *cellCap, and adds the capacitance to the ground capacitance.

Small capacitances only: `(misc)`
: Appears on the line after the uncertainty.

Combination: `(total)`
: Appears on the line after the uncertainty, and subsequent lines are comments (beginning with an asterisk) that list the capacitances that were added together. Each line includes the designation `(misc)` or `(ground)`, describing the origin of the term.

The Summary File

QuickCap prints the summary file after QuickCap terminates. The first line in the summary file is the QuickCap version and built time, and so on. Coupling and total capacitances are reported with net name, capacitance value, and 1 sigma error. The following is the coupling capacitance format:

```
<net1_name> <net2_name> <coupling_cap_value> <1 sigma error>
```

If **-asymmetric** is specified, the coupling capacitance format is:

```
<net1_name> <net2_name> <coupling_cap1> <1 sigma error of cap1>
<coupling_cap2> <1 sigma error of cap2>
```

For the total capacitance, the report format is:

```
<net name> <total capacitance> <1 sigma error of this total capacitance>
```

The summary file is named according to the input files, the input file name with a .summary suffix. For example, if the input file is alu or alu.hdr, the summary file name is alu.summary. If there are two input files, hbt.header and alu, the summary file name is hbt.header.alu.summary.

Runtime

Before any capacitance values are output, QuickCap outputs a line that contains the *root file name* (the name of the input file or the concatenated names of the input files), the elapsed time, and the CPU time.

The elapsed time is the total time elapsed on the problem. If this is a continued run, it includes the elapsed time from the previous runs. The elapsed time does not include time spent waiting for you to input a run length or the time you spent examining the problem graphically.

The CPU time is the total time spent by the random-walk kernel (including the CPU time from any previous runs) and does not include the overhead required to read in the problem, generate the surfaces of integration, write the summary file, and so on.

Convergence Information

The convergence information printed depends on whether you specify **-v** on the command line and whether any goals are met (see **-g**, “Convergence-Goal Options” on page 5-2).

If you specify **-v** on the command line and any specified goals are not met, Quick Cap outputs the least known capacitance value, which is the value that requires the most number of walks to achieve its goal.

If you specify **-v**, QuickCap outputs its incremental and cumulative concentration. These values show how much QuickCap is concentrating its effort on refining the capacitance values. A concentration of 1 is “uniform.” As the QuickCap efforts become more focused, the concentration increases.

If any goal was specified, QuickCap outputs either an estimate of the time left to achieve the goal or a message stating that the goal has been achieved.

Parallel Operation

For parallel operation (see **-LSF** on page 5-5 and **-MPP** on page 5-6), the parent process is aware of the number of walks performed by the child processes, but does not collect statistics until the end of the run. Until the statistics are collected, the summary includes a note indicating that results from LSF, MPP, and NOW processors are not included. If you specify **-v** on the command line, the

QuickCap report is more detailed, including the number of outstanding walks and the efficiency (including loss of efficiency due to system load and due to the QuickCap time for backing up statistics).

At the end of a parallel run, after statistics are gathered from remote processes, the elapsed time is the largest elapsed time of any of the processes (generally the parent process) and the CPU time is the sum of the CPU times—generally a larger figure than the elapsed time.

Table Output

If you specify **-table** on the command line, QuickCap outputs a tab-delimited output file just before it terminates. The output consists of header information, a header line, and table data.

The table output file is named according to the input files—the input-file name with a .table suffix. For example, if the input file is alu or alu.hdr, the table output file name is alu.table. Or, if there are two input files, hbt.header and alu, the table output file name is hbt.header.alu.table*.*

Table File Header Information

The header information in the table output file contains information created by QuickCap and any header information from the input files that QuickCap used. All lines in the header have an asterisk (*) in the first column.

The first line indicates the QuickCap version number and the time the table output file was created. QuickCap then prints to the table output file the name of each input file used, followed by any header information from the QuickCap input file. The header information in a QuickCap input file consists of any comments before the first command.

If you use a gain file, QuickCap prints the name of the gain file and its contents to the table output file.

Terminal

Unless you specify **-p** on the command line (see page 5-7), output to the terminal basically consists run status and time estimation.

9. Graphics Preview Mode

When you specify **-x** as a command-line option, QuickCap shows a top view of the structure after it processes the input file. You control the graphics display by using the keyboard and the mouse.

When you exit the graphics preview mode (Ctrl+q), QuickCap exists without normal operation.

