

Allocating decoupling capacitors to reduce simultaneous switching noise on chips

by

Adam Granich Unikowsky

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of
Master of Engineering in Electrical Engineering and Computer Science
and

Bachelor of Science in Electrical Engineering and Computer Science
at the


MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2004

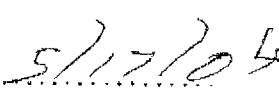
© Adam Granich Unikowsky, MMIV. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part, and to grant others the right to do so.


Signature of Author


Adam Granich Unikowsky
Department of Electrical Engineering and Computer Science
May 20, 2004

Certified by

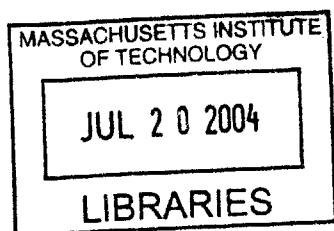

Paul Stabler
VI-A Company Thesis Supervisor

Certified by ..


Christopher J. Terman
Senior Lecturer, Electrical Engineering and Computer Science
M.I.T. Thesis Supervisor

Accepted by


Arthur C. Smith
Chairman, Department Committee on Graduate Theses



BARKER



Room 14-0551
77 Massachusetts Avenue
Cambridge, MA 02139
Ph: 617.253.2800
Email: docs@mit.edu
<http://libraries.mit.edu/docs>

DISCLAIMER OF QUALITY

Due to the condition of the original material, there are unavoidable flaws in this reproduction. We have made every effort possible to provide you with the best copy available. If you are dissatisfied with this product and find it unusable, please contact Document Services as soon as possible.

Thank you.

Some pages in the original document contain text that runs off the edge of the page.

Using decoupling capacitors to combat simultaneous switching noise in circuits

by

Adam Granich Unikowsky

Submitted to the Department of Electrical Engineering and Computer Science

on May 20, 2004, in partial fulfillment of the

requirements for the degrees of

Master of Engineering in Electrical Engineering and Computer Science

and

Bachelor of Science in Electrical Engineering and Computer Science

Abstract

$L \frac{dI}{dt}$ noise due to simultaneous switching of circuits on chips is a growing problem in VLSI design. This kind of noise can lead to timing errors and significant circuit slowdowns, if not kept within reasonable bounds. The most common way of reducing this form of noise is the addition of decoupling capacitance on chip. However, adding decoupling capacitance can take significant area and can make routing very difficult. The goals of this thesis are twofold: first, to characterize the relationship between noise propagation on chip and parameters such as on-chip resistance and capacitance; and second, to develop an algorithm which will minimize the number of decoupling capacitors on chip while simultaneously reducing the noise to within acceptable boundaries.

Thesis Supervisor: Christopher J. Terman

Title: Senior Lecturer, Department of Electrical Engineering and Computer Science

Acknowledgments

I would first like to thank IBM Microelectronics as a whole for their support of my thesis. They provided extremely generous financial support for all three years I worked at IBM, and provided me with easy access to any resource I ever needed. Numerous engineers took their time to answer my questions and provide with guidance about my summer internships and my thesis. Working at IBM was a truly outstanding experience and I am truly appreciative.

I would like to specifically thank two people at IBM. First, this thesis would not have been possible without Tim Budell, my thesis supervisor at IBM. He spent countless hours teaching me, guiding me, reading my often impenetrable drafts, and putting up with my frequent silly questions. Without his support, this thesis would have been a pipe dream; with his support, it became a reality. I am indebted to him for his advice and his guidance.

I would also like to thank Paul Stabler. Paul was my manager at IBM for all three summers, during which I had the opportunity to learn a tremendous amount about engineering and feel truly integrated into the IBM community. He provided invaluable support in choosing a thesis topic and a supervisor, and was a watchful eye throughout the thesis process. He always welcomed my questions and constantly gave me advice and guidance.

I was extremely privileged to have Professor Christopher Terman as both my undergraduate advisor and my thesis advisor at MIT. For the past four years, Professor Terman has been a constant source of support as I navigated the waters of Course VI. As an undergraduate, his office doors were always open to me for any advice I ever wanted. As a graduate student, he provided with invaluable advice and direction as I struggled through the thesis process. I cannot thank him enough for his support.

Finally, I would like to thank my mother, father, and sister, for their unending love and support. They have been a rock of stability in my life and have helped me throughout my MIT career in ways unimaginable. It is to them that I am proud to dedicate this thesis.

Contents

1	Introduction	8
2	Methodology	10
2.1	The power grid	10
2.2	Current source methodology	12
3	Mathematical discussion	20
3.1	Effect of increasing current	22
3.2	Effect of decap on magnitude of peak noise	23
3.3	Effect of decap on time of peak noise	23
3.3.1	Application to determination of peak noise	24
3.4	Effect of pulse width	28
4	Noise propagation trends	35
4.1	Background capacitance and resistance	35
4.2	Noise modeling on chip	37
4.2.1	Superposition	37
4.2.2	Noise propagation	38
4.2.3	Effect of decap on noise propagation	41
4.2.4	Distribution of capacitance among nodes	45

5	Noise approximation	49
5.1	Problem description	49
5.2	The naive algorithm	53
5.3	The peak time approximation	58
5.4	Implementing the peak time approximation: a first pass	61
5.5	Calculating the local noise	65
5.6	Calculating the non-local noise	69
5.6.1	Measurement point at (2,0) coordinate	70
5.6.2	Measurement point at (2,1) coordinate	72
5.6.3	General methodology	74
5.7	Relaxing assumptions	75
5.8	Precharacterizing the entire curve	76
5.9	Edge effects	82
6	Decap allocation algorithm	85
6.1	High-level picture	85
6.2	Justification for algorithm	87
6.3	Description of algorithm	89
7	Results	92
7.1	Two by two configuration	93
7.2	Three by three configuration	95
7.3	3 by 3 grid with different decap maxima	97
7.4	3 by 3 grid with no middle source and different noise tolerances	98
7.5	Full chip	100
8	Conclusions and future work	106

List of Figures

2.1	Grid in one dimension	11
2.2	Grid in one dimension with current sources	12
2.3	Overhead view of densely packed area in two dimension	13
2.4	Smaller power grid	15
2.5	Comparison of noise on distributed RC circuit, as compared to smaller power grid	16
2.6	Current signature as compared to current with transistor model	18
2.7	Noise with current signature as compared to noise with transistor model	19
3.1	Simple power grid model	21
3.2	Range of maximal noise	25
3.3	Effects of decap on noise	26
3.4	Constant charge, changing peak current	30
3.5	Constant peak current, changing charge	31
3.6	Constant peak current and charge comparison	32
3.7	Linearly increasing noise	33
4.1	Superposition of several curves	39
4.2	Inverse square root of voltage as a function of distance from current source	40
4.3	Hypothetical current source setup	42
4.4	Placing decaps in various locations	43

4.5	Noise propagation model	44
4.6	Five situations	46
4.7	Simulated noise in five situations	48
5.1	Scenario with three current sources and decap at each node	50
5.2	Scenario with one current source and decap at multiple node	50
5.3	Actual birds eye view of section of chip	51
5.4	Idealized birds eye view of section of chip	52
5.5	Representative example of peak time approximation	60
5.6	New problem statement	61
5.7	Measurement point at (2,0) coordinate	70
5.8	Measurement point at (2,1) coordinate	73
5.9	Approximate curve given fixed point	78
5.10	0, 100, and 200 decaps with identical peak times and voltages	79
7.1	Two by two current source configuration	94
7.2	Three by three current source configuration	96
7.3	Three by three current source configuration with no middle node	99
7.4	Current sources on a small chip	102
7.5	Histogram of noise values before adding decap	103
7.6	Histogram of noise values after adding decap	104
7.7	Histogram of decap values in eventual solution	105

Chapter 1

Introduction

The problem of noise on chip power rails has been cited in the literature for many years. However, as chips grow faster and as supply voltages grow lower, noise is becoming an increasingly troubling problem for chip designers. On-chip noise has been known to cause logical errors and severe timing slowdowns.

On-chip noise arises from parasitics inherent in the pins connecting chips to packages. When CMOS devices remain in a static state, power consumption is minimal. One of the transistors will always be in cutoff, allowing only minimal current to pass through the device. However, when the transistors switch, both transistors are in saturation for a certain period of time. Accordingly, a large amount of current must pass between the power and ground rails of the chip. Normally, this does not pose a problem; the power rails are connected to an off-chip power supply, which can supply current to the chip. However, there is a small amount of inductance in the pins connecting the chip to the package. These inductors resist the large changes in voltage by creating a reverse EMF by the equation $V = -L \frac{dI}{dt}$. Accordingly, there can be significant voltage droop on chip when the chip demands current. This effect is especially noticeable when large numbers of circuits switch on chip at the same time. In fact, on modern chips, this inductance is often so significant that virtually all of the charge used to supply this current comes from on-chip capacitance. As charge leaks off these capacitors, the potential across them tend to decrease, causing substantial noise.

There are numerous strategies that can be used to confront this problem. The first strategy is to

change on-chip parameters in order to reduce noise. Increasing on-chip stored capacitance, decreasing on-chip resistance, and decreasing pin inductance can all lead to decreased noise. It is important to develop an understanding of the precise relationship between on-chip noise propagation on these parameters beyond their general trends. Part of this thesis will consist of theoretical explanations of trends observed in simulation with different circuit parameters.

However, it is often expensive or impossible to change on-chip circuit parameters. Accordingly, on-chip decoupling capacitance must be used. Since most of the charge used to supply current to the current sources comes from stored on-chip capacitance, storing charge on decoupling capacitors can substantially reduce the noise. However, area constraints make it difficult to place these capacitors, especially in the proximity of large macros such as registers. Adding too many capacitors can lead to significant layout difficulty. In general, different nodes on chip might have different noise tolerances, and might be able to fit different amounts of decap. The goal is to find a placement of capacitors on chip that minimizes the total amount of capacitance, while simultaneously ensuring that all nodes are within reasonable noise limits, and no node is populated with decaps above its maximum capacity.

This thesis will be in five parts. First, I will discuss the basic methodology I used to conduct all these simulations, and justify the models I used in simulation. Second, I will present a model for on-chip noise, and explain the interesting features of a mathematical analysis of this model. Third, I will describe the trends observed in simulation using different on-chip parameters. Fourth, I will describe two different methods that can be used to estimate the amount of noise at any node on chip. Fifth, I will present the optimization algorithm I designed to place capacitors on chip, and analyze the results as compared to SPICE simulation.

Chapter 2

Methodology

2.1 The power grid

The primary tool used to generate simulation results for this thesis was an RC network based on a 225 micron grid. Each node of the grid represented a location where a current source could be connected; nodes were located 225 microns apart. There have been several attempts in the literature to construct such models; these frequently involve distributed RLC networks, as can be seen in [2], [13], and [19]. However, a simple RC model was found to be effective at IBM Microelectronics. I performed no tests to compare it to the chip distribution network of an actual chip; I exclusively used the model with the given parameters. A value of $750 \frac{pF}{mm^2}$ was used for the background capacitance, and a value of 0.22Ω was used for each resistor. These values are believed to be typical of IBM's Cu-11 technology.

C4 package pins were placed near every node on chip. These pins consisted of an inductor of $2 nH$ and a resistor of 0.125Ω , and were connected to a $1.8 V$ power supply. In reality, C4's are more sparsely located on chip, but this was done to ensure that every node was symmetrical. The reason I could do this was that the overwhelming majority of the charge provided to switching transistors actually comes from on-chip capacitance, rather than from the package. (The package eventually recharges the on-chip capacitance, of course, but only after the noise peaks have already occurred.) This assumption contradicts methods such

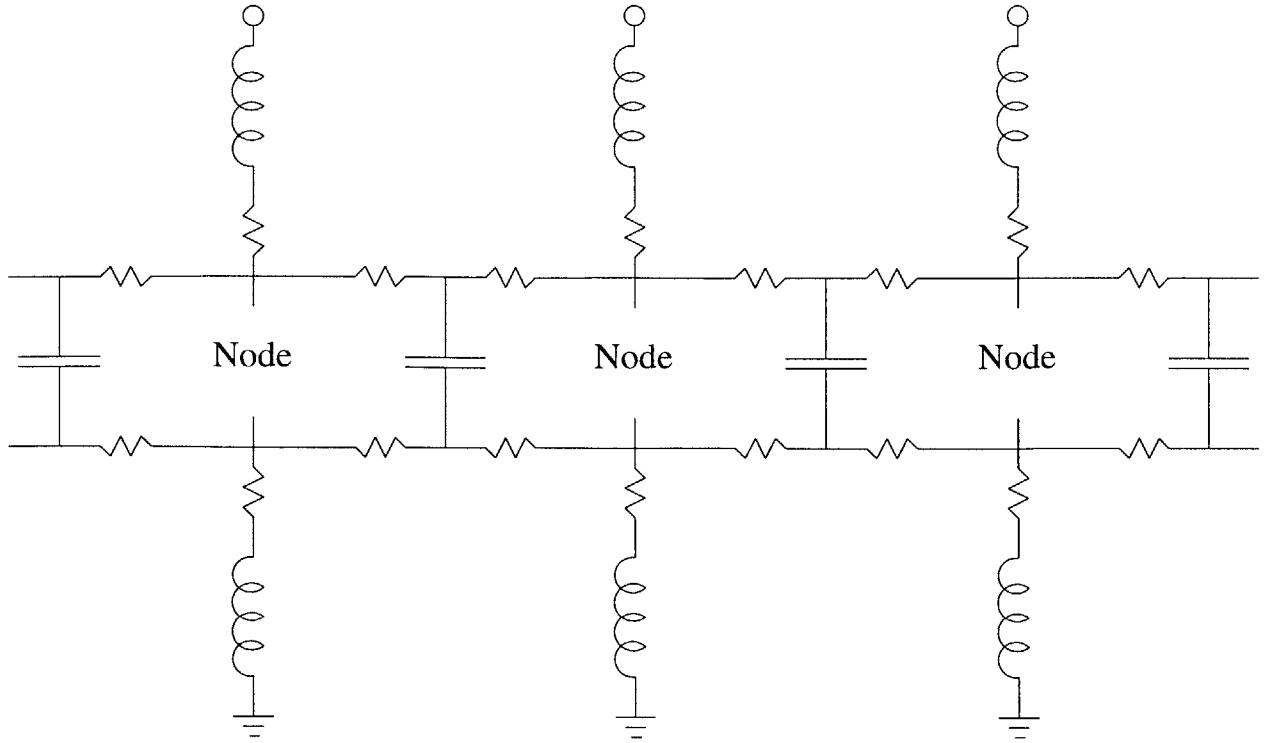


Figure 2.1: Grid in one dimension

as that seen in [19], which assume that all of the charge comes from the package, and which calculate the noise on the basis of the number of IR drops between the pin and the current source node. However, in simulation, it did appear that with the large values for pin inductance, the placement of the pins made little difference in altering the peak noise.

As mentioned above, the grid was arranged so that there were certain nodes where current sources could be placed, with an identical amount of resistance and capacitance between any two nodes. The grid was implemented as shown in Figure 2.1; the same grid with current sources can be seen in Figure 2.2. A two-dimensional overhead view of the grid, with the package pins not shown, can be seen in Figure 2.3.

At every resistor intersection in this figure in which there is no current source, a capacitor is present. The area in Figure 2.3 is densely packed, meaning current sources are present at every available node. By placing current sources in this manner, we can simulate the effects of large numbers of transistors firing at the same time.

A 30-by-30 node chip, measuring 2.7 mm by 2.7 mm , was used in simulation. This would correspond in

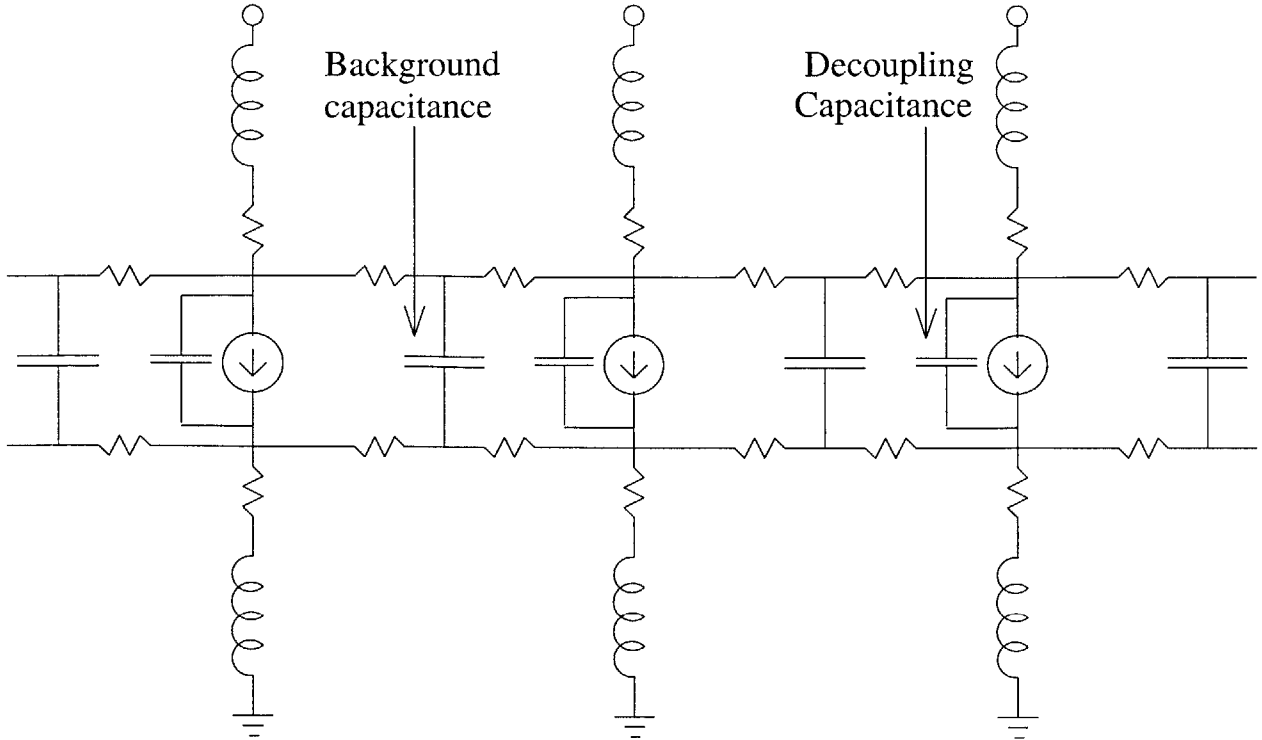


Figure 2.2: Grid in one dimension with current sources

reality to a chip that is small, but still on the same order as chips actually being manufactured. (The size of the chip was limited in order to save on simulation time; in reality, the size of the chip matters little when simulations are being done in the middle of the chip).

2.2 Current source methodology

When a transistor is in its saturation state, the voltage between the drain and the source has little effect on the current between drain and source. Accordingly, the transistor can, in principle, be modeled as a current source. Simulating transistor-level models of very large macros can be very resource intensive. Therefore, modeling these models as triangular current spikes is very desirable, and indeed the use of triangular current spikes to substitute for transistor level models was one way that simulations were done at IBM. Current source models for complicated macros are cited extensively in the literature. Generally, these take the form of complicated piecewise linear current sources, and there are numerous papers such as [17] which explain

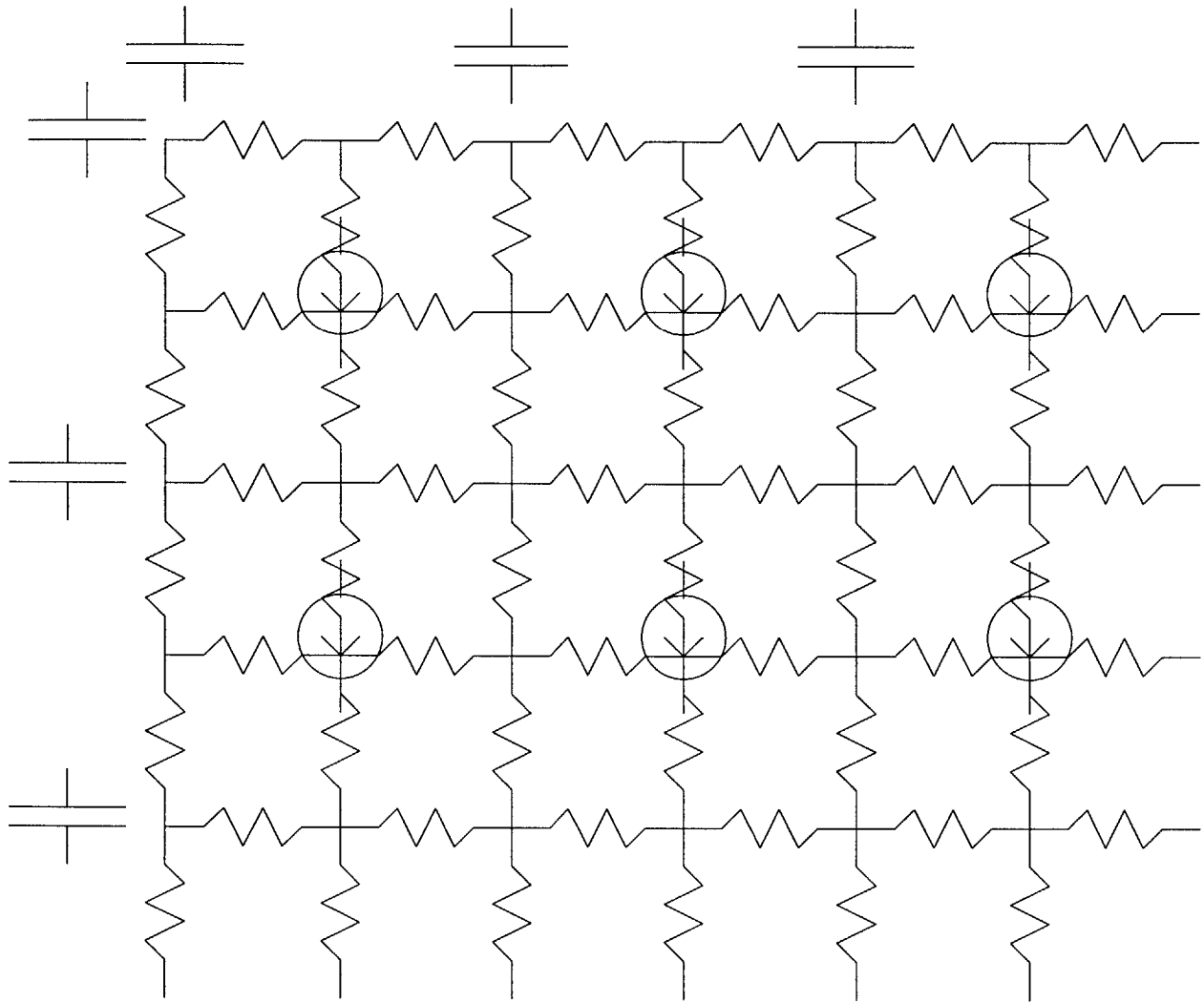


Figure 2.3: Overhead view of densely packed area in two dimension

methods of obtaining worst-case current signatures from particular macros. However, for more generic models, the use of triangular current source pulses is also cited extensively [2] [13] [3]. Now it is not entirely clear that using a current source can capture the characteristics of a transistor-level model; for instance, there are multiple papers such as [1] [6] which describe a negative feedback effect between noise and current. In particular, as more inverters attempt to draw current, the voltage compresses, causing each inverter to consume less current; therefore, a current waveform might conceivably not be able to be reused on power grids with different levels of noise. To resolve this, I performed a series of simulations, using a transistor-level model of a register array, to determine whether or not such models were appropriate.

The steps in running these simulations were as follows.

1. It was necessary to create a small power grid to mirror the properties of the larger power grid. Simulating transistor-level macros on large distributed RC -meshes took intractable amounts of time (on the order of hundreds of hours). Therefore, I took an old current signature from a macro, tested it on a distributed RC mesh, and saved its noise plot. Then, I simulated the current signature on a smaller grid, altering the values of the parasitics on the smaller grid until the two noise plots resembled one another. The parasitics of the smaller grid were deliberately kept very low - roughly 2% of the supply voltage.
2. I simulated a transistor-level register array on the smaller power grid. This resulted in the creation of a current signature. I extracted and saved this current signature for future tests.
3. I altered the parasitic values in the smaller power grid so that the parasitics would achieve a level of about 10% of the supply voltage. This was done just by repeatedly simulating the transistor-level register array in the smaller noise grid, and altering the parasitics by hand until the noise rose to about 10% of the supply voltage.
4. Using the new, noisier power grid, I could test the current signature derived in step 2. But first, it was necessary to add some capacitance to the new power grid, to compensate for the parasitic capacitance inherent in the register array. The value used for the capacitance, 12.7 pF, was taken directly from the

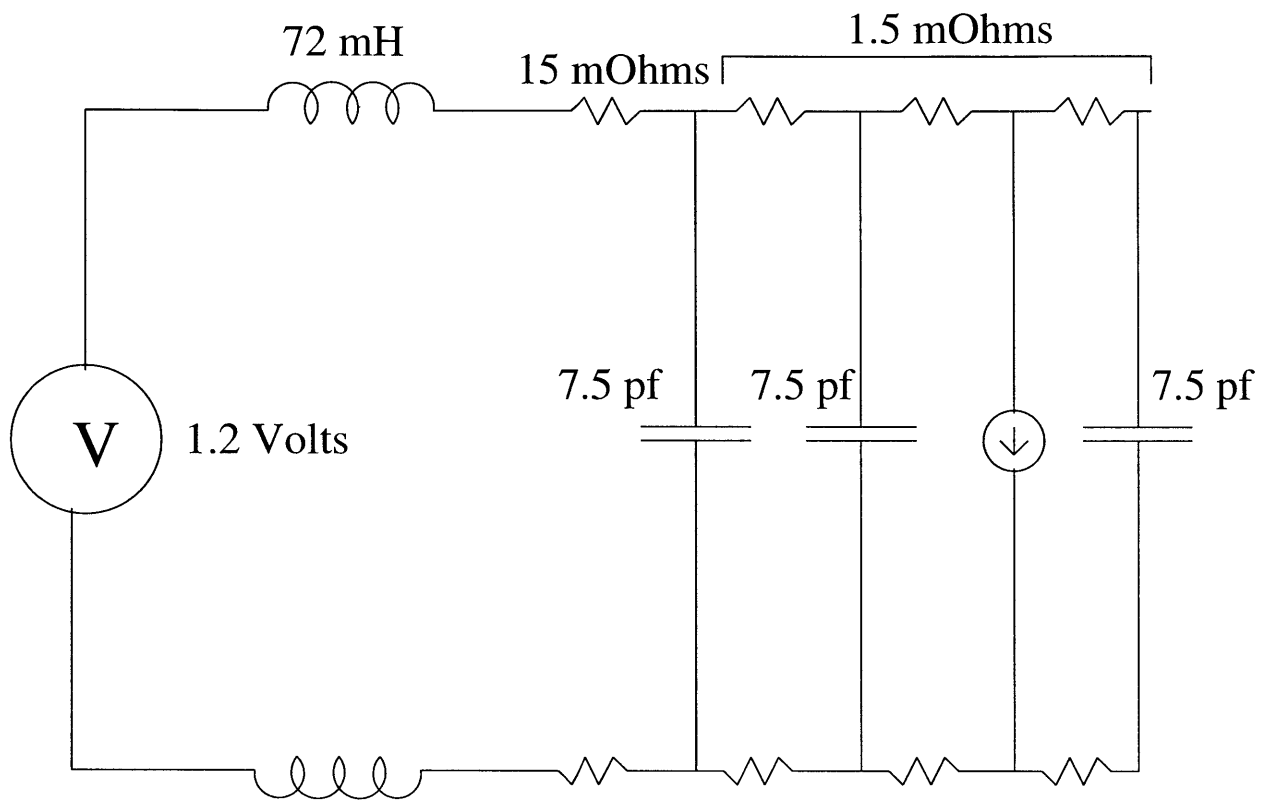


Figure 2.4: Smaller power grid

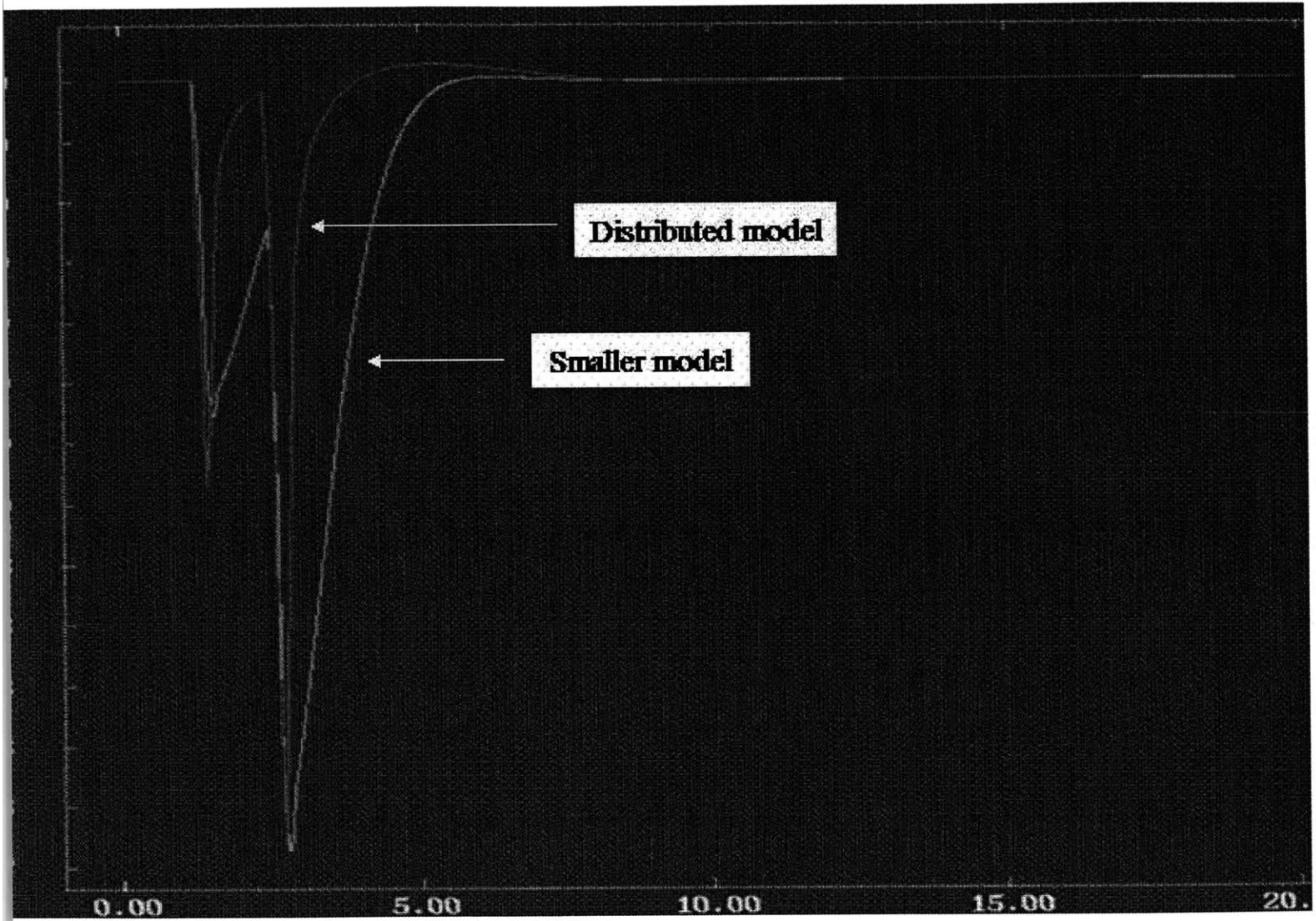


Figure 2.5: Comparison of noise on distributed RC circuit, as compared to smaller power grid

register array specification.

