POWER SUPPLY PARTITIONING FOR PLACEMENT OF BUILT-IN CURRENT SENSORS FOR I_{DDQ} TESTING

A Thesis

by

ABHIJIT PRASAD

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2003

Major Subject: Computer Engineering

POWER SUPPLY PARTITIONING FOR PLACEMENT OF BUILT-IN CURRENT SENSORS FOR I_{DDQ} TESTING

A Thesis

by

ABHIJIT PRASAD

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

December 2003

Major Subject: Computer Engineering

ABSTRACT

Power Supply Partitioning for Placement of Built-In Current Sensors for I_{DDQ} Testing.
(December 2003)

Abhijit Prasad, B.E., Regional Engineering College, Trichy, India
Chair of Advisory Committee: Dr. Duncan Moore Henry Walker

I_{DDQ} testing has been a very useful test screen for CMOS circuits. However, with each technology node the background leakage of chips is rapidly increasing. As a result it is becoming more difficult to distinguish between faulty and fault-free chips using I_{DDQ} testing. Power supply partitioning has been proposed to increase test resolution by partitioning the power supply network, such that each partition has a relatively small defect-free I_{DDQ} level. However, at present no practical partitioning strategy is available. The contribution of this thesis is to present a practical power supply partitioning strategy.

We formulate various versions of the power supply partitioning problem that are likely to be of interest depending the constraints of the chip design. Solutions to all the variants of the problem are presented. The basic idea behind all solutions is to abstract the power topology of the chip as a flow network. We then use flow techniques to find the min-cut of the transformed network to get solutions to our various problem formulations. Experimental results for benchmark circuits verify the feasibility of our solution methodology. The problem formulations will give complete flexibility to a test engineer to decide which factors cannot be compromised (e.g. area of BICS, test quality, etc) for a particular design and accordingly choose the appropriate problem formulation. The application of this work will be the first step in the placement of Built-In Current Sensors for I_{DDQ} testing.

DEDICATION

To my parents – Sandhya & Yashwant

ACKNOWLEDGMENTS

Returning to graduate school after a break of three years I was apprehensive as to what it would be like. At the end of these two years I can say that I have thoroughly enjoyed my two years here as a grad student. This section is to acknowledge the help, guidance and support of all the people who made this thesis and my two years here possible.

First and foremost I would like to thank my advisor Dr. Hank Walker. Besides his financial support and technical guidance, he was always a friend to me. I am grateful to him for all his help, guidance and encouragement both in technical work and on personal matters. I have learned a great many things from him and consider myself fortunate to have been one of his students.

I would like to thank Dr. Weiping Shi and Dr. Jianer Chen for being on my committee. They were both always ready to spend time with me to discuss my work. In addition, I am grateful to both of them for the excellent courses offered by them. I learned a great deal from both their courses. I would also like to thank Dr. Jiang Hu for all his help. Even though he was not on my committee he was always there to guide and advise me.

I have greatly enjoyed my time spent in my office with Wangqi Qiu, Sagar Sabade, Bin Xue, Siddharth Choudhuri and Jing Wang. I thank them for all their help and the numerous conversations with them, ranging from VLSI to politics. I would also like to thank Sagar for giving me access to some of his figures and references for use in this thesis.

I am extremely grateful to all my friends who always were there for me and have made graduate life memorable. In particular, I thank Shravan, Kaushik, Anuj, Rishi, Vimalan, Abhi and Murali.

I thank my family in the United States, Gyan and Preeti Hajela, Dinkar, Meenakshi, Sharad and Abhas for all their support during my stay here.

I thank all the Computer Science Department staff for taking care of all administrative issues especially Sandy Morse, Wanda Allen and Margaret Dunaway.

I am thankful to the Semiconductor Research Corporation for funding my project.

Finally, I thank my parents and grandparents for having faith in my abilities and who have been infinite sources of encouragement to me. I am what I am because of them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

Ideal CMOS circuits should consume negligible power in the stable state, as the NMOS and PMOS circuits do not conduct at the same time. This state is called the quiescent state of the circuit. In reality, all transistors leak when off, and this leakage current is called the quiescent current (I_{DDQ}). This current is typically low and hence an appreciable increase in its value indicates a defective circuit. Such a form of testing is known as I_{DDQ} testing.

I_{DDQ} testing has been highly useful in detecting defects that are hard to detect using functional or stuck-at testing. With the arrival of Deep Submicron (DSM) technology, leakage currents are rising rapidly [1]. This is primarily due to sub-threshold conduction and gate oxide leakage in CMOS MOSFETs. With this it is becoming increasingly difficult to distinguish between a defective and defect-free chip. This is illustrated in Figure 1. For earlier technologies (> 1 µm) there is a clear distinction between good and faulty chips. However, for DSM technologies it is getting tougher to draw a line between faulty and fault-free chips based on full chip I_{DDQ}. In addition, manufacturing variations in the I_{DDQ} level will be difficult to control. The combination will make it increasingly difficult to distinguish defect-free from defective chips via I_{DDQ} tests. As I_{DDQ} tests have proven to be a very effective test screen for chips, it is desirable that its usefulness be extended into the DSM era.

1. Reducing Background I_{DDQ}

Several approaches have been proposed to extend the life of I_{DDQ} testing. Delta-I_{DDQ} [2], Current Ratios [3], Current Signatures [4, 5], Neighbor Current Ratios (NCR) [6] and Immediate Neighbor Difference IDDQ Test (INDIT) [7]. These methods can increase I_{DDQ} test resolution by 30-100 times, however these only extend the usefulness of I_{DDQ} testing for a few more technology nodes. Recent work also examine ways to lower the intrinsic leakage current: temperature reduction, substrate backbiasing, lowered quiescent V_{DD}, multiple transistor thresholds, stacked transistors, and Silicon on Insulator (SOI) [8, 9, 10,

---

The journal model is *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems.*

11]. These approaches suffer from drawbacks like the use of a specific technology, design modifications, or process modification. They are also insufficient to keep the background leakage low enough to permit effective I_{DDQ} testing.

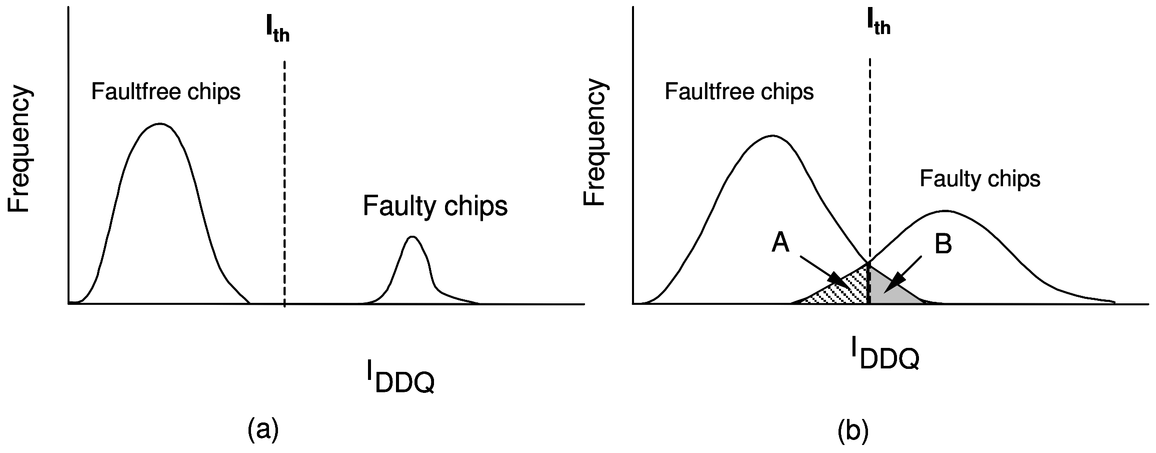

Figure 1. Limit Setting Problem in (a) Earlier Technologies (b) DSM.

Power supply partitioning has been proposed to increase test resolution by partitioning the power supply network, such that each partition has a relatively small defect-free I_{DDQ} level [12, 13]. Every partition has a BICS that measures the current in that single partition. Thus the fault free leakage current (also called background leakage) in each partition is low enough to achieve adequate I_{DDQ} test resolution. We can externally partition the power supply too. However, this might only be feasible for the current technology node and that too for low power chips. Analysis in [14] shows that the only feasible long-term test approach would be to combine power supply partitioning with resolution enhancement methods like Delta-I_{DDQ}.

2. Partitioning Requirements Based on ITRS Predictions

The 2002 update of the International Technology Roadmap for Semiconductors [1] projections are given in Table I. In the first row we have the technology node and the year it is likely to be in production. The next two rows give the DRAM ½ pitch and the gate length for high performance microprocessors. Power supply voltage and transistor leakage current estimates are in the next two rows. Following that we give the number of transistors projected in a high performance microprocessor and the total number of power and ground pads. The total power and ground pads are expected to be two-thirds of the total pads on the chip. The I_{DDQ} per transistor is calculated assuming the width to length ratio (W/L) of transistors is 3/1. Total chip I_{DDQ} is calculated assuming that 25% of the transistors are leaking. Assuming a background leakage current of 100 µA is permissable per partition, the number of partitions and transistors per partition are presented in the next two rows. We use the value of 100 µA since many chips with leakage of 100 µA are successfully screened with I_{DDQ} test today [15]. It can be seen that the leakage current rises from 70 mA in the 130 nm technology node to the over 300 A in the 22 nm node. To be able to maintain test resolution the number of partitions also rises very steeply from 600 in the 130 nm technology node to over 3 million in the 22 nm node. It is clear that this partitioning cannot be done manually and a practical partitioning strategy needs to be developed to automate this task. The contribution of this thesis is to present such a power supply partitioning strategy.

The requirements for a practical BICS are described in [14] and a prototype sensor is described in [16]. This BICS senses the magnetic field generated by the $I_{DDQ,}$ and so does not affect the supply network and can sense current flowing in either direction. This work has been extended to measure the voltage drop due to the I_{DDQ} flowing through the parasitic supply line resistance [17]. This sensor is about 500 transistors in size, including self-calibration and scan chain readout circuitry. The scan chain contains approximately 400 out of the total of 500 transistors and hence we can reduce the number of transistors per sensor by sharing scan chains across sensors. If four sensors share one scan chain, the number of transistors becomes 200. However this will result in a four-fold increase of test application time.

TABLE I. ITRS 2002 PROJECTIONS RELATED TO I_{DDQ} TESTING

Year Technology Node	2001 130 nm	2004 90 nm	2007 65 nm	2010 45 nm	2016 22 nm
DRAM 1/2 Pitch (nm)	130	90	65	45	22
MPU Physical Gate Length (nm)	90	53	35	25	13
Nominal Vdd (V)	1.2	1	0.7	0.6	0.4
High-perf. SubThreshold Leakage at 25°C (µA/ µm)	0.01	0.1	1	3	10
High-perf transistors (Millions)	97	193	386	773	3 092
Power and ground pads	2 048	2 048	2 048	2 560	2 944
I_{DDQ}/transistor (nA)	2.70	15.90	105.00	225.00	390.00
Chip I_{DDQ} at 25°C (A) MPU High-perf	0.07	0.77	10.13	43.48	301.47
Number of Partitions per chip	655	7 672	101 325	434 813	3 014 700
Transistors per Partition	148 092	25 157	3 810	1 778	1 026

Partitioning at the logic level is easiest to implement. However, it has the inherent drawback that the place/route tools would need to be modified to be able to handle these chunks of logic in a way that all the power to a given partition be monitored by a BICS on a single branch. In such a case, the number of partitions would be the number of sensors required. Such a routing may not meet all IR-drop constraints of the power network and thus in general the only way to do power supply partitioning is once it has actually been planned and routed on the chip. Power gating [18] to reduce power dissipation provides a natural partition to insert a BICS, but such partitions may be too large or too small for BICS insertion, so they are not considered here.

3. Organization of the Thesis

The organization of this thesis is as follows. In Section II we give a brief discussion of the background required to appreciate the rest of this thesis. It includes an introduction to the VLSI testing problem and I_{DDQ} testing. In Section III we formally state the power supply partitioning problem. We present all the different formulations of the problem. Section IV discusses the methodology used to solve the different problem formulations stated in the previous section. It also gives in pseudo code format the various algorithms used. We present the experimental data and results in Section V. Finally in Section VI we give the conclusion and direction for future work.