Commands in the **View** menu control the size and location of the view window as well as the window type, whether the view is 2-D or 3-D. See the following figures for the following window types:

- Figure 9-1 on page 9-2: A top-view window that shows the top view of the structure.
- Figure 9-2 on page 9-2: A cross section window that shows a cross section of the structure in the xy, xz, or yz plane.
- Figure 9-3 on page 9-3: A wireframe window that shows a projected view of the structure viewed from a user-defined orientation.
- Figure 9-4 on page 9-3: A stereo window that shows two side-by-side projected views of the structure from orientations that you define.
- Figure 9-5 on page 9-4: A view orientation window that does not show the structure, but is used to define the orientation of the wireframe and stereo windows.

Figure 9-1: Top View

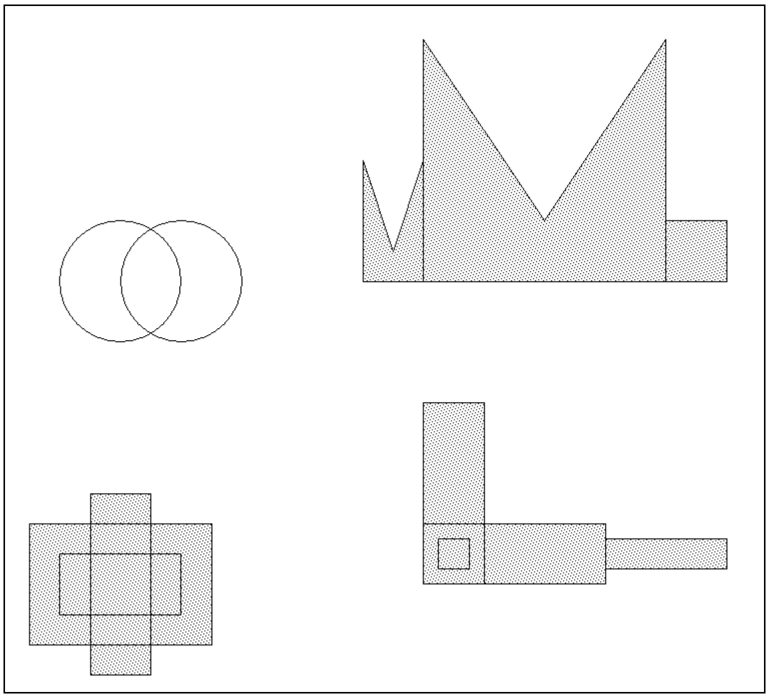


Figure 9-2: Side View

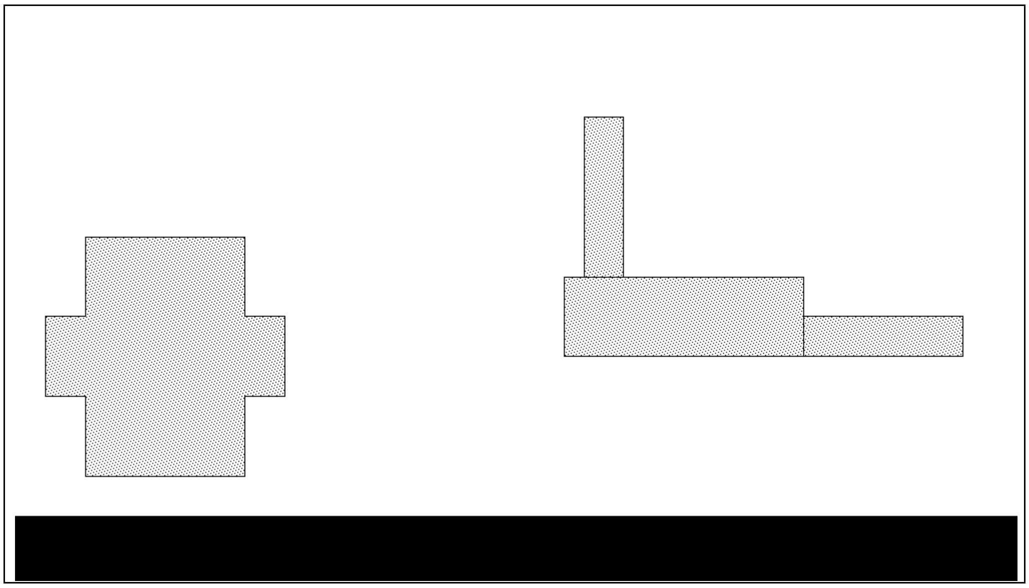


Figure 9-3: Wireframe View (Hidden Lines Removed)