5. I simulated the current signature on the noisier power grid, and compared the noise to the plots taken in step 3.

The results are shown in Figure 2.6 and Figure 2.7. While there were slightly higher currents using the current signature, and therefore slightly higher noise voltages, the noise values and current values were certainly comparable. Accordingly, performing simulations using current signatures rather than a transistor-level model seemed to be a reasonable way of simulating the chip.

Needless to say, the current signatures acquired in the previous step were very complicated, piecewise linear functions. I found in simulation that merely using different data inputs, or even reading different addresses in a register array, could lead to different current signatures. This fact is also noted in the literature as a difficulty of obtaining worst-case current signatures [17]. Thus, the critical question was, which current signatures would be used in simulation?

To solve this problem, engineers at IBM typically used triangular current spikes (a model that is used in numerous places in the literature, as cited above). The width of the pulse would be equivalent to the width of the current signature obtained in simulation; the height of the pulse would be chosen so that the total area under the pulse (i.e. the total amount of charge passing through the transistor) would be equivalent to the total area under the current signature curve. I performed several simulations to test whether this was a valid assumption - that is, whether or not a current signature produced a similar noise peak to its triangular equivalent. I found that the answer depended on particular characteristics of the power grid. I will discuss this question in more detail after I present a mathematical analysis of some of the properties of the current sources on chip.

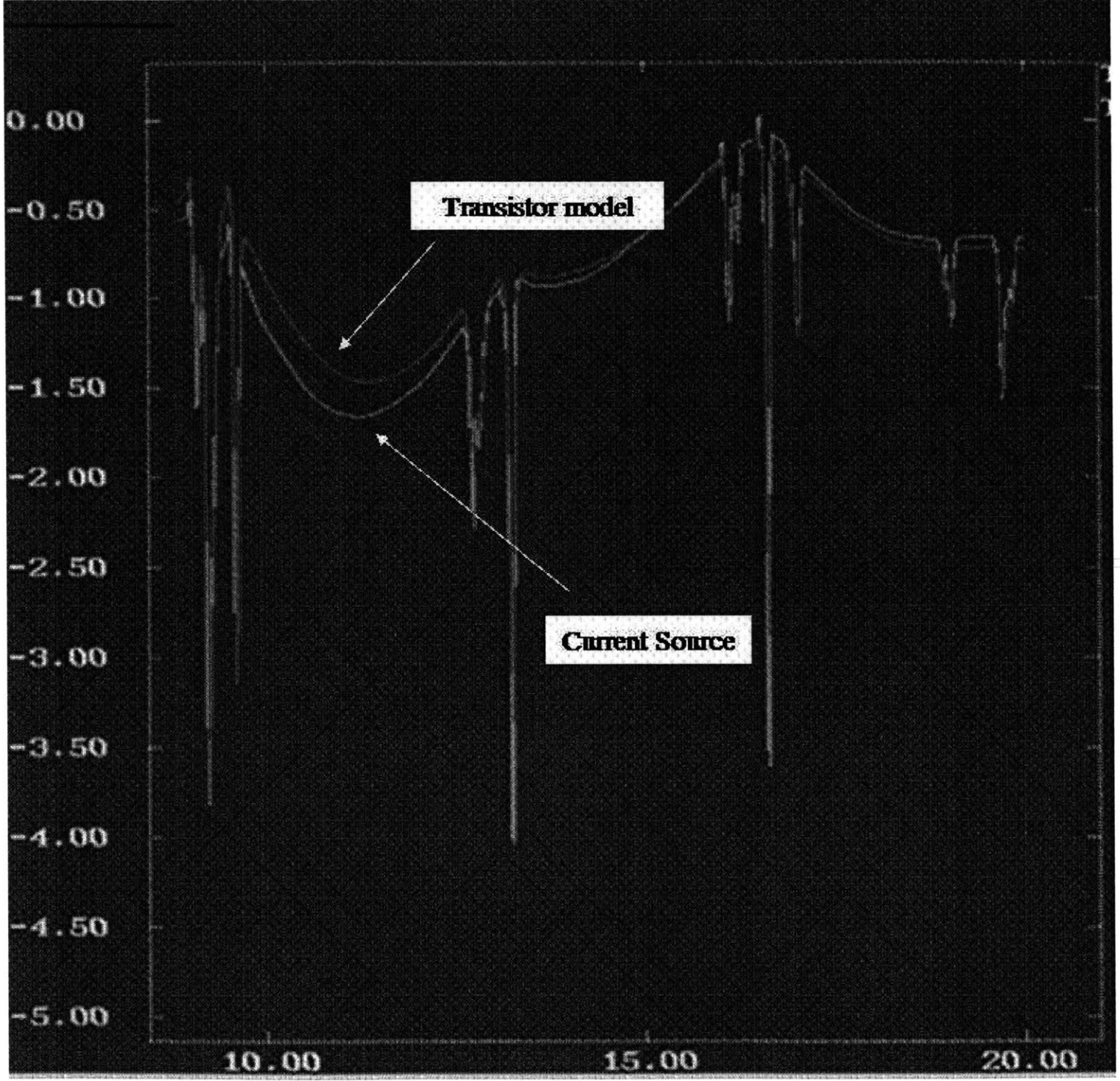


Figure 2.6: Current signature as compared to current with transistor model

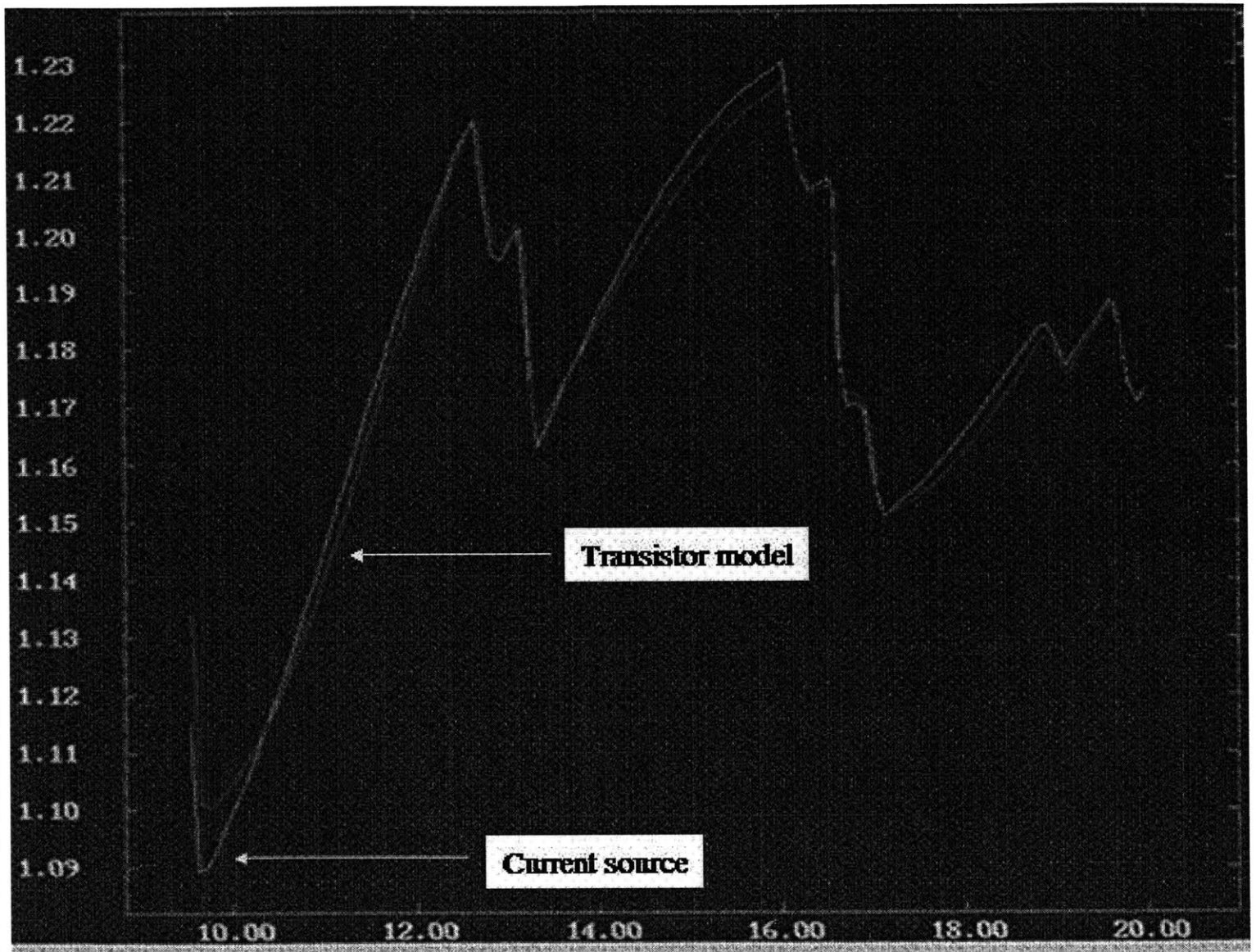


Figure 2.7: Noise with current signature as compared to noise with transistor model

Chapter 3

Mathematical discussion

There are several attempts in the literature [11] [14] [8] [16] to find a closed-form expression for maximum noise due to a particular transistor firing, involving the voltage-current relations for CMOS circuits. Here, I present instead an analysis of a circuit involving an actual current source. I use a current source in this analysis instead of a transistor model for several reasons:

- All of my simulations were done with current sources, so doing a mathematical analysis with current sources is more illuminating in explaining the results.
- The current source analysis leads to a simpler closed-form result, while still explaining many of the observations seen in simulation.
- We are not trying to simulate the effect of a single transistor, but rather thousands of transistors firing in an irregular pattern, so using a transistor might not be illuminating.

Therefore, consider the circuit in Figure 3.1. We can consider this kind of circuit a rough approximation of what happens on chip when a resistive-capacitive mesh must provide a certain amount of current to a macro. The resistor and left capacitor represent the on-chip resistance and capacitance in the power grid. The capacitor on the right represents decoupling capacitance. There is no resistance between the decap and

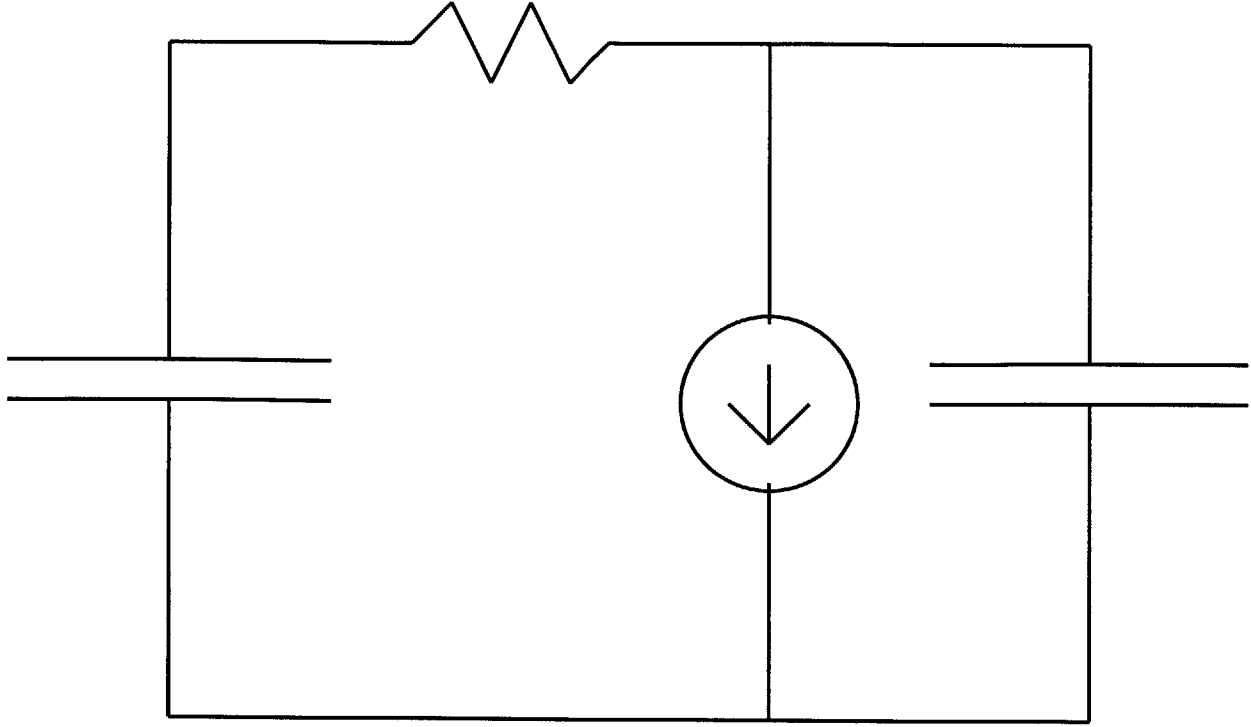


Figure 3.1: Simple power grid model

the current source because the resistance between the added decap and the macro is smaller than the on-chip resistance. (The simulations performed also used this model, for simplicity.)

Let I_1 be the current through the resistor, i be the current through the current source, and V_{cs} be the voltage across the current source. Then $-\frac{Q_{back}}{C_{back}} - I_1 R - \frac{Q_{decap}}{C_{decap}} = 0$. Differentiating with respect to time and putting $I_{decap} = I_1 - i$, we get

$$\frac{I_1}{C_{back}} + R \frac{dI_1}{dt} + \frac{I_1}{C_{decap}} = \frac{i}{C_{decap}} \quad (3.1)$$

We can also write an equation for the voltage across the current source, which is identical to the voltage across the decoupling capacitor:

$$(I_1 + i) = C_{decap} \frac{dV_{cs}}{dt} \quad (3.2)$$

Assuming we can put $i = k_1 t + k_2$ (the rising half of a triangular current source), we can solve these differential equations for V_{cs} , yielding

$$V_{cs} = -I_0 R \frac{C_{back}}{C_{back} + C_{decap}} e^{\frac{-t}{RC_1 C_2}} - \frac{k_1}{C_{back} + C_{decap}} \frac{t^2}{2} - \frac{k_2}{C_{back} + C_{decap}} t - \frac{k_1 C_{back}^2 R}{(C_{back} + C_{decap})^2} + K \quad (3.3)$$

where I_0 is the initial current that would have been seen without any current source, and K is a constant that allows you to set the initial voltage to any value.

This equation is not exact; the power grid is a distributed network of resistors and capacitors, along with inductive pins connecting the grid to power sources. Therefore, we should not expect this equation to produce an exact solution, but rather a general intuition on how various variables affect the voltage across the current source.

There are several important lessons we can take from this equation.

3.1 Effect of increasing current

First, as the current increases, the voltage across the current source decreases. This can be understood intuitively. There are two different effects that would cause the voltage to decrease:

1. The current across the resistor must be increasing as a function of time. Since the current is proportional to the voltage, we would also expect the voltage across the resistor to increase as a function of time, serving to depress the voltage across the current source with respect to ground.
2. Because the capacitors are releasing charge, their voltage with respect to ground is decreasing, and so the voltage across the current source will decrease as well.

3.2 Effect of decap on magnitude of peak noise

Second, the addition of decoupling capacitance decreases the extent to which the voltage across the current source will decrease. This can be seen in the equation, as C_{decap} appears in the denominator of every single term; as C_{decap} increases, these negative terms become smaller, and therefore there is a lower noise voltage. It can also be understood intuitively. With more capacitance in the system, providing an equivalent amount of charge will result in less of a voltage collapse, due to the equation $Q = CV$. Establishing a more systematic relationship between the voltage droop and the decoupling capacitance is more difficult; I will discuss this after the next section provides the necessary background.

3.3 Effect of decap on time of peak noise

Third, the above equation provides useful knowledge on the time of the voltage peaks. Clearly, as the current rises, the noise voltage will increase. However, eventually the current will reach its peak, and then decrease again. The question is, after the current reaches its peak and begins decreasing, will the noise voltage increase, or decrease? The two effects listed above will now have opposite effects on the noise voltage: Effect 1 will cause the noise voltage to decrease (because the current is decreasing), whereas effect 2 will cause it to increase (because the current is still positive, so charge is leaking off the capacitors). We must therefore examine the equation in more detail. To do this, we find the rate of change of V_{cs} with respect to time. For simplicity, we can assume that t is relatively large, so we can ignore the exponential term:

$$\frac{dV_{cs}}{dt} = \frac{-1}{C_{back} + C_{decap}} \left(k_1 t + k_2 + \frac{k_1 C_{back}^2 R}{C_{back} + C_{decap}} \right)$$

If we set $\frac{dV_{cs}}{dt}$ to 0, and assuming k_1 is negative and k_2 is positive (that is, a positive but linearly decreasing current), we get

$$t = \frac{k_2}{k_1} - \frac{RC_{back}^2}{C_{back} + C_{decap}}$$

There are two important things to notice about this equation. First, as C_{decap} increases, the time at which the maximum noise voltage occurs increases. As C_{decap} goes to infinity, the time at which the maximum voltage occurs goes to $\frac{k_2}{k_1}$, which is precisely as we'd expect, since that is the time for which the current is 0 in the equation $I = -k_1 t + k_2$.

In practice, without any decap, when the current starts to decrease, the noise voltage starts to decrease almost immediately. This occurs for two reasons:

1. The package can begin recharging the background capacitance, since the current is decreasing.
2. Other background capacitance further away starts providing charge to recharge the closest background capacitance.

In the context of the above equation, when C_{decap} is low, the $\frac{RC_{back}^2}{C_{back} + C_{decap}}$ term dominates, so in the above equation, t approaches 0 - that is, when the current starts to decrease, the noise voltage decreases as well. However, when decoupling capacitance is added, the $\frac{RC_{back}^2}{C_{back} + C_{decap}}$ term decreases. Thus, t rises, and with large amounts of decap, t does approach $\frac{k_2}{k_1}$, or the end of the current spike. This effect will have importance significance, as well be discussed later on.

It is important to explicitly spell out one consequence of this observation. At a node containing a current source, the earliest possible time that the peak voltage can occur is the apex of the current source, and the latest possible time is at the end of the current source triangle. Clearly, increasing the amount of current will always cause the voltage droop to increase. Meanwhile, as time elapses with zero current, the capacitors are going to be recharged, so the voltage droop will decrease. Therefore, the peak voltage will always occur in this range of Figure 3.2. The effect of adding decap in simulation, in terms of both decreasing the peak noise and making the peak noise occur later, is shown in Figure 3.3.

3.3.1 Application to determination of peak noise

We can now discuss establishing a quantitative relationship between the peak noise and the decoupling capacitance. Given the equation $Q = CV$, we would want, in principle, to establish a inverse relationship

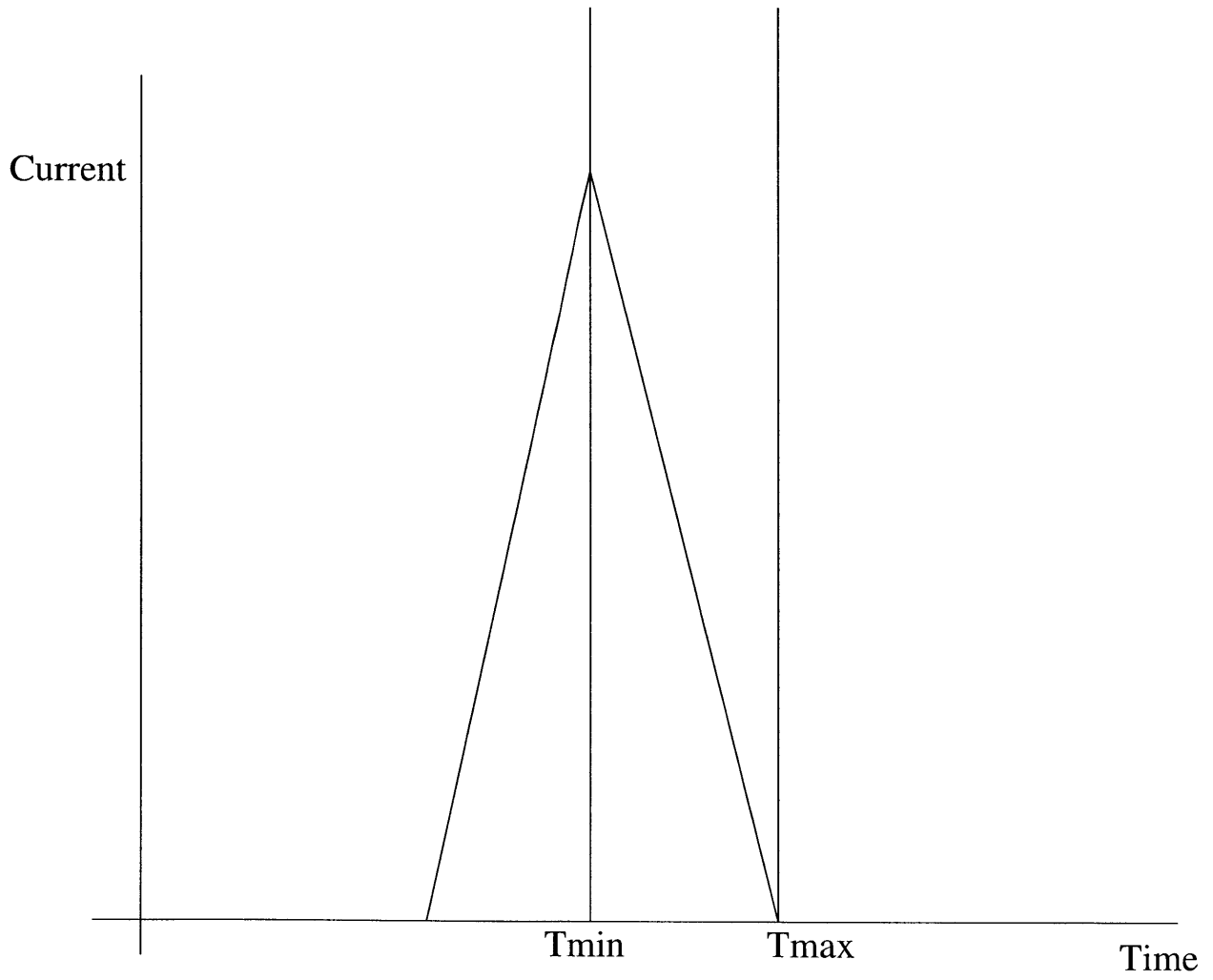


Figure 3.2: Range of maximal noise

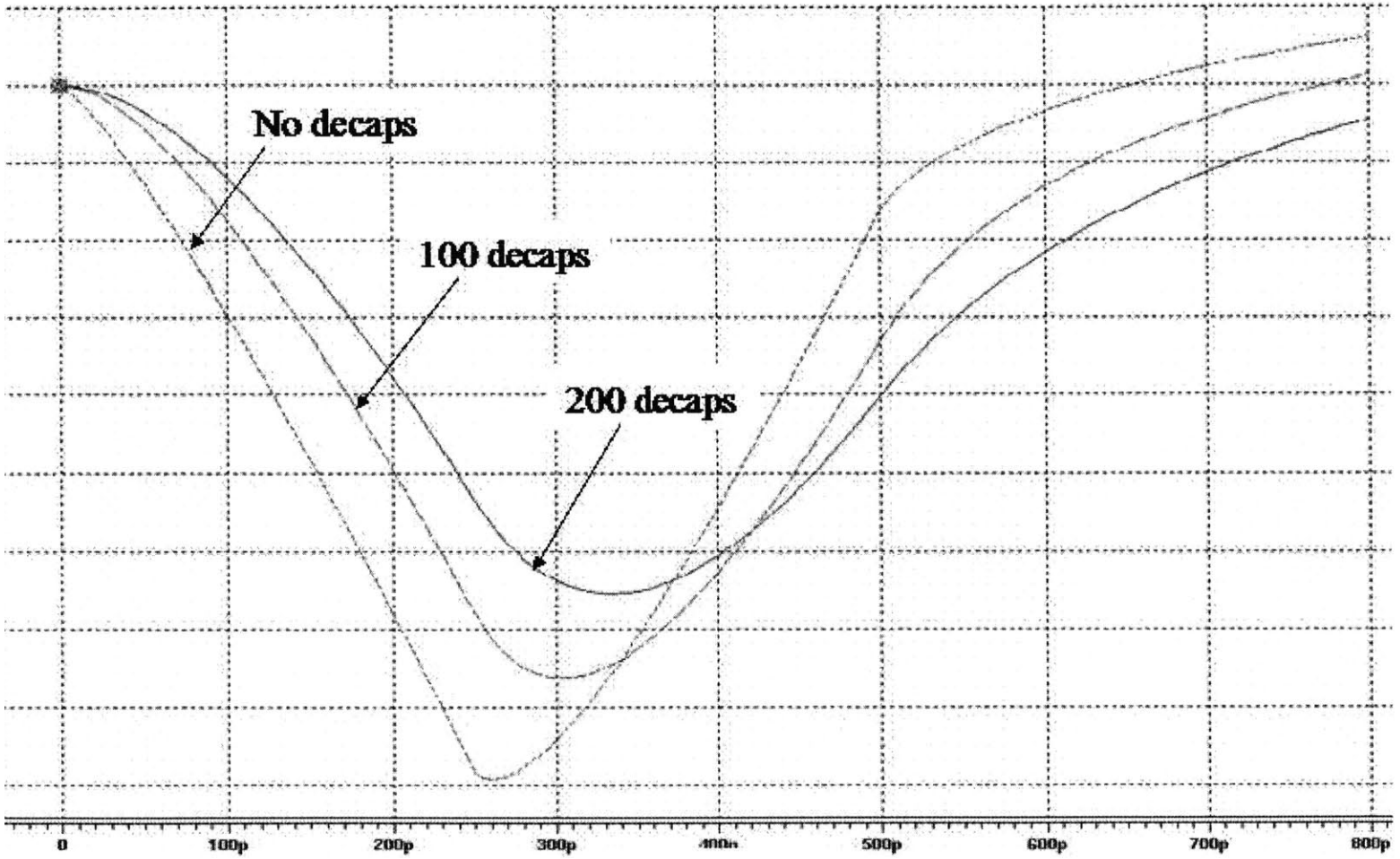


Figure 3.3: Effects of decap on noise

between the voltage droop and the capacitance. This equation should hypothetically be of the form $V = \frac{Q}{C_{decap} + C_{background}}$; this can be rewritten as $V = \frac{1}{mC_{decap} + b}$. However, according to the above equation, at any specific time t during the ascending part of the current triangle, the voltage will not follow such a relationship. This is because of the presence of the $\frac{k_1 C_{back}^2 R}{(C_{back} + C_{decap})^2} t$ term, which is an inverse-squared function with respect to the decoupling capacitance. After the triangle reaches its apex and the current begins decreasing, the sign of k_1 may change, but the format of the equation will remain the same.

If there is very little decoupling capacitance, as discussed above, the peak voltage occurs at approximately the same time as the peak current. If we try to plot the inverse of the noise voltage as a function of capacitance at this time, the $\frac{k_1 C_{back}^2 R}{(C_{back} + C_{decap})^2} t$ term will cause this graph to be superlinear. This means that an inverse function of the form $V_{time-of-peak-current} = \frac{1}{mC_{decap} + b}$ will actually overpredict the amount of noise at high values of the decoupling capacitance.

After extensive simulation, I observed that if we are to put the peak voltage, rather than the voltage at a specific time, then the equation $V_{peak-voltage} = \frac{1}{mC_{decap} + b}$ would hold correctly. Recall that as we add more decoupling capacitance, the time at which the peak voltage occurs increases. Therefore, at low values of decap, $V_{peak-voltage}$ is about equal to $V_{time-of-peak-current}$. As the amount of decap increases, the time of the peak current increases, and therefore $V_{peak-voltage}$ becomes increasingly greater than $V_{time-of-peak-current}$. Therefore, since $V_{time-of-peak-current} = \frac{1}{mC_{decap} + b}$ overestimates the amount of noise at large amounts of decap, inserting $V_{peak-voltage} = \frac{1}{mC_{decap} + b}$ would probably be a closer fit than $V_{time-of-peak-current} = \frac{1}{mC_{decap} + b}$.

As it turns out, this produces a nearly perfect fit. So far as I can tell, there is no principled reason for this, other than the heuristic argument above. It just turns out to be an extremely useful approximation, since peak voltages are, very frequently, precisely what we are interested in calculating.

3.4 Effect of pulse width

Fourth, this equation can lend some insight on how noise scales with respect to the width of the pulse. Imagine we were to move the identical amount of charge, but in a shorter window of time; say, instead of a triangle of width 500 ps and height 300 mA, we had a triangle of width 250 ps and height 600 mA. We know that the same amount of charge is being moved, so we would expect the voltage across the background capacitor to be unchanged. However, the charge is moving more quickly, leading to a larger current, so one might expect the voltage across the resistor (and accordingly the voltage across the current source) to compress. Now, imagine we were to have a current spike with the identical peak current, but with a wider base, and hence more charge moved; say, instead of a triangle of width 500 ps and height 300 mA, we had a triangle of width 800 ps and height 300 mA. We know that the peak current is the same, so we will need to maintain the same potential across the resistor. However, we are moving more charge, so the potential across the capacitor will go down. Accordingly, we would expect the voltage across the current source to compress as well. It seems that there are two distinct effects - one related to the peak current, and one related to the total charge.

This intuition is borne out by the equation. Assuming that $k_2 = 0$ and the exponential component is small, there are two terms which are governed by the slope of the function. The first is the $\frac{k_1}{C_{back} + C_{decap}} \frac{t^2}{2}$ term. This term is only affected if the amount of charge under the curve changes; multiplying the peak current by N and dividing the peak time by N cause the slope to increase as N^2 so the increased slope and the decreased time will cancel out. The second is the $\frac{k_1 C_{back}^2 R}{(C_{back} + C_{decap})^2} t$ term. In this term, since the t is not squared, multiplying the peak current by N and dividing the time by N will cause the peak time to increase by a factor of N . Therefore, even if you have the same amount of charge being passed through the chip, the voltage droop at the current source will increase. However, the $\frac{k_1 C_{back}^2 R}{(C_{back} + C_{decap})^2} t$ term is constant with respect to the peak current. If one were to divide the rise time and fall time of the spike by a factor of N , but keep the peak current the same, then k_1 would increase by a factor of N , but the time would decrease by a factor of N . Therefore, unlike the other term which stayed constant for the same amount of charge, this term stays constant for the same peak current.

In a real-world chip, there is a second reason the voltage droop at the current source increases as you make the current pulse narrower. The charge has to be provided in a given amount of time. RC time constants delay the amount of time in which capacitors far away on the chip can provide charge. Therefore, as the pulse width decreases, the effective capacitance that can provide charge to the current source decreases as well, and so the local voltage droop increases.

However, back to the original equation; we have two factors, one of which changes if the amount of charge changes and is left unchanged by changes in the peak current; and the other of which changes if the peak current changes and is left unchanged by changes in the amount of charge. We can call these terms the “charge term” and the “current term”. The question is, which of these terms dominates? This answer is extremely significant, since we need to know how to transform an arbitrary current signature into a triangular current pulse.