II. BACKGROUND

1. VLSI Testing

Keeping up with Moore's Law [19] for the past couple of decades, the number of transistors on a chip has been doubling approximately every eighteen months. This has been a result of advances in semiconductor manufacturing technology. As a result of this greater transistor density, integrated circuits (ICs) have become faster, cheaper and smaller. Defects or damage to the chip can be introduced during the fabrication, or assembly of silicon. Substantial progress has been made in IC processing and assembly toward reducing the frequency of defect occurrence, but devices with defects will still exist. It is from this fact that the need for IC testing is established. Testing is used to identify faulty chips in an otherwise good population and ensure that they are not shipped to the customer.

One way to test a chip is to verify that the outputs of a chip are correct for all possible combinations of inputs. If the outputs match the expected values then we can say that the chip is functionally fault free. This form of testing is known as *functional* Testing. The problem with this is that it results in a very long test sequence. For example, for a combinational circuit with 64 inputs, a complete functional test requires that we observe its output for 2^{64} input combinations. If we can apply these inputs at a rate of 1 GHz, we need 585 years to fully test this circuit. It is obvious that such testing is not practical. Thus there is a need for more intelligent testing methodologies.

The most common form of VLSI testing is called *structural* testing. Unlike functional tests, structural tests depend on the specific structure of the circuit. Primary to these tests are fault models. These models abstract the behavior of a physical defect on the chip to its effect on the functioning of the chip. The most common fault model is the *stuck-at* fault model. It assumes that defects cause gate inputs to behave as stuck-at-1 or stuck-at-0. In a given circuit if there are *n* gate terminals (or lines), then using the Stuck-at-fault model, there are $2*n$ possible faults that can occur in the chip (each terminal can be either stuck-at-1 or stuck-at-0). To test a given stuck-at fault, we need to set the value of the line to opposite of the stuck-at-fault, i.e. to test a stuck-at-0 fault at a line we need to set its value to 1. Finding the right set of inputs to generate a 1 at the required line requires tracing back from the faulty line to the circuit inputs (termed primary inputs). The degree of difficulty in setting a terminal to a given value is called the c*ontrollability* of the line. Now, to be able to detect the fault, we must be able to propagate the faulty value from the fault site to a circuit output (termed primary output). For this we again need to assign values to primary inputs such that the faulty signal gets propagated to the primary output. The degree of difficulty in making assignments to the primary inputs such that a fault at a given terminal is observed at a primary output is called the *observability* of a line. This task of generating a test is known as Automatic Test Pattern Generation (ATPG) and is essentially the *satisfiability* problem that is known to be NP-complete [20]. There has been considerable research on ATPG and various heuristic algorithms [21, 22, 23] have been proposed which are widely used to perform test generation.

Apart from functional and structural tests, there can be tests that consider various parameters of the chip like delay, transient current, quiescent current, etc. These tests typically target faults that do not affect the logical behavior, but degrade the performance and reliability of the circuit. Such faults are called parametric faults and these tests are called *parametric* tests. Delay testing and I_{DDQ} testing are parametric tests. In the next section we explain I_{DDQ} testing in greater detail.

2. I_{DDQ} Testing

In ideal CMOS circuits the NMOS and PMOS circuits do not conduct at the same time. Hence at any given time there is no path from V_{DD} to V_{SS} (or GND) and thus there is no current flow. In reality transistors are not ideal, and CMOS and NMOS do conduct together for a short period of time when the input values switch and capacitors are charged and discharged. This state is called the transient state and the current that flows in the circuit is called the transient current (I_{DDT}). Once the input value has changed, the circuit settles down to a stable state. This state is called the quiescent state of the circuit and I_{DDQ} is the current that flows in a CMOS circuit when all nodes are in a quiescent state. In ideal CMOS circuits I_{DDQ} should be zero. However, due to various transistor leakage mechanisms [24] there is always a current that flows from V_{DD} to V_{SS}. In a fault free circuit this current is typically very small. For a circuit in which a fault exists this current increases appreciably. Thus by measuring this current we can identify a faulty circuit from a fault-free one.

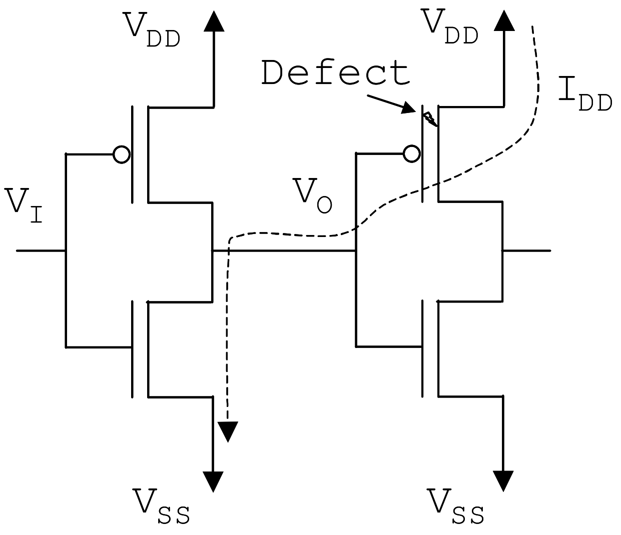

Figure 2. CMOS Invertor Circuit with Defect.

Figure 2 shows two CMOS inverters back to back with a defect in the PMOS transistor of the second invertor that causes its gate impedance to drop from an infinitely high value to a finite value. Now a DC current flows in steady state along the path indicated by the dashed line and this elevates the steady state leakage current. Figure 3 shows the input and output voltages and the drain current I_{DD} that flows through the transistors. After switching completes, this current is referred to as I_{DDQ}. In the good circuit I_{DDQ} falls to a negligible value, whereas in the defective circuit I_{DDQ} remains elevated long after switching is over.

I_{DDQ} testing can detect various types of faults in CMOS circuits. Any fault that causes an increased quiescent current flow can be detected. Bridging faults and gate oxide short failures (as shown in Figure 2) in most cases can be detected by I_{DDQ} testing. Stuck-open faults (when lines are in a floating state due to a broken wire) cause elevated I_{DDQ} if the floating line is at an intermediate voltage. Since stuck-at faults are essentially bridging faults to V_{DD} or V_{SS} lines they can also be detected using I_{DDQ} testing.

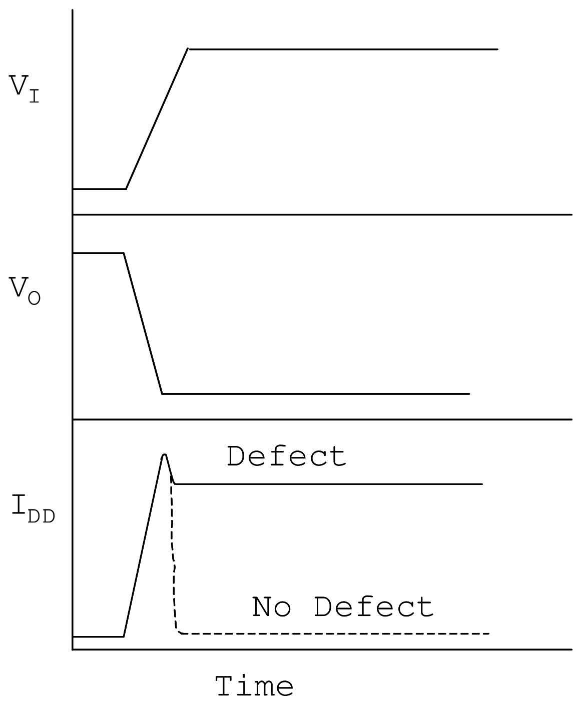

Figure 3. Voltages and Currents at Input and Output for Figure 2.

The current limit setting problem with relation to I_{DDQ} testing is deciding a pass/fail current value that can discriminate between good and bad chips. This problem has received considerable attention recently [25, 26] and is still an active problem. Setting the threshold limit (I_{th}) to a high value will allow faulty chips to be shipped (called test escapes). Conversely setting the limit too low will cause good chips to be discarded (called overkill). Figure 1 shows how the I_{DDQ} limit-setting problem is getting tougher with Deep Submicron (DSM) technologies. Region A shows the overkill and region B the test escapes.

One drawback of I_{DDQ} testing is that we need to wait for the circuit to settle down to its quiescent state and then take the current measurement. As a result, the rate of application of test vectors for I_{DDQ} testing is much slower than stuck-at testing. However, test generation for I_{DDQ} testing is much easier since faults only need to be excited and we do not need to worry about propagating them to a primary output, as there is global observability in the supply network.

The measurement of current can be done in two ways, off the chip using the tester precision measuring unit (PMU) or a load board current sensor and on the chip using a built in current sensor (BICS). The advantage of using off chip measurements is that there is no silicon overhead on the chip for the BICS and associated circuitry, and there is no extra delay due to a series impedance in the supply lines. However, off chip current testing has an inherent resolution limit that is getting worse with advancing technology.

III. POWER SUPPLY PARTITIONING PROBLEM

In this section we formally define the power supply partitioning problem. For any chip design we can easily represent its power supply network as a linear electrical network. It is this power network that we refer to in the following discussion. We first list the variables that define the problem and then give three possible formulations, by varying the objective function to be optimized and the constraints on the problem.

1. Variables

Given a power network, the task is to place BICSs on various branches of the power network such that these BICSs together monitor the leakage current of the chip while at the same time keeping background leakage current levels low in each BICSs. In the following discussion, unless otherwise mentioned, whenever we talk about current in a branch, we are referring to the fault free background leakage current.

The problem formulation is dependant on the following three variables:

- The maximum background leakage current on a branch being monitored by any BICS on a single branch (I_{max}).
- The number of sensors used to monitor the current (n).
- The total current that is not being monitored by the sensors (I_u).

As can be seen all three variables are crucial factors in determining the success of the on-chip current sensing strategy. The first variable I_{max} addresses the problem of reduction in test resolution as background leakage current increases. The higher this value, the lower the resolution of the sensor in detecting faults. The second variable n, is the number of sensors used on the chip. It addresses the issue of area overhead due to the additional sensor circuitry on the chip, and increased routing complexity. It is obvious that this should be kept as small as possible. The last variable I_u helps in deciding the test escape probability. The greater the current that is not monitored, the higher the probability of missing detection of a defect using I_{DDQ} test, and thus the lower the quality of the test solution. Ideally I_u should be zero, thereby not allowing any current to go unmonitored.

However, in real scenarios the constraints on silicon area might be a stronger driving factor, resulting in a solution that causes some current to go unmonitored.

2. Problem Formulations

By using one of the above variables in the objective function and the remaining two as the constraints, we can formulate various optimization problems, all of which are interesting from an implementation standpoint. The first problem formulation is:

Minimize I_u

Such that $n <= n_{max}$;

And $I_{max} <= I_{Dmax}$.

In this formulation, the chip area overhead and test resolution are not compromised, however, we do allow current to go unmonitored. The term n_{max} is the maximum number of BICSs allowed and I_{Dmax} is the maximum allowable background leakage current on any branch that is being monitored by a BICS.

Alternatively, the number of sensors that can be used is fixed and we need to find the lowest possible maximum current on any single branch being monitored by these BICSs. It is also required that there is no unmonitored current. Such a problem can be stated formally as:

Minimize I_{max}

Such that $n <= n_{max}$;

And $I_u = 0$.

The above formulation is likely to be a very common case as the area overhead permitted for BICS sensors is likely to have a limit for most chip designs. With this constraint we want to find the set of branches that monitor all the current on the chip such that the maximum current through any of these branches is minimum.