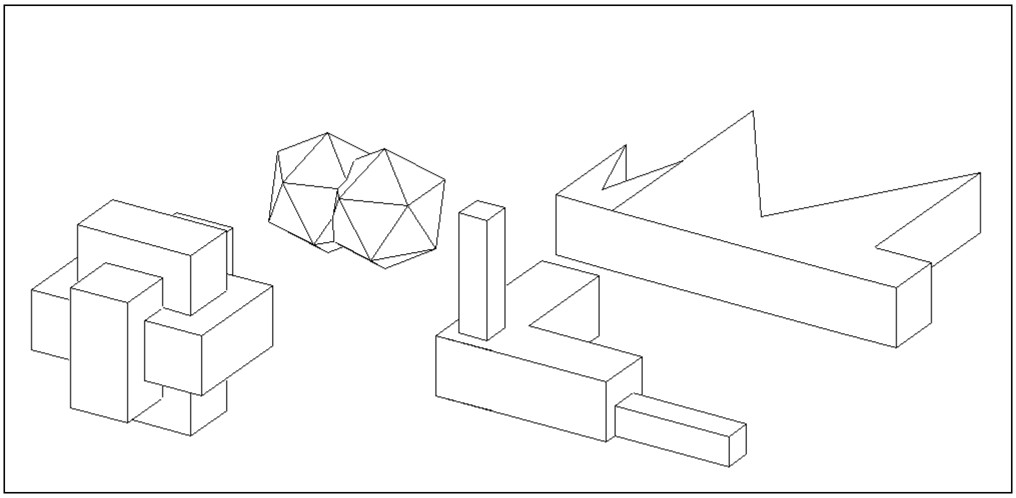


Figure 9-4: Stereo View (Hidden Lines Removed)

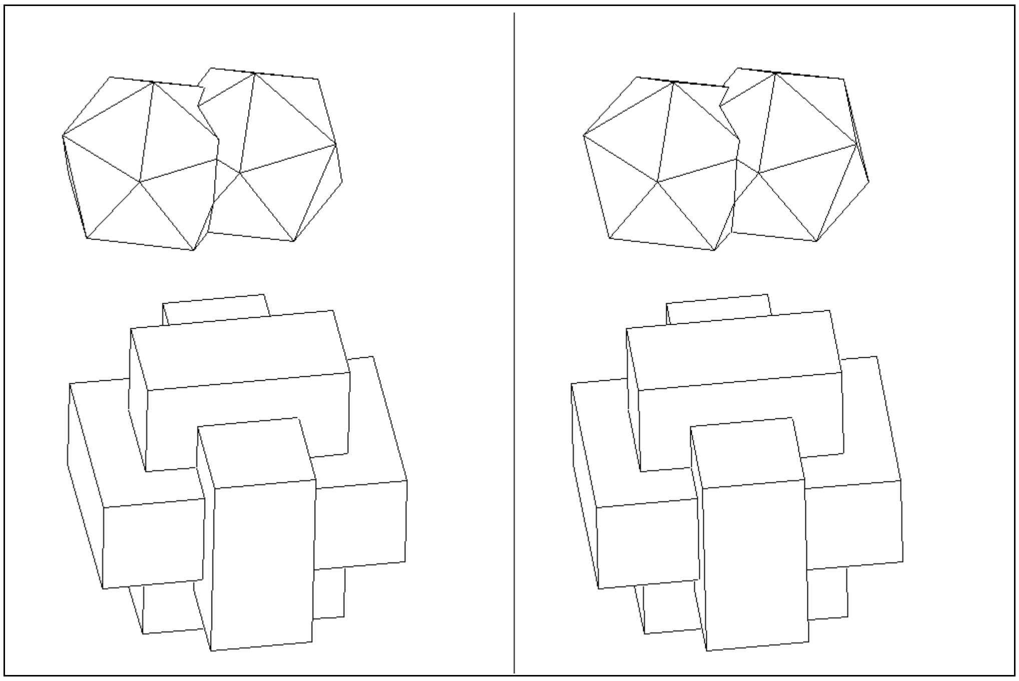
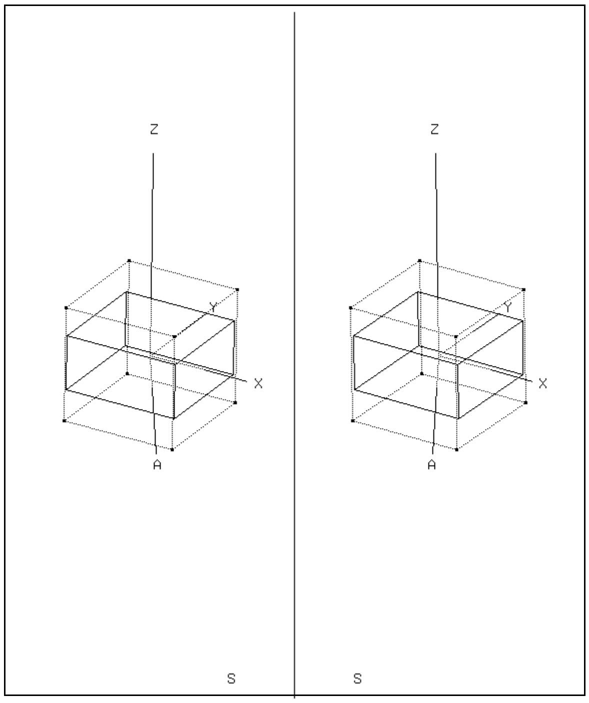