To test this, I first performed a series of simulations in which the charge stayed constant, but the peak current changed (Figure 3.4) and then a series of simulations in which the peak current stayed constant, but the charge changed (Figure 3.5). I then compared how the noise changed under either circumstance. According to simulation, it generally appears that the current had a far bigger effect on the noise than the charge under the curve. Put differently, the $\frac{k_1 C_{back}^2 R}{(C_{back} + C_{decap})^2} t$ term dominates, as shown in Figure 3.6. Thus, keeping the current source constant, but altering the width of the pulse, has a relatively small impact on the noise on the chip.

Incidentally, using this analysis, if we were to keep the time constant, and increase the peak current by a factor of N , as shown in Figure 3.7, the peak voltage would also increase by a factor of N . This is because every term in the equation would scale by a factor of N . This is also observed in SPICE simulations.

This knowledge is significant because it calls into question the fundamental assumption of this research - that we can approximate any current source by a triangular pulse. If the only relevant value is the area under the curve, then any arbitrary piecewise linear current source can be transformed into an equivalent triangular pulse. The same is true if the only relevant value is the peak current. However, if both the peak current and the amount of charge contribute to the noise, then this transformation becomes more difficult.

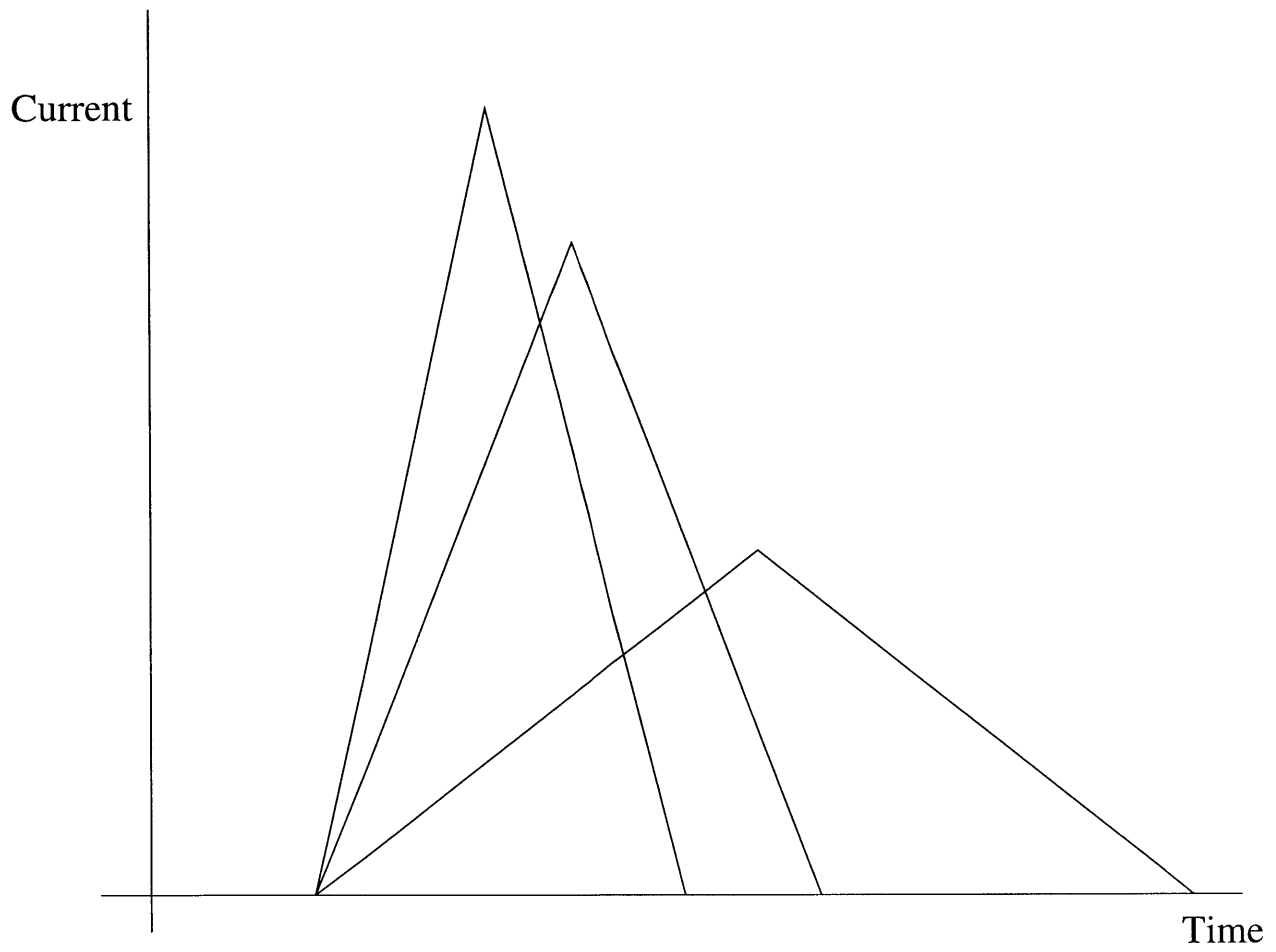


Figure 3.4: Constant charge, changing peak current

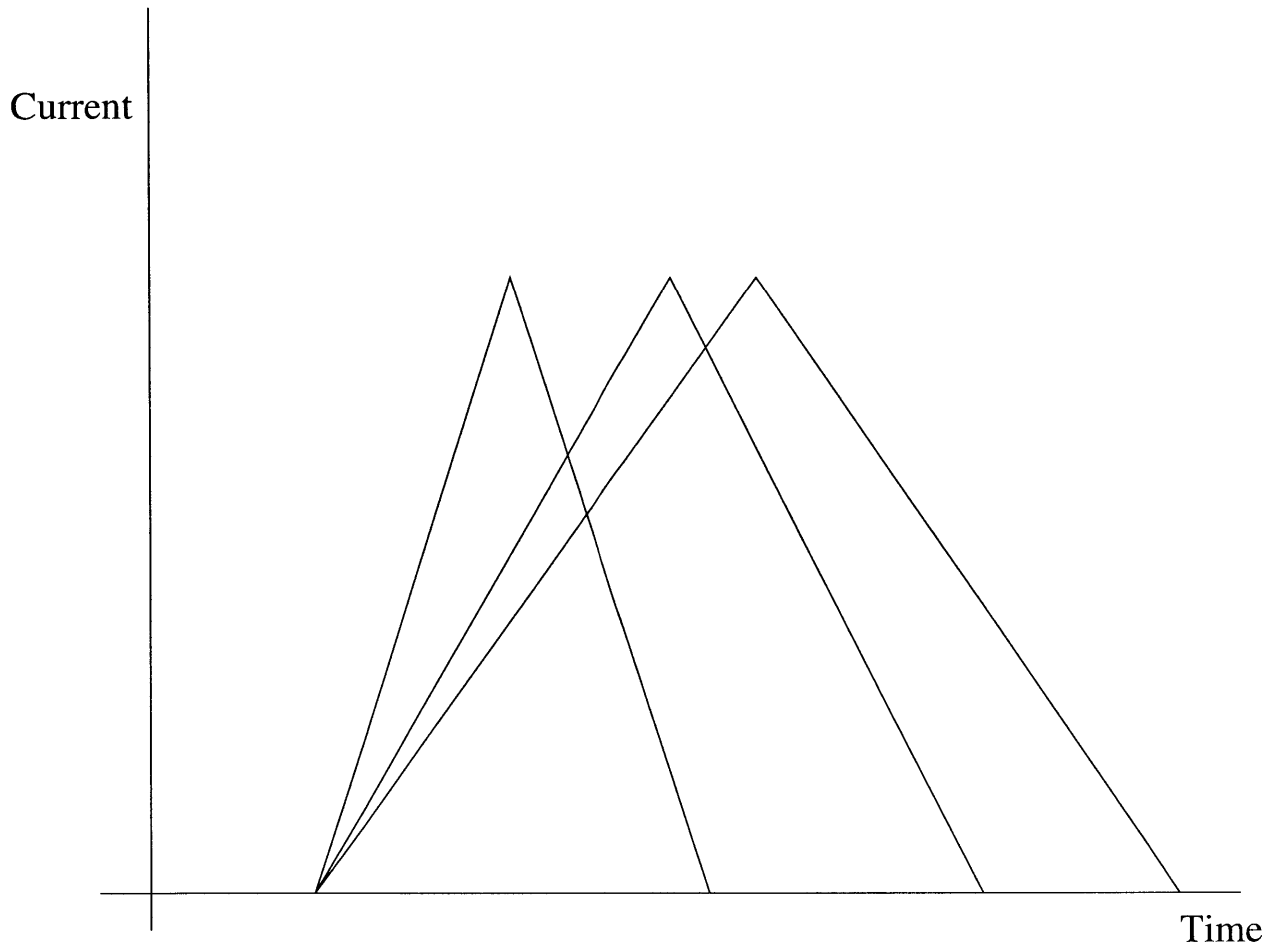


Figure 3.5: Constant peak current, changing charge

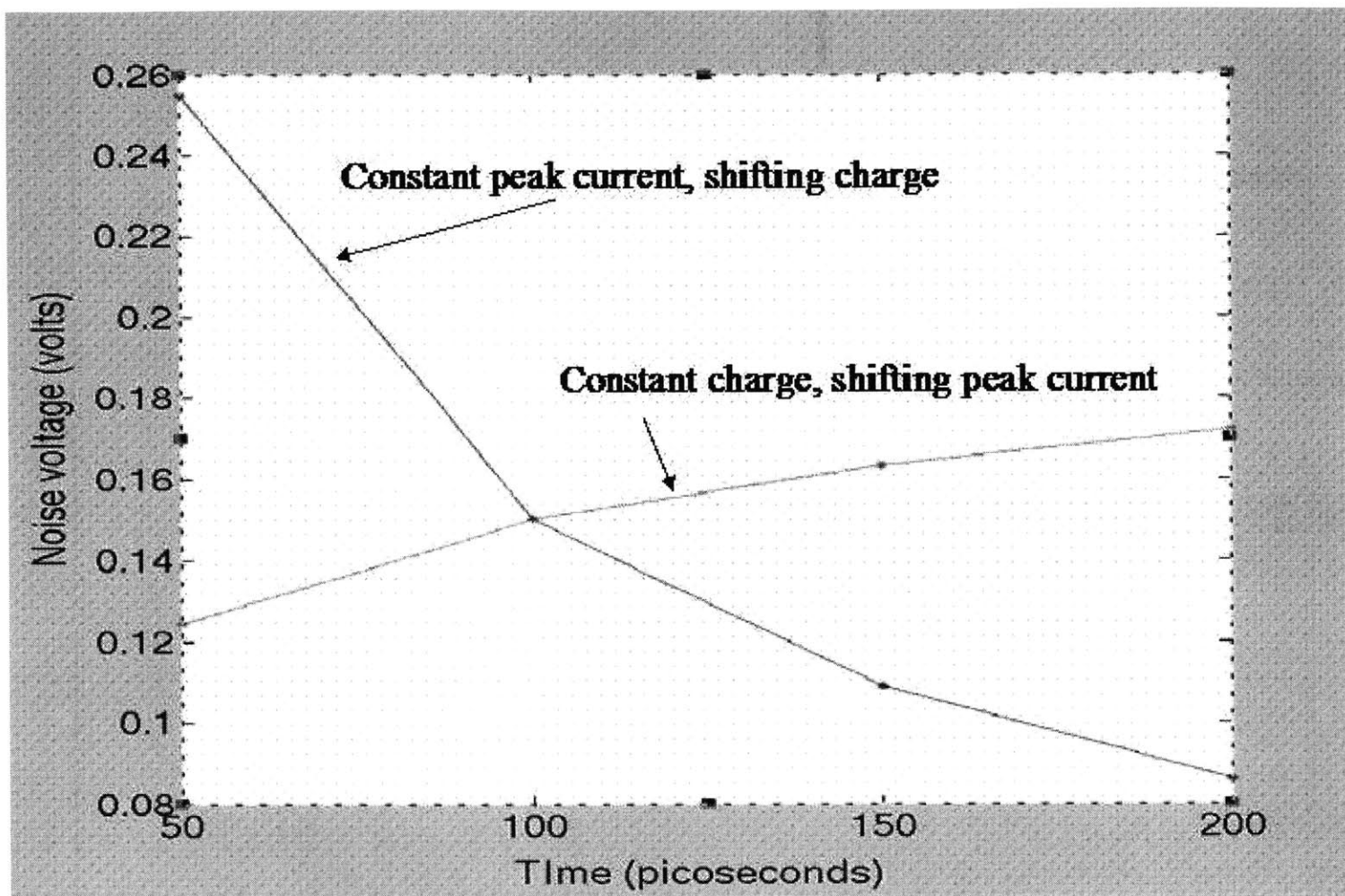


Figure 3.6: Constant peak current and charge comparison

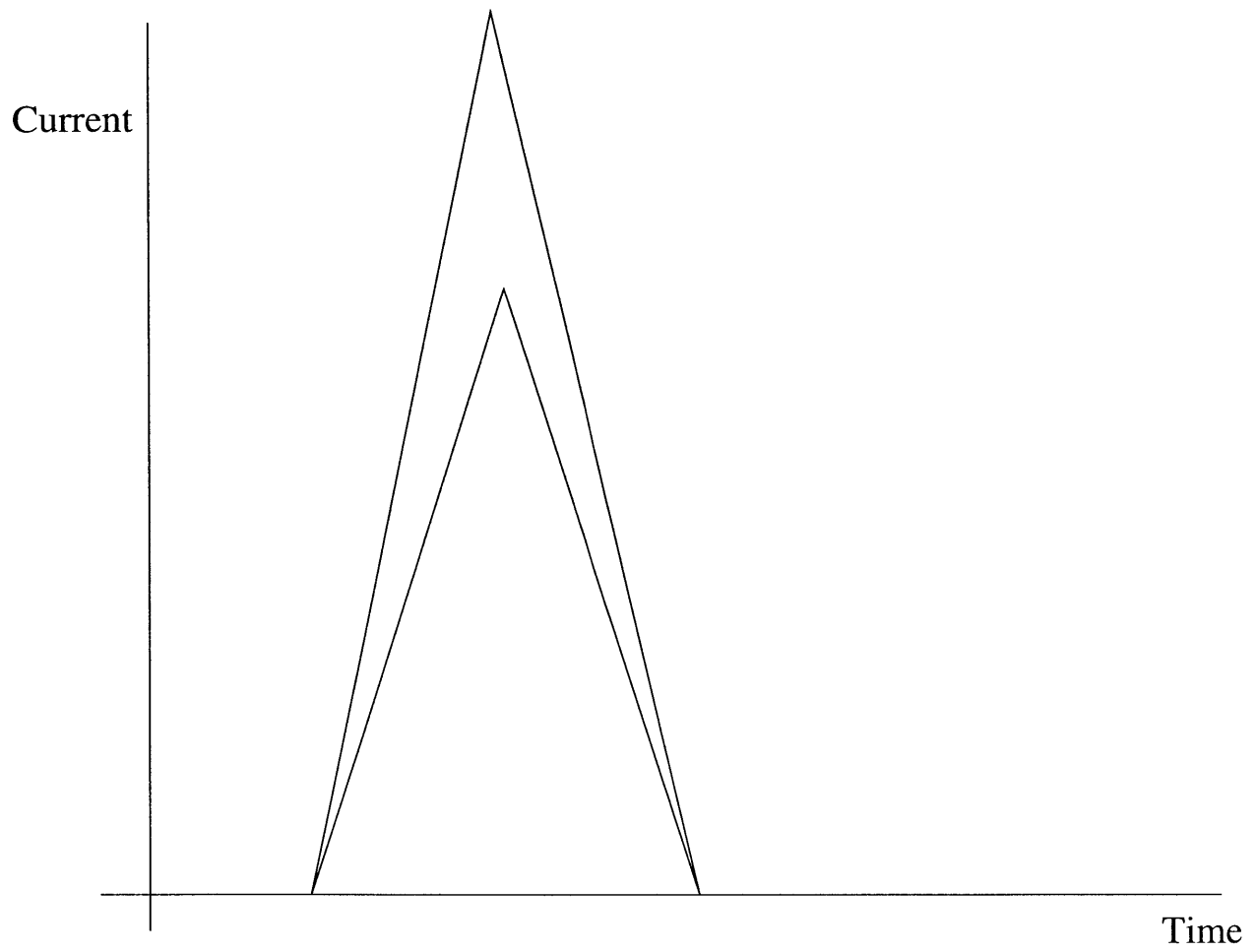


Figure 3.7: Linearly increasing noise

It is unclear whether or not the effects above are actually fundamental to the chip technology, or whether they are artifacts of the way the SPICE model was constructed. In my model, when we place current sources next to resistors, the peak current seems to play a bigger role than the charge. In an alternate model in use at IBM, where current sources are placed next to capacitors, the charge seemed to play a bigger role. The literature cited earlier seemed to suggest that one could determine the amount of charge that the macro would require, and arrange it into a triangular pulse. This would imply that the area under the curve is the only relevant consideration.

Finally, if the peak current (and not just the amount of charge) played a substantial role to the amount of noise, it would have serious implications for circuit design. This is because of the fact that frequently, speeding up a chip results in narrower current pulses with identical amounts of charge under the curves, but taller peaks. If doing this actually caused noise to go up, even with the same amount of charge being moved, it would make speeding up chips difficult.

Chapter 4

Noise propagation trends

In this section, I shall explore the impact of changing chip parameters such as parasitic resistance and parasitic capacitance on noise propagation. I shall also discuss the effects of decap placement on noise. The concepts in this chapter will be critical towards developing a strategy to estimate noise on chip.

4.1 Background capacitance and resistance

Background capacitance and resistance on chip are well-known to slow down chip operation. However, intuitively, increasing background capacitance would also decrease noise, by increasing the amount of charge stored on chip ([1] demonstrates this convincingly). The question is: what is the precise quantitative relationship between background capacitance, resistance, and noise?

The simulated results of this are complicated. The relationships between capacitance, resistance, and noise are quite disparate depending on the distance from the current source. To find these relationships, I did simulations involving a single triangular current pulse, and checked the noise at various locations in its vicinity. The following were the main simulated observations:

- When I checked the noise *close* to the current source, the noise increased roughly linearly as a function of resistance. The noise decreased when background capacitance increased, but much slower than

inversely; it decreased roughly as an inverse square root.

- As we moved away from the noise source, the noise increased sub-linearly as a function of resistance. Roughly three to four nodes away, the resistance ceased to have almost any effect on the noise whatsoever. Meanwhile, as I moved away from the noise source, the relationship between background capacitance and noise fit increasingly well to an inverse curve.

These observations have a relatively straightforward theoretical justification. At the current source, there are two main effects: first, charge is discharging from the on-chip capacitance, and second, it is necessary to maintain a certain level of current across the resistors. Both of these effects will contribute to noise. Earlier, I presented a graph indicating that the resistive effect was more significant than the capacitive effect, by comparing the effects of altering the charge (holding the peak current constant) and altering the current (holding the charge constant). Now, if the resistive effect totally dominated, then increasing the resistance would linearly increase the noise (due to $V = IR$) and increasing the capacitance would have no effect. If the capacitive effect completely dominated, increasing the capacitance would inversely decrease the noise (due to $V = \frac{Q}{C}$ and increasing the resistance would have no effect. In reality, increasing the resistance has a slightly sublinear effect, and increasing the capacitance has a sub-inverse effect. The implication is that the resistive effect is more significant than the capacitive effect.

However, as the noise ripples throughout the chip, there are no other current sources which need to maintain a particular current. Accordingly, the main source of noise will arise from capacitors discharging. Now, increasing resistance will result in some additional noise, because increasing the RC time constant will decrease the amount of charge that is released in a certain stretch of time, so voltages will droop in order to compensate. However, the main observed impact of increasing the resistance is that the noise just propagated more slowly through the chip. Accordingly, as one moves away from the noise source, the capacitive effects will dominate. Indeed, several nodes away from the noise source, the capacitive relationship becomes inverse, and increasing the resistance begins to have minimal effect on the actual noise.

Package effects also contribute to these trends. While the package provides only small amounts of charge

under normal conditions, as the on-chip resistance rises, the package begins to contribute a greater proportion of the charge. This trend was also observed in simulation; increasing the resistance at small values of the resistance had a greater effect than increasing the resistance at large values of the resistance, and at a certain point increasing the resistance had minimal effect.

The implications of this analysis are as follows:

1. In general, the relationship between capacitance, resistance, and noise is complicated at various nodes on the chip. This was one of the factors that led me to use precharacterization, rather than closed-form mathematical analysis, in order to create a noise estimation procedure.
2. Increasing background capacitance or background resistance can be an effective strategy to battle noise voltage, but only under certain circumstances. If the goal is to minimize the noise around a sensitive node far away from the main sources of noise, then increasing the surrounding background capacitance might have a pretty fair impact, but decreasing the resistance in the power grid would not have an effect. Conversely, close to sources of noise, decreasing the resistance in the power grid can have very significant impacts on the noise.

4.2 Noise modeling on chip

In this section, I will expand on the initial discussion of noise in the context of a single RC circuit. I will discuss experimental observations and corresponding theoretical models of noise propagation on chip. All of the insights in this section will be necessary in the next section, which will discuss accurate ways of evaluating the total noise at any node on chip. This noise evaluation technique is the center of the decap estimation algorithm discussed at the end of this thesis.

4.2.1 Superposition

We saw in the last section that the total noise in the RC circuit was linearly proportional to the magnitude of the current peak, assuming the current width remains constant. In general, when solving circuits with

numerous current sources, their effects can be added linearly; we can treat each current source individually, and sum their resultant noise values. This suggests a general principle of superposition, in which the total noise due to several current sources is equivalent to the total noise due to each individual current. Indeed, this principle of superposition holds in SPICE analysis. The significance of this is that to determine the noise at any point on chip, we only need to fully characterize the noise due to a single current source. We can then linearly add the noise due to each current source to obtain the true value. An example of superposition at work can be seen in Figure 4.1. In this figure, the bottom curve represents the noise due to three current sources firing together; the other three curves represent the noise due to each individual current source firing on their own. The bottom curve can be seen to represent the sum of the other three curves.

However, superposition does not work for capacitors. It is apparent from the equations determined in chapter 3 that adding capacitance does not have corresponding linear effects on the noise; indeed, there is a diminishing marginal impact of adding additional capacitance. Consequently, knowing the effect of placing N capacitors alone in location 1, and the effect of placing M capacitors alone in location 2, cannot be used to determine the effects of placing N capacitors in location 1 and M capacitors in location 2. This is the main difficulty of the noise estimation problem.

4.2.2 Noise propagation

In simulation, it generally appeared that the peak noise due to a current pulse decreased roughly as the inverse square of the distance to the current pulse, as can be seen in Figure 4.2. This trend can be explained by the following theoretical model. We can imagine that the noise travels along the chip at a certain speed. When the noise wave reaches some node N at a distance d of the source, all nodes of distance d or closer to the source will have been affected. In reality, what is occurring is that charge stored on capacitors on outlying nodes recharge capacitors on inner nodes until an equilibrium is reached. So for instance, the capacitors at distance 2 will recharge the capacitors at distance 1 until an equilibrium between these two sets of capacitors is reached; as the capacitors at distance 2 discharge, the capacitors at distance 3 will recharge them until an equilibrium is reached; and so the noise is spread through the chip.

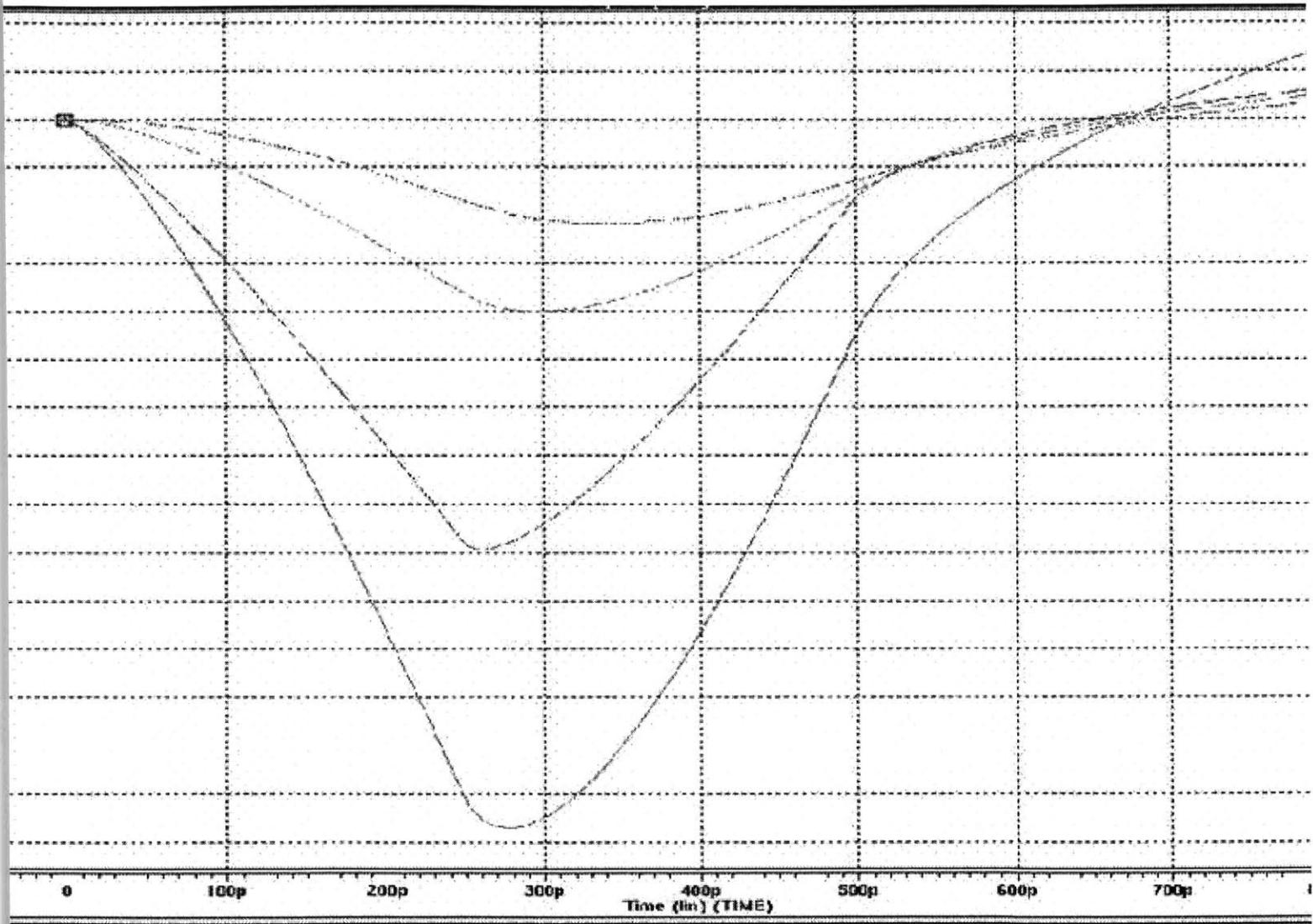


Figure 4.1: Superposition of several curves

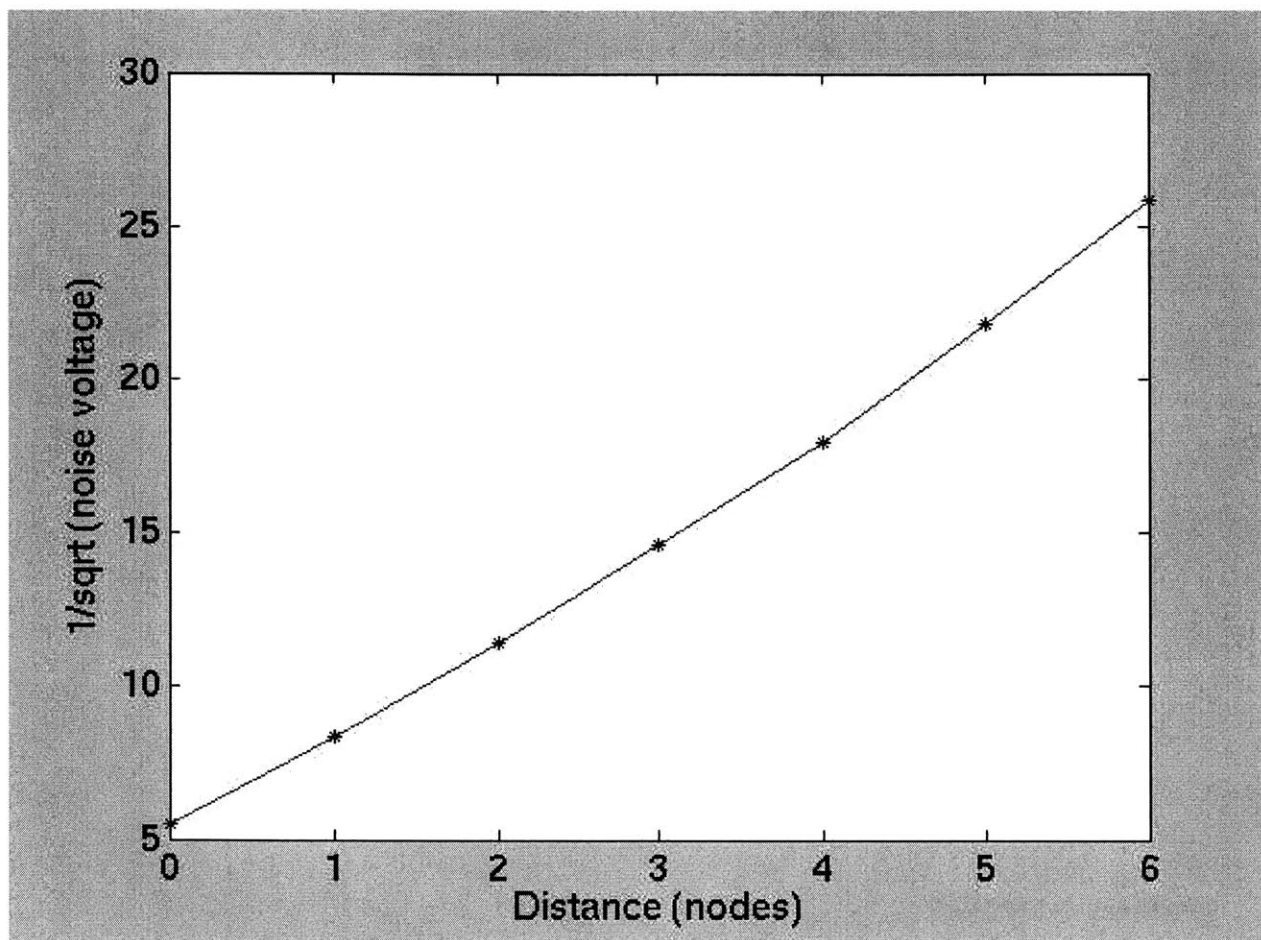


Figure 4.2: Inverse square root of voltage as a function of distance from current source

It is therefore reasonable to imagine a model where, as noise spreads to a distance d , the noise is distributed across all nodes of a distance d . Since we are using a two-dimensional grid, the total number of nodes within a distance d is proportional to the area of the circle of radius d , and thus increases with the square of the distance from the source. Accordingly, we would expect the noise to drop off as roughly an inverse square, as the above graph indicates.