The third formulation is to minimize the number of sensors used while keeping the current on all branches below the specified threshold I_{Dmax}. We also ensure that there is no unmonitored current. This means that the test resolution and test quality is not compromised, and we only try to minimize the area overhead of the BICS circuitry. This problem can be formally stated as:

$$\text{Minimize } n$$
$$\text{Such that } I_{max} <= I_{Dmax};$$
$$\text{And } I_u = 0.$$

Consider the case when there is no unmonitored current. This means that if we remove all the branches on the power network that are being monitored by the BICSs there will be no current flow in the network. Hence we would have essentially partitioned the power network into parts, with the removed edges forming the "cut" edges. For the last problem formulation, we want to minimize the number of BICSs. Thus, what we are looking for is the smallest set of branches that partition the given power network into two parts such that none of these branches has a background leakage of more than I_{Dmax}. If we call the number of branches along the boundary of the two partitions the "cut size", our objective is to find the partition with the minimum cut size. The classical circuit partitioning heuristics of Kernighan-Lin [27] and Fidducia-Mattheyses [28] are for balanced bipartition of circuits, which is an NP-Complete problem [20]. However, in our transformed problem there is no constraint on the sizes of the partitions. We can thus apply known polynomial time flow techniques to solve our problem optimally [29, 30, 31, 32]. Such an approach has been used in [33] for balanced circuit partitioning.

3. Tradeoffs in Objective Function/Constraints

So far in our problem formulations, the constraint on the current limit I_{Dmax} has been fixed. However, this can be relaxed to allow a small increase in I_{Dmax} if it results in a large reduction in the number of sensors n. For example, if I_{Dmax} is 10 μA and there is an 11 μA branch feeding 22 branches each carrying 0.5 μA, then it might be preferable to have a single BICS monitoring the 11 μA line rather than 22 sensors monitoring each of the 0.5 μA lines. Such tradeoffs need to be decided based on how tight the constraints need to be. For example in the case considered above, if it is acceptable to have currents more than the

10 μA limit, we need to decide how much reduction in the number of sensors is worth the loss of resolution.

4. Power or Ground Network

Another factor that we have not considered so far is whether to use the ground network to monitor current. So far we have talked about monitoring current only on the power (V_{DD}) lines. Depending on the type of built-in current sensors used, either one of the power or ground networks will be preferable. For some types of sensors (like magnetic sensors [16]) it does not matter whether a power or a ground line is being monitored.

There are two ways in which the ground network can be included in our analysis. The first is to use only the ground network instead of the power network. In this case, our methodology works identically as it does for power networks, the only difference being that instead of the power network we consider the ground network. In a typical design, there will not be too much difference in the topology of the power and ground networks and hence it should not matter considerably which network is chosen.

Another way would be to use a combination of both power and ground lines to monitor current. This would mean that the number of branches in the network would double and thus the number of potential BICS locations also would increase (need not double though, as it would depend on the capacity labeling strategy employed for the transformed flow network). The solution generated when using both networks can only be better than that obtained when using only one network. This is because it is just two networks connected together with the cells being the nodes between the corresponding branches in the two networks. In the worst case the solution obtained will be same as the solution when only one of the networks is considered.

IV. SOLUTION METHODOLOGY

This section explains the methodology used to solve the problems defined in the previous section. Given the chip design, the first step is to obtain the topology of the power supply network. We assume that the chip design is in the layout format of the chip after the placement and routing stage of the design flow.

From the chip layout the power network can be easily obtained by doing a parasitic extraction of the power net of the chip. We also need to know the leakage current estimates of the cells. The technology design parameters give us this data. The extracted parasitic resistances form a resistive network and the leakage current of the cells can be modeled as current sources hanging off the nodes in the network. An example is shown in Figure 4. This linear circuit problem can be solved to obtain the fault free leakage current magnitude and direction for each branch of the power network.

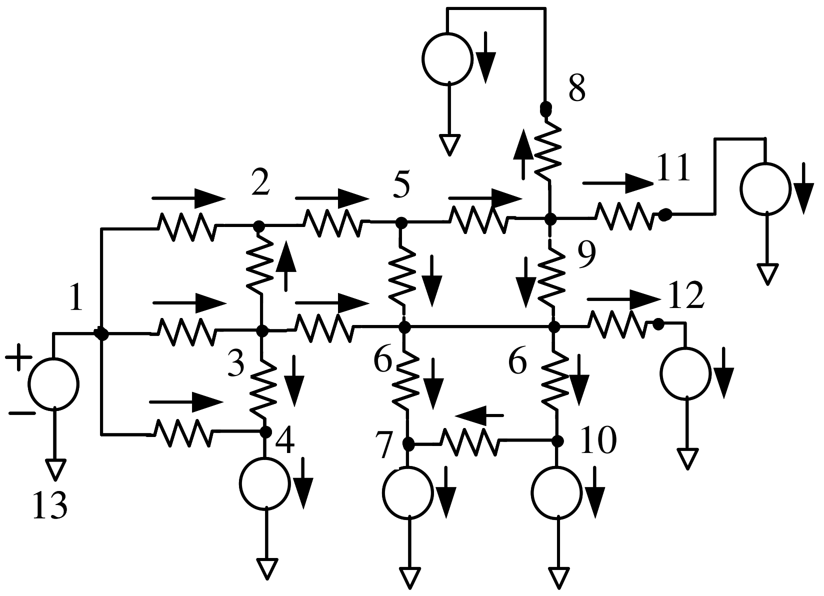

Figure 4. A Linear Electrical Network.

1. Power Supply as a Flow Network

We model this power network as a directed graph $G=(V, E)$, where V is the set of nodes (i.e. fanout points in the resistive network), and E is the set of branches. The direction of edges in E is determined by the direction of current flow, i.e. if current flows from node v_1 to v_2, then the edge direction will be from v_1 to v_2. Each edge e has two values associated with it, a flow $F(e)$ and a capacity $C(e)$. The capacity of edges is set using the labeling scheme described in the following subsection. The flow of all edges is set to zero initially.

We can then reduce all the problems described in the previous section to that of finding an s-t (source to target) cut of minimum capacity with appropriate s and t values in the transformed flow network. Figure 5 shows the transformation of the power supply network in Figure 4 to its equivalent flow network. The numbers shown on the circuit are the node numbers and the arrows show the direction of current flow. The values of s and t in this particular case are 1(V_{DD}) and 13(GND) respectively. We use the cut given by the min-cut algorithm to choose BICS locations.

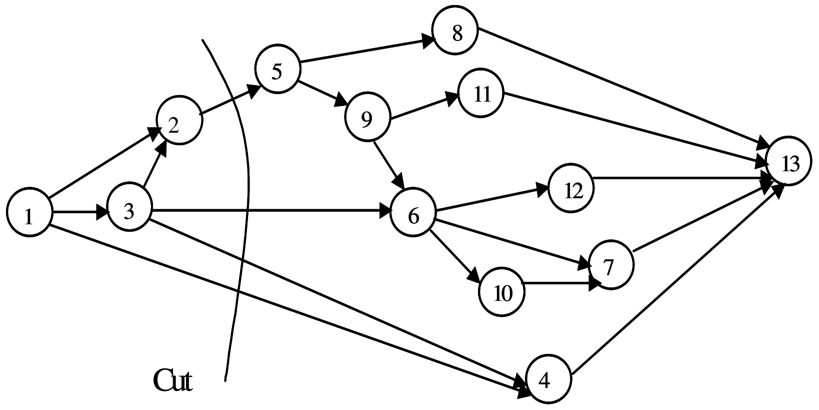

Figure 5. The Electrical Network Abstracted as a Flow Network.

2. Capacity Labeling Scheme

The capacity $C(e)$ of an edge e is assigned by comparing the current on the corresponding branch of the power network with I_{Dmax}. If the current through e is less than or equal to I_{Dmax}, we set $C(e)$ to 1, otherwise it is set to a much higher value (assume this to be infinity for the time being – we will later explain how it can take other values). The value of $F(e)$ is initially assigned 0 for all edges.

The reason behind the capacity labeling scheme is as follows. All branches that have current less than I_{Dmax} are potential sites for the BICS placement and are thus potential cut edges. Since it is the number of such cut edges that we want to minimize, we normalize all the edges such that all potential cut edges have the same probability of being chosen. Correspondingly, all edges that are not potential edges are given a high capacity that forces the min-cut algorithm to avoid choosing them.

These high values can either be set to a single infinitely high value or scaled up according to the edge weight. As an example, all edges with current from I_{Dmax} to twice that of I_{Dmax} can be assigned weight 10. This would mean that exceeding I_{Dmax} by up to 100% would only be acceptable if it requires 9 fewer sensors than otherwise. This capacity scaling scheme is flexible, allowing us to make tradeoffs in the solution.

This labeling scheme can also be extended to eliminate those branches on which it would be difficult to monitor current. For example, in a mesh type power network, it might be hard to monitor current on a branch that is in the higher metal layers and thus this branch could be given a higher weight. In all the discussions that follow we follow the simple labeling scheme by marking all branches carrying current greater than I_{Dmax} as infinity. However, it should be remembered that any type of labeling scheme could be used depending on the design constraints.

3. Min-Cut Partition

A flow network is represented as $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each edge has two values associated with it, a flow $F(e)$ and a capacity $C(e)$. If s and t are the source and target in V, then an s-t cut (or simply *cut*) (V_1, V_2) is a bipartition of

V into sets V_1 and V_2 such that $s \in V_1$ and $t \in V_2$. An edge whose starting node is in V_1 and ending node is in V_2 is defined as a forward edge from V_1 into V_2. The capacity of the cut (V_1, V_2) is the sum of capacities of the forward edges from V_1 to V_2.

The residual capacity of an edge $R(e)$ is defined as $C(e) - F(e)$, i.e. the additional flow that can be pushed along e. If $R(e)$ is 0, then the edge e is saturated and no additional flow can be pushed along it. This edge is removed from the network and the resulting network is called the residual network. An augmenting path from node u to v in G is a simple path from u to v in the undirected graph using the edges in the residual network that can be used to push additional flow from u to v. A max-flow in G is a flow of maximum value from s to t.

The max-flow min-cut theorem [29] states that the value of a maximum flow is equal to the capacity of a minimum cut. More formally stated, given a max flow f in G, let $V_1 = \{v \in V: \exists$ an augmenting path from s to v in $G\}$, and let $V_2 = \{V - V_1\}$, then (V_1, V_2) is a cut of minimum capacity (which is equal to $|f|$), and f saturates all forward edges from V_1 to V_2.

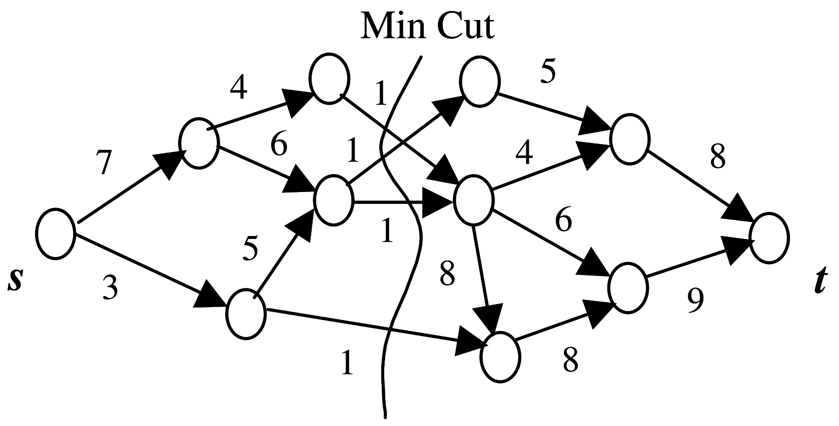

Figure 6. Min-Cut of a Flow Network.

An example illustrating the max-flow and min-cut is shown in Figure 6. The numbers are the capacities of the edges and the source and sink are marked s and t. It is obvious that the flow from s to t will be restricted by the edges with capacity 1. The network will have a max-flow of 4 with all the edges of capacity 1 being saturated. To find the min-cut once the flows are known, we start from s and traverse the network until we reach a saturated edge. The set of saturated edges found this way gives us the min-cut.

There are numerous polynomial time algorithms that exist to find the max-flow in a network [29, 30, 31, 32]. The fastest such algorithm is the Goldberg-Tarjan algorithm [31] which has a worst case time complexity of $O(|V|^3)$, which we have implemented in this work.

4. Partition Using Min-cut

The three different problem formulations in the previous section can be solved using the min-cut algorithm. For all the three problems the initial steps are the same. The first step is to calculate the currents in all branches of the power network. Once the branch currents are known, the network can be transformed into a graph using the appropriate labeling scheme. Since we also know *s* (V_{DD}) and *t* (GND) for the graph, we can find the *s-t* cut of the graph. Using the cut obtained this way we can obtain solutions to the problem depending on its formulation. We now present solutions to the different problem formulations. For the following sections we assume that we know the topology of the power network.

a. Minimize number of sensors (*n*)

We consider this problem formulation first because it is simplest to implement and is the basis for solving the other two problem formulations. As a reminder, the problem is to find the minimum number of sensors such that the entire current in the chip is monitored and that current in any branch must be below I_{Dmax}.