Figure 9-5: Orientation View



The following terms are discussed in this chapter:

- 2-D window refers to a top-view window or a cross section window.
- 3-D window refers to a wireframe window or a stereo window.
- Structure window refers to a 2-D window or a 3-D window (any graphics window except a view-orientation window).

QuickCap shows one of the following views:

- A layer-based view: Objects are colored and patterned by layer
- A net-based view: Objects are colored and layered by net.

In the QuickCap deck , use **layer** declarations to define the default layer colors and patterns, and use the **setLayer** declaration to associate objects with layers. If objects are not assigned with a layer, layers are inferred from the z values. Commands from the **Layer** menu result in a layer-based view, whereas commands from the **Nets** menu result in a net-based view.

Graphics Log Files

Log files of the low-level graphics commands that QuickCap uses to display data can be created during the graphics preview mode. QuickCap starts recording a graphics log file when you use the **-xRecord** option. However, QuickCap does not record when you use the **-x** option. To use the **-x** option to start recording to a log file, you must type an open parenthesis (**(**).

For more information about the **-xRecord** and **-x** options, see the description on page 5-12.

The first graphics log file is named ***root***.xLog. You can close the log file and open a new log file by typing a close parenthesis (**)**) and then an open parenthesis (**(**). Subsequent graphics log files are named ***root***.xLog2, ***root***.xLog3, and so on.

To ensure that the graphics log file starts with a command to open the graphics window, type "**(**" just before a command that reopens a window, such as changing the view window (**0**, **1**, **2**, or **3**). The graphics log file is not currently in any standard graphics language. The language is likely to change when such a standard is recognized.

Graphics Record and Play Files

The **-xRecord** option causes QuickCap to record all keyboard and mouse graphics entries, until you type **Ctrl+.** or you exit graphics preview using **Ctrl+q**. This record of graphics entries is maintained in a file named ***root***.xRec. For verbose output (**-v**), each graphics entry is described as a comment in the file.

While recording, use a comma (**,**) or semicolon (**;**) to insert a short or long pause, respectively, into the graphics record file. On playback, the display pauses at any such points.

To create a graphics play file, rename the file to ***root***.xPlay and run QuickCap on the same input deck with the **-x** or **-xRecord** option. The playback continues until the end of the record file, or you interrupt by clicking the mouse or using the keyboard.

Mouse Input

Mouse control consists of using left-click or right-click, optionally followed by dragging the mouse before releasing. Right-click at the top of a text box to open the text box for editing or to move the box. For more information, see "Markup Text" on page 9-11.

The mouse position is also used for measuring distances, perimeters, and areas. For more information. see "Measurement Points" on page 9-11. Table 9-1 shows mouse procedures and their use.

Table 9-1: Mouse Procedures

Procedure	Task
Click and release	In a structure window, center the view at the indicated point.
Click and drag horizontally or vertically	In a top-view window, show a cross section.
Click and drag diagonally	In a structure window, zoom to the specified rectangular view.
Click and drag a handle	In a view-orientation window, change the value of the corresponding handle: **A**: Altitude (viewing height), in degrees relative to the xy plane **S**: Stereo angle (degrees) between the two stereo-view panels **X**: Azimuth (rotation counterclockwise around the z-axis) **Y**: Azimuth (rotation counterclockwise around the z-axis) **Z**: Altitude (viewing height), in degrees relative to the xy plane Any cube corner: Distortion, a function of z
Right-click on object	In a top-view window or a 3D-view window, select the net. In a side-view window, select the layer.
Right-click on text	In a 2D-view window, open text
Right-click and drag text	In a 2D-view window, move text

Keyboard Input

This section describes keyboard commands that are categorized into eight menus: **View**, **Show**, **Colors**, **Markup**, **Nets**, **Layers**, **Surface**, and **Help**.