This fact is of extreme importance in developing a noise propagation model, for two reasons. First, it can allow us to estimate the noise from far-away current sources with reasonable accuracy, given the noise from nearby current sources. However, more significantly, it implies that noise drops off extremely rapidly, so that the vast majority of the noise from some current source will come at the very node at which the current source is located. Experimentally, consider a setup like Figure 4.3. This produces significantly more noise at the side, top and bottom nodes than at the middle node. (The middle node does experience more noise than the corner nodes, however). The implication of this is that for any realistic distribution of current sources on chip, if we are able to substantially reduce the noise exclusively at the current source nodes, it is very likely that we have substantially reduced the noise everywhere. Put differently, assuming that our noise tolerance is constant at every node on chip, it suffices to reduce the noise to within acceptable limits exclusively at nodes which contain current sources, because we can then infer that all nodes are within acceptable limits. (If nodes have different noise tolerances, however, this assumption becomes invalid.) This fact will be used to substantially reduce the runtime of the eventual decap allocation algorithm.

4.2.3 Effect of decap on noise propagation

As discussed in the previous section, noise propagates through the chip radially from the original source of noise. Adding decoupling capacitance has the effect of reducing the noise by a certain fraction. One critical question in the placement of decoupling capacitors is: to what extent does the location of the decaps affect the noise propagation?

The first series of experiments to answer this question were performed by placing a current source at some node N , and measuring the noise at some measurement point M . I placed capacitors at every inter-

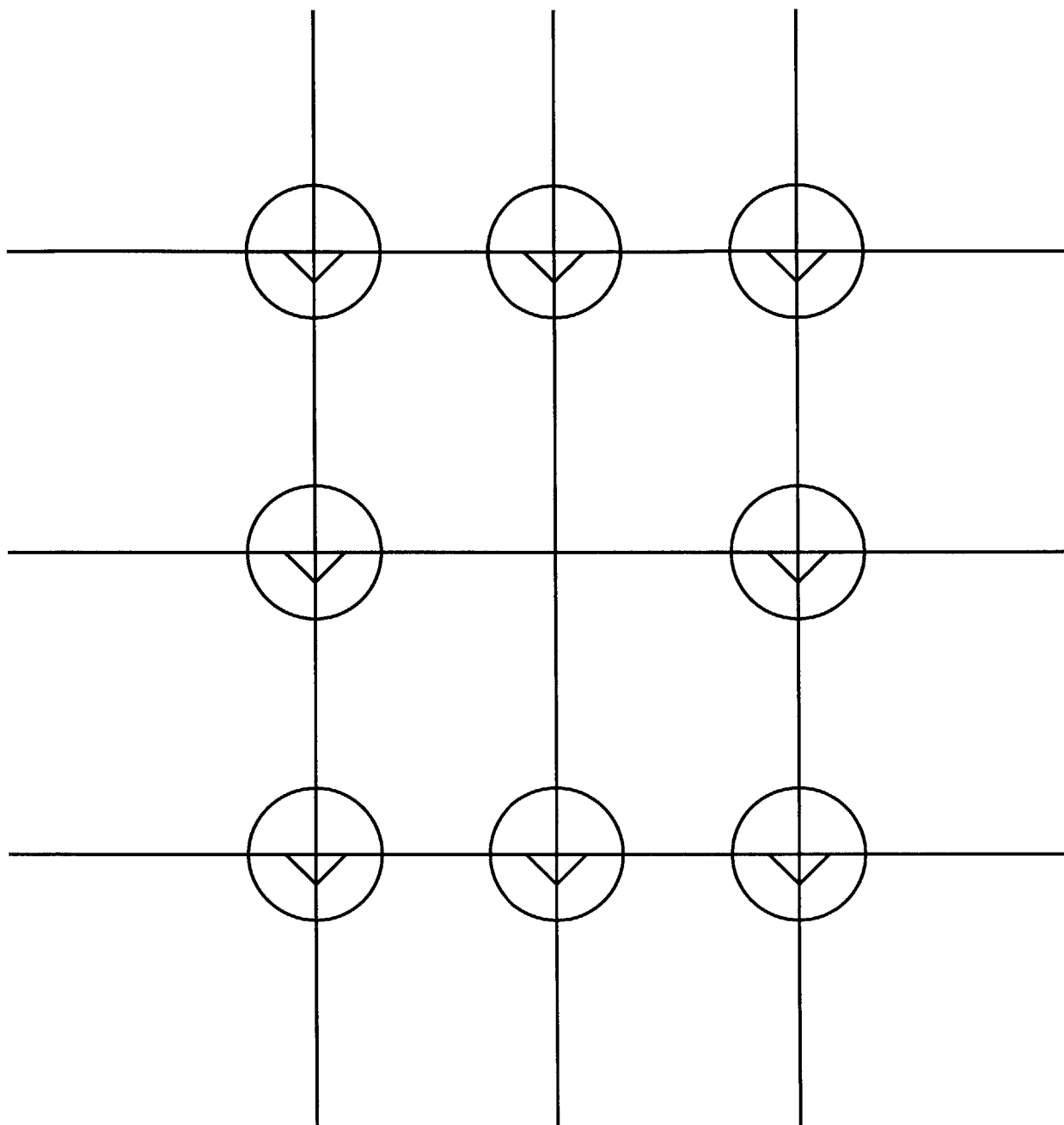


Figure 4.3: Hypothetical current source setup

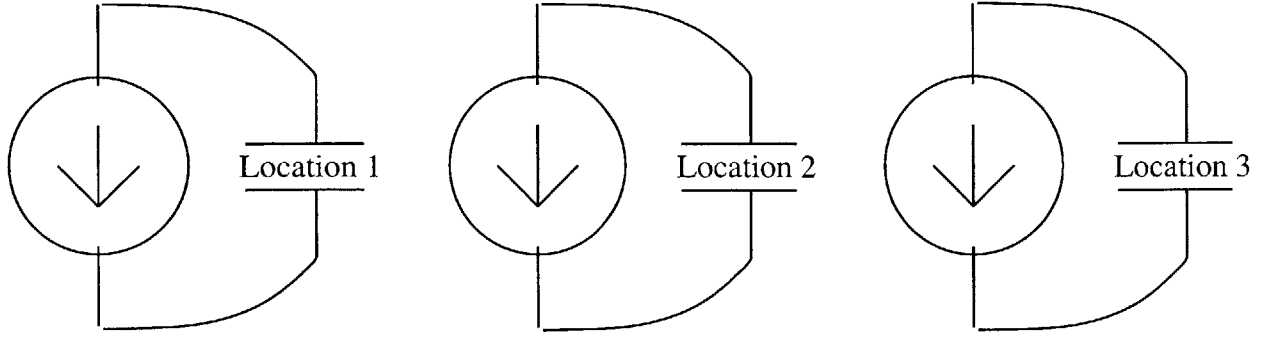


Figure 4.4: Placing decaps in various locations

mediate location. So, in Figure 4.4, the effect of decoupling capacitance at each location would be measured individually.

The following observations were made:

- Decoupling capacitors placed at the current source (Location 1) produced the identical effect as decoupling capacitors placed at the measurement point (Location 3).
- Decoupling capacitors placed at intermediate locations (e.g. Location 2) produced a slightly smaller effect than decoupling capacitors placed at either Location 1 or Location 3.
- Decoupling capacitors placed outside of the line of sight between the current source and the measurement point (e.g. in a hypothetical node to the left of Location 1) had a far smaller effect.

These observations can be explained using a theoretical model suggested in Figure 4.5. Adding decoupling capacitance attenuates the noise wavefront by a certain ratio. Whether this attenuation occurs early or late in the wavefront propagation does not affect the degree of attenuation. So in the above example, imagine that moving a single node away from the current source causes the noise to attenuate by a factor k , i.e.

$$N \rightarrow \frac{N}{k}$$

Imagine that adding some decap causes the noise to attenuate by a factor q . Consider the noise two nodes away from the current source. Then, placing the decoupling capacitance at the current source would

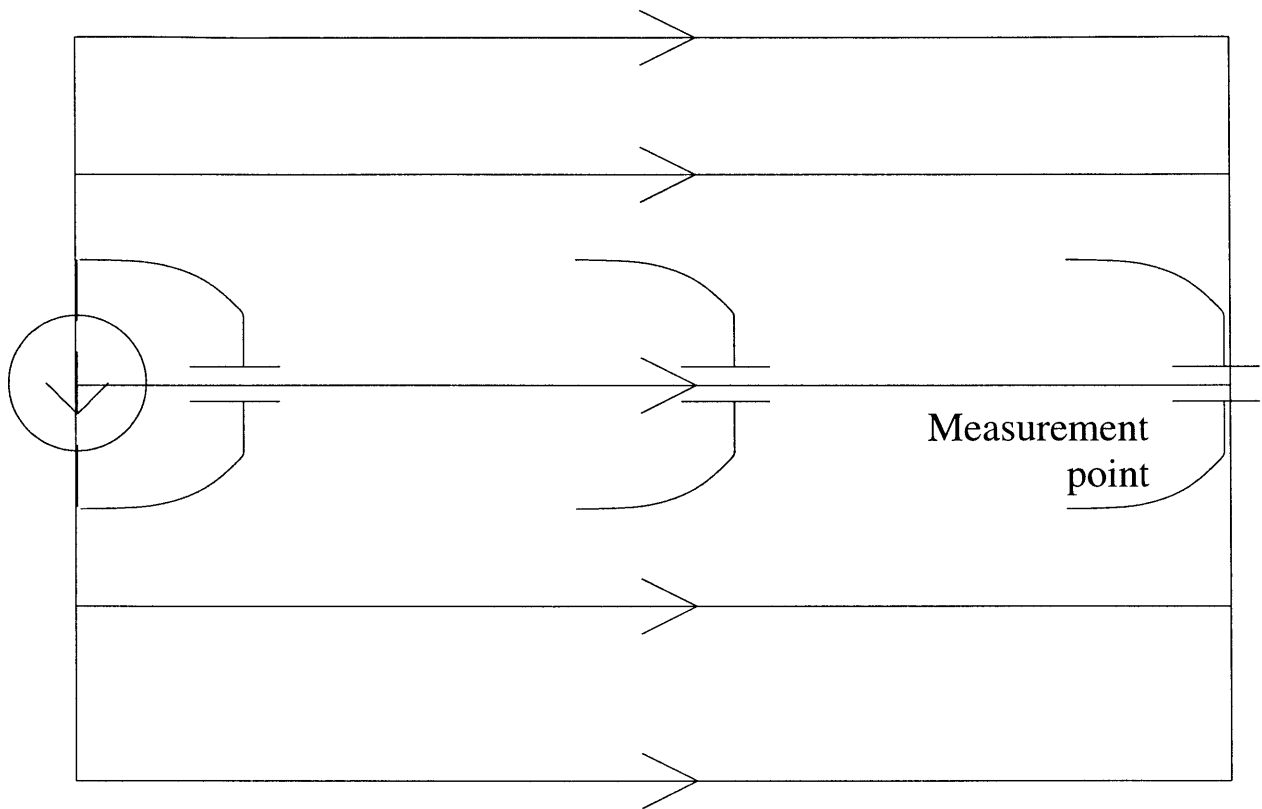


Figure 4.5: Noise propagation model

cause the noise to attenuate as

$$N \rightarrow \frac{N/q}{k} = \frac{N}{k^2q}$$

Whereas, placing the decap at the measurement point would cause it to attenuate as follows:

$$N \rightarrow \frac{N}{k}/q = \frac{N}{k^2q}$$

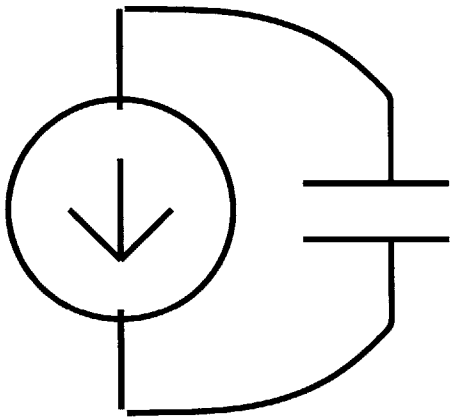
Both of these equations, however, result in a total noise $\frac{N}{k^2q}$. That is, placing decap at the measurement point is equivalent to placing it at the current source.

One might wonder why the same would not be true for decaps placed in intermediary positions between the current source and the measurement point (for instance, Location 2 in the above diagram). The answer is, as the diagram suggests, there are numerous paths for the noise to take between the current source and the measurement point. Placing the decap at Location 2 only attenuates one path, while placing it at Locations 1 or 3 attenuates all of the paths. However, the shortest path travels fewer nodes than the second shortest path (three as opposed to five in the above example). Accordingly, adding decap in Location 2 has a very similar impact to adding decap in Location 1, a fact we will exploit in the noise estimation algorithm.

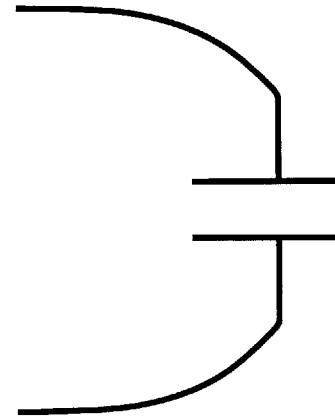
The final question is the effect of decaps outside of the line of sight. According to the wave propagation model, a decap placed to the left of the current source would have no impact whatsoever. This is false; adding decap there does help replenish the capacitors at the current source, and therefore does have an effect on the noise throughout the chip. However, decaps outside of the line of sight have a less significant impact than decaps in the line of sight, which is also a fact that will have to be taken into account.

4.2.4 Distribution of capacitance among nodes

Consider the five different situations in Figure 4.6. In the earlier section, I described how situations 1 and 5 would produce the identical impact at the measurement point. Similarly, situations 2 and 4 produce the identical impact. However, do situations 1,2, and 3 differ? Assuming the total amount of decoupling



N1 decaps



N2 decaps

Situation 1: $N1 = 0, N2 = 20$

Situation 2: $N1 = 5, N2 = 15$

Situation 3: $N1 = 10, N2 = 10$

Situation 4: $N1 = 15, N2 = 5$

Situation 5: $N1 = 20, N2 = 0$

Figure 4.6: Five situations

capacitance is the same, does the distribution of this decoupling capacitance affect the total noise at the measurement point?

The answer to this question is yes, which complicates the question considerably. The reason is that we've already seen that there is a diminishing marginal impact to adding capacitors at a node. Consequently, consider a situation in which there are 10 decaps at the current source, and 0 decaps at the measurement point. Adding 10 additional decaps at the current source (yielding situation 5) will have less of an impact on the noise than adding 10 decaps at the measurement point (equivalent to situation 3), because there is a diminishing marginal impact to adding more decap at Location 1.

Accordingly, spreading decap out evenly (i.e. situation 3) has two impacts:

1. The noise at the measurement point will tend to be lower.
2. The peak at the measurement point will occur somewhat later.

We will call this effect the “double node effect”.

In SPICE simulation, effect 2 was far more noticeable than effect 1, as Figure 4.7 indicates. In fact, there was minimal impact on the noise peak, and in fact the noise appears slightly greater in situation 3 (for reasons that are not entirely clear). However, it is very clear that the time at which the peak occurs is far greater. To explain the significance of this, let the time of the peak noise in situation 1 and 5 be t_p . At time t_p , the noise associated with situation 3 is significantly less than the noise associated with situations 1 and 5. Accordingly, even if the peak noise is not much affected by the distribution of capacitors, the noise at specific times is affected.

In general, on chips, we may have several dozen current sources, all firing at the same time. The noise at any given node is the superposition of the noise due to each of these current sources at that node. It turns out that it is fairly easy to approximate the peak noise due to each of these current sources at that node. However, attempting to add these values linearly will grossly overestimate the noise, since these peaks occur at radically different times. Overcoming these time shifts is the main challenge in approximating the noise at any node.

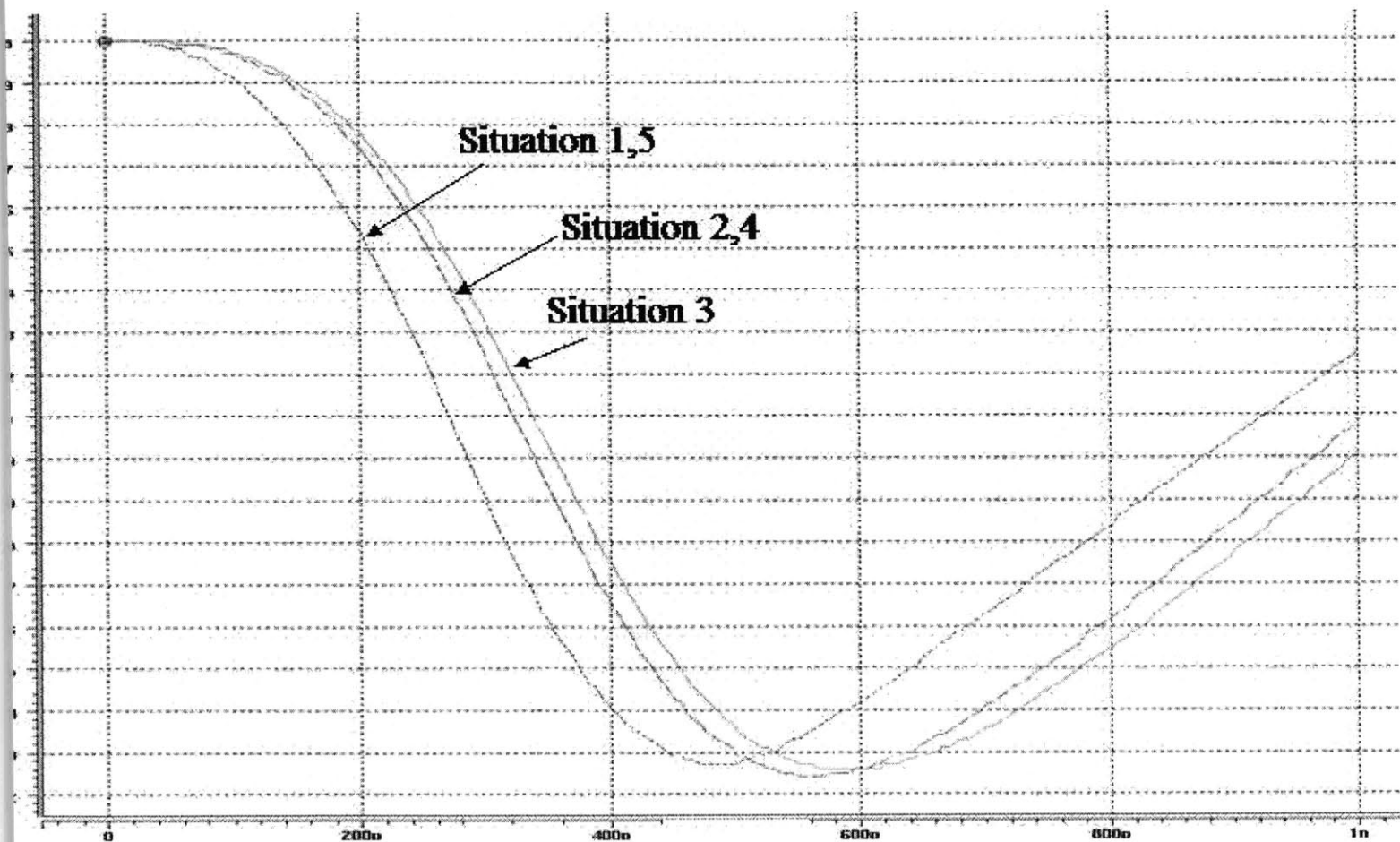


Figure 4.7: Simulated noise in five situations

Chapter 5

Noise approximation

5.1 Problem description

In this section, I will describe my solution to the problem of finding the noise at any node, due to an arbitrary collection of current sources and decaps scattered throughout a chip. This problem is not merely academic; this approximation is the core of the next chapter, which describes an algorithm to actually place the correct amount of decaps at every node on the chip.

Let us spell out the problem in more detail. Consider the scenario in Figure 5.1. The question we are trying to answer is as follows. What is the value of the noise, at any of the nodes, as a function of N_1 , N_2 , and N_3 ? N_1 will affect the value of the noise at all three nodes, N_2 will affect the value of the noise at all three nodes, and N_3 will affect the value of the noise at all three nodes. We need to determine the noise at all nodes, as a function of the decap at all nodes.

Noise estimation with decap has been studied numerous times in the literature. [11], [14], [8] and [16] all make accurate calculations at the transistor level, a level of abstraction which is too low for our purposes. A direct differentiation model is suggested in [10], which does not appear to take into account the diminishing effect of decaps at different distances from the noise source. [19] calculates the noise with the assumption that all of the charge comes from the pins, and calculates the IR drops from the pins to the current sources;

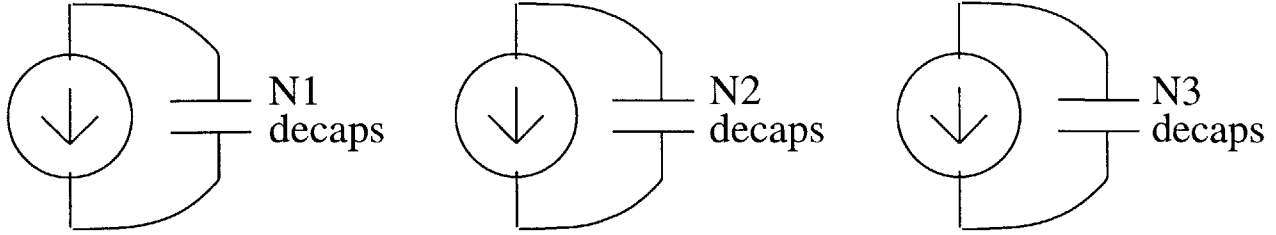


Figure 5.1: Scenario with three current sources and decap at each node

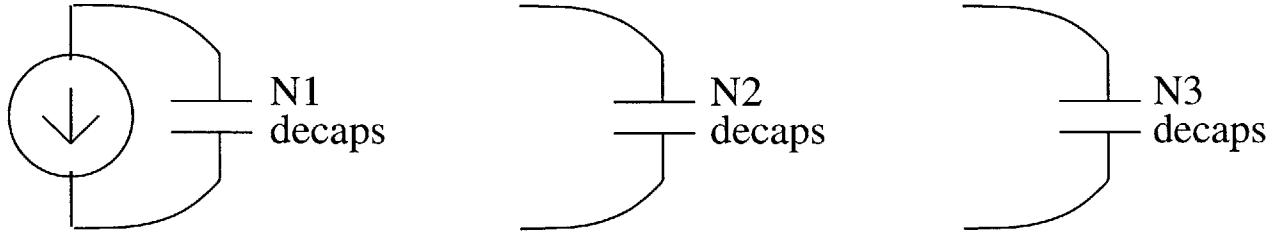


Figure 5.2: Scenario with one current source and decap at multiple node

this is highly different from our model, which assumes that all of the charge comes from the background capacitance. Additionally, it assumes the sole source of the noise comes from charge being removed from decap, rather than the need to maintain a certain current across a resistor. Accordingly, approximating the noise according to the particular model and set of assumptions in this case appears relatively unique in the literature.

We can simplify the problem. Recall that the total noise at any node is equal to the superposition of each individual current source. Therefore, consider the scenario in Figure 5.1. Consider, again, the problem of finding the noise at any node. If we can solve this problem involving only *one* current source for any arbitrary distribution of decaps, then we will have solved the general problem, since we need to only superimpose the contributions of each individual current source. Therefore, solving the problem in Figure 5.2 - finding the noise at any location, with an arbitrary distribution of decap, but only a single current source - is what we will concern ourselves with.

For the rest of this thesis, we will view the chip as an idealized two-dimensional grid. Decaps and current sources can be placed at any node on the grid. Therefore, the model you see in Figure 5.3 will be looked at in the idealized form of Figure 5.4.

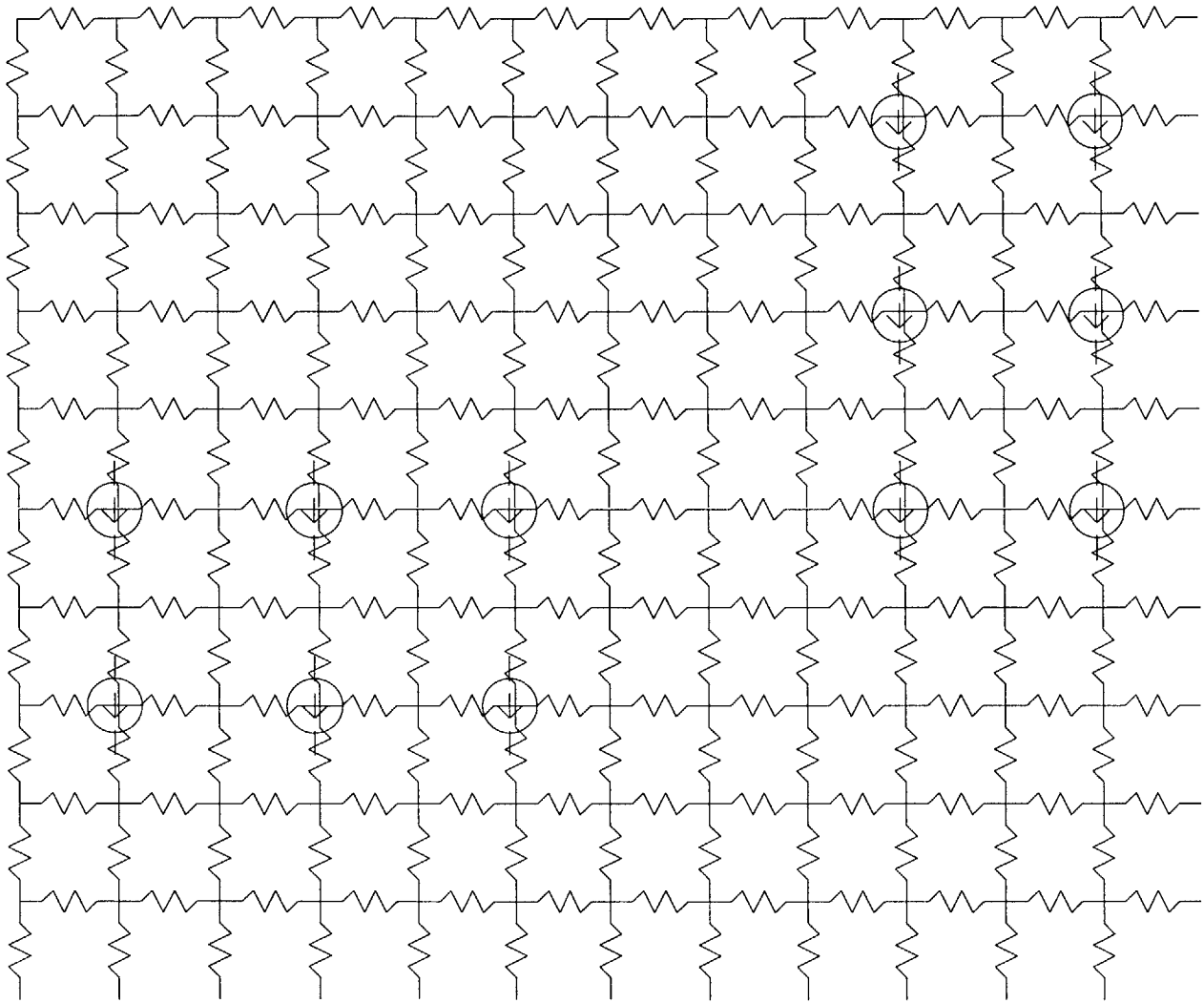


Figure 5.3: Actual birds eye view of section of chip

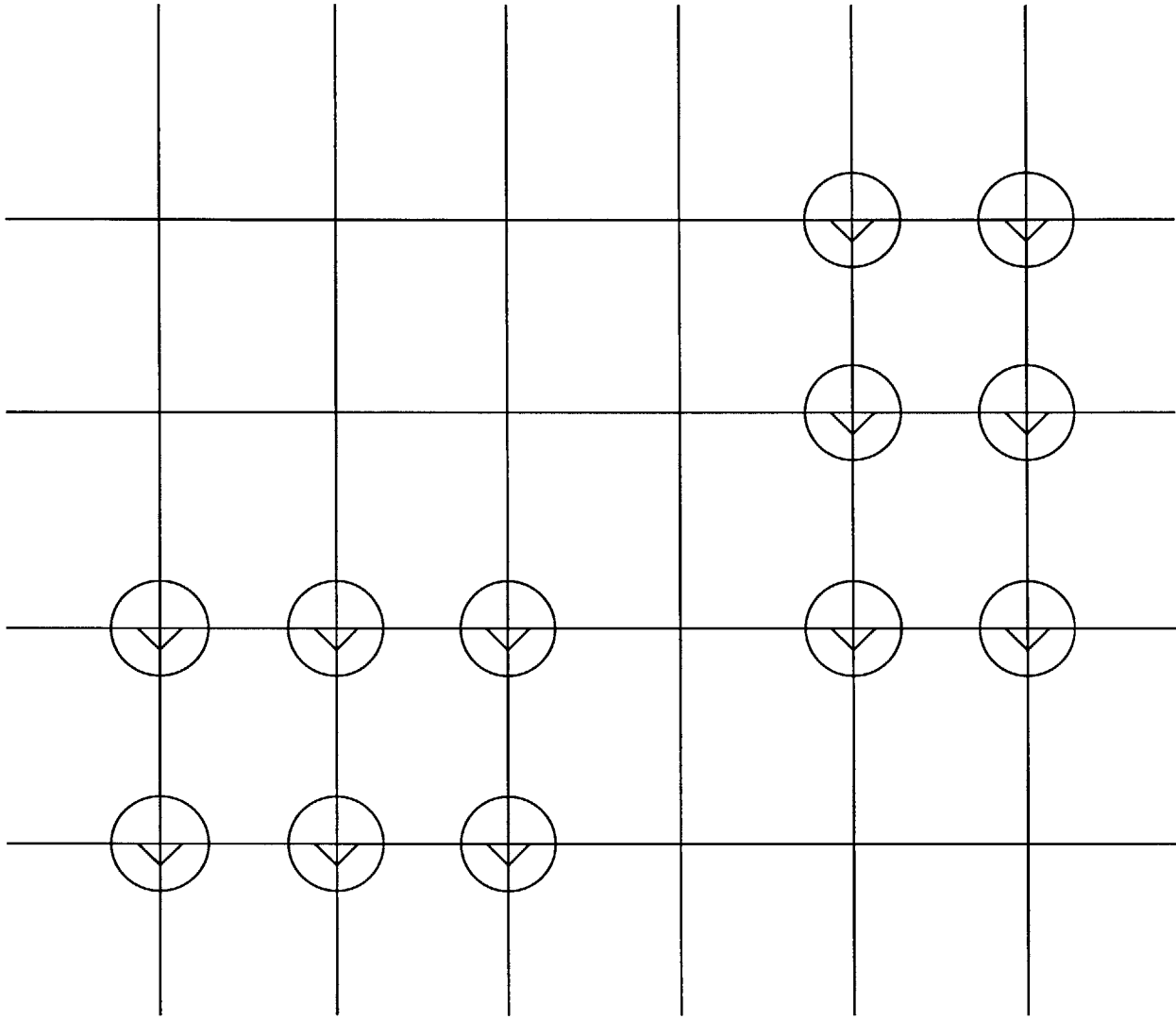


Figure 5.4: Idealized birds eye view of section of chip

The fundamental premise of this chapter is the idea of precharacterization. In the eventual algorithm to place decaps on chips, an iterative process is used, in which capacitors are placed and the noise is measured until the layout of capacitors on chip is optimized. However, neither of the two most intuitive methods of measuring the noise - calculating it by circuit analysis, or simulating it in SPICE - is usable. Calculating the noise due to an enormous, distributed chip by means of mathematical analysis is intractably difficult. Calculating the noise by means of repeated simulation is often far too slow. We must therefore arrive at a third method of calculating the noise, which is to precharacterize the chip by means of a few, simple simulations, and use the results to predict the noise as a result of more complicated setups. The challenge is to create a systematic methodology for exactly what simulations to perform, and what data to collect, in order to generate plausible results. The rest of this section will describe different methods of precharacterization for noise analysis.

