Toward PDN Resource Estimation: A Law of General Power Density

Kwangok Jeong
ECE Department, UC San Diego
La Jolla, CA 92093, USA
kjeong@ucsd.edu

Andrew B. Kahng
ECE and CSE Departments, UC San Diego
La Jolla, CA 92093, USA
abk@ucsd.edu

ABSTRACT

The power distribution network (PDN) is an increasingly significant consumer of on-chip interconnect resources. Thus, PDN estimation is increasingly central to system-level interconnect prediction for modern ICs. PDN design and verification require accurate power estimation and realistic current source distribution across a die. However, at early design stages, detailed placement or switching information is rarely available, so that designers either rely on pessimistic overdesign, which can lead to severe routing congestion, or encounter unexpected voltage noise problems at late design stages, which can lead to costly design iterations. In this work, we seek to identify a general trend for power density. From both empirical and analytical studies on random activity distributions, we propose a power law of activity density, which can potentially enable estimates of power density and voltage noise, as well as of required power distribution network (PDN) resources, in early design stages.

1. INTRODUCTION

Due to the increase in transistor density and operating frequency, power consumption has also been increased. While operating voltage reduction can quadratically reduce power consumption, voltage margin is also reduced and the impact of voltage noise (e.g., IR-drop, and Ldi/dt) increases. To suppress the voltage noise, a large portion of on-chip interconnect resources is dedicated to the power distribution network (PDN). Major issues with PDN design are (1) how *accurately* and (2) how *early* power can be estimated: overestimation wastes valuable routing resources, while underestimation results in parametric yield loss as low-Vcc faults. Since PDN design is a very early step in the IC implementation flow, excessive voltage noise observed in late design stages due to insufficient PDN design will cause costly design iterations.

Power is estimated at all levels of design abstraction. Electronic system level (ESL) estimation typically exploits a history of measured power of IPs. Register-transfer level (RTL) tools can apply quick-synthesis method before power estimation. For gate-level netlists, more accurate power estimation techniques are available given accurate parasitic estimates and SPICE-characterized power models.

At all design levels, the most important input is the switching activity information. Switching activity information can be dumped from stimuli-based functional simulations and then fed to power estimation tools via .vcd, .saif, .fsdb etc. formats. However, functional simulation requires large runtime and substantial data storage. Finding a representative stimulus is time-consuming, and guarantees of quality or relevance are difficult to establish. Due to these difficulties, at early design stages (or even at signoff stages) *vectorless* power estimation is used. This approach relies on statistical assumptions for the switching probabilities of primary input ports; the switching probabilities are then propagated into internal logic with consideration of functionality. Existing statistical techniques in power estimation include transition density [7], probabilistic waveform [9] and *binary decision diagram* (BDD) [4] based techniques [8]. Probabilistic waveforms [9] provide more details in time domain compared with signal probabilities on signal switching activities. BDDs [4] better capture signal correlations arising from reconvergence in a netlist.

For early-stage PDN design, another important issue is where the estimated power values should be distributed in a die. An estimated lumped power value of a functional block can be assumed to be evenly distributed to all nodes in an R(L)C PDN network, or a single current source can be assumed to be placed at the center of a target region of the PDN network. However, the former does not account for worst-case power distribution, and the latter can be too pessimistic and result in overdesign. To achieve a realistic current source distribution assumption, trends for current density must be understood at multiple length scales. Coarse-grain current densities determine C4 bump pitch and the amount of on-chip routing resources required for the power distribution network. Local power densities contribute to voltage noise hotspots, with increased voltage gradients then causing local on-chip performance variations.

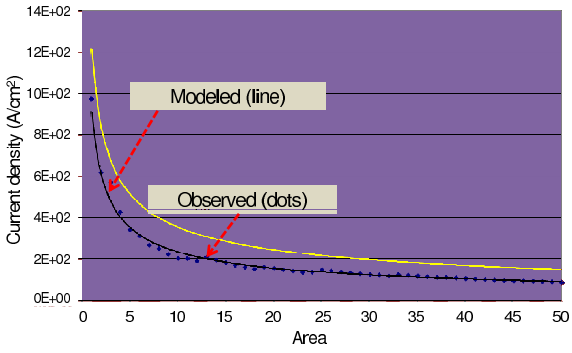


Figure 1: Power density with respect to sample area [11].