View Menu

The **View** menu commands control the window type, and the view scale and position. The window size on the screen is adjusted to fit most of the screen. You can specify the screen size using the environment variables **QUICKCAP_DISPLAY_HEIGHT** and **QUICKCAP_DISPLAY_WIDTH** (in pixels). The window size in default is 800x600.

Table 9-2 on page 7 lists the keyboard keys for using to perform tasks in the **View** menu. Table 9-1 on page 6 shows the related mouse procedures for the **View** menu.

Commands that rotate a 3D view (>, <, +, -) rotate around a point in a reference plane that is set by the following operations:

- Centering highlighted objects (**c**) in a structure view.
- Selecting a layer (**l**) in a structure view.
- Executing a left-click or a right-click in a cross section.

Table 9-2: View Menu Keyboard Commands

Keyboard Command	Task
0	Set window to top (2D) view.
1	Set window to wireframe (3D) view.
2	Set window to stereo (3D, two-panel) view.
c	S Center view on selected object.
Ctrl+c	Input the center and size of a window.
Ctrl+M	Move coordinates
o	Zoom out by 133 percent (structure window).
i	Zoom in by 50 percent (structure window).
h	Zoom and move structure window to home view.
H	Set home view to current view (structure window).
>	Rotate right (3D or view-orientation window).
<	Rotate left (3D or view-orientation window).
+	Rotate up (3D or view-orientation window).
-	Rotate down (3D or view-orientation window).
r	Repaint (structure window).
R	Toggle removal of hidden lines (3D window).
~	Toggle background between white and black.
Ctrl+o	show overlapped polygons in the cross section

Using Cross Section in Difference Dimensions

- By default, displays an xy cross section. This is the same orientation as the top view, but only shows objects passing through the plane (a z value) of the cross section. You can display an xy cross section from the top view by moving the view to a critical z value (Ctrl-M), and type a z value as z [=]coordinate, or from a side view by holding down the mouse, dragging horizontally, and releasing. You can now switch between an xz and an yz cross section by holding down the mouse, dragging vertically, and releasing.
- From the top view or any cross section (xy, xz, or yz), you can now center the view on an x-, y-, or z-coordinate by using Ctrl-M, and entering a coordinate and value as x|y|z[=]coordinates. If the coordinate is perpendicular to the plane of the view (for example, x in an yz view), the cross section at that value is drawn.
- Visualization window now displays the coordinate of the plane shown in an xy, xz, or yz cross section in square brackets [coordinates], rather than an asterisk (*).

Specifying the Center and Size of a Window

Use Ctrl+c to specify the center coordinates and the window size. The text you type displays on the top left of the graphics window. The coordinates indicate the following:

- First two coordinates, separated by a comma, define the center of the window. Type @ at the beginning if you want visualization window to automatically insert the current center.
- Third and fourth coordinates, also separated by commas, indicate the width and height of the window. Type @ after the second coordinate if you want the visualization window to automatically insert the current width and height. This command is useful for displaying the same view of similar layouts in different field solver runs to compare.

Use the following keys to manage the window,

- Backspace or Delete key: Deletes the last character or return.
- Esc key: Exits without accepting the specified window.
- Enter or Return key: Selects the Delete window.

Show Menu

Table 9-3 on page 9 lists **Show** menu commands to change objects to display. The commands apply only to a structure window.

Table 9-3: Show Menu Keyboard Commands

Keyboard Command	Task
*	Toggle to show the highlighted net or layer.
d	Highlight device regions or the next dielectric region (side-view window)
D	Toggle display of dielectric and device regions (side-view window)
t	Toggle display of trapezoid details.

Dielectric regions are displayed in the side-view window (for 3-D problems, **dimensionality 3D**) or top-view window (for 2-D problems, **dimensionality 2D**). Dielectric regions appear as patterned black regions. Dielectric regions with lower dielectric constants appear using lighter patterns, except that any dielectric region matching the background dielectric (eps) is blank.