5.2 The naive algorithm

As a first pass, I will discuss a model in which an arbitrary setup of current sources can be used, but with two restrictions:

1. Current pulses must be of a single uniform width.
2. Decoupling capacitors are only placed at a single node on the entire chip.

I will explain how to determine the noise at the node with the decoupling capacitors through a very simple set of precharacterization steps. Obviously, these assumptions do not hold true on real chips; this discussion is include only to give an intuition for the difficulties involved in the more general problem of when decaps are laid out arbitrarily on chip.

As mentioned above, adding decoupling capacitance at a certain node results in the peak noise voltage decreasing by the function $\frac{1}{V_{peak}} = mC + b$, where V is the peak noise voltage, C is the amount of decoupling capacitance, and m and b are arbitrary constants. m and b will change as a function of the amount of background capacitance and background resistance, and will also change depending on how far the measurement

point is from the current source. After some simulation, it appeared that the function $\frac{1}{V_{peak}} = mC + b$ held at the node with the decoupling capacitors, regardless of the layout of capacitors on chip, though obviously the values for m and b changed depending on how current sources were placed. If we can determine an equivalent value for m and b in this function for any arbitrary collection of current sources on chip, we have solved the problem, because given m_{eff} and b_{eff} we can solve for V_{peak} for any input value of C_{decap} . Therefore, the goal is to determine the value for m_{eff} and b_{eff} through precharacterization.

The precharacterization procedure is as follows. Place a current source, of arbitrary magnitude, at some location near the center of the chip. Simulate this circuit, noting the peak resultant voltages at every node in the vicinity. (In my experience, “the vicinity” means perhaps one to two millimeters; beyond that, the noise becomes minimal.) Add a certain amount of decoupling capacitance at the node with the current source, and simulate again, noting the peak resultant voltages everywhere in the vicinity. Repeat with varying amounts of decoupling capacitance; in principle, you only have to simulate twice, but simulating more times will lead to more accurate results.

At every single node at which we are measuring noise, we now have a series of (C, V) pairs - capacitance values at the current source node, and associated maximum noise values at the particular measurement point. We can therefore find the line of best fit of the function $\frac{1}{V_{peak}} = m_{eff}C + b_{eff}$. This will result in separate m and b values for every node in the vicinity. (If the chip is symmetrical in latitude and longitude, you only need to take values in one quadrant of the chip; that is, if one node going north is identical to one node going east, only one of these values needs to be taken. If not, you may have to find m and b values in more than one direction.) For best results, do not take reciprocals of the voltages, and find the best fitting line; instead, use a least squares curve-fitting algorithm to find the best fitting inverse curve.

Following this step, you have m and b values for every node on the vicinity. So for instance, if we originally had the current source at the node at the coordinate $(0, 0)$, we would have $m_{(0,0)}$, $b_{(0,0)}$, $m_{(0,1)}$, $b_{(0,1)}$, $m_{(0,2)}$, $b_{(0,2)}$, $m_{(1,1)}$, $b_{(1,1)}$, and so forth, (i, j) is a node of a distance i nodes longitudinally and j nodes latitudinally from the current source, $m_{(i,j)}$ is the best fitting slope at node (i, j) , and $b_{(i,j)}$ is the best fitting intercept at node (i, j) . For the remainder of this thesis, we will use this coordinate scheme; that is, we will assume

that the current source is at coordinate $(0,0)$, and a node at $(0,1)$ is directly adjacent to the current source.

Using these m and b values, we can now construct an expression for m_{eff} and b_{eff} discussed in the previous section. It is important to note at this point that the values of m and b are measured at a particular peak current; clearly, they will change based on the value of the peak current. In particular, imagine we increase the peak of the current spike (recall that we are assuming that all of the current sources have the same current width). Then, since superposition indicates that increasing the peak current proportionally increases the noise, m and b need to be normalized by the ratio of the actual value to the value used in the original measurement. So for instance, if a current source has a peak value of 500 mA, and m and b were measured with the current source at 250 mA with the same rise-time, the values of m and b would have to be halved.

The key principles to generate this estimate are:

1. Superposition - we can linearly add the effects of two current sources at a given node.
2. Symmetry - the values of m and b at a measurement point calculated by adding capacitance at the current source will be identical to the values of m and b calculated by adding capacitance at the measurement point.
3. Rapid propagation - without any decap on the grid except at one node, the noise propagates rapidly through the grid. (Recall we saw earlier that adding decap delays the noise maximum at a particular node.)

Assume that there is some arbitrary collection of current sources laid out on the chip, and we are trying to find m_{eff} and b_{eff} at some arbitrary node n . Again, n will be the only place on chip at which we can put decap. The noise at n will be the sum of all of the noise curves produced by each source individually; we know this by superposition. However, because of the rapid propagation on the grid (assumption 3) and the absence of the time delays due to decaps in multiple locations (because decap appears in only one place), the peak voltages of each curve occur at roughly the same time. Therefore, the peak voltage of the sum of the curves is approximately equal to the peak voltage. That is, imagine V_i is the voltage curve due to the

current source at node i , and $V_{peak,i}$ is the noise peak of the curve at node n due to the current source at node i . Then V_{peak} at $n \approx \sum_i V_{peak,i}$. (This assumption does not hold when the double node effect begins to appear.)

So, consider for a moment exclusively the current source at some node i . We already determined m and b values above for the curve we'd get at n if we placed decap at node i ; this is precisely what our precharacterization step accomplished. However - and this is the key step - because of symmetry, this curve is actually equivalent to the curve we'd get at n if we placed decap at node n . (This symmetry argument will constantly be used in the algorithms that will follow).

Now, let C be the capacitance at node n , and let i_1, i_2 , and i_3 , be locations of current sources. We know that $V_{peak,i_1} = \frac{1}{m_{i_1}C+b_{i_1}}$, $V_{peak,i_2} = \frac{1}{m_{i_2}C+b_{i_2}}$, $V_{peak,i_3} = \frac{1}{m_{i_3}C+b_{i_3}}$ and so on, where m_{i_k} and b_{i_k} are the slope and intercept measured in the initial precharacterization step at that position. We can thus straightforwardly add the equations: $V_{peak,n} \approx \frac{1}{m_{i_1}C+b_{i_1}} + \frac{1}{m_{i_2}C+b_{i_2}} + \frac{1}{m_{i_3}C+b_{i_3}}$.

Unfortunately, this can not be rewritten as a function in the form : $\frac{1}{V_{peak,n}} \approx m_{eff}C + b_{eff}$, and accordingly easily solved for C . However, upon experimentation, V and C do, in fact, appear to vary inversely, and so we can create a very useful approximation. The approximate value of b_{eff} is actually the exact value of the peak noise with zero decoupling capacitance; we have $V_{peak} = \frac{1}{b_1} + \frac{1}{b_2} + \frac{1}{b_n}$ or $\frac{1}{V_{peak}} = \frac{1}{\frac{1}{b_1} + \frac{1}{b_2} + \frac{1}{b_n}}$. It turns out that we can approximate the slope as well. We imagine a situation where there is an infinite amount of noise at $C = 0$, i.e., $b = \infty$ so $\frac{1}{b} = 0$. In that case, we would have $V_{peak} = \frac{1}{m_1C} + \frac{1}{m_2C} + \frac{1}{m_nC}$ and thus $\frac{1}{V_{peak}} = \frac{1}{\frac{1}{m_1C} + \frac{1}{m_2C} + \frac{1}{m_nC}}$, that is, we have an inverse function. It turns out that if we sum these two contributions, we get an extremely close approximation to reality. That is, under the approximate capacitances we can expect on a chip, the function $\frac{1}{V_{peak}} = \frac{1}{\frac{1}{m_1C} + \frac{1}{m_2C} + \dots + \frac{1}{m_nC}} + \frac{1}{\frac{1}{b_1} + \frac{1}{b_2} + \dots + \frac{1}{b_n}}$ comes extremely close to correctly predicting the chip noise, to within 0.1% or less in almost all cases.

Therefore, we can use this function to calculate the necessary amount of decoupling capacitance at a particular node. For a particular node, we now have the equation : $\frac{1}{V_{peak}} \approx m_{eff}C + b_{eff}$. We can now determine the noise as a function of decoupling capacitance. This concludes the “naive algorithm”.

Unfortunately, this naive algorithm must be significantly modified to take into account the existence of other decoupling capacitors on the grid. More specifically, consider current sources I_1 and I_2 at nodes N_1 and N_2 . Let N_1 be at the coordinate $(0,0)$ and N_2 be at the coordinate (i,j) . Consider placing capacitance C_1 at location N_1 , and C_2 at location N_2 . (Because of the symmetry property discussed above, we can put C_1 at location N_2 , and C_2 at location N_1 , without changing the resultant voltage/time curve at N_1 due to I_2 . Obviously this would affect the resultant voltage/time curve at N_1 due to I_1 .) Let us assume that I_1 and I_2 are very far away, so that the capacitance C_2 will have minimal impact on the noise at N_1 due to I_1 , because the radius of effect for capacitors to affect nodes with current sources is very small. (Later, we will describe how to take into account this radius of effect; for now, let us just assume that C_2 will have minimal impact on the noise at N_1 due to I_1 .) However, the noise at N_1 due to I_2 will be affected by C_2 as well as C_1 , since C_2 will decrease the value of the peak noise in the spreading noise wave, prior to it reaching N_1 .

At N_1 , we are trying to superimpose two curves - the voltage due to I_1 (call it V_{N_1,I_1}) and the voltage due to I_2 (call it V_{N_1,I_2}). Now, we could easily generate the peaks of the functions V_{N_1,I_1} and V_{N_1,I_2} . The peak noise due to I_1 can be derived directly from the function $\frac{1}{V_{N_1,I_1}} = m_{0,0}C_1 + b_{0,0}$. The peak noise due to the current source at B can be derived by using the equation $\frac{1}{V_{N_1,I_2}} = m_{i,j}(C_1 + C_2) + b_{i,j}$. This is because we saw earlier, in the discussion of the double node effect, that the peak voltage is unaffected by the distribution of the capacitors, even though the peak time is significantly affected. If the double node effect did not exist, we could add these two curves to calculate the peak voltage. However, because of the double node effect, adding the peak voltages does not produce an accurate approximation for the actual peak voltage of the superposition of the curves, because we are superimposing curves with vastly different peak noise times.

The question is, how can we create a method to superimpose numerous noise curves which reflects the fact that different peaks occur at different times? There are two possible solutions. One is to determine a way to create an approximation for the entire noise curve for the length of the entire spike. If we could do that, then we could construct curves due to every individual current source, sum them, and then simply search the resultant curve for the peak value. The other is to develop an approximate value for the time of the peak noise, and add approximations of all the curves we are superimposing at that time. I developed

methods to implement each of these two strategies, and these methods are the core of this thesis. First I shall describe the latter method, in which an approximate value for the peak time is used, and then I shall describe the far more complicated former method, in which the entire curve is approximated.

5.3 The peak time approximation

In order to calculate the peak voltage at a particular node, we need to superimpose the effects of numerous current sources. Unfortunately, because of both the distance between nodes and the effects of decoupling capacitance, these peaks often occur at different times. Accordingly, the easiest solution - just summing the peak value of each curve - provides a very crude upper bound for the amount of noise. We need to find a more effective algorithm.

Let us first consider the narrower problem of finding the total noise at a node which has a current source at that node. As discussed in the last section, solving this problem is generally sufficient to allocate capacitors on any chip. This is because if we can reduce the noise voltage at nodes with local current sources, the noise will presumably also have been reduced further away from the current source, due to the very rapid dropoff of the noise magnitude as the noise propagates throughout the chip. We will narrow the problem further; throughout this entire algorithm, until explicitly stated otherwise, we will assume that all current source spikes are of identical width.

So, consider a node N_1 with current source I_1 , and other current sources I_2, I_3, I_4 at nodes N_2, N_3, N_4 . Now, we will be superimposing several different voltage curves at N_1 . However, *most* of the noise is likely to come from the curve due to I_1 . This is because no matter how high the peak current at nodes N_2, N_3, N_4 happens to be, in our final allocation of decaps, the noise at nodes N_2, N_3, N_4 will have been reduced to within acceptable limits. Therefore, since the noise due to I_2, I_3, I_4 will have been reduced to tolerable levels at nodes N_2, N_3, N_4 , and since they will also have been reduced by the inverse-square of the distance to N_1 by the time it gets to N_1 , we can infer that the local current source (i.e. the current source at N_1) is likely to be the greatest contributor to the noise at N_1 . (A note on terminology: for the remainder of this

thesis, the “local” current source will be the current source positioned directly at the measurement point. A “non-local” current source will be a current source positioned some distance away from the measurement point.)

With that in mind, we can make the assumption *that the peak noise of the sum of curves will occur at a time very close to the peak noise due to the local current source alone.* To use an example with real-life numbers, imagine that we are superimposing three different current sources which will be producing noise at a particular node. The curve due to the local current source will reach its peak at (say) 300 ps; the other two curves will reach their peaks at roughly 500 ps. We will assume that the peak voltage will occur at 300 ps; we will calculate it by summing the peak voltage due to the local current source, plus the value of the other two curves at 300 ps. Experimentally, this results in a close approximation of the maximum noise. A fairly representative example of such a sum occurs in Figure 5.5; the noise due to the local current source is the noisiest curve, while the noise due to three nearby current sources is also shown. The idea of taking the values of all of the noise curves at the peak of the local curve will be referred to as the “peak time approximation”.

The key question we must ask now becomes: when will that peak time occur? Even in the context of current spikes with identical widths, the peak time will be a function of the decoupling capacitance, as we saw in the last chapter. Therefore, imagine that putting 100 decaps at some node causes the noise due to the local current source to reach its peak at time t_1 . Then, we would use the values of the non-local curves at t_1 . Let’s say we were to now increase the number of decaps to 150 decaps, causing the noise due to the local current source to reach its peak at t_2 . We would then have to use the values of the non-local curves at t_2 . The critical idea is that the time at which we take the non-local noise is a function of the amount of amount of decap at the measurement node. This concept will be critical in the precharacterization process.

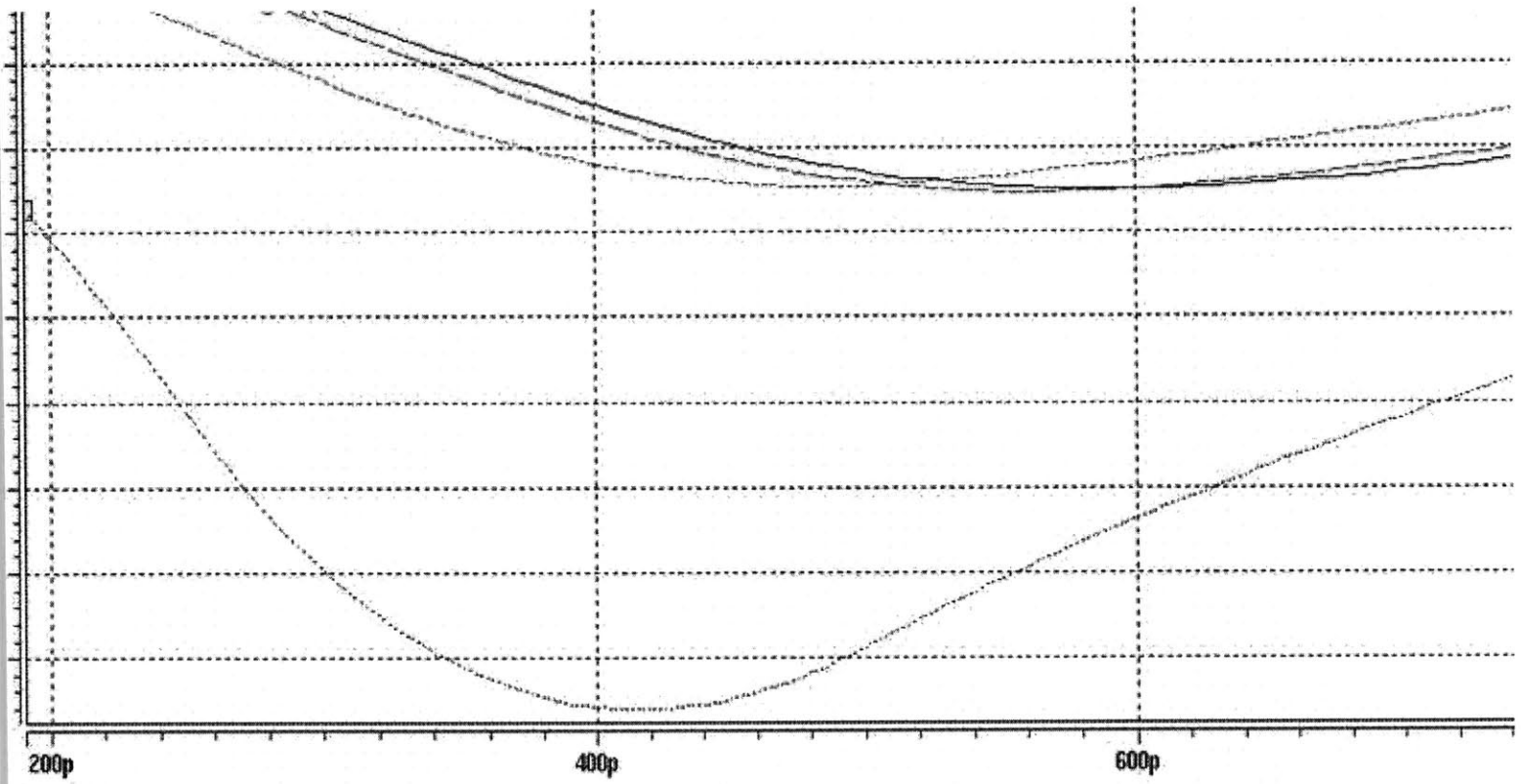


Figure 5.5: Representative example of peak time approximation

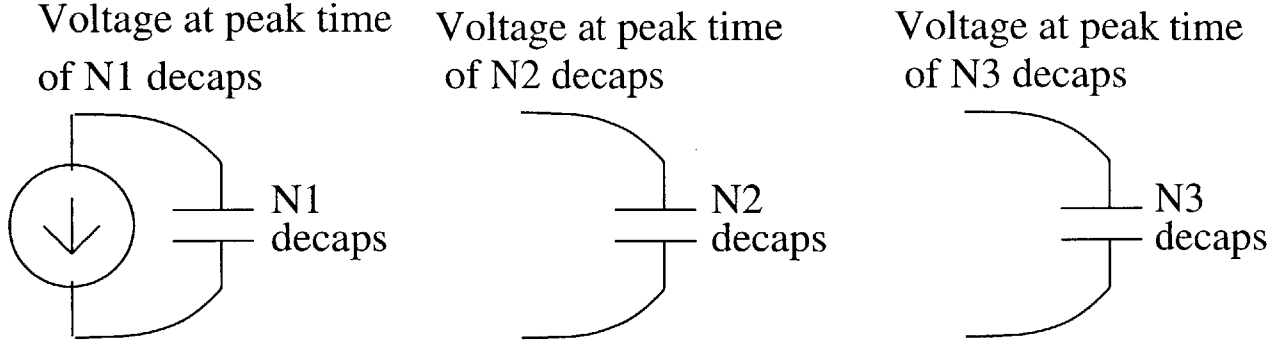


Figure 5.6: New problem statement

5.4 Implementing the peak time approximation: a first pass

Let us now consider a chip with two current sources I_1 and I_2 at nodes N_1 and N_2 with decap C_1 and C_2 . For simplicity, imagine they are sufficiently far away from each other that the decap at N_2 will not affect the noise due to I_1 at N_1 , and vice versa. Let us say we want to estimate the maximum noise at node N_1 . The noise at node N_1 will be the sum of the following two functions: a function $V_{N_1, I_1}(C_1)$ (i.e. the noise at node N_1 due to the current source I_1 , as a function of the capacitance at N_1) and a function $V_{N_1, I_2}(C_1, C_2)$ (i.e. the noise at node N_1 due to the current source at N_2 , as a function of both the capacitance at N_1 and N_2). $V_{N_1, I_1}(C_1)$ will just be the peak noise of the local current source, and will be immediately derived from the function $\frac{1}{V} = m_{0,0}C_1 + b_{0,0}$ using the values of $m_{0,0}$ and $b_{0,0}$ that we already calculated. The function $V_{N_1, I_2}(C_1, C_2)$ is more complicated. Recall that this function is *not* measuring the peak noise of the curve due to I_2 at N_1 . It is estimating the noise of the curve due to I_2 at N_1 at the precise time where the noise due to I_1 at N_1 reaches its peak! Thus, the magnitude of C_1 actually determines the time at which $V_{N_1, I_2}(C_1, C_2)$ will be measured. We now have a transformed problem statement, as shown in Figure 5.6.

Let us try to predict the form of the function $V_{N_1, I_2}(C_1, C_2)$. The following two conditions must hold for this function:

1. Adding capacitance at either node will cause the noise voltage to decrease.
2. For some fixed value of C_1 , we want the function to reduce to $\frac{1}{V} = mC_2 + b$. Similarly, for some fixed value of C_2 , we want it to reduce to $\frac{1}{V} = mC_1 + b$. (This is an experimentally observed trend, not a

mathematical fact.)

As a first pass, we might expect a function of the form $V_{N_1, I_2}(C_1, C_2) = \frac{1}{m_1 C_1 + m_2 C_2 + b}$. Indeed, this function would fulfill both of the above conditions. However, a function like this will not accurately reflect the contributions of the two capacitors. This is because, as we saw above, the distribution of the capacitors also makes a difference. Having $C_1 = 15$ and $C_2 = 5$ will result in the same curve as $C_1 = 5$ and $C_2 = 15$, but a different kind of curve than $C_1 = 10$ and $C_2 = 10$. Therefore, we must add a term reflecting the product of C_1 and C_2 . The function we will use is accordingly of the form $V_{N_1, I_2}(C_1, C_2) = \frac{1}{m_1 C_1 + m_2 C_2 + m_3 C_1 C_2 + b}$. This function still meets both of the above conditions. At a fixed value of C_2 , the function will be of the form $V_{N_1, I_2}(C_1) = \frac{1}{(m_1 + m_3 C_2) C_1 + (m_2 C_2 + b)}$, which is still of the form $\frac{1}{V} = m C_1 + b$, and ditto for a fixed value of C_1 . Again, we must also recall that this function will *not* represent the peak value of the noise due to I_2 at N_1 . This will represent the value of the noise due to I_2 at N_1 *at the time at which a node with a single current source, and capacitance C_1 , would reach its peak noise.*

In order to find values for these parameters, we must perform a several simulations, and then plot the best-fitting curve to extract values for m_1 , m_2 , m_3 and b . (Recall that there will be separate values for these four numbers at each relative position from the current source; so if the current source is located at $(0, 0)$ we would have values for these four variables at $(0, 1)$, $(1, 1)$, $(0, 2)$ and so on.) So for instance, imagine the maximum number of decaps that could be placed at a point was 200. Imagine further that we were trying to find values for m_1 , m_2 , m_3 and b at a node of a distance $(0, 2)$ from the current source. Then, we would precharacterize as follows. First, we would generate a mapping between amounts of decap, and times of the peak at the current source. This is done to determine what times we will take voltage data to conform to the peak time approximation. Second, we would actually measure the voltages with various combinations of decoupling capacitance at the two nodes. These two steps are shown in Table 5.1. Third, we would take all of the data, and find the best fitting m_1 , m_2 , m_3 and b in the function $V_{N_1, I_2}(C_1, C_2) = \frac{1}{m_1 C_1 + m_2 C_2 + m_3 C_1 C_2 + b}$.

The alert reader will note that we used capacitance values significantly greater than 200, despite the fact that we cannot place more than 200 decaps at any specific point. This feature will be explained somewhat later; it is due to the necessity of transforming arbitrary decap layouts into “equivalent” decap layouts where

Measurement number	CS location	Measurement	Decaps at CS	Decaps at measurement	Value we are measuring
1	(0,0)	(0,0)	0	N/A	Time of peak (call it t_1)
2	(0,0)	(0,0)	100	N/A	Time of peak (call it t_2)
3	(0,0)	(0,0)	200	N/A	Time of peak (call it t_3)
4	(0,0)	(0,0)	300	N/A	Time of peak (call it t_4)
5	(0,0)	(0,0)	400	N/A	Time of peak (call it t_5)
6	(0,0)	(0,2)	0	0	Value of noise at time t_1
7	(0,0)	(0,2)	0	100	Value of noise at time t_2
8	(0,0)	(0,2)	0	200	Value of noise at time t_3
9	(0,0)	(0,2)	0	300	Value of noise at time t_4
10	(0,0)	(0,2)	0	400	Value of noise at time t_5
11	(0,0)	(0,2)	100	0	Value of noise at time t_1
12	(0,0)	(0,2)	100	100	Value of noise at time t_2
13	(0,0)	(0,2)	100	200	Value of noise at time t_3
14	(0,0)	(0,2)	100	300	Value of noise at time t_4
15	(0,0)	(0,2)	100	400	Value of noise at time t_5
16	(0,0)	(0,2)	200	0	Value of noise at time t_1
17	(0,0)	(0,2)	200	100	Value of noise at time t_2
18	(0,0)	(0,2)	200	200	Value of noise at time t_3
19	(0,0)	(0,2)	200	300	Value of noise at time t_4
20	(0,0)	(0,2)	200	400	Value of noise at time t_5
21	(0,0)	(0,2)	300	0	Value of noise at time t_1
22	(0,0)	(0,2)	300	100	Value of noise at time t_2
23	(0,0)	(0,2)	300	200	Value of noise at time t_3
24	(0,0)	(0,2)	300	300	Value of noise at time t_4
25	(0,0)	(0,2)	300	400	Value of noise at time t_5
26	(0,0)	(0,2)	400	0	Value of noise at time t_1
27	(0,0)	(0,2)	400	100	Value of noise at time t_2
28	(0,0)	(0,2)	400	200	Value of noise at time t_3
29	(0,0)	(0,2)	400	300	Value of noise at time t_4
30	(0,0)	(0,2)	400	400	Value of noise at time t_5

Table 5.1: Precharacterizing the node at the (0,2) coordinate

decap is placed at only two locations.

Measurements 5-30 gave us 25 (C_1, C_2, V) combinations. We can insert them into a least squares curve fitter and try to fit them to the function $V_{N_1, J_2}(C_1, C_2) = \frac{1}{m_1 C_1 + m_2 C_2 + m_3 C_1 C_2 + b}$. The resultant values of m_1, m_2, m_3 and b can accordingly be extracted; these values are specific to the coordinate $(0, 2)$. This measurement must be repeated for other positions.

All in all, this seems like a forbidding number of simulations to do, especially considering that we are repeating them for each coordinate. However, we can reduce the number of simulations in two important ways.

1. Recall the symmetry property discussed earlier. Any measurement with C_1 decaps at the current source and C_2 decaps at the measurement source will produce the same curve as one with C_2 decaps at the current source and C_1 decaps at the measurement source. Therefore, for instance, instead of performing both simulations 13 and 17, we can perform exclusively simulation 13 and measure the data at both t_2 and t_3 . This cuts down the number of simulations by half.
2. Any measurement done with zero decaps at one of the nodes (i.e. measurements 6-11, 16, 21, 26) is equivalent to a measurement performed in the naive precharacterization process (where we just placed a certain number of decaps at the current source and simulated the voltage at all nearby nodes). Recall that in the naive precharacterization process, a single simulation could yield (C, V) values for any coordinate in the vicinity. Therefore, we can obtain measurements 6-10 (and implicitly 11, 16, 21, and 26, because of the symmetry condition discussed above) for every coordinate using only a single simulation, instead of distinct simulations for every coordinate.

Using these extracted values, we can now determine the noise at a measurement point due to a distant current source. Again, remember our assumptions were as follows:

1. The two current sources were sufficiently far away that the decap at one current source did not substantially affect the local noise due to the other current source.
2. The only decap on the grid is the decap at the current source and the decap at the measurement point.

3. All the current source spikes are of the same width.
4. There is also a current source at the measurement point (so the peak time approximation is appropriate).

We can also determine the noise at a measurement point due to several current sources, provided that they are sufficiently spread out that the decap at each current source does not significantly affect the noise at any other decap. We can do so by superimposing the noise due to each individual current source.

Our current task is to eliminate the assumptions above. Let's start with assumption 1.

5.5 Calculating the local noise

This section will discuss determining the noise at a node due to the local current source, as a function of nearby decaps laid out arbitrarily. This is very important to do accurately; the peak time approximation is based on the premise that most of the noise is due to the local current source, so we would presumably want to measure this value accurately.

Let's start with a narrower sub-problem. Consider a chip with two current sources I_1 and I_2 at nodes N_1 and N_2 with decap C_1 and C_2 . This time, we will *not* assume that they are sufficiently far away from each other that the decap at N_2 will not affect the noise due to I_1 at N_1 . Let us say we want to estimate the maximum noise at node N_1 (without loss of generality). The noise at node N_1 will be the sum of the following two functions: a function $V_{N_1, I_1}(C_1, C_2)$ (i.e. the voltage at node N_1 due to the current source I_1 , as a function of the capacitance at N_1 and the capacitance at N_2) and a function $V_{N_1, I_2}(C_1, C_2)$. We saw above how to calculate $V_{N_1, I_2}(C_1, C_2)$. It turns out that $V_{N_1, I_1}(C_1, C_2)$ must be calculated through precharacterization as well. We can construct a new function, again of the form $V_{N_1, I_1}(C_1, C_2) = \frac{1}{m_1 C_1 + m_2 C_2 + m_3 C_1 C_2 + b}$. This time, C_1 will be the amount of decap at the local node, and C_2 will be the amount of decap at some nearby node. Again, we will need to find individual parameters for each coordinate value. This means that at (for instance) coordinate $(0, 2)$ we need m_1, m_2, m_3 and b values for non-local current sources (where C_1 is the amount of decap at the measurement point, and C_2 is the amount of noise at the current source, which is

Measurement number	Current source location	Measurement location	Decaps at current source	Decaps at measurement location	Value we are measuring
1	(0,0)	(0,0)	0	N/A	Time of peak (call it t_1)
2	(0,0)	(0,0)	100	N/A	Time of peak (call it t_2)
3	(0,0)	(0,0)	200	N/A	Time of peak (call it t_3)
4	(0,0)	(0,0)	300	N/A	Time of peak (call it t_4)
5	(0,0)	(0,0)	400	N/A	Time of peak (call it t_5)
6	(0,0)	(0,2) and (0,0)	0	0	Value of noise at (0,0) at peak and value of noise at (0,2) at time t_1
7	(0,0)	(0,2) and (0,0)	0	100	Value of noise at (0,0) at peak and value of noise at (0,2) at t_1 and t_2
8	(0,0)	(0,2) and (0,0)	0	200	Value of noise at (0,0) at peak and value of noise at (0,2) at t_1 and t_3
9	(0,0)	(0,2) and (0,0)	0	300	Value of noise at (0,0) at peak and value of noise at (0,2) at t_1 and t_4
10	(0,0)	(0,2) and (0,0)	0	400	Value of noise at (0,0) at peak and value of noise at (0,2) at t_1 and t_5
11	(0,0)	(0,2) and (0,0)	100	0	Value of noise at (0,0) at peak
12	(0,0)	(0,2) and (0,0)	100	100	Value of noise at (0,0) at peak and value of noise at (0,2) at t_2
13	(0,0)	(0,2) and (0,0)	100	200	Value of noise at (0,0) at peak and value of noise at (0,2) at t_2 and t_3
14	(0,0)	(0,2) and (0,0)	100	300	Value of noise at (0,0) at peak and value of noise at (0,2) at t_2 and t_4
15	(0,0)	(0,2) and (0,0)	100	400	Value of noise at (0,0) at peak and value of noise at (0,2) at t_2 and t_5

Table 5.2: Precharacterizing the local node and the node at the (0,2) coordinate

by assumption located at (0,0) - two nodes away). and separate m_1, m_2, m_3 and b values for local current sources. Fortunately, we can reuse the simulations performed in the other precharacterizations and just take more measurements. We can do this as seen in Table 5.2 and Table 5.3.