Intuitively, local power density should increase exponentially as area reduces. For example, an LVT clock buffer (occupying only a couple of square microns) may dissipate up to tens of microwatts, even as the entire SOC chip (occupying a square centimeter) dissipates just one watt. One motivation for our work is from Figure 1, which shows local power density with respect to normalized area, reported from an industry source [11]. The curve is described as

$$I/A = c_1 A^{c_2 e^{-c_3 \alpha}} \quad (1)$$

where I/A denotes the current consumption I within area A, and α represents the activity factor of the design. Coefficients c_1, c_2 and c_3 capture design characteristics [11]. Although the curve does

not explain any physical mechanisms, it is shown that the measured data (i.e., blue dots) closely follow the fitted power function (i.e., blue curve).[1]

A second motivation for our work is the well-known *Rent's rule*, a simple, empirical power-law relationship between the number of I/O terminals T for a logic block and the number of gates N contained in that block [6]:

$$T = kN^p$$

where k and p are empirical parameters. In the field of interconnect prediction, Rent's rule has provided a powerful foundation to various subfields such as wirelength distribution estimation [2] [3], fanout distribution estimation [13], and via count estimation [12] [5].

Motivated by the power law relationship between the power density and area as shown in Figure 1, and the variety of applications of the power law-based Rent's rule, we seek a power law for power density. In this work, we consider (1) the maximum possible activity density for a given area, and (2) how activity density changes with time. We also (3) seek to identify a general trend for the activity density, so as to enable estimates of power density and voltage noise, as well as of required power distribution network (PDN) resources, in early design stages.

2. DENSITY OF RANDOM ACTIVITY

Intuitively, given an average activity factor of a design, the activity density can change with the area and the location of the sampling area. Figure 2 illustrates the sampling of area and the changes of the activity density. In addition, activity density can change with time. Figure 3 illustrates the temporal dependency of activity density. At each time (e.g., clock cycle) t the activity can change due to the design's functional activity.

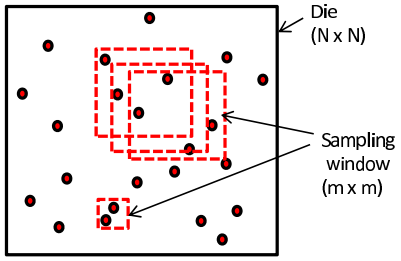


Figure 2: Spatial sampling for power density calculation. Power density changes due to the sampling area and location.

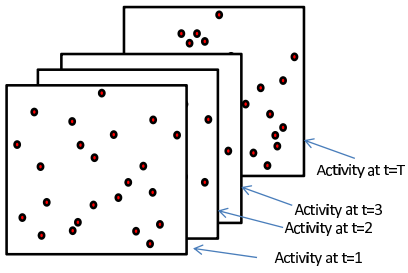


Figure 3: Temporal sampling for power density calculation. Power density changes due to the sampled timeframe.

[1]Coefficients in Equation (1) can vary depending on the design types and activity factor α. The yellow curve shows a 3σ maximum power density observed from various designs with different activity factors.

We evaluate maximum activity density of a design using an uncorrelated activity model, as follows. Average activity factor ($0 \leq p \leq 1$) of a design is the average number of toggled gates (or nets) per cycle in the entire design. The maximum activity density $d(m)$ is defined as the maximum activity observed within $m \times m$ ($1 \leq m \leq N$) windows among all possible $m \times m$ windows in the design. To remove the dependency to the design size, we define a normalized maximum activity density $d_{norm}(m)$ as $d(m)/d(N)$.