Colors Menu

Use the Colors menu commands to mark the current 2-D (top or side) view. Symbols related to measurement (@, #,!) are erased if you redraw the view, for example, by zooming in or by selecting a net. However, text is preserved.

Table 9-4 lists the color menu keyboard commands. Table 9-5 on page 11 list the keyboard keys for using to perform tasks in the Markup menu. Table 9-1 on page 6 shows the related mouse procedures for the Markup menu.

Table 9-4: Color Menu Keyboard Commands

Keyboard Command	Task
Ctrl+r	Set color of selected objects (or open text box) to red.
Ctrl+g	Set color of selected objects (or open text box) to green.
Ctrl+b	Set color of selected objects (or open text box) to blue.
Ctrl+y	Set color of selected objects (or open text box) to yellow.
Ctrl+c	Set color of selected objects (or open text box) to cyan.

Table 9-4: Color Menu Keyboard Commands *(Continued)*

Keyboard Command	Task
Ctrl+m	Set color of selected objects (or open text box) to magenta.
Ctrl+d	Set color of selected objects (or open text box) to black.
Ctrl+i	Name the color of selected objects (or open text box). Selecting the color by name as explained in "Selecting a Color by Name" on page 9-10.
Ctrl+a	Assign arbitrary colors to distinct nets or layers that do not have assigned colors.
Ctrl+A	Clear arbitrary colors.
Ctrl+p	Set pattern to next in series
Ctrl+k	Reset all color information to default values.

Selecting a Color by Name

Use Ctrl+i to select a color by name. The QuickCap tool accepts the following formats when you enter the last character:

- Shaded basic colors
- Three-digit base-six values
- Decimal RGB representations
- Six-digit hexadecimal RGB colors.

The text you type displays on the top left of the graphics window along with any auto-completion. Use the Backspace or Delete key to delete the last character or return. Use the Esc key to exit without accepting the indicated color. Use the Enter or Return key to select a named color.

Note: Currently, color for floating nets are not supported.

Markup Menu

For description for the **Markup** menu command that is similar to colors menu command, see "Colors Menu" on page 9-9.

Table 9-5 lists the **Markup** menu keyboard commands. Table 9-5 on page 11 list the keyboard keys for using to perform tasks in the Markup menu. Table 9-1 on page 6 shows the related mouse procedures for the Markup menu.

Table 9-5: Markup Menu Keyboard Commands

Keyboard Command	Task
Ctrl+t	Inserts text at cursor.
Ctrl+T	Inserts default text name of selected net or layer at cursor.
Ctrl+B	Sets color of selected objects (or open text box) to blue.
Ctrl+X	Removes all text.
@	Marks as first measurement point.
#	Marks as additional measurement point.
!	Ends measurement.
=	Prints on the terminal a list of hidden and visible dielectric, conformal, polygon, or device regions at the cursor position.

Markup Text

A box appears around any edited text when you press Ctrl+t or after you select the text with the mouse. Use any of the color keys (Ctrl+r, Ctrl+g, Ctrl+b, Ctrl+y, Ctrl+c, Ctrl+m, or Ctrl+d) to change the text color and color of subsequent text. Printable characters you type are appended to the text. Use a carriage return (the Enter key) for a new line. Each line is centered. Use the Backspace or Delete key to delete the last character or return. Use the Esc key to exit from the editing mode. Text defined in a side view appears in a 3D view as well.

Measurement Points

When measurement is started (@), in addition to displaying the coordinates at the left corner of window, the Visualization window displays the distance from the last @ or # point and a line from that point to the cursor. As additional points are added (#), the Visualization window marks the path with a heavy line. When an additional point is on top of the first @ point, the Visualization window also displays the total distance traversed and the area enclosed.

Nets Menu

Use the **Nets** menu commands to show a net-based view in a structure window and to select nets to be colored or patterned (see “Colors Menu” on page 9-9).

Use asterisk (*) to toggle to see only the selected net displays. For information, see “Show Menu” on page 9-9. Table 9-6 on page 12 lists keyboard keys to perform tasks in the Nets menu.

Any net selected in a net-based view is shown with wide outlines. A single highlighted net without a defined color is colored by layer.

Table 9-6: Nets Menu Keyboard Commands

Keyboard Commands	**Task**
n	Highlight next (arbitrary) net.
N	Remove the highlight from any highlighted net.
Ctrl+n	Name the net to highlight. Selecting the net by name is explained in "Selecting a Net by Name" on page 9-12.
e	Highlight next (arbitrary) extracted net.
f	Highlight next floating structure.
F	Highlight all floating structures.