Note what we are doing here. First, we generate a relation between the amount of decap, and the time of the peak. Next, we use this relation to measure the noise at various times as a function of different amounts of decap at the noise source and the measurement node. This creates a set of C_1, C_2, V ordered pairs which we can plug into $V_{N_1, I_1}(C_1, C_2) = \frac{1}{m_1 C_1 + m_2 C_2 + m_3 C_1 C_2 + b}$, and a separate set of C_1, C_2, V ordered pairs which we can plug into $t_{V_{N_1, I_2}}(C_1, C_2) = \frac{1}{m_1 C_1 + m_2 C_2 + m_3 C_1 C_2 + b}$ to find the best fits for m_1, m_2, m_3 , and b for both functions.

Measurement number	Current source location	Measurement location	Decaps at current source	Decaps at measurement location	Value we are measuring
16	(0, 0)	(0, 2) and (0, 0)	200	0	Value of noise at (0,0) at peak
17	(0, 0)	(0, 2) and (0, 0)	200	100	Value of noise at (0,0) at peak
18	(0, 0)	(0, 2) and (0, 0)	200	200	Value of noise at (0,0) at peak and value of noise at (0,2) at t_3
19	(0, 0)	(0, 2) and (0, 0)	200	300	Value of noise at (0,0) at peak and value of noise at (0,2) at t_3 and t_4
20	(0, 0)	(0, 2) and (0, 0)	200	400	Value of noise at (0,0) at peak and value of noise at (0,2) at t_3 and t_5
21	(0, 0)	(0, 2) and (0, 0)	300	0	Value of noise at (0,0) at peak
22	(0, 0)	(0, 2) and (0, 0)	300	100	Value of noise at (0,0) at peak
23	(0, 0)	(0, 2) and (0, 0)	300	200	Value of noise at (0,0) at peak
24	(0, 0)	(0, 2) and (0, 0)	300	300	Value of noise at (0,0) at peak and value of noise at (0,2) at t_4
25	(0, 0)	(0, 2) and (0, 0)	300	400	Value of noise at (0,0) at peak and value of noise at (0,2) at t_4 and t_5
26	(0, 0)	(0, 2) and (0, 0)	400	0	Value of noise at (0,0) at peak
27	(0, 0)	(0, 2) and (0, 0)	400	100	Value of noise at (0,0) at peak
28	(0, 0)	(0, 2) and (0, 0)	400	200	Value of noise at (0,0) at peak
29	(0, 0)	(0, 2) and (0, 0)	400	300	Value of noise at (0,0) at peak
30	(0, 0)	(0, 2) and (0, 0)	400	400	Value of noise at (0,0) at peak and value of noise at (0,2) at t_5

Table 5.3: Precharacterizing the local node and the node at the (0,2) coordinate (continued)

The relevant features of this set of measurements are as follows.

- In simulation 6 and other simulations, we can take three measurements. Two can be used at the measurement point, and one at the current source point.
- However, even though the symmetry property may hold at the measurement point, it certainly does not hold at the current source point. Therefore, even though simulations 6 and 11 may be redundant if we are exclusively measuring at the measurement point, they must be performed separately to get different measurements at the current source point. Similarly, even though simulations 6-10 can be done using the original precharacterization method for the measurement point, they can certainly not be done using the original precharacterization method for the current source point. This would seem to eliminate all of the optimizations that we saw above. The only way to preserve these optimizations is to simply eliminate measurements in which only the peak noise at the source is measured - that is, only perform measurements in which three measurements can be taken. This will result in half the number of points being fit to the local parameters noise function as opposed to the non-local parameters noise function. In practice, the results obtained doing this were still tolerable.

Using the measurements from above at the current source, we can now extract parameters to fit to the function $V_{N_1, I_1}(C_1, C_2) = \frac{1}{m_1 C_1 + m_2 C_2 + m_3 C_1 C_2 + b}$. If we generate these parameters for several nearby coordinate values, we can determine the noise at a current source node due to the presence of decap at the noise source, and a single nearby location.

Of course, we want to be able to determine the noise at the current source due to an arbitrary layout of decaps. We can approximate it using the following algorithm:

1. Determine the noise at the current source assuming there is only decap at the current source (i.e. no surrounding decap). Call this V_0 .
2. For each nearby node where there exists decap, calculate the amount of noise there would be at the current source if there were only decap at that point and at the current source. If there are i nearby nodes with decap, we will generate i different noise values. Call each of these values V_i .

3. Note that each V_i is less than V_0 . Calculate $V_0 - V_i$ for each i , and then add up all of these differences.

Call the result $V_{sum-of-differences}$.

4. Find $V_0 - V_{sum-of-differences}$. This is our estimate for the noise at the current source.

The explanation for this algorithm is as follows. Each set of decaps at each individual node reduces the noise to a certain extent. The above algorithm linearly sums all of these reductions of the noise. Now, this is only an approximation; it underestimates the noise, since the marginal benefit of adding capacitors at a given node decreases when there are already capacitors at other nodes. However, it is the best way to take into account the contributions of the decap at each nearby node.

So we have now fully solved a very significant subproblem. Assuming a current source at node N and any arbitrary layout of decaps in other locations, we can determine the noise at node N due to the local current source. We now need to determine the noise at other nodes. Remember that we already know how to figure out the noise at a measurement point as long as the following three remaining assumptions hold:

1. The only decap on the grid is the decap at the current source and the decap at the measurement point.
2. All the current source spikes are of the same width.
3. There is also a current source at the measurement point (so the peak time approximation is appropriate).

Let us now relax assumption 1.

5.6 Calculating the non-local noise

Consider arbitrary layouts of decoupling capacitors on the chip when finding the non-local noise. Our strategy will be to reduce these arbitrary layouts to equivalent problems in which there is exclusively decap at the current source and the measurement point.

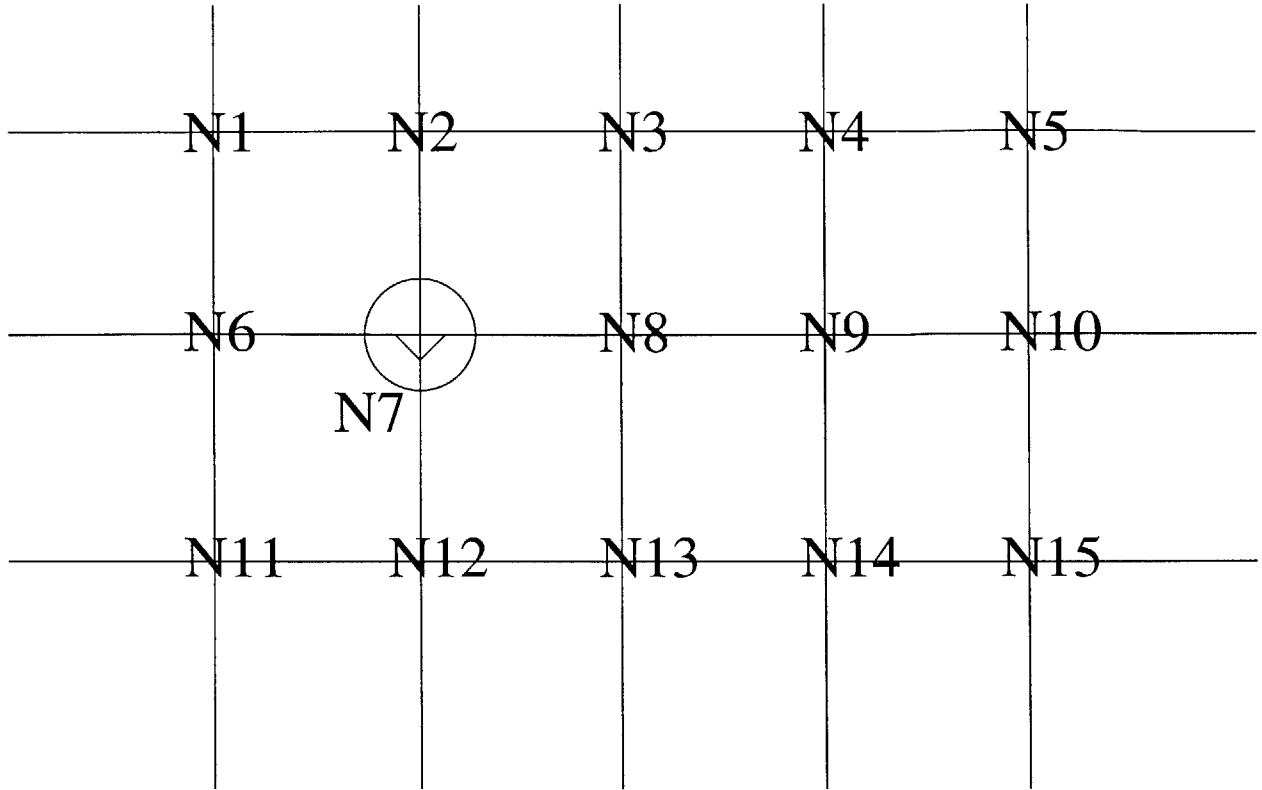


Figure 5.7: Measurement point at $(2,0)$ coordinate

5.6.1 Measurement point at $(2,0)$ coordinate

Consider the layout in Figure 5.7. (Any decap further away than these decaps will be ignored due to their comparatively small effect). Imagine we are measuring the noise at node N_9 . Let there be C_i decap at node N_i . Imagine therefore that the current source has C_7 decap and the measurement point has C_9 decap. We want to reduce this into an equivalent problem in which there is only decap at N_7 and N_9 . To do this, we need to do two things.

Line of sight decap

We must deal with decap directly in the line of sight. In this case, N_8 is directly in the line of sight. We saw earlier that decap in this position did not quite have the effect of decap at the current source node or at the measurement point, but it had a similar effect. Therefore, we will distribute this decap evenly among the current source node and the measurement point. We can define a weight constant meant to capture the

impact of line of sight impact (I generally set this constant to 1). We will make the following transformations:

$$C_7 \rightarrow C_7 + \frac{1}{2} * weight * C_8$$

$$C_9 \rightarrow C_9 + \frac{1}{2} * weight * C_8$$

$$C_8 \rightarrow 0$$

Non-line of sight decap

Now, we must deal with all of the other decap, outside of the line of sight. We will deal with it by the familiar method of precharacterization. Note that due to symmetry, there are really only seven different positions: N_1 - N_6 and N_{10} . (N_1 and N_{11} , N_2 and N_{12} , and so forth, are symmetrical. N_2 and N_4 are very similar, as are the N_6 and N_{10} pair and the N_1 and N_5 pair; they are not identical, because there are some non-linear effects when a decap is outside of the line of sight.) Here is an example of such a precharacterization to figure out the equivalent effect of N_4 .

1. Find the total noise at N_9 by placing C_{init} decaps at N_9 , where C_{init} is the maximum amount of decap that can fit at that node.
2. Place $\frac{C_{init}}{2}$ decap at N_9 . Determine, though trial and error, how much decap is necessary to be placed at N_4 so that the noise is equivalent to the noise in the previous step. If C_{init} decaps are necessary at N_4 , for instance, we can say that a decap at N_4 is worth half a decap at N_9 . Then we can say that $k_4 = \frac{1}{2}$.

We can repeat these steps to find out “equivalent decap values” or k_i values for the various nearby nodes, except for the nodes closer to the current source (such as N_1 and N_2), we’d find the equivalent amount of decap at the current source. For N_3 , we could choose arbitrarily whether to find the equivalent value at the measurement point or the current source.

We can thus make the following transformations:

$$C_7 \rightarrow C_7 + \frac{1}{2} * weight * C_8 + k_1 N_1 + k_2 N_2 + \frac{k_3 N_3}{2} + k_6 N_6 + k_1 N_{11} + k_2 N_{12} + \frac{k_3 N_{13}}{2}$$

$$C_9 \rightarrow C_9 + \frac{1}{2} * weight * C_8 + \frac{k_3 N_3}{2} + k_4 N_4 + k_5 N_5 + k_{10} N_{10} + k_4 N_{14} + k_5 N_{15} + \frac{k_3 N_{13}}{2}$$

$$C_i \text{ for } i \neq 7, 9 \rightarrow 0$$

Let's consider a slightly more complex example, where the current source and the decap are not in the same row.

5.6.2 Measurement point at (2,1) coordinate

Line of sight decap

Consider the setup in Figure 5.8. Assume our measurement point is at N_{14} . In this case, N_7 , N_8 , N_9 , N_{12} , N_{13} , and N_{14} are all in the line of sight. We want to group N_8 and N_{12} with N_7 , and N_9 and N_{13} with N_{14} . Consider the three shortest paths from N_7 to N_{14} . One goes through N_8 and N_9 , one goes through N_8 and N_{13} , and one goes through N_{12} and N_{13} . We would thus do

$$C_7 \rightarrow C_7 + \frac{1}{3} * weight * C_{12} + \frac{2}{3} * weight * C_8$$

since $\frac{1}{3}$ of the shortest paths go through N_{12} and $\frac{2}{3}$ of the shortest paths go through N_8 . We would similarly do

$$C_{14} \rightarrow C_{14} + \frac{1}{3} * weight * C_9 + \frac{2}{3} * weight * C_{13}$$

$$C_8, C_9, C_{12}, C_{13} \rightarrow 0$$

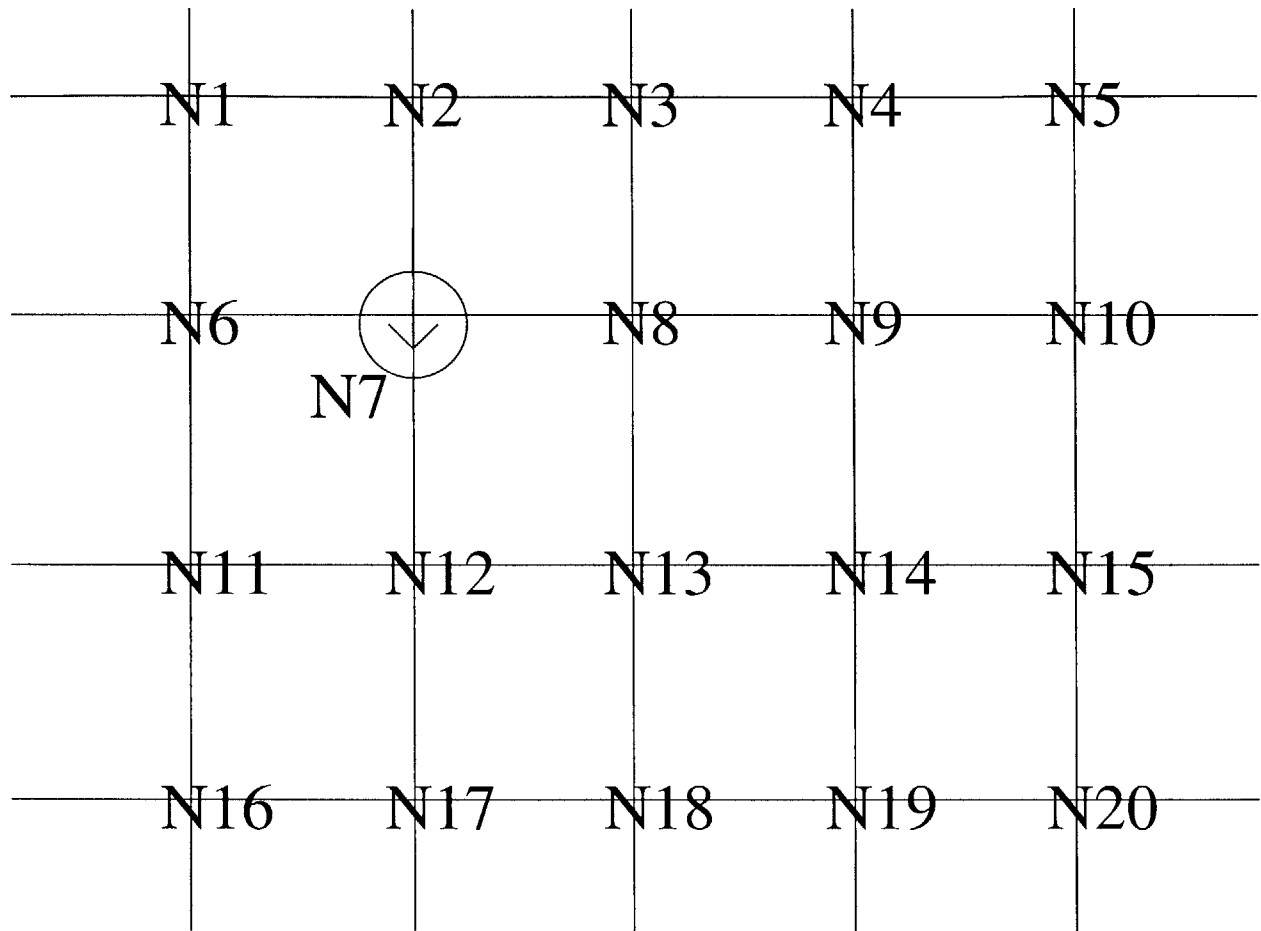


Figure 5.8: Measurement point at (2,1) coordinate

Non-line of sight decap

Next, we must take into account the other decaps. To save time, we can throw out the decaps that aren't within one vertical, horizontal, or diagonal node of N_7 or N_{14} . Thus, we can throw out the decaps at N_{16} , N_{17} , N_4 , and N_5 . Technically, we must then precharacterize the effects of decap at N_1 , N_2 , N_3 , N_6 , N_{11} , N_{15} , N_{18} , and N_{20} . If one wants to aspire to extreme accuracy, one can do this; if not, one can assume that (for instance) all nodes one diagonal unit of distance from the current source or measurement point produce roughly similar results, thus making N_1 , N_{11} , N_{15} , and N_{19} equal. Either way, we again do

$$C_7 \rightarrow C_7 + \frac{1}{3} * weight * C_{12} + \frac{2}{3} * weight * C_8 + C_{non-local-equiv}$$

$$C_{14} \rightarrow C_{14} + \frac{1}{3} * weight * C_9 + \frac{2}{3} * weight * C_{13} + C_{non-local-equiv}$$

$$C_i \text{ for } i \neq 7, 14 \rightarrow 0$$

In this sense, we have transformed the problem to one involving only decaps at the noise source and the measurement point. We can then use the precharacterized parameters from earlier to calculate the noise.

5.6.3 General methodology

These steps proceed roughly equivalently for other relative coordinate positions. I fully precharacterized the effects of decaps at four different relative coordinates: (0, 1), (0, 2), (1, 1) and (1, 2). However, the algorithm can generally accommodate any coordinate position, by doing the following series of steps:

1. For decaps in the line of sight, determine how many of the shortest paths from current source to measurement point go through that node as a fraction of the total number of shortest paths. Multiply the number of decaps at that node by this fraction, and add it to the closer of the measurement point and the current source node. If it is equidistant between the two (as in the first example above), split

the decap halfway.

2. For all relatively close non-line of sight nodes to either the current source and the measurement point, precharacterize the effects of adding initial decaps. Determine the “effective value” of individual decaps at these positions, and add them to the closer of the current source node or the measurement point. If they are equidistant between the two, split the decap halfway.
3. Now, we have an equivalent problem involving only decaps at the current source node and the measurement point. Use the precharacterized values to determine the noise.

It is now clear why, in the initial precharacterization step, we precharacterized using capacitor totals greater than the total decaps that could fit at that node. Even if only 200 decaps can fit at node N_7 , a value of 400 decaps might be being used in the “transformed” problem, so we need to have simulated data in that range.

5.7 Relaxing assumptions

Using the above methodology, we have solved another significant subproblem. We can calculate either the local or the non-local noise at any point, with any arbitrary layout of decaps on the grid. However, just by superimposing, we can determine the noise for any arbitrary layout of current sources. Therefore, we have solved the general problem. Recall that there remain two assumptions:

1. All the current source spikes are of the same width.
2. There is also a current source at the measurement point (so the peak time approximation is appropriate).

Neither of these assumptions can be easily resolved. We can modify the precharacterization to deal with the first assumption. Imagine that we were using two different spike widths: width W_1 and width W_2 . For each coordinate, we would have to make four distinct data tables:

1. A data table for current sources of width W_1 , assuming that there is a current source of width W_1 at the measurement point
2. A data table for current sources of width W_1 , assuming that there is a current source of width W_2 at the measurement point
3. A data table for current sources of width W_2 , assuming that there is a current source of width W_1 at the measurement point
4. A data table for current sources of width W_2 , assuming that there is a current source of width W_2 at the measurement point

Unfortunately, for n different current widths we'd need to do n^2 tables per coordinate, and for arbitrary current widths it's hopeless.

Similarly, the second assumption cannot be easily resolved either - the peak time approximation depends on there being a current source at the measurement point. We discussed earlier that this is a reasonable assumption, since most nodes without current sources can count on being within tolerable limits as long as the surrounding nodes with current sources are within tolerable limits. However, if certain nodes are less tolerant to noise than other nodes, that assumption goes away; we might need precise noise measurements at nodes without a current source. If so, the entire above algorithm breaks down.

The algorithm also suffers the weakness of inaccuracy. It is only as good as the peak time approximation, and as the figures above suggest, the peak time approximation is far from perfect. To get around all of these problems, we could use a noise calculator that finds the noise at any given point - i.e. that precharacterizes the entire voltage curve. Such a noise calculator is extremely slow and cannot be used in an iterative algorithm that does thousands of noise calculations to assign decaps, but it can be used to refine approximations considerably. This noise calculator is described in the next section.

5.8 Precharacterizing the entire curve

The full-curve precharacterization is based on three fundamental assumptions.

1. It is relatively easy to find the noise peak of a curve.
2. It is relatively easy to find the time of the noise peak of a curve.
3. With these two facts, we have already almost solved the problem.

Let's start with the first item. To calculate the peak noise at the local node of a current source, we can use the method described in the last section. (Since we are precharacterizing at the time of the peak due to the local current source, the local value for the noise that's used is going to be the peak noise due to the local current source.) To calculate the peak noise at the non-local node of a current source, recall that we described much earlier that the peak voltage is generally given by a function of the form $\frac{1}{V} = mC + b$. The initial, naive precharacterization method determined values for m and b for each coordinate. Now, we also saw that the double node effect certainly affected the time of the peak, but it doesn't much affect the magnitude of the peak. Therefore, to find the peak of a noise curve at a non-local node, we need to first transform an arbitrary decap layout into a layout with decaps only in the current source and the measurement point (as described in the last section). Then, it is simple a matter of calculating $\frac{1}{V} = m(C_{current-source} + C_{measurement-point}) + b$ to find the peak voltage.

Calculating the peak time is also relatively simple. Consider first the peak time due to a current source at the local node. The peak time monotonically increases as a function of decap; it doesn't increase as a particularly attractive function, but one can take (decap, peak time) pairs and linearly interpolate between them. It also takes time for noise to travel from closer nodes to further away nodes. We can precharacterize the time between the peak noise at the local node, and the peak noise at each nearby non-local coordinate. Thus, in order to calculate the time of the peak noise at any node, we must calculate the time of the peak at the local node (using linear interpolation), and add the time corresponding to the distance between the current source and the measurement point. Finally, we must take into account the double node effect. This is done by multiplying the product of $C_{current-source}$ and $C_{measurement-point}$ by a term different for each coordinate, determined by eyeballing simulations, and adding it to the time calculated assuming no double node effect.

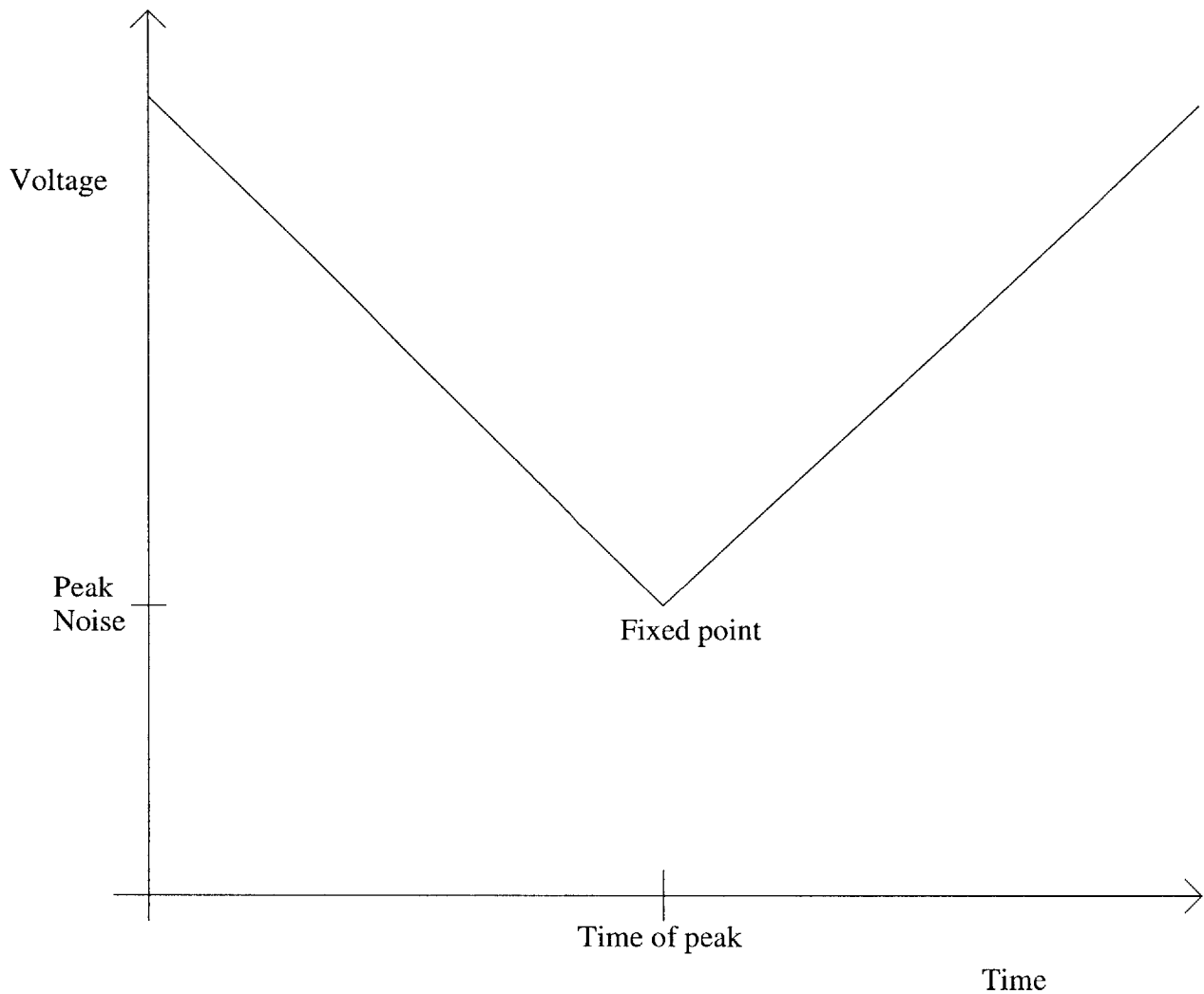


Figure 5.9: Approximate curve given fixed point

Thus, without much difficulty, we can pinpoint both the time and magnitude of the peak. But then, we know that the curve will look roughly as in Figure 5.9.

We need to refine this curve. The method for doing so was as follows:

1. Construct a series of curves, all of which have the same peak time and same peak noise, but which have different amounts of decap. To do this, imagine a node which has a peak of time t with N decaps. Imagine we want the same peak time and peak magnitude with $N - m$ decaps. Since decreasing the number of decaps tends to make the peak happen earlier, we would have to widen the spike. Since widening the spike would cause the peak noise to go down, we'd also have to make the peak of the

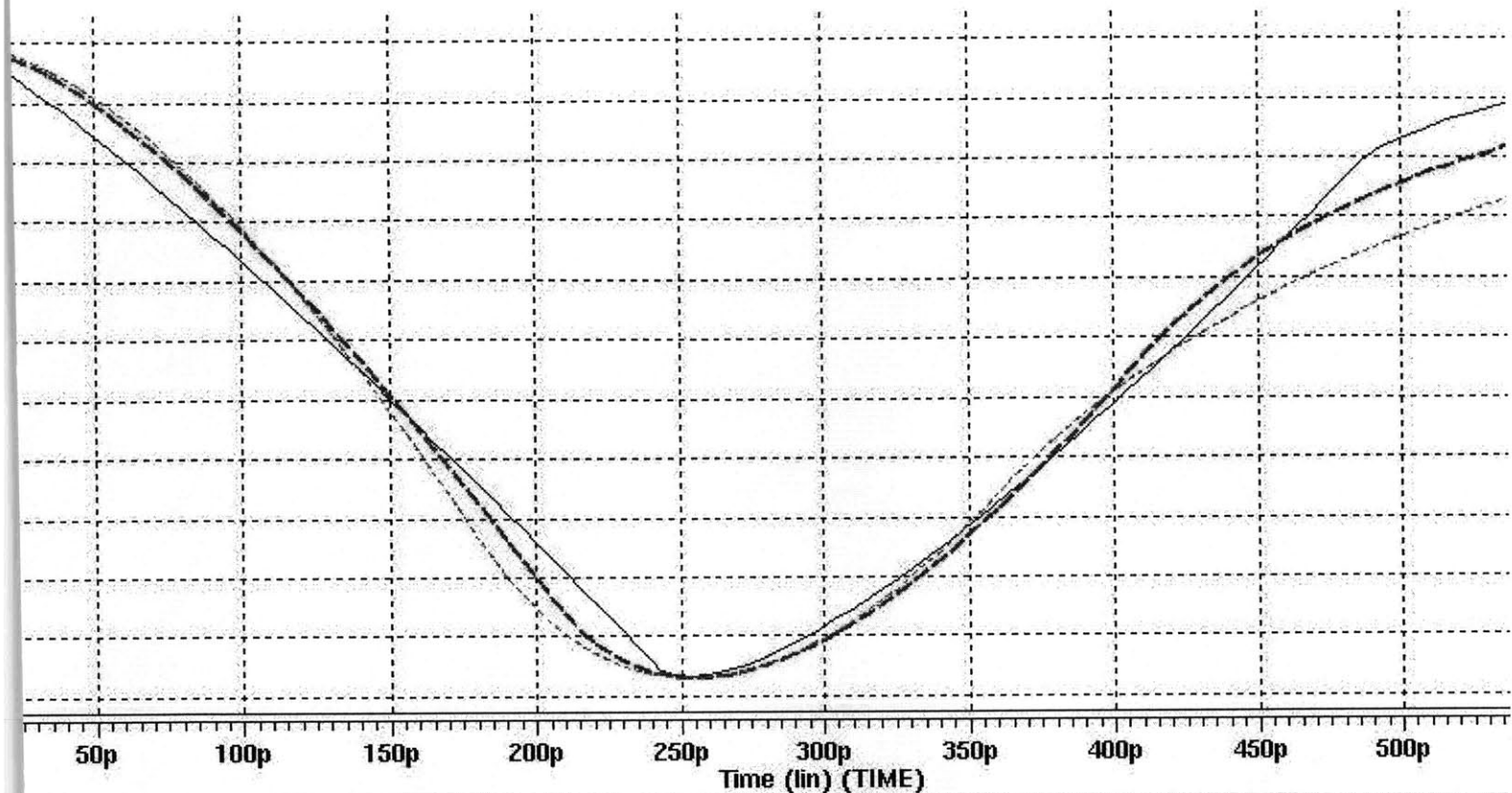


Figure 5.10: 0, 100, and 200 decaps with identical peak times and voltages

pulse higher. The example in Figure 5.10 have 0, 100, and 200 decaps at the current source node.