This is the simplest formulation and all we need to do is to transform the electrical network into a flow network, label the edges appropriately, and obtain the min *s-t* cut for the flow network. These cut edges give the locations for the BICS locations. Since we have labeled all branches with current less than I_{Dmax} as 1, all such edges are equivalent to the min-cut algorithm which will try to find the minimum set of such edges that partitions the network into two with source (V_{DD}) in one partition and sink (GND) in the other partition.

Also since these edges form a cut it means that there will be no current flow from V_{DD} to GND if these edges are removed from the network and thus monitoring current on these edges will monitor all the current in the circuit.

b. Minimize unmonitored current (I_u)

When the problem formulation is to minimize the unmonitored current, then the cut-size must be less than n_{max} and the current in each branch must be less than I_{Dmax}. The approach we use for this is just a small step beyond the approach used for minimizing the number of sensors. We transform the electrical network into a flow network, label the edges appropriately, and obtain the min *s-t* cut for the flow network. After that, since we want to minimize the unmonitored current, we sort the cut edges in descending order of current. Then the top n_{max} edges will give the BICS locations.

c. Minimize the maximum current through any BICS (I_{max})

For this problem formulation, we want the test resolution to be as good as possible while restricting the area overhead of the BICS, i.e. the number of branches is kept below n_{max}. To solve this problem we make use of a property of the transformed flow network. Depending on the value of I_{Dmax} used the number of edges in the generated network changes. The higher the value of I_{Dmax}, the number of edges in the transformed network with capacity 1 will be higher. Hence branches with higher current become potential cut locations. Since current flows from source (V_{DD}) to sink (GND) in the network and there exist fan-out points in the network, monitoring current on a branch closer to V_{DD} will not require more locations than if measured closer to GND. In fact, it is expected that it will require fewer locations if current is monitored closer to V_{DD}. This correspondingly means that the higher the value of I_{Dmax}, the number of sensor locations will not increase and is expected to decrease. We verify the monotonicity of the n vs. I_{Dmax} with our experiments.

Once we know that number of cut locations varies monotonically with I_{Dmax}, we do a simple binary search of the solution space until we get a cut set that is less than n_{max} and close enough (depending on the accuracy required). We run the algorithm as we do for the first problem formulation (to minimize the number of sensors) in a loop. For every iteration of the loop we change I_{Dmax} depending on the number of locations given by the previous run. The stopping criterion is if the number of locations generated by the min-cut algorithm is within a percentage of n_{max}.or if the number of BICS remains the same for 3 iterations. The algorithm has been stated in Figure 7. The functions used are in Figure 8 and Figure 9.

```
Power Supply Partition Algorithm
Input:
   1. Power network in SPICE netlist format
   2. Problem Formulation
   3. $I_{Dmax}$ , $n_{max}$ (depending on problem formulation)
   4. %Error $\varepsilon$ if problem formulation is to min $I_{max}$

Output:
   1. Locations on the Power network that BICS need to be placed

Algorithm:
   1. Solve power network to get currents in all branches
   2. If the problem formulation is to minimize $I_{max}$ then:
        a. set $I_{Dmax}$ = High value
        b. $I_{low} = 0$; $I_{high} = I_{Dmax}$
   3. FlowNetwork = Transform and Label edges(Power network, $I_{Dmax}$)
   4. Edges = Find min s-t cut(FlowNetwork, VDD, GND)
   5. If the problem formulation is
        a. To minimize n then:
             i. Return Edges as the locations for BICS placement
        b. To minimize $I_u$ then
             i. Sort Edges in descending order of current on the
                edges
            ii. Return first $n_{max}$ edges in Edges as the locations for
                BICS placement
        c. To minmize $I_{max}$ then
             i. EdgeCount = Number of branches in Edges
            ii. Diff = Absolute Value($n_{max}$ - EdgeCount)
           iii. Loop While( Diff/$n_{max}$ % > $\varepsilon$ or EdgeCount does not
                change for 3 iterations)
                   1. $I_{max} = (I_{low} + I_{high})/2$
                   2. FlowNetwork = Transform and Label edges (Power
                      network, $I_{max}$)
                   3. Edges = Find min s-t cut(FlowNetwork, VDD,
                      GND)
                   4. EdgeCount = Number of branches in Edges
                   5. If(EdgeCount > $n_{max}$) then $I_{low} = I_{max}$
                   6. If(EdgeCount < $n_{max}$) then $I_{high} = I_{max}$
                   7. Diff = Absolute Value($n_{max}$ - EdgeCount)
            iv. End While Loop
             v. Return Edges as the locations for BICS placement
   6. End If
   7. End Algorithm
```

Figure 7. Power Supply Partitioning Algorithm.

```
Tranform and Label Edges
Input:
   1. Power network in SPICE netlist format
   2. $I_{Dmax}$

Output:
   1. Flow network G=(V, E)  corresponding to the input power network

Algorithm:
   1. For each node X in the power network make a vertex in v in G
      such that vertex(X) = v
   2. For each branch between node X and node Y in the power network
      such that current flows from node X to node Y:
         a. Make two directed edges in G:
                i. Edge1 = From vertex(X) to vertex(Y)
                      1. If current on edge > $I_{Dmax}$ then set
                         Capacity(Edge1) = Infinity
                      2. If current on edge <= $I_{Dmax}$ then set
                         Capacity(Edge1) = 1
               ii. Edge2 = From vertex(Y) to vertex(X)
                      1. Set Capacity(Edge2) = 0
   3. Return G
```

Figure 8. Function to Transform the Power Network into a Flow network.

```
Find Min s-t Cut
Input:
   1. Flow network G=(V, E)
   2. Source s
   3. Sink t

Output:
   1. Minimum set of edges on the flow network such that the network
      is partitioned into parts with these edges removed, one
      containing the source and the other the sink

Algorithm:
   1. Run the Max flow algorithm on the network G
   2. Mark all vertices in the residual network generated by the max
      flow algorithm as Unreachable
   3. In the residual network find all the vertices reachable from the
      source and mark them as Reachable
   4. For all edges between vertex X and Y in the residual network:
         a. If X is marked Reachable and Y is marked Unreachable then
            add this edge to the CutSet
   5. Return CutSet
```

Figure 9. Finding the Min s-t Cut of a Flow Network.

5. Modifying the Power Network

So far in all our problem formulations we do not modify the power network of the chip. We take an existing power network and locate branches on this network based on the constraints given to us. However, most power networks designed in chips are not optimized and have lot of redundancy in them. We can remove this redundancy (i.e. remove branches in the power network), and thus get cuts with fewer edges. Since the min-cut of the transformed flow network gives us the branches that need to be monitored, if we remove any of these edges, we can guarantee a reduction in the number of sensors by 1 and hence these are the only branches that we consider for removal.

While we are removing edges of the power network we must ensure that we do not compromise the performance or the reliability of the circuit. To this end, we define two parameters to ensure that the performance and reliability of the power network is within acceptable limits. We make these checks when considering each branch of the power network and thus the complexity of this check step must be kept as small as possible.

The first parameter that we check is the *connectivity* of the circuit. It is obvious that at any time there must exist a path from all nodes in the network to V_{DD} and GND. However, even with this, we might get a case in which power pads get disconnected from each other. This is also not a desirable situation. We need to analyze what happens when we remove a branch from the power network. The current that was flowing through this branch needs to get re-routed through the circuit. This obviously affects the neighborhood of the two nodes that form the branch. The greater the number of nodes that are affected by this re-routed current, the greater is the change in node voltages. We want to keep the number of nodes that get affected as small as possible. To achieve this goal, we first remove the edge under consideration from the network, and then do a breadth first search from one of the branch nodes. If we do not reach the other node in a predefined number of levels, we say that the removal of the edge causes too many nodes to be affected and is this not accepted. In this way we ensure that this operation always takes constant time.

The second parameter we define is the *voltage drop* at the nodes feeding the cells. We check the value of the voltage at these nodes using the peak current values of the cells. We assume that all cells are drawing their worst-case currents at the same time. This is a pessimistic approach since it is unlikely that all cells will draw their worst-case peak currents at the same time. Typically each cell will have a given vector (set of inputs) for which it will draw its peak current and these vectors will be different for most cells. Doing a vector-by-vector analysis is too complex for even a small chip. We thus use the vector-less approach – i.e. we know the peak currents for all cells in the library and model these cells as current sources (with the value of their peak currents).

For every branch that we consider for removal, we need to recalculate the voltage at the affected nodes and use this newly calculated value to make sure that no cell tap off point is less than some percentage of V_{DD}. For this, we make use of the fact that the removal of the edge affects the current distribution in the neighborhood and hence only these nodes need to be considered. We define the neighborhood region that is affected by the removal of the branch, and model the nodes bordering this region as voltage sources with value equal to their current voltage (since we assume that their voltages will not be affected). The rest of the neighborhood is kept as it is and this is again fed to SPICE [34] to recalculate the new voltages of all nodes in the neighborhood. Experimental results show that the percentage error in using the neighborhood to calculate the new voltages as compared to the full network is a maximum of 5% in the worst case.

We require connectivity of the power network since otherwise power pads are connected to each other only at the package level and not on the chip network which may cause a high voltage difference between the disconnected regions. For example, if the maximum allowable voltage drop is 10% and two pads are disconnected from each other, then there is the possibility that there is a large voltage difference at the boundaries of the two disconnected regions. This is not a desirable situation and might cause problems such as latchup.

The algorithm to remove branches has been formally stated in Figure 10, Figure 11 and Figure 12.

```
Remove Branches In Power Network
Input:
   1. Branches on the Power Network where BICS need to be placed
   2. $I_{Dmax}$

Output:
   1. A reduced set of branches on the power network

Algorithm:
   1. Sort the branches in ascending order of leakage current in them
   2. For each branch e between node X and node Y in the sorted list:
         a. ReturnVal1 = CheckConnectivity(e)
         b. ReturnVal2 = CheckVoltageDrop(e)
         c. If removal of edge causes current in any of the branches
         d. If (ReturnVal1 == FALSE OR ReturnVal2 == FALSE OR
            ReturnVal3 == FALSE) then do not remove branch e from the
            network else remove e from the network.
   3. End Algorithm
```

Figure 10. Algorithm for Removing Branches from the Power Network.

```
CheckConnectivity
Input:
   1. Branch e between node X and node Y

(LevelMax is the parameter giving the maximum levels the nodes can be
apart on removal of the given edge.)

Output:
   1. Return FALSE if removal of the branch results in loss of
      connectivity of the network else return TRUE

Algorithm:
   1. Do a BFS from node X.
   2. If node Y is not reached within LevelMax levels then Return
      FALSE else Return TRUE
```

Figure 11. Check the Connectivity of the Power Network on Removal of a Given Edge.

```
CheckVoltageDrop
Input:
   1. Branch e between node X and node Y
   2. Min voltage allowed $V_{min}$

Output:
   1. Return FALSE if removal of the branch results in unacceptable
      voltage drop at any tap-off node of the network, else return
      TRUE

Algorithm:
   1. Do a BFS from node X and Y upto LevelMax and get the list L of
      all nodes.upto this level
   2. Represent all nodes neighboring the nodes in L as voltage
      sources with value as the voltage at that node
   3. Keep the network for all the nodes in L the same
   4. Run SPICE on the generated network
   5. Read the new voltage values of nodes
   6. If the new calculated voltage of any node in L is less than $V_{min}$
      Return FALSE else Return TRUE
```

Figure 12. Checking Voltage Drop Violations on Removing a Branch.

6. Optimality of Solutions

Network flow techniques can find the min-cut of a network optimally. Hence, the solution for the BICS locations when the objective is to minimize the number of sensors is optimal. However, if the objective is to minimize the total unmonitored current or to minimize the maximum current in any branch the solution cannot guarantee optimal solutions. The solution approaches are intuitive heuristics to solve these problems.