There is only one related mouse procedure. Select the net by right-clicking a net in a top-view or 3D view window.

Selecting a Net by Name

Use Ctrl+n to select a net by name. The text you type displays on the top left of the graphics window along with any auto-completion. Use the Backspace or Delete key to delete the last character or return. Use Enter or Return key to select the indicated net, or use the Esc key to exit without selecting the net.

Layers Menu

Use the **Layers** menu commands to show a layer-based view in a structure window and to select a layer to be colored or patterned. For more information, see "Colors Menu" on page 9-9.

Use asterisk (*) to toggle to see whether the selected layer displays. For information, see "Show Menu" on page 9-9. Table 9-7 on page 13 lists keyboard keys to perform tasks in the Layers menu.

Any layer selected in a layer-based view is shown with wide outlines.

Table 9-7: Layers Menu Keyboard Commands

Keyboard Command	Task
l	Highlight next layer.
L	Remove the highlight from layers.
Ctrl+l	Name the layer to highlight. Selecting the layer by name explained in “Selecting a Layer by Name” on page 9-13.

There is only one related mouse procedure. You select the layer by right-clicking a layer in an xz or yz cross section window.

Selecting a Layer by Name

Use Ctrl+l to select a layer by name. The text you type displays on the top left of the graphics window along with any auto-completion. Use the Backspace or Delete key to delete the last character or return. Use Enter or Return key to select the indicated layer, or use the Esc key to exit without selecting the layer.

Surface Menu

Use the **Surface** menu commands to show integration surfaces around any extract nets in a 2D view. See Table 9-8 on page 13. In a net-based view when nets are selected, only the integration nets for those nets appear. In a layer-based view when a layer is selected, only integration surfaces corresponding to objects in the selected layer displays.

Table 9-8: Surface Menu Keyboard Commands

Keyboard Command	Task
Z	Show top integration surfaces.
z	Show bottom integration surfaces.
s	Show all integration surfaces.
S	Show no integration surfaces.
Ctrl+s	Toggles whether to color all integration surfaces red, or to vary by net (net-based view) and layer (layer-based view).

Help Menu

Use the **Help** menu command to access help for using the graphics viewer.

The **Help** menu contains a single key command. Click the question mark (?) to view a brief description of each key.

Glossary

absolute uncertainty

The statistical uncertainty of an estimate in the same units as that estimate.

bias

The bias of a statistical estimate due to approximations to the problem (1 Å resolution, for example), truncation error, nonideal random-number generation, boundary approximations, and so on.

capacitance matrix

Capacitances from each extracted net to each other net. The diagonal elements in QuickCap's capacitance matrix are the self (or total) capacitances.

capacitance netlist

A QuickCap output file that contains extracted capacitance values and that can be merged with the netlist for the circuit.

capacitance to infinity

In finite problems (no groundplane and not infinitely periodic), this is the capacitance to a large distant net (such as earth ground).

checksum

An estimate of "zero" based on a column of the raw capacitance matrix. Typically, it has an uncertainty on the order of the estimate. An uncertainty much smaller than the estimate can indicate some kind of error, probably a program error, but possibly a source of truncation error due to averaging together many statistics.

command line

A line entered by the user to execute QuickCap.

confidence interval

Upper and lower "soft" limits, within which the answer might lie (with a certain *confidence level*).

confidence level

The confidence that the answer is within a specified *confidence interval.*

dielectric resolution

The spatial resolution of the dielectric interfaces.

dummy metal

See *floating net.*

effective capacitance

The capacitance of a net, including device capacitance **Cd** defined in a **net** or **node** declaration. Rather than using just the parasitic component, QuickCap uses the effective capacitance as a basis for the goal, since circuit analysis is more directly related to the effective capacitance.

effective parasitic resistance

The effective resistance of a net. This is the resistance of the net scaled by a factor that depends on details of the logic family and is the resistance used to estimate a delay time due to loading an ideal driver with the parasitic line resistance and capacitance.

effective resistance

The effective resistance of a net plus the effective resistance of a driver. This is the resistance used to estimate a delay time due to loading with the parasitic line resistance and capacitance.

error

The difference between the correct answer and the calculated answer, caused by bugs, truncation error, approximation to the problem being solved, and so on.

estimate

The statistical estimate (of the capacitance), accompanied by an uncertainty.