2. Eyeball these curves to try to locate different parameters that could capture their relevant differences, then find these parameters as a function of decap, coordinate, and pulse width.

Purely upon observation, it turned out that the curves had the following characteristics.

1. From the initial time, to 20% of the peak, we have a curvy line between 0 and some proportion of the peak noise, which is a function of coordinate, width, and decap. A “curvy” line is defined as a straight line, plus a rotated sine curve, the amplitude of which is also a function of coordinate, width, and decap.
2. Connect a hypothetical straight line from the noise at 20% of the peak to the peak. Take the portion of that line that runs from 20% of the peak to 57% of the peak. The curvy version of this line runs

from 20% of the peak to 57% of the peak.

3. From $t = 57\%$ of peak to $t = t_s$ of peak, where t_s is determined by a constant which is a function of coordinate, width, and decap, we have a straight line. This straight line will be a slightly rotated version of the straight line connecting the noise at 57% of the peak to the peak; the amount of rotation is a function of coordinate, width, and decap.
4. Determine the best fitting parabola connecting the noise at t_s and the peak noise, assuming that the peak noise is the vertex of the parabola. This can be done by inverting a 3x3 matrix. This parabola represents the noise up until the peak time.
5. Let t_{p1} and t_{p2} be some time determined as a function of width, decap, and coordinate. Let r be some parameter which is a function of width, decap, and coordinate. Let t_{150} be the time representing 150% of the time of the peak noise. Assume the noise is constant from the peak time to t_{p1} . Then, imagine a straight line between t_{p1}, v_{peak} and $t_{150}, r * v_{peak}$. Find the noise value at t_{p2} . Determine the best fitting parabola connecting the noise at t_{p2} and the peak noise, assuming that the peak noise is the vertex of the parabola. This can be done by inverting a 3x3 matrix. This parabola represents the noise up until t_{p2} .
6. Use the line imagined in the previous step to connect $(t_{p2}, v(t_{p2}))$ and $(t_{150}, r * v_{peak})$.
7. From t_{150} to t_{195} (where t_{195} is 195% of the peak), connect a curvy line whose angle with the above line is a function of width, decap, and coordinate
8. From t_{195} to t_{250} (where t_{250} is 250% of the peak), connect a curvy line whose angle with the above line is a function of width, decap, and coordinate
9. From t_{250} to t_{end} , connect a curvy line whose angle with the above line is a function of width, decap, and coordinate

This seems like an ad-hoc collection of characteristics, and indeed it is, but I found that it could capture the full range of curves at different coordinates and decap values quite well. The benefit of this scheme is that

the amount of precharacterization is extremely limited. This method hearkens back to the original, naive algorithm. We need to perform the following steps:

1. Place a single current source somewhere in the middle of the chip. Add 100, 200, N decaps to it, and repeatedly simulate. (N is not the total amount of decap that can fit at a node; it's the total amount of decap that might be used in the calculations at that node after the decaps on the chip are rearranged using the equivalency method discussed above.) Collect data from all nodes in the vicinity. Repeat at various widths. (For simplicity, I performed the full precharacterization at only a single width.)
2. Eyeball the resultant noise curves, altering the parameters discussed above until the curve looks like the simulated curves.
3. Perform some simulations involving the double node effect to get approximate values for the fudge factor in determining the peak time.

Having generated these parameters, it is a simple matter to construct the curves to simulate the noise.

We use the following method:

1. For each current source, transform the problem involving an arbitrary layout of decaps into a problem involving only decaps at the noise source and at the measurement point.
2. Find the resultant noise curve using the above algorithm and store it.
3. After all the noise curves have been generated, sum them all.
4. Search this summed curve for the maximum noise.

This method can deal with continuous values of curve width, and makes no assumptions about the presence of a current source at a node. Accordingly, it is capable of handling any arbitrary allocation of current sources and decaps. There is also no fundamental restriction on the accuracy due to the peak time approximation.

The only problem is that it is resource intensive. The process of generating “curvy lines” is quite complicated; we must construct a sine curve, rotate all of the points using a rotation matrix, and then

translate the new (x, y) coordinate pairs where x is an arbitrary real number to (x, y) coordinate pairs where x is an integer. The process of generating two parabolas requires inverting 3x3 matrices. In practice, it takes a few seconds to generate this curve, and if we need to repeat this process thousands of times, the time requirements are impracticable. Therefore, this method will be used in the next section to refine more coarse estimates derived using the peak time approximation.

Some comments are in order about the practicality of using the precharacterization method. Intuitively, it seems like an extremely tedious process. In practice, it turns out not to be impractical, for the following reasons.

1. These simulations need to be performed once for a given chip technology. As long as the on-chip resistance and capacitance does not significantly change, the method can be reused for numerous chips.
2. These simulations involve only one current source, and therefore are unlikely to be resource-intensive.
3. Only current sources in relatively close proximity to the measurement point produce significant amounts of noise. I used the coarse method of only precharacterizing nodes of coordinates $(0, 1)$, $(0, 2)$, $(1, 1)$, and $(1, 2)$ from the noise source, and ignoring all other nodes. This proved to produce reasonable estimates, as will be discussed in the next section. Alternately, we can assume that noise rolls off as $\frac{1}{d^2}$ (where d is the distance from the noise source) and make approximations that way for distant noise sources.

5.9 Edge effects

Near the edge of a chip, there is less local capacitance to supply charge to macros. Accordingly, noise voltages increase significantly. This effect increases near the corner of the chip to an even greater extent. In theory, additional simulations must be done to take these effects into account.

However, there need not be a large number of additional simulations. Consider the scenarios in Table 5.4. If we weren't near the edge, all of these would produce identical outcomes. Near the edge, we might

Measurement number	Current source location	Decoupling Capacitor Location	Measurement location
1	One node from edge	One node from edge	Two nodes from edge
2	One node from edge	Two nodes from edge	Two nodes from edge
3	Two nodes from edge	One node from edge	One node from edge
4	Two nodes from edge	Two nodes from edge	One node from edge

Table 5.4: Tests of edge effects

expect them to produce different outcomes. The question is, to what extent is that true?

The answer is that measurements 1 and 3 will produce the same curve, and measurements 2 and 4 will produce the same curve. The relevant factor is the location of the decoupling capacitance, not the location of the current source. We can explain this observation in physical terms. As the noise propagates near the edge, the noise is reduced at each node as a function of the total capacitance it sees at that node. Near the edge, different nodes have different amounts of background capacitance. Therefore, consider a current source that results in noise N . Imagine if the background capacitance one node from the edge will result in the noise attenuating by a factor k_1 . Imagine if adding decap will cause it to attenuate by a factor $k_1 + c_1$. Similarly, imagine if the background capacitance two nodes from the edge will result in the noise attenuating by a factor k_2 , and that adding decap will cause it to attenuate by a factor $k_2 + c_2$. Then, in case 1, we have:

$$N \rightarrow \frac{\frac{N}{k_1 + c_1}}{k_2}$$

In case 3, we have

$$N \rightarrow \frac{\frac{N}{k_2}}{k_1 + c_1}$$

Both of which amount to

$$N \rightarrow \frac{N}{(k_1 + c_1) * k_2}$$

Meanwhile, in case 2 we have

$$N \rightarrow \frac{\frac{N}{k_1}}{k_2 + c_2}$$

and in case 4 we have

$$N \rightarrow \frac{N}{\frac{k_2 + c_2}{k_1}}$$

Both of which amount to

$$N \rightarrow \frac{N}{(k_2 + c_2) * k_1}$$

Therefore, to precharacterize edge effects, we need to only perform sets of simulations altering the location of decap near the edge, and generate alternate sets of parameters. (These additional simulations, for reasons of time constraints, did not make it into this thesis.)

Chapter 6

Decap allocation algorithm

6.1 High-level picture

Using decoupling capacitors is the most effective way to reduce noise on chip. It is by no means the only way; there is considerable literature on other methods of solving the noise problem. [9] discusses changing pin assignments to reduce noise; [7] discusses adjusting the slew rate of the signal being outputted; and [12] describes a method of optimizing the location of output pads. However, the use of decaps is the most frequently cited method of reducing noise. In this section, I will describe the decap allocation algorithm I designed.

The decap algorithm operates according to a relatively simple methodology. It initially assumes that the chip has no decap on it whatsoever. It finds the “worst” node, or the node with the greatest amount of noise, and adds a small amount of decap to that node. These steps are repeated until every node on the chip is either (a) within tolerable noise limits, or (b) has had its vicinity fully populated by decoupling capacitors. At this point, the algorithm will stop, and output the number of capacitors at each node, and the amount of remaining noise at each node.

More formally, the algorithm takes the following inputs:

- The x-y coordinate of each node

- The magnitude (in milliamperes) and duration (in nanoseconds) of the triangular pulse at that node
- The tolerable noise threshold at that node
- The number of decaps that can fit at that node
- The total number of decoupling capacitors to place at a node in the initial iteration. (This can be used if an area happens to have excessive background capacitance)
- There is a seventh input, known as the “adjacent weight”, which is used to quantify the weight placed on line-of-sight decaps (this was discussed in the section on noise estimation and is typically set to 1).

The algorithm returns:

- The number of decaps to place at every node.
- Optionally, the amount of noise at every single node in the final allotment of decaps.

Numerous algorithms exist in the literature for solving this problem, but most of them uses different assumptions from this thesis. In [10], a potentially useful method involving moving along the gradient of a decap vector function is suggested. However, this method attempts to optimize for the smallest total noise and total number of decaps on chip, instead of rigidly adhering to distinct decap and noise requirements at every node. [2] optimizes during the floorplanning stage, rather than following the floorplanning stage, and also uses repeated simulations in its iterations, which is undesirable for our purposes; [5] and [19] use very interesting algorithms for decap allocation, but it is all pre-floorplanning optimization. [13] optimizes the decap by assuming that the sole source of the noise is the charge removed from capacitors nearby in its calculations, instead of the more complicated model employed in these simulations. [4] uses a similar power grid model involving current sources, but iterates in an ad-hoc manner using repeated simulations instead of trying to optimize the amount of decap precisely. Similarly, [15] uses a promising iterative approach, in which it locates hot-spots and calculates the sensitivity of each node with respect to adding decap, but it calculates the noise by repeated simulation as well. This kind of methodology could conceivably work with this problem, but as I will discuss, the greedy algorithm actually employed is guaranteed to produce an

optimal solution. [18] is the most similar approach in the literature to my algorithm. It tries to fully quiet the worst node to below the noise threshold before moving to the second-worst node. This is far faster, but also less optimal, than the greedy approach described in more detail below.

6.2 Justification for algorithm

The decap algorithm is a “greedy” algorithm, meaning that at any given point, we can optimize the “worst” node without reference to any future or past iteration of the algorithm. Here, I will present a slightly modified algorithm and prove that it results in the optimal solution in terms of minimizing the total amount of decap on chip. I will then explain how the algorithm used in this thesis closely approximates the ideal algorithm.

First, I will discuss the ideal algorithm under the following two assumptions:

- all nodes have the same voltage tolerance
- all nodes can fit an infinite number of capacitors.

Then, I shall relax these assumptions and demonstrate that the proof is still valid.

If these assumptions hold, consider the following algorithm (described in pseudo-code)

While there exists a node with greater noise than the noise threshold:

1. Find the worst node and add an infinitesimal amount of capacitance.
2. If more than one node has the same amount of noise, add decap to each node in such a way that the noise after the addition of decap remains the same after decap is added. Certain nodes will have a higher $\frac{dF}{dC}$ than other nodes, meaning that adding a unit of decap at some nodes will cause a greater decrease in noise than adding a unit of decap at other nodes. Thus, we may have to add varying amounts of decap to each node to ensure that each node will continue having an equal amount of noise. (Calculating the amount of decap at each node necessary to do this may be difficult in practice, since adding decap at any node has an effect on the surrounding nodes as well. However, it is theoretically possible, and any such calculations will turn out to be unnecessary).

Claim: If decaps are allocated in this manner, the eventual arrangement of decaps will be optimal. No smaller total amount of decap on the chip will cause all noise to be within tolerable noise margins.

Proof: A node will only get decap added to it if it is, at some point while the algorithm is running, the noisiest node on the chip. Moreover, all nodes with decap have the same amount of noise, since we always add decap in such a way that the noise stays equal at each node. Therefore, once the algorithm terminates, all nodes with at least one decap will have noise identical to the noise tolerance.

Now, imagine some theoretical rearrangement of the decaps currently on chip which would use less than or equal to the amount of decap that is currently on chip. Certain nodes will have had decaps removed. Now, if a node has had a decap removed, it is still possible that its noise after the rearrangement is within acceptable limits; decaps could have been added to the surrounding nodes to compensate. However, earlier, we discussed that the optimal placement of a decoupling capacitor to quiet down a current source is at the local node. Thus, if an amount of decap C was removed from a certain node, then some amount of decap greater than C would have had to be added to the surrounding nodes.

However, this cannot possibly be the case for every single node for which we removed decap, since the total amount of decap cannot increase on the chip. Therefore, in any possible rearrangement of decap on chip, the noise at some node which previously had decap will increase. However, all nodes which previously had decap were previously directly at the tolerable noise limit. Thus, in any rearrangement, some node will exceed the tolerable noise limit. QED.

We can now relax the assumptions in this algorithm. First, consider the case where different nodes can have different noise thresholds. In that case, we can modify the algorithm so that instead of adding decap to the node with maximum noise, we can add decap to the node with the maximum difference between the noise at that stage in the algorithm, and the tolerable noise at that node. If we apply the algorithm this way, then all nodes with decap will have an identical value for ΔV at every step in the algorithm (i.e. the difference between the desired noise and the noise at the present stage in the algorithm). Accordingly, when the algorithm terminates, all nodes with decap will have $\Delta V = 0$ - that is, all nodes with decap will have exactly the tolerable amount of noise. In that case, we can apply the identical reasoning as above to prove

that this is the optimal allocation of decoupling capacitors.

Now, consider the case where each node has a maximum number of decaps that can fit at that node. We modify the algorithm so that if a node has reached its maximum number of decaps, decap is instead distributed evenly across the closest nodes for which additional decap can fit. (Closest is here defined technically as “nodes which will reduce the noise to the greatest possible extent at the fully populated node”, and in fact tends to correspond with nodes which are physically closest in proximity.) To demonstrate that the greedy algorithm with this modification still produces the optimal solution, consider any rearrangement of decoupling capacitors. Every node with some decap will either (a) be at precisely the noise threshold, or (b) be sufficiently close to some node at the noise threshold so that any closer node is fully populated with decap. We can then proceed with the same argument. Any removal or rearrangement of some amount of decap C from the vicinity of a given “noisy” node N will result in an increase in the noise at that node, which could only be counteracted by adding some amount of decap greater than C arranged differently in the vicinity of N . Since the total amount of decap on chip cannot increase, some node will have its noise increase.

6.3 Description of algorithm

The algorithm used in real life operates quite similarly to the last generalization of the ideal algorithm. It operates as follows:

1. Set all nodes on chip to “active”.
2. While there exists an active node:
3. Calculate the noise of all the active nodes on the chip. If the noise at a node is now below its noise threshold, flag it as inactive. Additionally, find the node on the chip for which $V_{present} - V_{threshold}$ is maximal.
4. For this node, determine the best place to add decap to lower the noise. If the node is not fully populated

with decap, clearly it will be the most effective place to add decap. If the node is fully populated with decap, test the result of adding a single decap to all the surrounding nodes to determine which would be the most effective location to add decap. If all of the surrounding nodes in the vicinity are also fully populated, flag the node as inactive.

5. Now that we have selected a node to which to add decap, we must determine how much decap to add. The amount of decap to add is related to how noisy the node is with respect to its target noise. If it is extremely noisy, we can make a large decap jump; if it is not so noisy, we will make only a small decap jump. The precise relationship between noise and the magnitude of the jump is an ad-hoc relation determined by observation.
6. When all nodes are inactive, we are done.

Some comments about the performance of the algorithm:

- This algorithm exclusively used the fast noise calculator. Unfortunately, it was still somewhat slow in performance. Since every node is checking the noise due to every other node, the time grows as $\Theta(N^2)$ with the number of nodes. To reduce the time, we can split the chip into several parts along areas of white-space, since current sources in one area of the chip have only a small impact on current sources in another area on the chip. It typically took an hour to allocate decap on the chip shown in Figure 7.4.
- We are trying to optimize to minimize the total amount of decap on chip. It's possible that we might not want to optimize for this; for instance, we might be willing to have more decap on chip, but limit the decap at any individual node to a certain quantity. This can be easily incorporated into the algorithm, since we can set the "maximum decap value" at any node to whatever we want, and the algorithm will try to find a solution.
- In cases where the peak time approximation fails (such as sensitive nodes without current sources), the best method is to generate an initial approximation using the algorithm, and then test various decap values using the more accurate noise estimator until an acceptable allocation is found.

- In practice, it is best to do some initial noise estimates to see if there is any plausible solution, or where the hotspots (without any decap) might be. If only a small area of the chip requires decap, one can input only the nodes in that area into the algorithm in order to save time.

Chapter 7

Results

In this section, I will use the noise and decap algorithms to study a set of current source configurations. I will present the values that the algorithm predicts, and compare them to SPICE simulation. I will perform the following series of tests:

1. Run the decap allocation algorithm to determine how many decaps are needed for each node. (Recall that this uses the faster method of noise estimation for efficiency purposes). Mark down how much noise the algorithm estimates will exist at each node.
2. Test the current formation in SPICE with the given value for the decap, and note the amount of noise it generates.
3. Iteratively use the slower noise estimation algorithm, using the values generated in the decap allocation algorithm as an initial guess, until we arrive at a new estimate for the amount of decap necessary at each node. Mark down how much noise the algorithm estimates will exist at each node.
4. Test the current formation in SPICE with the updated value for the decap, and note the amount of noise it generates.
5. Iteratively perform SPICE simulations until the correct amount of decap at every node is determined.

Decap prediction method	Decaps	Predicted noise	Actual noise	Percent error
Fast	171	0.18 V	0.1808 V	0.44%
Slow	177	0.18 V	0.1781 V	1.07%
SPICE	173	N/A	0.18 V	N/A

Table 7.1: Predictions for 2 by 2 grid

Obviously, with very complicated configurations, steps 3 through 5 are impossible (indeed, that's the whole reason why one would want a decap allocation algorithm that can handle any arbitrary situation.) For the last test, which involves the simulation of a full chip, I will only perform the first two steps.

7.1 Two by two configuration

Consider the configuration of current sources in Figure 7.1. Each current source will have a width of 500 ps and a height of 400 mA. A maximum of 200 decaps will be placed at any node; since each decap is 1.4 pF, this corresponds to 280 pF per node, consistent with values for IBM's CU-11 technology. The maximum noise at any node will be 10% of the supply, which in this case is 0.18 V for a supply voltage of 1.8 V; again, these values are on the order those used in the real world. Since each node is symmetrical, each node will have an identical amount of decap in the final solution.

We can see the results in Table 7.1; the values are identical for all four nodes. The first row cites the predicted decap value using the decap allocation algorithm, and compares the predicted amount of noise to the simulated amount of noise in SPICE. The second row cites the predicted value for the decap using the more accurate noise prediction model, and compares the predicted amount of noise to SPICE. The third row cites the correct amount of decap determined by iteratively performing SPICE simulations.

In this relatively simple current source configuration, without many nodes to take into account, both algorithms do extremely well. The faster algorithm is actually slightly more accurate in this case than the slower algorithm, but both come close to the right answer.

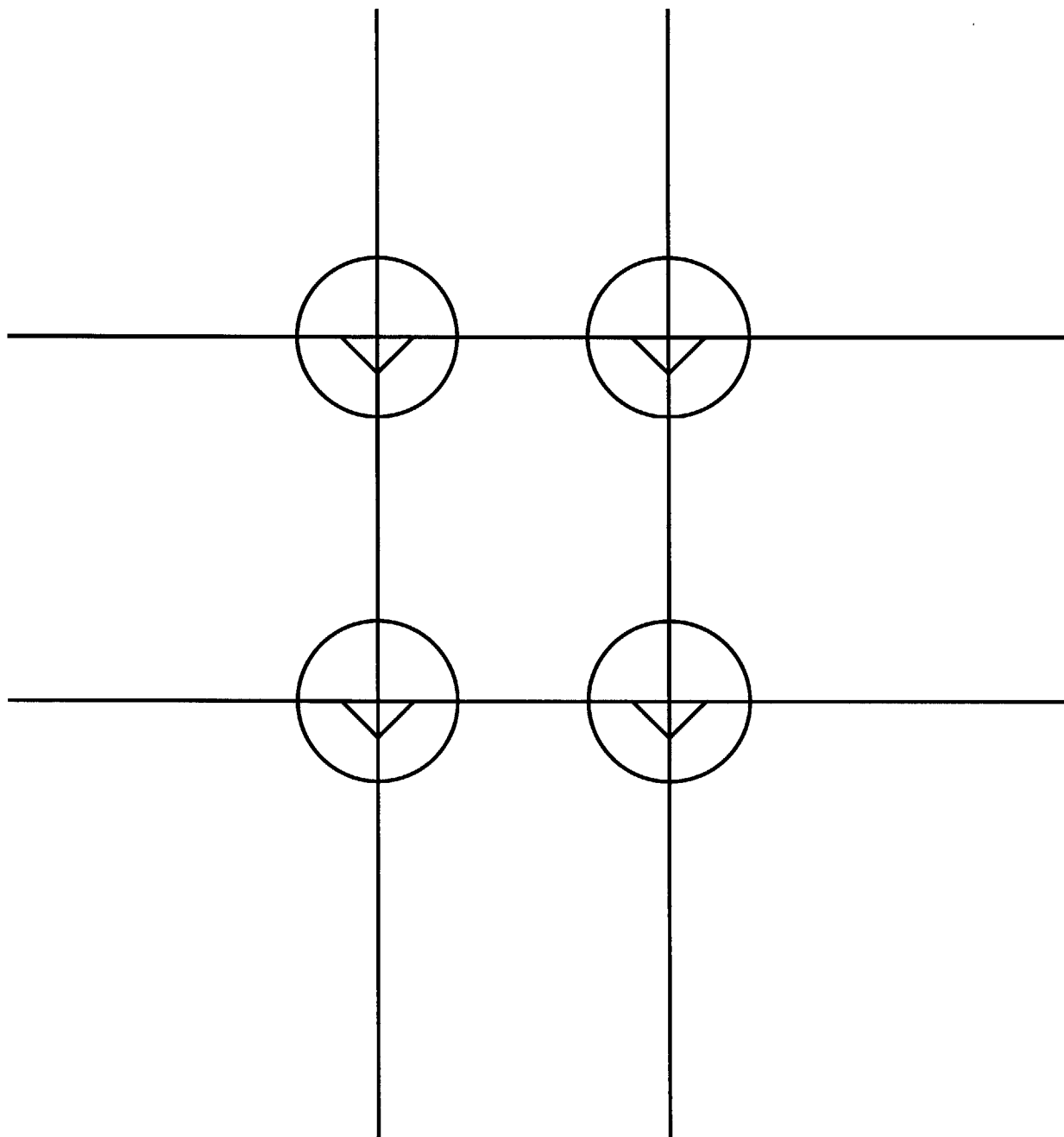


Figure 7.1: Two by two current source configuration

Decap prediction method	Node	Decaps	Predicted noise	Actual noise	Percent error
Fast	Corner	0	0.1397 V	0.1585 V	11.86%
Fast	Top	80	0.1661 V	0.1719 V	3.37%
Fast	Middle	200	0.18 V	0.1836 V	1.96%
Slow	Corner	0	0.1484 V	0.1571 V	5.53%
Slow	Top	84	0.1686 V	0.1704 V	1.06%
Slow	Middle	200	0.18 V	0.1820 V	1.10%
SPICE	Corner	0	N/A	0.1571 V	N/A
SPICE	Top	89	N/A	0.1686 V	N/A
SPICE	Middle	200	N/A	0.18 V	N/A

Table 7.2: Predictions for 3 by 3 grid

7.2 Three by three configuration

We now take the more complicated scenario seen in Figure 7.2. In this scenario, several of the complicated (1, 2) coordinate comparisons discussed in the last section are performed. Note that since I only precharacterized the chip for coordinate values of (0, 1), (0, 2), (1, 1), and (1, 2), the predicted noise values for the corner nodes will be slightly inaccurate, since they will not take into account the current sources at the opposite corners (which are at a distance of (2, 2)). The results are shown in Table 7.2. In this case, due to symmetry, there are three nodes that will have different noise values: the corner nodes, the nodes directly between the corner nodes (which we will call the “top” nodes), and the middle node. The “top” nodes have the same amount of decap and the “corner” nodes have the same amount of decap. (Some “tweaking” had to be done to the output of the decap algorithm to achieve this. Because the algorithm does not add infinitesimal amounts of decap at each step, occasionally the algorithm will produce an asymmetrical allotment of decaps - more decaps at the node directly above the current source than the node directly below the current source, for instance, even though they are symmetrical. To correct this, I ran the algorithm, averaged the outputted values of decaps at each of the “top” nodes as an initial guess for the amount of decap at each of the “top” nodes, and then iteratively guess and checked until I found a solution which had the same amount of decap at each of the “top” nodes.)

These results merit some discussion. First, the fast noise algorithm underestimates the noise. This is what one would predict from the peak time approximation - we are assuming that the peak noise occurs at

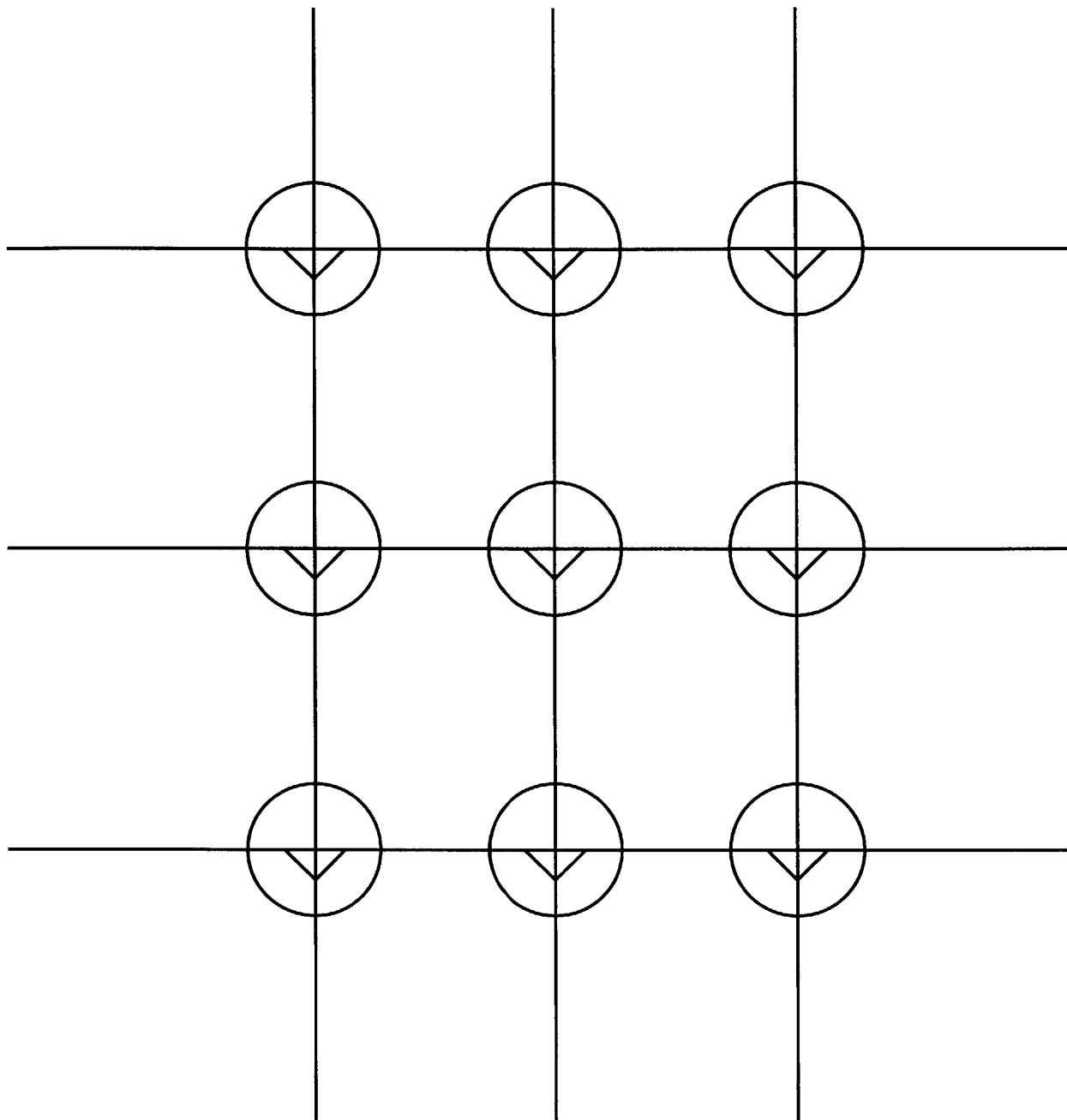


Figure 7.2: Three by three current source configuration

at a particular time, when in fact the noise at other times may in general be greater.

Second, the fast noise algorithm is far less accurate at the corners than at the middle. This has two explanations. First, most obviously, the corner nodes are not taking into account the opposite corners, which results in an underestimate of the noise. Second, there are many distant current sources which we are taking into account at each corner (two current sources of a distance $(1,2)$, and another two at a distance $(0,2)$). Each of these current sources experiences a delay in its peak at the corner node, due to the many capacitors in between these nodes and the corner node. In principle, the algorithm takes these effects into account, as recounted in the past sections; in practice, the fast algorithm tends to have trouble in these types of situations.

Third and most importantly, the above inaccuracy doesn't much matter. This is because the goal of this algorithm was to determine an optimal amount of decap for each node. Since the noisiest node is the middle node, our goal is to determine the number of decaps to place around the grid until the middle node has been quieted to 10% of the supply voltage. Indeed, that the only reason that there is decap at the nodes adjacent to the middle node - the middle node reached its limit of 200 decaps, and had to add decaps nearby in order to lower its noise to below the threshold. Since the noisiest node is at $0.1820 V$ - very close to the desired value of $0.18 V$ - we can be sure that we are close to the correct solution.