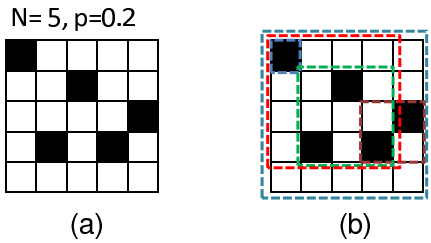
2.1 Activity Density Calculation

Given an $N \times N$ array that represents a design having an average activity factor p, we find the maximum activity density over all $m \times m$ regions using the procedure in Figure 4. For instance, for a design with $p = 0.2$, and $N = 5$ as shown in Figure 5(a), we find the maximum activity density for each value of m from 1 to 5, while moving the $m \times m$ window. (Trivially, when $m = 1$, $d(1) = 1$.) Examples of maximum activity density windows are shown in Figure 5(b), and the corresponding maximum density and normalized density are shown in Figure 5(c).

Procedure: MaximumActivityDensity
Inputs: activity information $\in \{0,1\}$ for each element in $N \times N$ array

for $m = 1$ to N
  $c(m) \leftarrow$ maximum activity count enclosed by $m \times m$ window
  $d(m) \leftarrow c(m)/m^2$
end
for $m = 1$ to N
  $d_{norm}(m) \leftarrow d(m)/d(N)$
end

Figure 4: Calculation of maximum activity density.



c(1) = 1 → d(1) = 1.00 → d_{norm}(1) = 5.00
c(2) = 2 → d(2) = 0.50 → d_{norm}(2) = 2.50
c(3) = 3 → d(3) = 0.33 → d_{norm}(3) = 1.32
c(4) = 4 → d(4) = 0.25 → d_{norm}(4) = 1.25
c(5) = 5 → d(5) = 0.20 → d_{norm}(5) = 1.00

(c)

Figure 5: Activity density calculation. (a) 5×5 grids of a given design with average activity factor $p = 0.2$. (b) Maximum density windows for different window sizes (m). (c) Maximum activity density and normalized maximum activity density calculation.

2.2 Maximum Activity Calculation for Artificial Data

To study activity density in large grid arrays and over many timesteps, an efficient counting method is required. Figure 6 shows an efficient algorithm to calculate the switching activity in an $m \times m$

window with bottom-left corner (x, y), from the given activity information *countInBox*1. The value of *countInBox*1 for location (x, y) is 1 if there is switching activity at that location, and 0 otherwise. The proposed algorithm iterates over m from 1 to N for all possible locations of m-sized windows within the $N \times N$ grid array, and computes the counts of activities in larger-sized windows from the previously-computed counts of smaller-sized windows, based on the dynamic programming approach.

Procedure: MaxActivityCount (N)
Inputs: Size of the grid N

```
for x = 1 to N
    for y = 1 to N
        CA[x][y][1] = CountInBox1 (x,y)
    end
end
for m = 2 to N
    for x= 1 to N
        for y = 1 to N
            if ( m % 2 == 0 )
                lb ← CA[x][y][m/2]
                lt ← CA[x][y+m/2][m/2]
                rb ← CA[x+m/2][y][m/2]
                rt ← CA[x+m/2][y+m/2][m/2]
                CA[x][y][m] = lb+lt+rb+rt
            else
                lb ← CA[x][y][m/2]
                lt ← CA[x][y+m/2][m/2+1]
                rb ← CA[x+m/2][y][m/2+1]
                rt ← CA[x+m/2+1][y+m/2+1][m/2]
                cn ← CA[x+m/2][y+m/2][1]
                CA[x][y][m] = lb+lt+rb+rt−cn
            end
        end
    end
end
```

Figure 6: Dynamic programming-based activity counting.

3. EMPIRICAL MODEL FROM POWER DENSITY CALCULATIONS

Due to the random nature of activity distribution, we find an average maximum activity density from multiple trials of activity distribution. Figures 7 and 8 show our design of experiments for a single timeframe and multiple timeframes, respectively, for various activity factors (p) and various sizes of timeframes (w).

3.1 Normalized Activity Density

Figure 9 shows the normalized maximum activity density for different average activity factors. We observe that when m = 1, the normalized maximum activity density $d_{norm}(1)$ is roughly $1/p$, and as m increases, $d_{norm}(m)$ decreases and converges to 1. Figure 10 shows the normalized activity density changes with respect to the number of timeframes. We observe that as the number of timeframes considered increases, the normalized maximum activity density $d_{norm}(m)$ decreases, with the rate of decrease depending on the given average activity factor (p).