7. Complexity Analysis

The complexity of this entire process can be analyzed as follows. Calculation of currents given the topology of the power network is essentially solving a set of linear equations and thus can be done in $O(|V|^2)$ in the worst case. In our implementation, we use SPICE to perform the calculations of currents on the power network branches rather than a special purpose solver or IR analysis tool. Transforming the power network into a flow network and labeling it can be done in time $O(|E|)$. The max-flow algorithm takes $O(|V|^3)$

and finding the min-cut from this max-flow takes O(|E|). If the problem formulation is to minimize I_{max} we need to do a binary search and thus need to do the transformation and min-cut calculation multiple times. However, experimental results (see Table IV) show that we do not require more than 11 iterations for any case. Hence for all practical purposes we can take this to be a constant factor. For the case where we remove the edges, each time the number of nodes being considered is bounded by a constant factor of *LevelMax* and hence does not affect the asymptotic complexity of the algorithm. Since in realistic power networks, the number of edges is linear in the number of nodes, the total complexity of the algorithm is O($|V|^3$). However, in the next section we show that in our experiments the algorithm runs in O($|V|^2$) time although we cannot prove this result.

V. EXPERIMENTAL RESULTS AND ANALYSIS

This section describes the details of the experiments conducted and does an analysis of the results. We first give the details of the implementation followed by the test cases and results of various experiments.

1. Implementation Details

We have implemented the above algorithms using the C/C++ languages. All experiments are run on a Pentium 4, 1 GB RAM, 2.26 GHz PC running RedHat Linux 7.3. We solved our circuits using SPICE 3f5 [34] running on this machine.

2. Test Cases

Since most large chip designs are moving towards a hierarchical mesh type supply network these are the types of power networks that we consider. The structure of the power networks is shown in Figure 13 and Figure 14. The first figure gives the structure of the power network at the higher level metal layers. The shaded lines are the power lines and the un-shaded lines are the ground lines. The horizontal and vertical lines are connected together using vias as shown. Power pads are positioned at regular intervals. Similar to the power pads, ground pads are also positioned at regular intervals. For the sake of clarity the ground pads have not been shown in the figure.

The second figure gives the structure on the lower level network. The thin lines show the metal wires on the lower level network and the thick gray lines are the metal lines on the higher-level network. The cells shown in the figure are actually groups of cells that tap off the lower level metal layer. It is assumed that the tap off points are at the intersection points of vertical and horizontal metal lines. Thus each tap off point corresponds to a single square formed by the intersection of the vertical and horizontal lines at the lower metal layer. As shown in Figure 14, the number of rows and columns in the lower level layer per square of the upper layer is kept constant at 10. We change the number of rows and columns in the higher layers to obtain different power networks of different sizes. The details of these power networks are given in Table II.

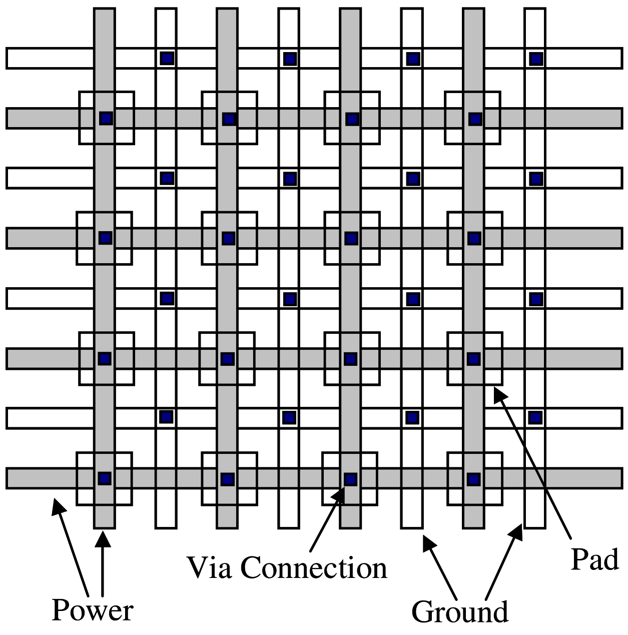

Figure 13. Structure of the Power Network at the Higher Metal Layers.

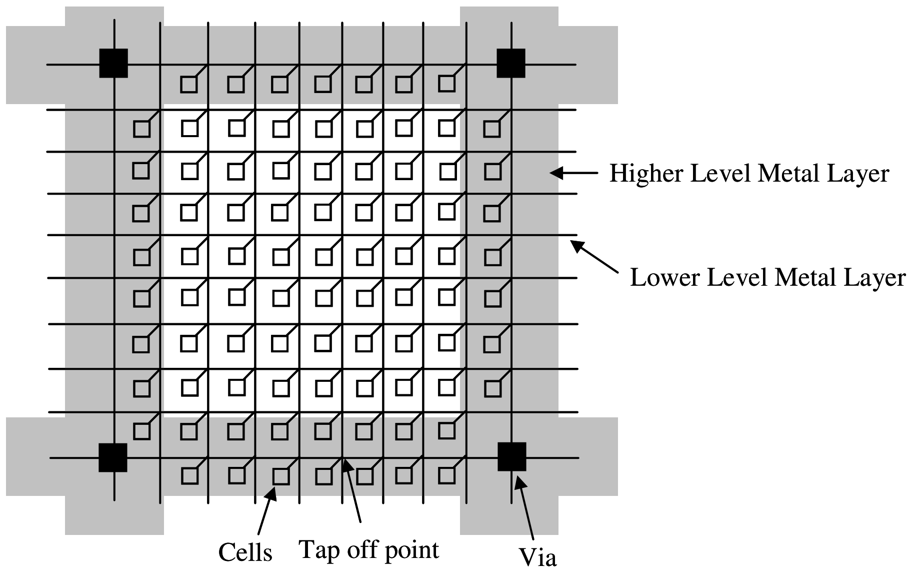

Figure 14. Structure of the Power Network at the Lower Metal Layers.

TABLE II. TEST CASES

Chip No.	Number of Rows/Columns in Higher Layer	No. of Tx. (Millions)	No. of Power Pads	Total Fault Free I_{DDQ} Current (mA)
1	5	1.5	9	0.69
2	10	5.8	19	2.78
3	15	12.8	56	6.23
4	20	22.8	76	11.1
5	25	35.5	141	17.3
6	30	51.1	171	24.9
7	35	69.6	264	33.9
8	40	90.9	304	44.3
9	45	115.1	425	56.1

The chip number given in the first row of Table II is used to identify each network in subsequent tables and figures. The second column of the table gives the number of rows (or columns, both are kept the same) at the higher metal layers. The third column gives the total number of transistors in the chip. In the next row we give the number of power pads in the chip. In the last row we give the total chip leakage current.

While generating the networks we fill cells in each small square of the lower layer from a cell library that has details of the number of transistors and leakage current. There is an upper bound on the maximum number of transistors in such a small square. Hence, the number of transistors in each square will vary depend on the cells in that region. Accordingly the current being tapped off at any point will also vary. The higher layer corresponds to Metal 5 & 6 on a real chip. Metal 3 & 4 are formed using the lower level layers. The tap off points correspond to anything at Metal 2 and below.

The cell library is built using details from the data given for the 130 nm technology in Table I (i.e. 2.7nA leakage/transistor). It contains 20 cells containing 2 to 80 transistors.

The average leakage current per cell is assumed with 25% of the transistors leaking. Cells are randomly selected and placed in each square at the lower levels.

3. Experimental Results

We now present results for various experiments conducted on these test cases for the different formulations of the power supply partitioning problem.

a. Minimizing number of BICS

This is the most straightforward problem formulation where we find the BICS locations for all the chips with a single I_{Dmax} value. Table III shows the results for such an experiment with I_{Dmax} = 100 µA. Besides giving the number of BICS locations, the percentage area overhead of the solution is also calculated. The first such column is calculated assuming each BICS has an overhead of 500 transistors. However, in a typical BICS, the majority of the transistors are in the scan chain readout (~400 transistors). It is possible then to share these scan chains across BICSs. If we assume that 4 BICSs share a single scan chain then the overhead for each BICS is only 200 transistors. However, it must be understood that only one of these four can be operated at a given time. In such a case the test time will increase 4 times. Depending on which is more crucial for a chip the designer will need to make a decision as to which method to adopt. From the table we can see that the area overhead for all chips is low.

The basic cell density throughout the chip is approximately uniform since the cells are randomly placed. As a result it is expected that for a fixed value of I_{Dmax} similar area overheads be obtained for all the circuits. The slight variations that are seen are only due to random circuit differences.

TABLE III. POWER SUPPLY PARTITIONING TO MINIMIZE THE NO. OF BICS (I_{Dmax} = 100 µA)

Chip Number.	Number of BICS locations	% overhead (500 tx/BICS)	% overhead (200 Tx/BICS)
1	19	0.88%	0.35%
2	43	0.49%	0.20%
3	138	0.71%	0.28%
4	226	0.65%	0.26%
5	367	0.68%	0.27%
6	485	0.62%	0.25%
7	706	0.66%	0.27%
8	854	0.61%	0.25%
9	1 155	0.66%	0.26%

b. Minimizing the maximum current through any BICS

Table IV below gives the results when the problem formulation is to minimize the maximum current through any BICS. The number transistors for 1% area overhead is calculated by taking 1% of the third column in Table II. Using this calculated value we can compute the number of BICS that correspond to that number of transistors (for both configurations of 500 transistors/BICS and 200 transistors/BICS). In the fourth and eight columns in Table IV we give the number of BICS locations given by our algorithm. In columns five and nine we give the maximum current through any of the BICS locations. The sixth and last columns give the number of iterations taken while doing the binary search for the number of BICS locations. In these experiments the acceptable percentage error we use is 10%, i.e. the algorithm terminates when it finds a solution where the number of BICS locations is within 10% of the maximum calculated value of given BICS locations (n_{max}).

TABLE IV. MINIMIZING I_{max} FOR A 1% OVERHEAD OF BICS AREA

Chip No.	1% Tx Overhead (thousand of tx.)	500 Tx./BICS				200 Tx./BICS			
		No. of BICS for 1% overhead	No. of BICS given by Algo	I_{max} (µA)	Iters	No. of BICS for 1% overhead	No. of BICS given by Algo	I_{max} (µA)	Iters
1	11	22	19	45.2	7	55	52	23.6	3
2	42	84	85	67.6	5	210	210	29.4	7
3	93	186	186	61	4	465	449	27.2	8
4	165	330	332	64.8	6	825	866	28.1	6
5	257	514	543	61.4	4	1 285	1 245	27.6	9
6	369	738	761	65.4	6	1 845	1 947	28	6
7	503	1 006	962	64.8	8	2 515	2 575	27.6	9
8	657	1 314	1 354	65.6	6	3 285	3 387	28	6
9	832	1 664	1 631	64.8	8	4 160	3 848	27.8	11

c. Minimizing the total unmonitored current

The objective in this problem formulation is to minimize the total unmonitored current (I_u) with the BICS locations such that the current in each location is below I_{Dmax} and the number of such locations is below n_{max}. Figure 15 gives the change in the percentage I_{DDQ} that goes unmonitored with the number of BICS for chip number 5. Similar results were obtained for other chips as well. For this experiment, we use an I_{Dmax} value of 61.4 µA taken from column five in Table IV.