expression

A mathematical expression in QuickCap's main input file. This can have units of length and can be used in place of any length in the input file. An expression can be used to define a length or another value such as a dielectric constant or an angle of rotation.

external net

In a periodic structure, a net that is not in the unit cell but only exists in one of the replications of the unit cell.

floating metal

See *floating net.*

floating net

A net that can accumulate no net charge.

floating random walk

A series of hops (steps), where each hop is from the center of a volume that contains no boundaries onto a point on the surface of that volume. The point is randomly selected according to certain probability rules. This differs from a classic *fixed* random walk, where the step size is constant. For a floating random walk, the step size is dictated by the distance to the nearest object.

gain file

A QuickCap input file that describes any simultaneous signals that occur on different nets.

goal

The maximum allowable uncertainty, specified by the user. Often, this is a function of the capacitance estimate. For example, a 10 percent goal for a 1 fF capacitance estimate is 0.1 fF, whereas a 10 percent goal for a 2 fF capacitance estimate is 0.2 fF.

groundplane

An infinite plane that is part of the *ground* net. Any structures below the groundplane are essentially ignored.

header

All beginning comment lines in the main QuickCap input file, and all beginning comment lines that QuickCap outputs to a capacitance netlist.

host file

A QuickCap input file that lists host computers on which to extract capacitances in parallel.

infinitely periodic

Geometric periodicity that extends to *infinity*.

integration surface

An arbitrary surface enclosing a net; used to find a column of the capacitance matrix.

license file

The file used by QuickCap to confirm that it is licensed.

local net

In a periodic structure, a net that is in the unit cell.

log file

An output file that contains some information on the problem being extracted, and a list of start and stop times.

Manhattan

A shape with each surface on an *xy*, *xz*, or *yz* plane.

Monte Carlo method

The method of statistically solving a problem with a solution that can be represented as the average of random events.

multihost controller

In parallel operation, the version of QuickCap that coordinates the efforts of QuickCap programs on other hosts.

net

An electrode or conductor that can be represented by lumped-element circuit parameters.

node

A capacitance node on a *signal* (an RC representation of a net).

non-Manhattan

Any shape that is not Manhattan.

planar dielectric

A dielectric with only top and bottom interfaces (horizontal).

proximity point

A point halfway between objects on different nets that are nearly touching, or within the overlap if the objects are overlapping.

relative uncertainty

The statistical uncertainty of an estimate shown as a fraction of that estimate.

RMS error

The square root of the average of the error squared. For statistical extractors, this is equivalent to the standard error if no bias is present. For deterministic extractors, this is the analog of standard error.

scale of the problem

The approximate median distance between nearest nets. The scale is used by QuickCap to set up default values for QuickCap length parameters.

self-capacitance

The total capacitance of a net. In QuickCap's capacitance matrix, this is the diagonal element. This is also the same as the *Miller capacitance* when all gain terms between nets are 0.

sigma

The normalized discrepancy of a statistical estimate of zero, found when taking the difference between two independent statistical estimates of the same value. One sigma represents one standard deviation of statistical error. The average sigma in the absence of bias is 0, and the standard deviation of sigma is 1 (found by evaluating many independent sigmas).

signal

The RC representation of a net. In QuickCap, a signal consists of a collection of capacitance nodes. The resistors, if any, are represented in an associated netlist. Signals are generated by running gds2cap with the **-RC** or **-RLC** option.

signal ground

The reference net on which voltage is, by definition, 0. This is assumed to be the same as earth ground. Discrepancies might arise when there is significant capacitance to *infinity,* and signal ground is floating with respect to earth ground.

standard deviation

A value that describes the mean variation of a number of samples.

standard error

The calculation of one standard deviation of the *average* of a collection of data, based on the standard deviation of the data itself.

statistical result

Represented in QuickCap by a pair of values: an estimate, and an uncertainty that is a calculation of one standard deviation of the estimate and can be used to establish confidence intervals and confidence levels.

summary file

A QuickCap output file that contains the most recent summary of capacitance estimates.

time constant

The time (*R* times *C*) associated with a resistor *R* and a capacitor *C*. For a signal path, the time constant is the sum of the individual time constants.

total capacitance

See *self-capacitance*.

uncertainty

See *standard error*.

uncertainty goal

See *goal.*

unit cell

A cell containing the definition of one period of a periodic structure. Other cells are inferred from the unit cell based on its edges.

unweighted sum

As opposed to *weighted sum*, a straight sum.

walk rate

The number of walks per second.