We can see that the slow noise estimator is substantially more accurate than the fast noise estimator. Its inaccuracies are related only to errors in the data fitting, rather than any fundamental inaccuracies like the peak time approximation. Its prediction of 84 decaps at each of the "top" nodes is roughly halfway between the "fast" predicted value (80 decaps) and the correct value (89 decaps).

7.3 3 by 3 grid with different decap maxima

We can now envision a situation identical to the one in the previous section, except that the maximum number of decaps that can fit at any node is 100. (In general, you can define any maximum number of decap at any node.) Here are the results we obtain:

Decap prediction method	Node	Decaps	Predicted noise	Actual noise	Percent error
Fast	Corner	37	0.1348 V	0.1505 V	10.43%
Fast	Top	100	0.1579 V	0.1665 V	5.17%
Fast	Middle	100	0.1798 V	0.1848 V	2.71%
Slow	Corner	53	0.1418 V	0.1466 V	3.27%
Slow	Top	100	0.1626 V	0.1624 V	0.12%
Slow	Middle	100	0.18 V	0.1804 V	0.22%
SPICE	Corner	54	N/A	0.1463 V	N/A
SPICE	Top	100	N/A	0.1621 V	N/A
SPICE	Middle	100	N/A	0.18 V	N/A

Table 7.3: Predictions for 3 by 3 grid with different decap maxima

The notable thing about these results is the decap in the corners of the grid. These decaps are included purely for the purpose for reducing the amount of noise at the middle node. The algorithm fully populates the middle node with decap; fully populates each of the top nodes with decap; and then moves to the corner nodes, evenly populating them with decap until the middle node is quieted below the threshold. Once again, the slower noise algorithm is substantially more accurate than the faster noise algorithm; this time, it comes close to achieving almost the perfect solution.

7.4 3 by 3 grid with no middle source and different noise tolerances

In this section, I present a modified version of the 3 by 3 grid. The middle node does not have a current source, but has a noise tolerance of 0.15 V . (All the rest of the nodes still have noise tolerances of 0.18 V). In this case, even though the middle node does not have a current source, it will still be the noisiest node in relation to its noise tolerance at most steps in the algorithm. We would expect the slower algorithm to be more accurate than the faster algorithm, because the peak time approximation does not hold for the middle node. The results are shown in Table 7.4. Again, because of the asymmetrical effect discussed above, some iteration was required after running the decap allocation algorithm to arrive at the solution that was actually tested.

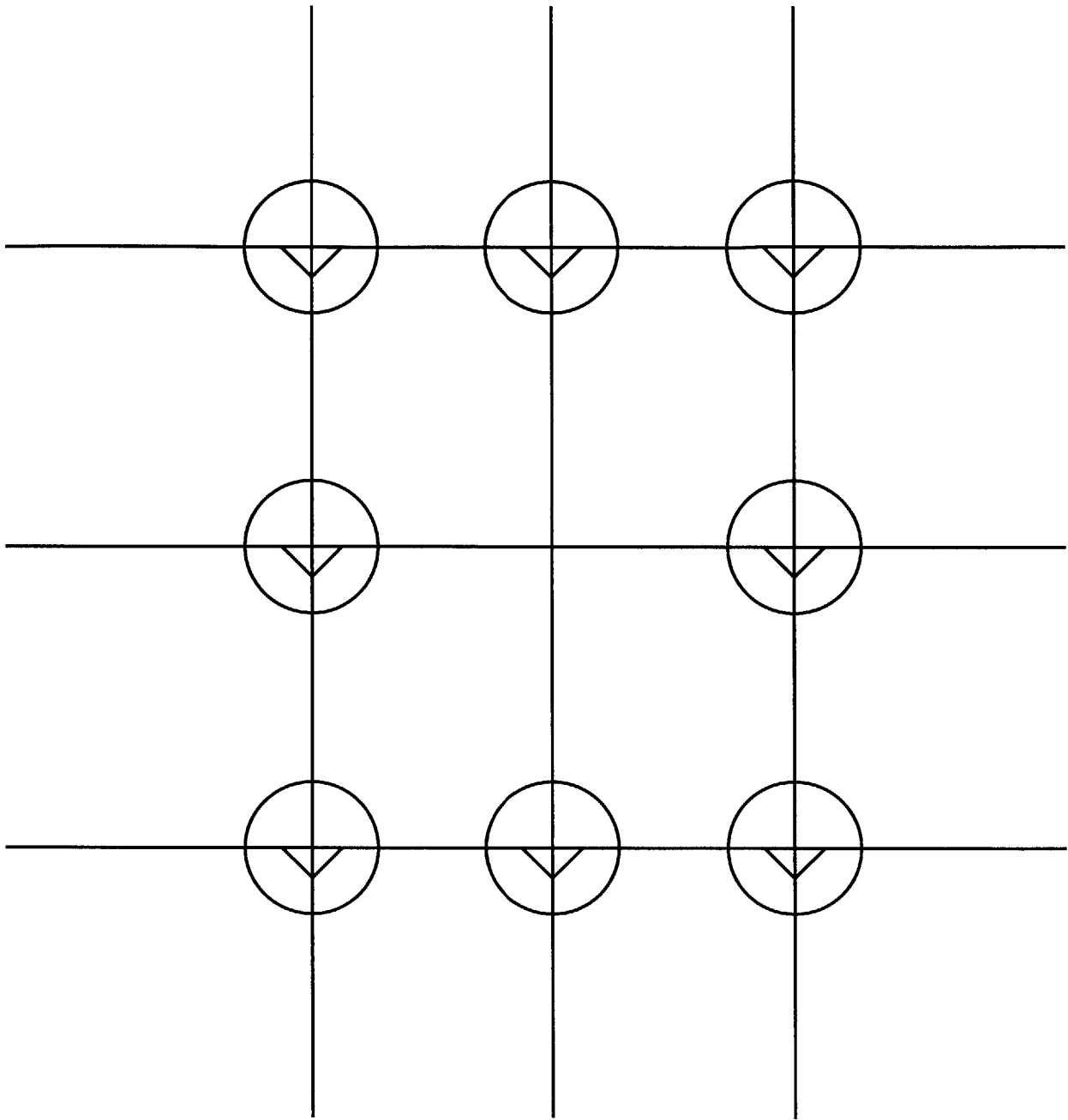


Figure 7.3: Three by three current source configuration with no middle node

Decap prediction method	Node	Decaps	Predicted noise	Actual noise	Percent error
Fast	Corner	0	0.1497 <i>V</i>	0.1666 <i>V</i>	10.14%
Fast	Top	127	0.1719 <i>V</i>	0.1718 <i>V</i>	0.06%
Fast	Middle	200	0.15 <i>V</i>	0.1600 <i>V</i>	6.25%
Slow	Corner	0	0.1518 <i>V</i>	0.1632 <i>V</i>	6.99%
Slow	Top	138	0.1670 <i>V</i>	0.1679 <i>V</i>	0.54%
Slow	Middle	200	0.15 <i>V</i>	0.1566 <i>V</i>	4.21%
SPICE	Corner	0	N/A	0.1571 <i>V</i>	N/A
SPICE	Top	160	N/A	0.1606 <i>V</i>	N/A
SPICE	Middle	200	N/A	0.15 <i>V</i>	N/A

Table 7.4: Predictions for 3 by 3 grid with no middle node

As expected, the slower algorithm is more accurate than the fast algorithm. It is not entirely clear why both of them underestimate the noise; these inaccuracies are an inevitable result of the data-fitting and approximation that characterize the noise estimation. However, both the fast and the slow algorithm clearly come within the right ballpark of the solution.

7.5 Full chip

The final test involves the simulation of an entire chip. This chip is 3 mm by 3 mm - a relatively small chip, but on the same order of magnitude as what one might encounter in the real world. All of the nodes have tolerances of 0.18 *V* and can fit 200 decaps. The current sources are of various different magnitudes, and are arranged as in Figure 7.4. Now, this calculation will be inherently inaccurate, since only current sources of a distance (0, 1), (0, 2), (1, 1), and (1, 2) were precharacterized. (Again, one could precharacterize for other distances too, but in the interests of time only those four coordinates were precharacterized). This means that any current sources of a distance greater than (1, 2) from any node will be ignored. However, since noise drops off relatively quickly, this strategy still approximates a reasonable solution.

In this case, we must rely exclusively on the results of the decap allocation algorithm; iterations with dozens of nodes is impracticable. Additionally, instead of presenting a node-by-node analysis of the noise, I will present it in histogram format. Figure 7.5 shows a histogram of noise values prior to adding any decap, and Figure 7.6 shows a histogram of noise values after adding decap. We can see that the noise tends to

cluster around $0.18 V$. which is precisely what we want; we would prefer if nodes didn't have significantly less than $0.18 V$ of noise, because that would imply that we were using too much decap, but of course we don't want nodes which have greater than $0.18 V$ of noise either. Finally, Figure 7.7 shows a histogram of the amount of decaps at each node. The striking thing is that most nodes either are fully populated, or have zero decap. This is because of the greedy nature of the algorithm; it tends to fully populate the noisiest nodes before moving onto nearby nodes.

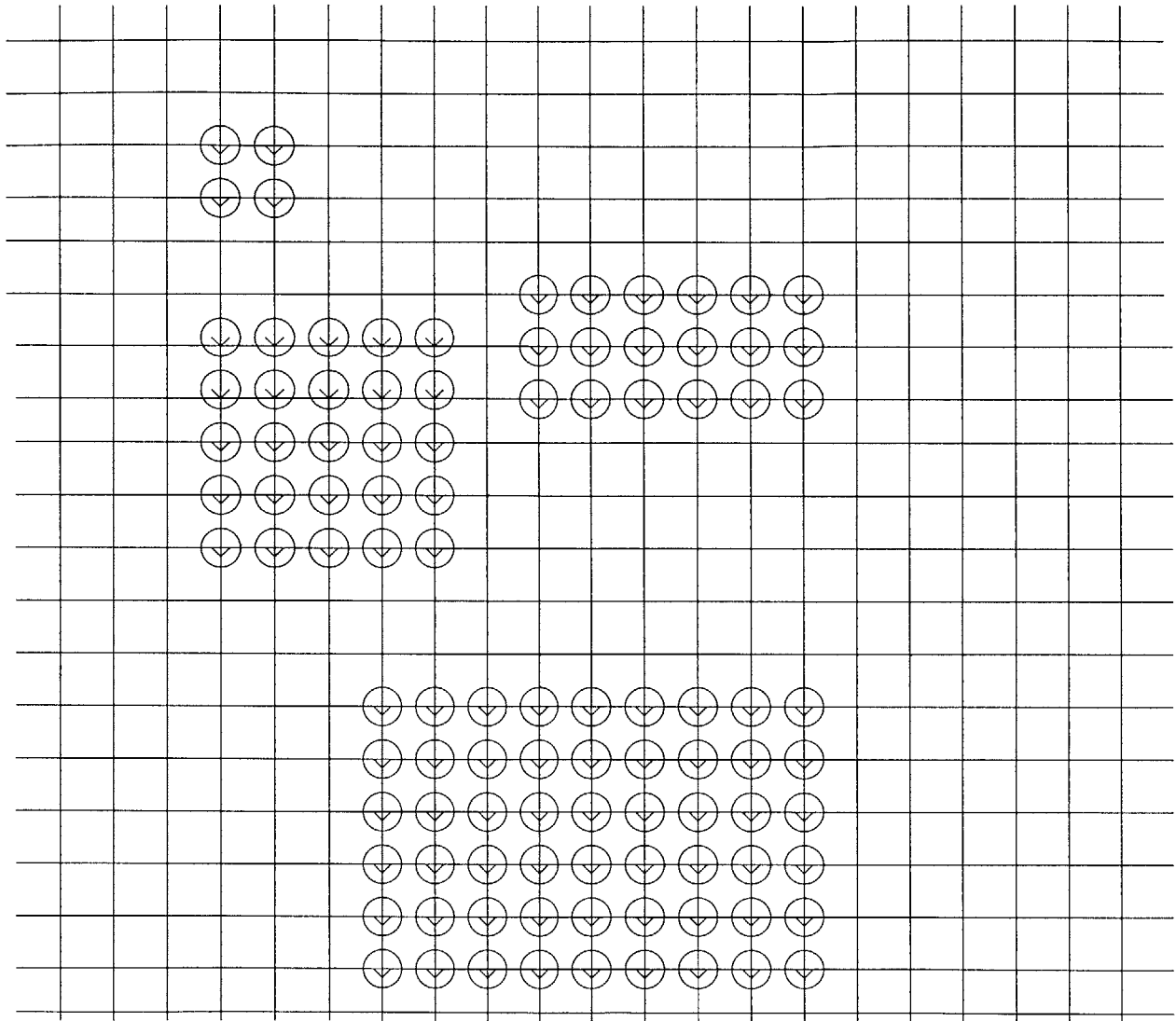


Figure 7.4: Current sources on a small chip

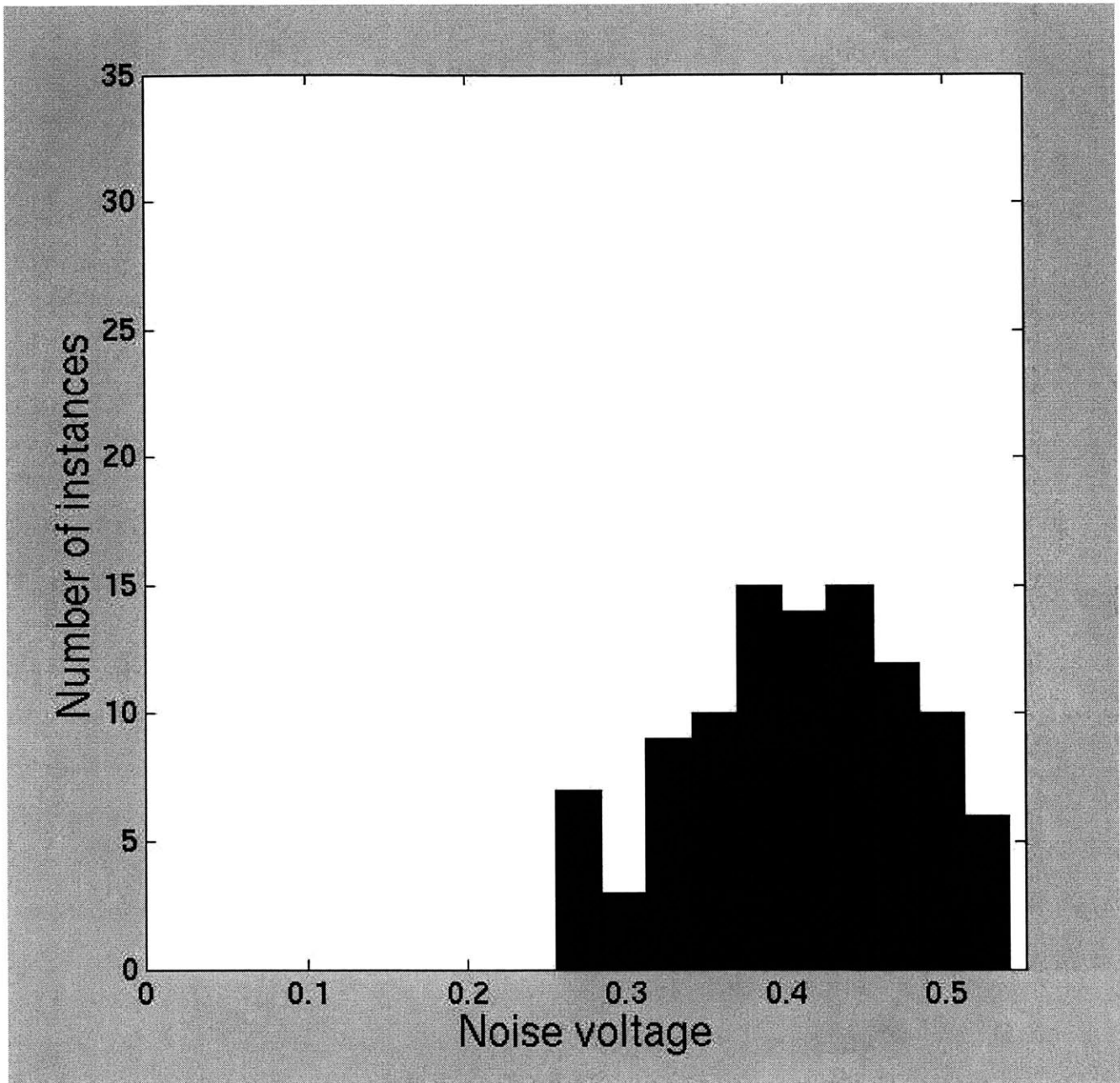


Figure 7.5: Histogram of noise values before adding decap

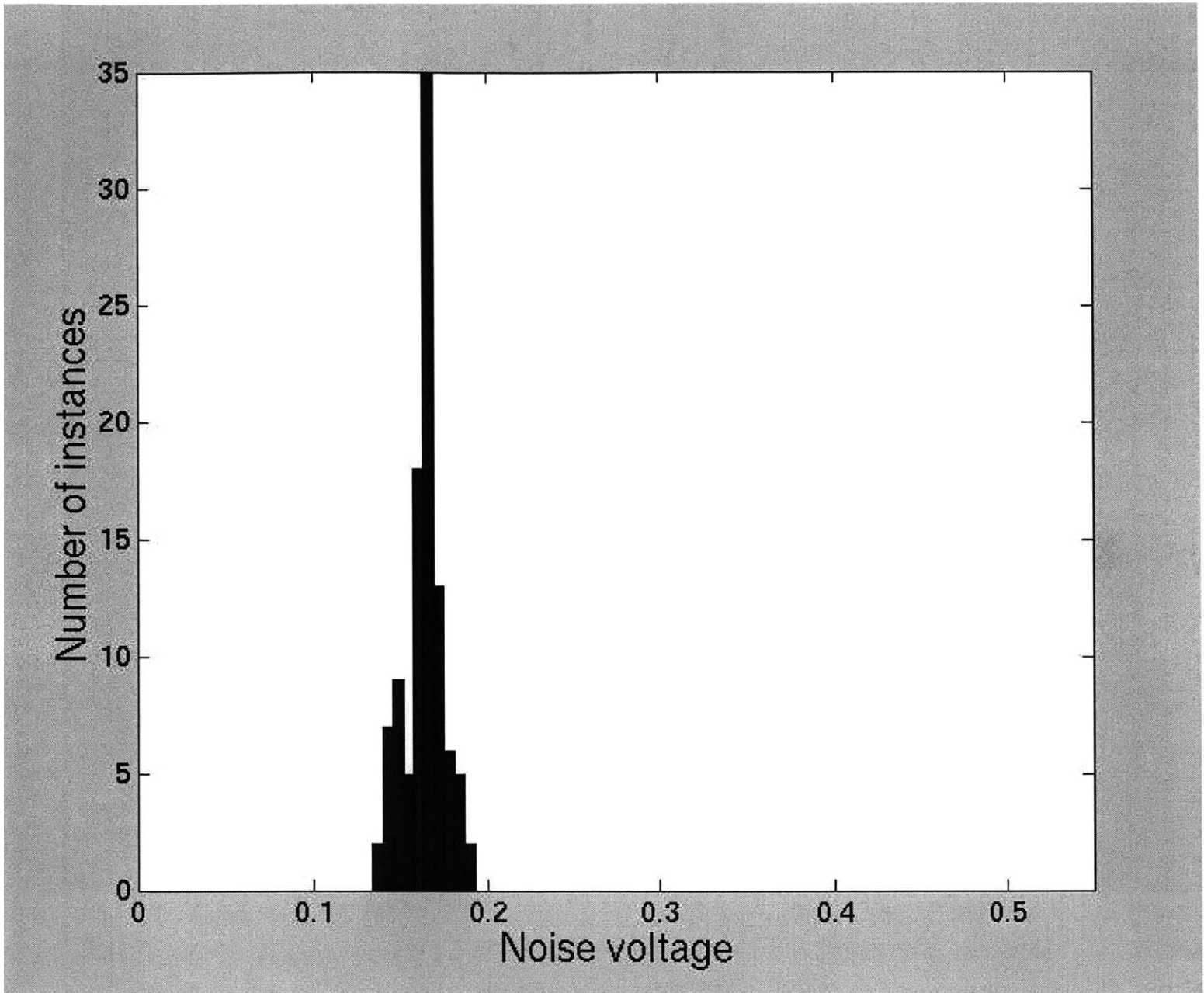


Figure 7.6: Histogram of noise values after adding decap

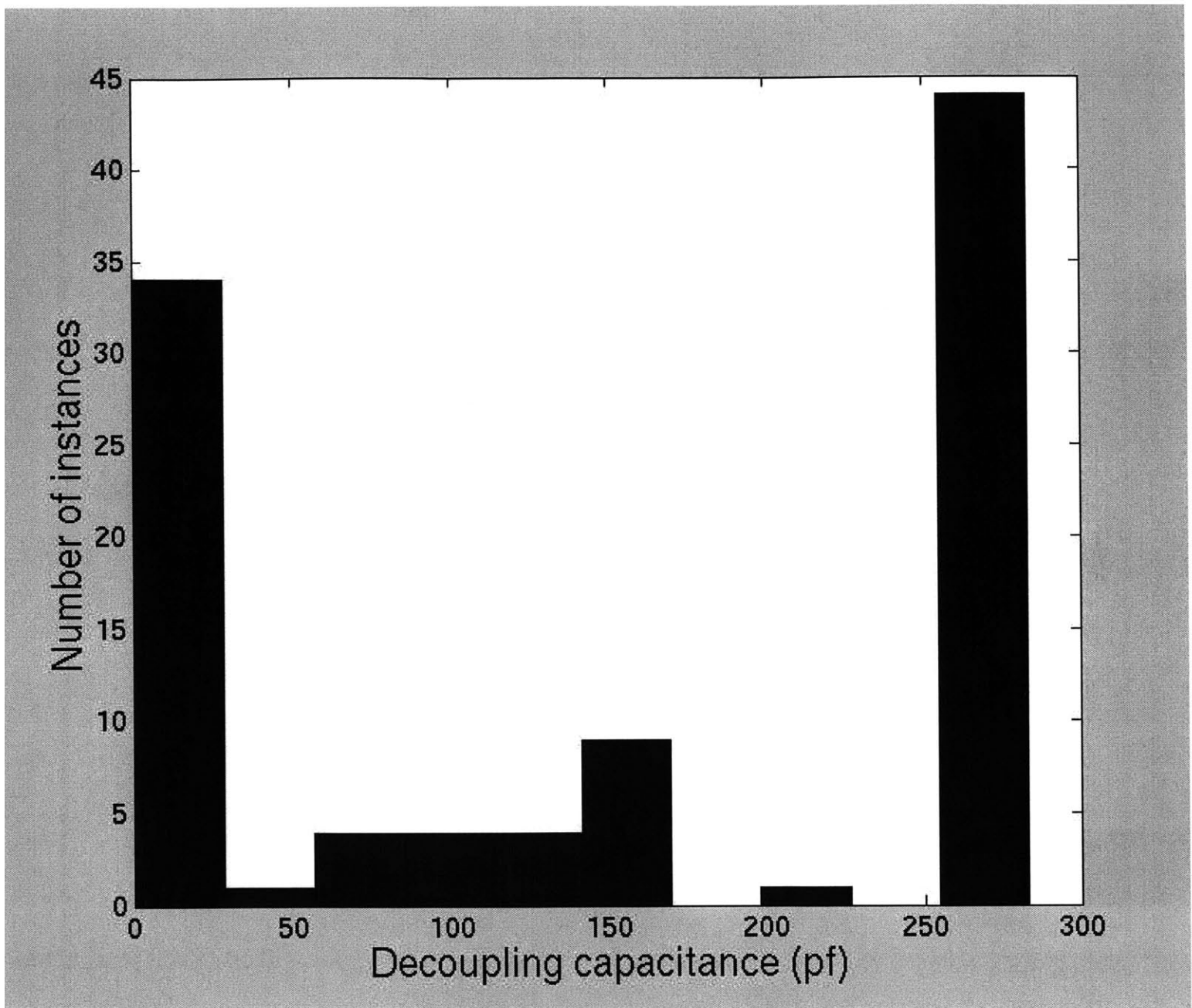


Figure 7.7: Histogram of decap values in eventual solution

Chapter 8

Conclusions and future work

There were two main goals in this thesis. The first goal was to develop a better understanding of quantitative aspects of noise propagation on chip, both through circuit analysis and through interpreting the results of SPICE simulation. The second goal was to develop a method for assigning decaps on chip, which preferably could be demonstrated to be optimal for a particular cost function. To achieve these goals, I used a simple circuit model to predict some of the non-trivial characteristics of noise propagation. I then synthesized several aspects of noise propagation observed in SPICE simulation into a more extended model. This model used three fundamental assumptions which vastly simplified the problem: first, that current sources could be linearly superimposed; second, that decap acted symmetrically whether placed at a current source or at a measurement point; and third, that the effects of decap on noise could be linearized. I then used these models to describe a method for predicting the exact amount of noise at any node on a chip for any allocation of decap. Finally, I used this method as the core of an algorithm that could allocate decap on chip in such a way as to minimize the total amount of decap.

The algorithms in this thesis are fairly effective at allocating decap within the context of the assumptions on which they are based. The problem of accurately calculating the noise due to an arbitrary collection of current sources and capacitors is quite difficult and non-linear, and any approximation method is bound to be somewhat inaccurate for complicated formations of current sources and capacitors. However, the previous

section indicates that the algorithms come within the ballpark of the solutions determined using SPICE. At worst, the algorithm provides a useful initial guess which makes iteration using SPICE simulation far easier.

In order to be useful, the algorithm depends on the ability to convert complicated current signatures into triangular current pulses. As discussed earlier in the thesis, it is not clear how easy this is; most models in the literature indicated that one can just sum the area under the curve, but in my simulations, the value of the peak current had a significant impact on the amount of noise, independent of the amount of charge. SPICE simulation cannot really be used to resolve this dilemma, since the problem amounts to determining which SPICE model is most accurate. Simulating actual, real-world chips in action would be useful in determining the best way to construct a power grid model.

Methods other than precharacterization could be used to simplify the algorithms used in this thesis. For instance, instead of complicated ad-hoc data fitting, a set of simulation results could be stored into a lookup table, and one could interpolate between them to predict the noise in a given situation. This method would be especially useful for a complicated piecewise linear current signature, in which it might be very difficult to precharacterize the curve using simple equations. Such a technique would eliminate the need to perform complicated nonlinear fits, some of which were of questionable quality, and instead use real simulated data.

Finally, the project could take into account other aspects of noise issues on chip. One effect that circuit designers have reported is ringing on the power rails. Because of the *RLC* circuit that forms between the package pins and the on-chip resistance and capacitance, the voltage will often oscillate. This may cause unpredictable effects on the noise, especially if the ringing is at the resonant frequency of the clock. The algorithms in this thesis assume that the power supply was stable prior to the current pulse. Relaxing that assumption might be helpful in increasing the utility of the thesis.

Bibliography

- [1] Yi-Shing Chang, Sandeep K. Gupta, and Melvin A. Breuer. Analysis of ground bounce in deep sub-micron circuits. *VLSI Test Symposium, 1997., 15th IEEE*, pages 110–116, 1997.
- [2] Howard H. Chen and David D. Ling. Power supply noise analysis methodology for deep-submicron vlsi chip design. *Design Automation Conference 1997, Proceedings of the 34th*, pages 638–643, 1997.
- [3] Howard H. Chen and J. Scott Neely. Interconnect and circuit modeling techniques for full-chip power supply noise analysis. *Components, Packaging, and Manufacturing Technology, Part B: Advanced Packaging, IEEE Transactions on, Volume: 21 , Issue: 3 , Aug. 1998*, pages 209–215, 1998.
- [4] Howard H. Chen, J. Scott Neely, Michael F. Wang, and Gricell Co. On-chip decoupling capacitor optimization for noise and leakage reduction. *Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings*, pages 251–255, 2003.
- [5] Howard H. Chen and Stanley E. Schuster. On-chip decoupling capacitor optimization for high-performance vlsi design. *VLSI Technology, Systems, and Applications, 1995. Proceedings of Technical Papers*, pages 99–103, 1995.
- [6] John A. DeFalco. Ground bounce, what you simulate may not be what you get. *ASIC Conference and Exhibit, 1992., Proceedings of Fifth Annual IEEE International , 21-25 Sept. 1992*, pages 479–482, 1992.

- [7] T.J. Gabara. Ground bounce and reduction techniques. *ASIC Conference and Exhibit, 1991. Proceedings., Fourth Annual IEEE International , 23-27 Sept. 1991*, pages T13 – 2/1–2, 1991.
- [8] Payam Heydari and Massoud Pedram. Analysis and optimization of ground bounce in digital cmos circuits. *Computer Design, 2000. Proceedings. 2000 International Conference on , 17-20 Sept. 2000*, pages 121–126, 2000.
- [9] Naohiko Hirano, Masayuki Miura, Yoichi Hiruta, and Toshio Sudo. Characterization and reduction of simultaneous switching noise for a multilayer package. *Electronic Components and Technology Conference, 1994. Proceedings., 44th*, pages 949–946, 1994.
- [10] Charles Hough, Tianxiong Xue, and Ernest Kuh. New approaches for on-chip power switching noise reduction. *IEEE 1995 Custom Integrated Circuits Conference*, pages 133–136, 1995.
- [11] Adnan Kabbani and Asim J. Al-Khalili. Estimation of ground bounce effects on cmos circuits. *Components and Packaging Technologies, IEEE Transactions on, Volume 22, Issue 2, June 1999*, pages 316–325, 1999.
- [12] Jaewon Oh and Massoud Pedram. Multi-pad power/ground network design for uniform distribution of ground bounce. *Design Automation Conference, 1998. Proceedings , 15-19 June 1998*, pages 287–290, 1998.
- [13] Mondira Deb Pant, Pankaj Pant, and Donald Scott Wills. On-chip decoupling capacitor optimization using architectural level current signature prediction. *ASIC/SOC Conference, 2000. Proceedings. 13th Annual IEEE International*, pages 288–292, 2000.
- [14] David P. Steele. Ground bounce in cmos asic's. *ASIC Seminar and Exhibit, 1989. Proceedings., Second Annual IEEE , 25-28 Sept. 1989*, pages P9 – 4/1–4, 1989.
- [15] Haihua Su, Sachin S. Sapatnekar, and Sani R. Nassif. Optimal decoupling capacitor sizing and placement for standard-cell layout designs. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , Volume: 22 , Issue: 4 , April 2003*, pages 428–436, 2003.

- [16] Srinivasa R. Vemuru. Simultaneous switching noise estimation for asics. *ASIC Conference and Exhibit, 1995., Proceedings of the Eighth Annual IEEE International , 18-22 Sept. 1995*, pages 7–10, 1995.
- [17] Shiyou Zhao and Kaushik Roy. Estimation of switching noise on power supply lines in deep sub-micron cmos circuits. *VLSI Design, 2000. Thirteenth International Conference on , 3-7 Jan. 2000*, pages 168–173, 2000.
- [18] Shiyou Zhao, Kaushik Roy, and Cheng-Kok Koh. Decoupling capacitance allocation and its application to power-supply noise-aware floorplanning. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , Volume: 21 , Issue: 1 , Jan. 2002*, pages 81–92, 2002.
- [19] Shiyou Zhao, Kaushik Roy, and Cheng-Kok Koh. Power supply noise aware floorplanning and decoupling capacitance placement. *Design Automation Conference, 2002. Proceedings of ASP-DAC 2002. 7th Asia and South Pacific and the 15th International Conference on VLSI Design*, pages 489–495, 2002.