As we increase the number of BICS, the percentage of unmonitored I_{DDQ} decreases. From the graph below, we can see that the amount of unmonitored current rises quickly as the number of sensors is reduced. In these benchmark circuits, I_{DDQ} is relatively evenly distributed in the power grid, so there are few monitored branches with insignificant I_{DDQ} current

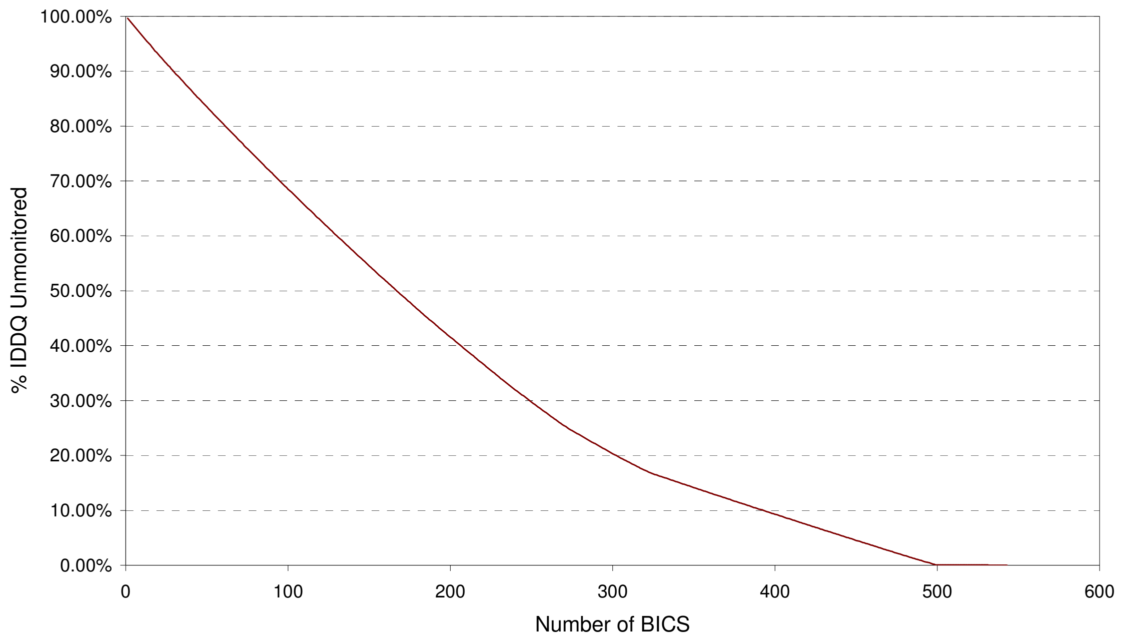

Figure 15. Percentage Unmonitored I_{DDQ} with Change in No. of BICS for Chip 5.

d. Change of BICS locations with I_{Dmax}

We show the variation of the BICS locations generated with different values of I_{Dmax} in the following tables and graphs. In these experiments there is no unmonitored current. Table V gives the change in the BICS locations as I_{Dmax} is varied. It gives the locations as the lower grid (low metal layers), upper grid (high metal layers) and the tap off locations for the cells. We make two observations from this table. Firstly, as I_{Dmax} is increased, the number of BICS locations decreases. The second observation is that with an increase of I_{Dmax} the BICS locations quickly move to the upper layers. Both these observations are expected as with higher I_{Dmax}, the min-cut algorithm is allowed to look at branches carrying greater current and hence considers the branches in the higher layers. Also, monitoring current on the higher metal layers means that fewer BICS will be required on the lower layers. Figure 16 and Figure 17 are plots showing the change in the number of BICS and their locations with change in I_{Dmax}. In Figure 16 the vertical line marks the place where the number of BICS reaches the lower bound of the number of power pads. If the current distribution was uniform we would have reached the lower bound at 122.7 μA (17.3 mA/141 power pads from Table II). However, due to non-uniform distribution of cells, and therefore branch currents, the lower bound is hit at 236.5 μA.

TABLE V. DISTRIBUTION OF BICS LOCATIONS WITH CHANGE IN I_{DMAX} FOR CHIP 5

I_{Dmax} (μA)	Distribution of BICS locations					% BICS on Lower layers
	Lower layers	Tap off locations	Lower + Tap	Upper Grid	Total BICS	
10	3 226	1 087	4 313	97	4 410	97.80%
20	1 306	336	1 642	297	1 939	84.68%
30	512	140	652	408	1 060	61.51%
40	464	124	588	397	985	59.70%
50	352	88	440	346	786	55.98%
60	224	56	280	311	591	47.38%
70	8	2	10	365	375	2.67%
75	0	0	0	367	367	0.00%

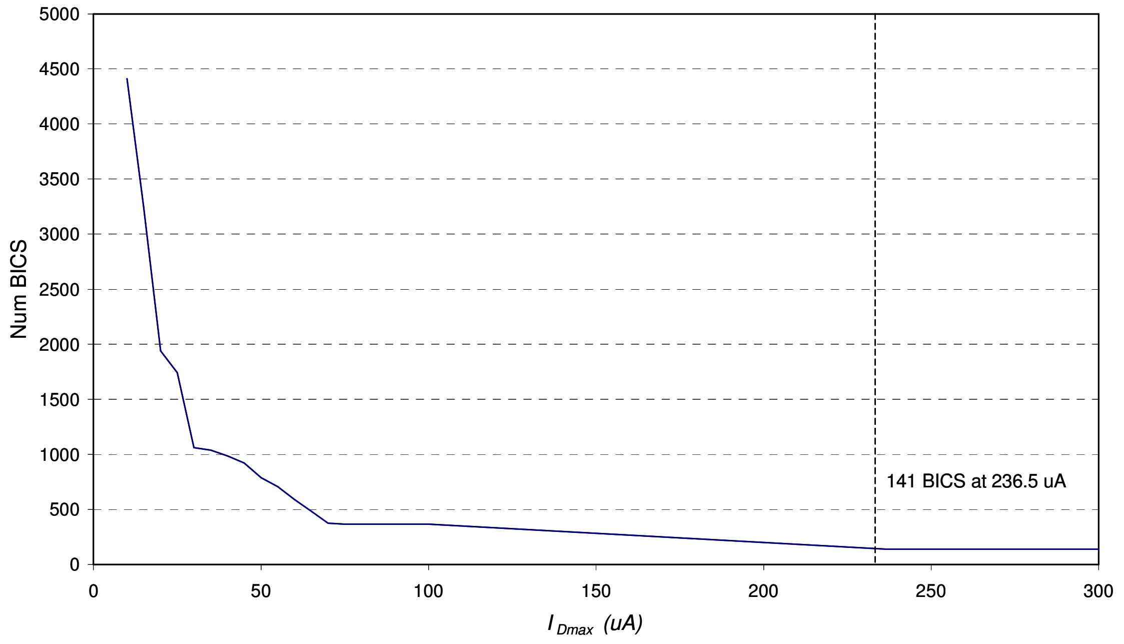

Figure 16. Change of Number of BICS with I_{Dmax} for Chip 5.

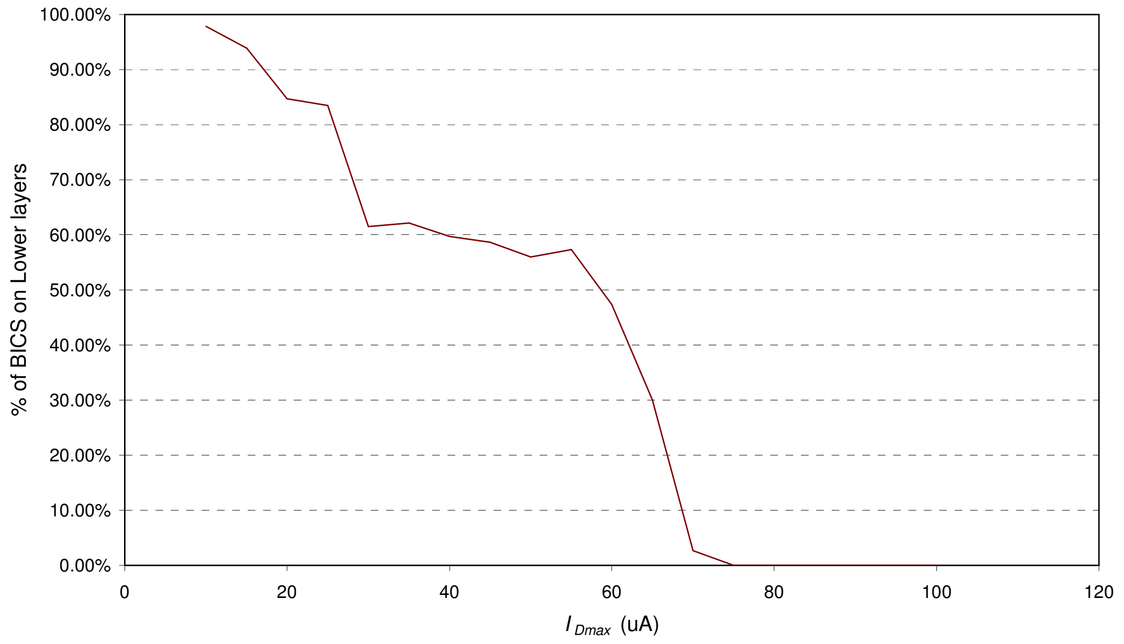

Figure 17. Percentage of BICS on the Lower Layers with Change in I_{Dmax} for Chip 5.

e. Change of I_{max} with allowable transistor overhead

In this experiment we change the allowable maximum percentage transistor overhead for the BICS and observe the change in the maximum current through any BICS. We do this for chip 5 and vary the allowable percentage overhead from 1% to 5%. As in the previous experiments we use an allowable error margin of 10%, i.e. if we obtain a cut within 10% of the calculated value we stop processing. The results are presented in Table VI. It is interesting to see that as the allowable percentage overhead of transistors is increased (i.e. n_{max} is increased), the value of I_{max} falls. This is expected, as the min-cut algorithm will look for cut locations lower in the hierarchy and hence will find smaller current branches.

TABLE VI. CHANGE OF I_{MAX} WITH ALLOWABLE TRANSISTOR OVERHEAD FOR CHIP 5

% Tx overhead	No. of Tx. in BICS	500 Tx./BICS			200 Tx./BICS		
		No. of BICS calculated	I_{max} (μA)	No. of BICS given by algo	No. of BICS calculated	I_{max} (μA)	No. of BICS given by algo
0.25	64 250	129	236.5	141	322	70	367
0.5	128 500	257	140	220	643	56.1	633
0.75	192 750	386	70	367	964	43.3	931
1	257 000	514	65.6	471	1285	27.6	1245
2	514 000	1028	37.5	1024	2570	16.3	2418
3	771 000	1542	27.3	1388	3855	14.1	3472
4	1 028 000	2056	18.6	1943	5140	8.5	5037
5	1 285 000	2570	16.3	2418	6425	7	6247

For the case of 0.25 % overhead and assuming 500 transistors per BICS the result is the lower bound on the number of BICS, i.e. the number of power pads of the chip (141 for Chip no. 5).

Figure 18 shows the same data in the form of a graph. This graph will help the designer decide the strategy to employ for BICS placement.

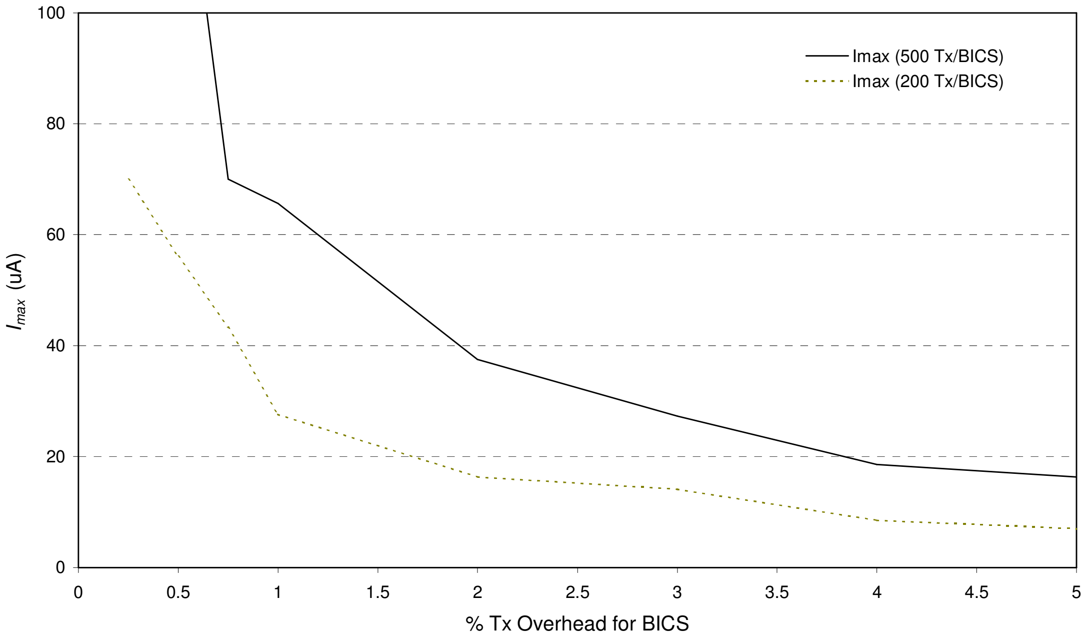

Figure 18. I_{max} with Change in Allowable Percentage Transistor Overhead for Chip 5.

f. Capacity Scaling Effect

The flexibility obtained using a scaling of capacities in the labeling step when transforming the power network into a flow network has been explained earlier. We conducted some experiments to show the effect of this scaling. The scaling scheme we use in this experiment is as follows:

0 - I_{Dmax} => Capacity of edges = 1,

I_{Dmax} – 1.5 · I_{Dmax} => Capacity of edge = 5,

1.5 · I_{Dmax} – 2 · I_{Dmax} => Capacity of edge = 10,

Above 2 · I_{Dmax} => Capacity of edge = Infinity.

To help illustrate the use of the capacity scaling we run our algorithm with I_{Dmax} of 10 µA and 30 µA and I_u = 0. The results for these experiments are given in Table VII. It can be seen from the table that there is a 10% or more reduction with a larger benefit for smaller I_{Dmax} values in the number of BICS with capacity scaling for most chips.

TABLE VII. CAPACITY SCALING EFFECT ON THE NUMBER OF BICS LOCATIONS

Chip Number	I_{Dmax} = 10 μA		I_{Dmax} = 30 μA	
	No. of BICS w/o scaling	No. of BICS w/ scaling	No. of BICS w/o scaling	No. of BICS w/ scaling
1	176	156	52	52
2	738	724	210	170
3	1 604	1 397	397	397
4	2 957	2 685	769	625
5	4 410	3 893	1 060	1 060
6	6 607	5 863	1 707	1 415
7	8 606	7 652	2 041	2 041
8	11 601	10 300	2 932	2 508
9	14 198	12 678	3 340	3 340

g. Using both V_{DD} and GND lines to monitor current

So far in all our discussions, we have only considered monitoring current on the power (V_{DD}) lines. However, for some circuits we extend our methodology to see the results when monitoring currents on both power and ground lines. For the first five chips, we find out the number of BICS locations for I_{Dmax} of 100 μA. We restrict our results to the first five chips due to the memory constraints of SPICE and the circuits it can handle. Having both power and ground circuits makes the entire circuit considerably larger and SPICE could not handle chips 6-9. These results are presented in Table VIII. It can be seen that there is a consistent reduction in the number of BICS required.

TABLE VIII. CHANGE OF NO. OF BICS BY CONSIDERING POWER AND GROUND LINES

Chip No.	No. of BICS using only Power lines	No. of BICS using both Power and Ground lines	% Reduction in no. of BICS using Power and Ground
1	19	13	31.58%
2	43	37	13.95%
3	138	110	20.29%
4	226	196	13.27%
5	367	317	13.62%

h. Removal of branches in the power network

In all the experiments so far, we do not modify the given power network of the chip. However, we observe that by modifying the power network we can obtain much better cuts for our algorithm and thus reduce the area overhead or increase test resolution for the same test quality. The two criteria of *connectivity* and *voltage drop* are used to decide whether a branch is removed from the power network. For the connectivity, we assume that if a removal of a branch disconnects two nodes such that the shortest path between these two nodes goes through more than 10 intermediate nodes, then the branch cannot be removed. For the voltage drop we allow a maximum drop of 10% of V_{DD}. This may seem too high, but we are using worst case analysis and is thus acceptable.

The results for I_{Dmax} value of 100 μA are presented in Table IX. We observe that there is a considerable reduction in the number of branches for all chips. Figure 19 presents a breakdown of the reasons for the BICS locations that are not removed. We observe that in most chips the reason the BICS location is not removed is equally divided between voltage drop violations and connectivity violations.

Figure 20 shows the sensitivity of the number of BICS locations with change in the maximum level of connectivity allowed for Chip 5. For this experiment, the number of edges not removed because of voltage drop violations remained the same. The number of edges not removed due to connectivity violations obviously reduces with looser connectivity, i.e. increasing value of MaxLevel in the algorithm. We see that increasing the connectivity level beyond 10 does not affect the results. This result is observed for all other chips (for some chips the results stop changing at a level of 8). This can be explained by examining the structure of the power grid in the benchmark circuits. Given a node in the grid, the maximum number of intermediary nodes between it and V_{DD} or GND is 10. It is for this reason that we use a MaxLevel value of 10 in the results shown in Table IX.

TABLE IX. REMOVAL OF BRANCHES ON THE POWER NETWORK (I_{DMAX} = 100µA)

Chip No.	No. of BICS		% Reduction in the no. of BICS
	W/o edge removal	W/ edge removal	
1	19	9	52.63%
2	43	34	20.93%
3	138	67	51.45%
4	226	167	26.11%
5	367	220	40.05%
6	485	358	26.19%
7	706	446	36.83%
8	854	637	25.41%
9	1 155	750	35.06%

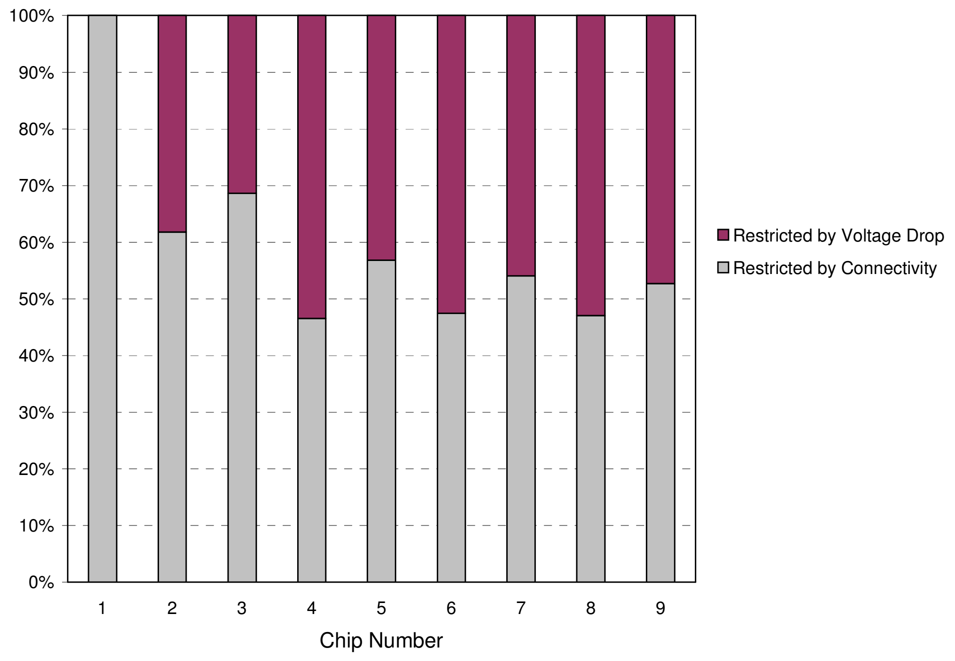

Figure 19. Reason for BICS Locations Not Removed.

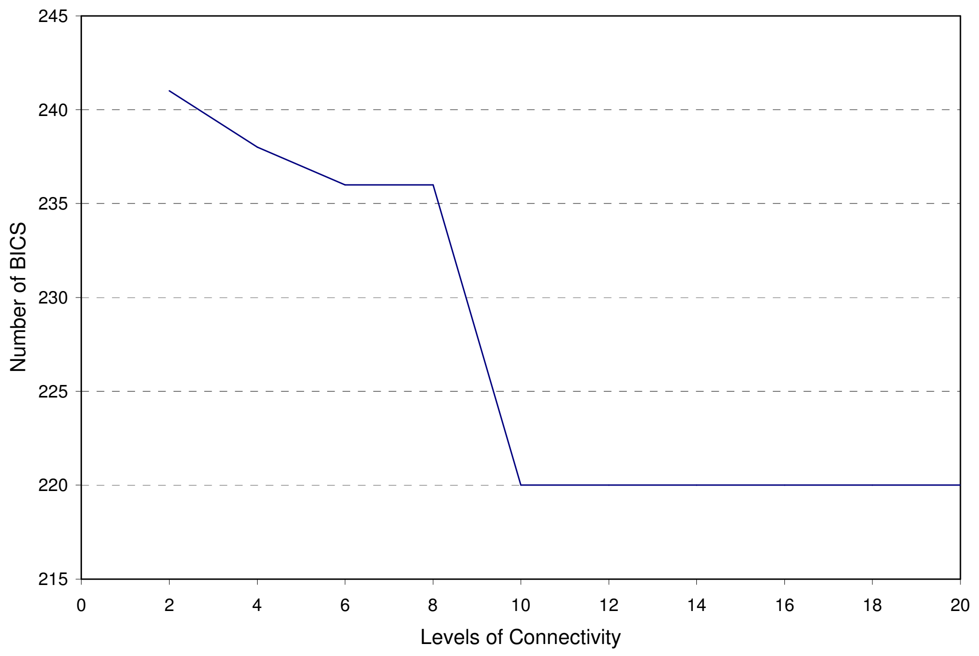

Figure 20. Variation of Number of BICS with Change in Connectivity for Chip 5.

i. Running time of the min-cut algorithm

The running time of the min-cut algorithm is O(|*V*|^3) in the worst case. However our experience with the algorithm shows that in practice the complexity is closer to O(|*V*|^2). The intuition is that the transformed flow network of the power grid will have bounded degree on the vertices that leads to this reduction in complexity.

In Figure 21 we show the comparison of the actual runtime vs. O(|*V*|^3) (V cube) and O(|*V*|^2) (V square). The plots of V cube and V square have been generated by using a constant, such that their time is the same for chip 5. This corresponds to number of vertices equal to approximately 64 000. The growth of the function is shown after that value. We can clearly see that the growth of the actual run time is less than the V square plot.

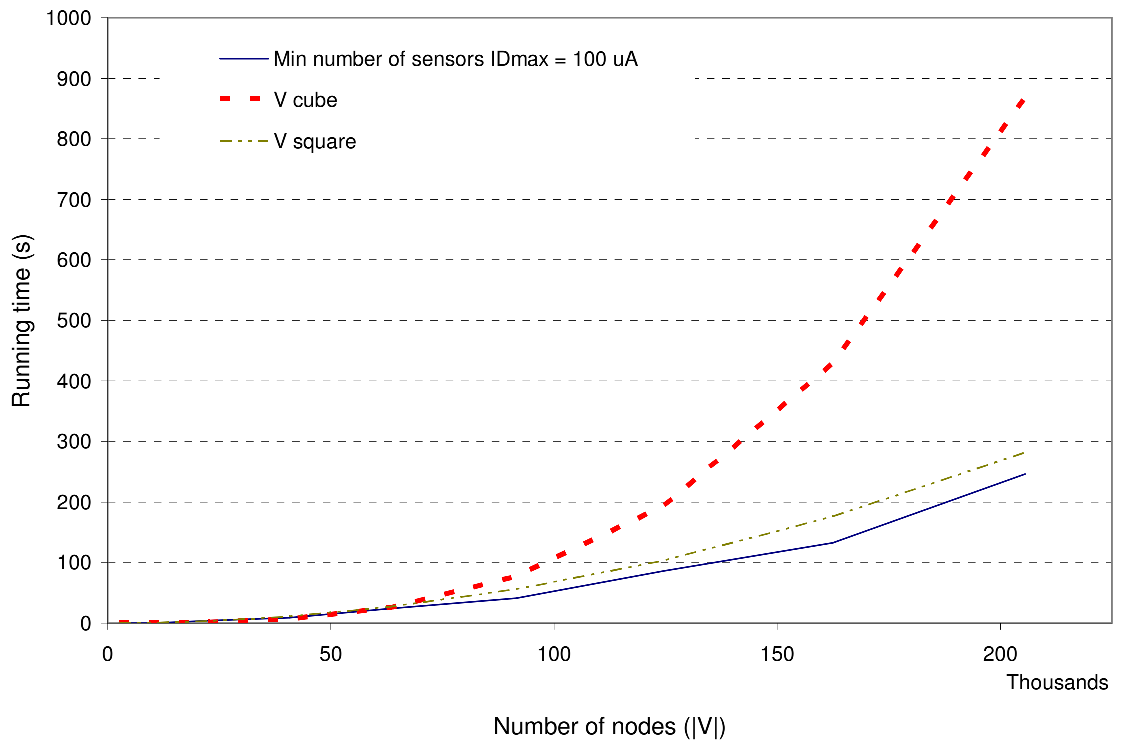

Figure 21. Comparison of Actual Running Time with V cube and V square plots.

j. Comparison of results for pads as a periphery and as an array

To show the difference in performance of the min-cut algorithm for different configurations of power pads we use a different set of benchmarks and show results for these benchmark circuits when the objective is to minimize the number of sensors. These benchmark circuits are described in Table X. The difference between these benchmark circuits and the previous benchmarks is that this is a power grid with a single hierarchy, i.e. the tap off points from the grid are assumed to be from the higher layers itself. The number of power pads in both the different configurations (power pads as an array and on the periphery of the chip area) is given. These circuits are again generated using ITRS 2002 predictions for 130 nm. Since the cells in a given chip are the same irrespective of the power pad configurations, the total chip leakage current given in the table is also same for both configurations.

TABLE X. BENCHMARKS FOR DIFFERENT POWER PAD CONFIGURATIONS

Chip No.	No. of Tx. (Millions)	Rows/ Cols on Mesh	No. of Power Pads using Periphery	No. of Power Pads using Array	Chip I_{DDQ} (mA)
1	2	20	40	80	2
2	13	60	120	240	17
3	37	100	200	400	48
4	72	140	280	560	92
5	118	180	360	720	151
6	146	200	400	800	186

Table XI and Table XII give the results for minimizing the total number of BICS for I_{Dmax}=100 μA. The mesh signifies the higher layer mesh. It is clear that the array type power supply network always requires fewer sensors than the peripheral configuration. We observe that for the periphery topology the percentage of sensors on the mesh is fewer than in the array topology. This is due to the structure of the periphery power network that causes non-uniform distribution of currents in the branches of the mesh. The middle regions of the chip carry small currents whereas those closer to the periphery carry larger currents. As a consequence many sensors will be underutilized, measuring currents less than I_{Dmax}. By making the pads as an array, current flow in the branches of the power network becomes more uniform. As a result there are fewer branches carrying very low currents and sensors can be pushed to their limits, thus decreasing the total number of required sensors. For the scan chain optimization, we assume we can share one scan chain across 4 sensors resulting in a 2.5 times reduction in the total transistor overhead of the BICS (assuming 400 out of the total of 500 transistors of the BICS in the scan chain).

TABLE XI. MINIMIZING NUMBER OF BICS FOR ARRAY PADS CONFIGURATION

Chip No.	Minimum Number of Sensors	Number of BICS given by our Algorithm			% Area Overhead without Scan Optimization	% Area Overhead with Scan Optimization
		On the mesh	Not on the mesh	Total		
1	40	40	0	40	1.6	0.6
2	170	815	177	992	3.9	1.5
3	480	3 513	1 274	4 787	6.4	2.5
4	920	7 596	3 436	11 032	7.6	3.0
5	1 510	13 299	6 503	19 802	8.4	3.4
6	1 860	16 528	8 558	25 086	8.6	3.5

TABLE XII. MINIMIZING NUMBER OF BICS FOR PERIPHERAL PADS CONFIGURATION

Chip No.	Minimum Number of Sensors	Number of BICS given by our Algorithm			% Area Overhead without Scan Optimization	% Area Overhead with Scan Optimization
		On the mesh	Not on the mesh	Total		
1	80	110	3	113	3.7	1.5
2	240	303	13	316	1.2	0.5
3	480	1 826	160	1 986	2.7	1.0
4	920	3 980	618	4 598	3.2	1.3
5	1 510	7 267	1 573	8 840	3.7	1.5
6	1 860	8 867	2 246	11 113	3.8	1.5

VI. CONCLUSIONS AND FUTURE WORK

In this research we have formulated and proposed solutions for various power supply partitioning problems. This will be the first step in the placement of BICS for I_{DDQ} testing as it identifies locations on the power grid where BICS should monitor current. Depending on the constraints and requirements of a particular design, one of the problem formulations can be chosen. Given a problem formulation and the required parameters (maximum permissible current if you want to minimize the number of BICS, or maximum number of BICS if you want to minimize the maximum current through any BICS, etc.) the algorithm will give the locations on the power grid where BICS need to be placed to meet the constraints. This will serve as an extremely useful tool whether the placement of BICS is to be automated or hand placed (which might be possible if the number of BICS is small). It also gives complete flexibility to the user to decide what the tightest constraint is in any particular case.

A logical extension of this work would be work on strategies to integrate this work into the physical design flow for a chip. This would require the following areas to be researched:

- Back-annotation of the power grid locations onto the actual layout. Since our algorithm will give the locations on a SPICE netlist, there must be a way of mapping this branch of the power grid onto the physical layout.
- Actual placement of the BICS into the layout. For this we will need to know the layout of the BICS cell and we must also be able to identify empty locations in the placed chip layout where these layouts can be placed. If the BICS is a magnetic sensor then it requires to be directly under the branch on the power network. In such a case the problem is very tough as it would involve perturbing the entire layout. In the case of a voltage drop sensor, all we need to do is find an empty space in the vicinity of the power branch to be monitored. This should be a relatively easy task given that the total area of the BICS will be less than 1% of the total area of the chip.

REFERENCES

[1] International Technology Roadmap for Semiconductors, 2002 Update, Semiconductor Industry Association, [online] http://public.itrs.net.

[2] C. Thibeault, "On the Comparison of ΔI_{DDQ} and I_{DDQ} Testing," in *Proc. VLSI Test Symposium*, Apr. 1999, Dana Point, CA, pp. 143-150.

[3] P. C. Maxwell, P.O'Neill, R.C. Aitken, R. Dudley, N. Jaarsma, et al., "Current Ratios: A Self-scaling Technique for Production I_{DDQ} Testing," in *Proc. International Test Conference*, Atlantic City, NJ, Sep. 1999, pp. 738-746.

[4] A. Gattiker and W. Maly, "Current Signatures: Application," in *Proc. International Test Conference*, Washington, DC, October 1997, pp. 156-165.

[5] C. Thibeault, "A Novel Probabilistic Approach for IC Diagnosis Based on Differential Qiescent Current Signatures," in *Proc. VLSI Test Symposium*, Monterey, CA, Apr. 1997, pp. 80-85.

[6] S. S. Sabade and D. M. H. Walker, "Neighbor Current Ratio (NCR): A New Metric for IDDQ Data Analysis," in *Proc. International Symposium on Defect and Fault Tolerant Systems*, Vancouver, Canada, Nov. 2002, pp. 381-389.

[7] S. S. Sabade and D. M. H. Walker, "Immediate Neighbor Difference IDDQ Test (INDIT) for Outlier Identification," in *Proc. VLSI Design Conference*, New Delhi, India, Jan. 2003, pp. 361-363.

[8] M. Sachdev, "Deep Sub-micron I_{DDQ} Test Options," in *Proc. International Test Conference*, Washington, DC, Oct. 1996, p. 942.

[9] A. Keshvarzi, K. Roy and C.F. Hawkins, "Intrinsic Leakage in Low Power Deep Submicron CMOS ICs," in *Proc. International Test Conference*, Washington, DC, Nov. 1997, pp. 146-155.

[10] T. Karnik, Y. Ye, J. Tschanz, L. Wei, S. M. Burns, et al., "Total Power Optimization by Simultaneous Dual-V_t Allocation and Device Sizing in High Performance Microprocessors," in *Proc. Design Automation Conference*, New Orleans, LA, Jun. 2002, pp 486-491.

[11] G. Sery, S. Borkar and V. De, "Life Is CMOS: Why Chase the Life After?," in *Proc. Design Automation Conference*, New Orleans, LA, Jun. 2002, pp. 78-83.

[12] W. Maly and P. Nigh, "Built-in Current Testing – Feasibility Study," in *Proc International Conference Computer Aided Design*, Santa Clara, CA, Nov. 1988, pp.340-343.

[13] W. Maly and M. Patyra, "Built-in Current Testing," *IEEE Journal of Solid State Circuits*, vol. 27, no. 3, pp. 425-428, Mar. 1992.

[14] D. M. H. Walker, "Requirements for Practical I_{DDQ} Testing of Deep Sub-micron Circuits," in *Proc. IEEE International Workshop on Defect and Current Based Testing*, Montreal, Canada, April 2000, pp. 15-20.

[15] W. R. Daasch, K. Cota and J. McNames, "Neighbor Selection for Variance Reduction in I_{DDQ} and Other Parametric Data," in *Proc. International Test Conference*, Baltimore, MD, Oct. 2001, pp. 92-100.

[16] H. Kim and D. M. H. Walker, "A Practical Built-in Current Sensor for I_{DDQ} Testing," in *Proc. International Test Conference*, Baltimore, MD, Oct 2001, pp. 405-414.

[17] H. Kim and D. M. H. Walker, "Current Sensor System to Measure Integrated Circuit Current (I_{DDQ})," Invention Disclosure 1539TAMUS00, U.S. Provisional Patent Application 60/250,735, Texas A&M University System, 2000.

[18] F. Li and L. He, "Maximum Current Estimation Considering Power Gating," in *Proc. International Symposium on Physical Design*, Apr. 2001, Sonoma, CA, pp. 106-111.

[19] G. Moore, "Cramming More Components Onto Integrated Circuits," *Electronics*, vol. 38, No. 8, April 19, 1965, [online] http://www.intel.com.

[20] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," W. H. Freeman, New York, 1979.

[21] J.P. Roth, "Diagnosis of Automata Failures: A Calculus and a Method," *IBM Journal of Research and Development*, vol. 10, no.4, pp. 278-291, July 1996.

[22] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Transactions on Computers*, vol. C-30, no. 3, pp. 215-222, Mar. 1981.

[23] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Transactions on Computers*, vol. C-32, no. 12, pp., Jun 1983, pp.1137-1144, Dec. 1983.

[24] A. Keshavarzi, K. Roy and C. Hawkins, "Intrinsic Leakage in Low Power Deep Sub-micron CMOS ICs," in *Proc. International Test Conference*, Washington, DC, Oct. 1997, pp. 146-155.

[25] T. A. Unni and D. M. H. Walker, "Model-based I_{DDQ} Pass/Fail Limit Setting," in *Proc. International Workshop on I_{DDQ} Testing,* San Jose, CA, Nov. 1998, pp. 43-47.

[26] S. Sabade and D. M. H. Walker, "Improved Wafer-level Spatial Analysis for I_{DDQ} Limit Setting," in *Proc. International Test Conference*, Baltimore, MD, Nov. 2001, pp. 82-91.

[27] B. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning of Electrical Circuits," *Bell System Technical Journal*, pp. 291-307, Feb. 1970.

[28] C. M. Fiduccia and R. M. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions," in *Proc. Design Automation Conference*, Jun. 1982, Las Vegas NV, pp.175-181.

[29] J. R. Ford and D. R. Fulkerson, "Flows in Networks," *Princeton University Press*, Princeton, NJ, 1962.

[30] J. Edmonds and R. M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *Journal of ACM*, vol. 19, no. 2, pp. 248-264, Apr. 1972.

[31] A. V. Goldberg and R. E. Tarjan, "A New Approach to the Maximum Flow Problem," *Journal of ACM*, vol. 35, no. 4, pp.921-940, Oct. 1988.

[32] A. V. Karzanov, "Determining the Maximal Flow in a Network by the Method of Preflows," *Sov. Math. Dokl.*, 15, pp. 434-437, 1974.

[33] H. Yang and D. F. Wong, “Efficient Network Flow Based Min-Cut Balanced Partitioning,” in *Proc. IEEE/ACM International Conference on Computer Aided Design*, San Jose, CA, Nov. 1994, pp. 50-55.

[34] SPICE, [online] http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/ accessed on 01 July 2003.

VITA

Abhijit Prasad was born in Mumbai, India in 1976. He completed his Bachelor of Engineering degree in electrical and electronics engineering from the Regional Engineering College, Trichy, India in May 1998. He subsequently worked for three years as a software engineer before starting his graduate studies as a computer engineering major at Texas A&M University in the Fall of 2001. His research interests are in computer aided design for VLSI physical design and testing. He can be reached at the following email address: abhijit_prasad@hotmail.com. His permanent address is c/o Dr. Gyan Hajela, 20926 Wolfe Way, Woodland Hills, CA 91364.