Procedure: FindAverageMaximumActivity-SingleTimeframe
Inputs: N = 100, P = {0.05, 0.10, 0.15, 0.25, 0.50, 0.90}, k = 50

```
foreach p ∈ P
    generate k random activity maps on the N × N grids
    calculate average activity count c(m) from k maps
    calculate average maximum activity density d_norm(m)
end
```

Figure 7: Calculation of average maximum activity density $d_{norm}(m)$ for a single timeframe.

Procedure: FindAverageMaximumActivity-MultipleTimeframe
Inputs: N = 100, P = {0.05, 0.10, 0.15, 0.25, 0.50, 0.90}, k = 50, W = {1, 2, 4, 8, 32 }

```
foreach p ∈ P
    generate k random activity maps on the N × N grid
    foreach w ∈ W
        select w activity maps from k random activity maps
        accumulate activities for each grid for w maps
          (i.e., pointwise-summation)
        calculate average activity count c(m) from w maps
        calculate average maximum activity density d_norm(m)
    end
end
```

Figure 8: Calculation of average maximum activity density $d_{norm}(m)$ for multiple timeframes.

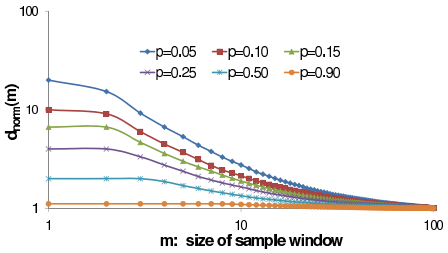


Figure 9: Average normalized maximum activity density $d_{norm}(m)$ from experiments in Figure 7, for a single timeframe.

Important characteristics of the maximum activity density are summarized as follows.

(1) $d_{norm}(m)$ exponentially decreases as the area ($m \times m$) increases.

(2) When m = 1 and w = 1, d_{norm} = $1/p$.

(3) When m increases, $d_{norm}(m)$ converges to 1.

(4) As the number timeframe (w) increase, $d_{norm}(m)$ decreases.

(5) The decay rate of $d_{norm}(m)$ depends on p; for large p, the rate is small, but for small p, the rate is large.

(6) $d_{norm}(m)$ decreases slowly as w increases.

(7) As p decreases to near zero, $d_{norm}(m)$ diverges to infinity.

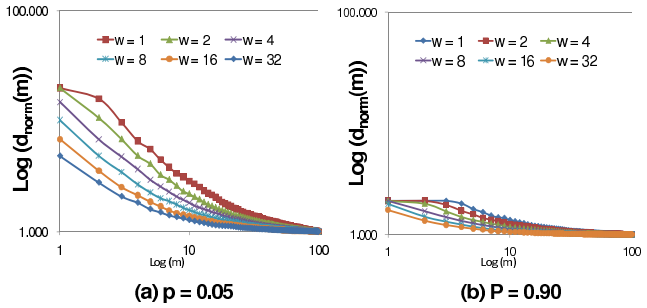


Figure 10: Average normalized maximum activity density $d_{norm}(m)$ from experiments in Figure 8, for multiple timeframes.

3.2 Normalized Activity Density Model

Based on the observations in the previous subsection, we propose an empirical activity density model.

$$d_{norm}(m,w) = \left(\frac{1-p}{p}\right)^{w^{c_1(p-1)}} \cdot m^{-c_2 p^{c_3}} + 1 \qquad (2)$$

In Equation (2), the term $\frac{1-p}{p}$ and the last constant 1 model the observed characteristics (2) and (3), the exponent $w^{c_1(p-1)}$ models characteristics (4) and (5), and together these terms represent characteristic (6). The term $m^{-c_2 p^{c_3}}$ models exponential decay of the density with respect to m, i.e., characteristic (1).

From curve fitting (using *nlinfit* in *MATLAB* [15]), we find model coefficients c_1, c_2 and c_3, and finally, we obtain the following model equation for the maximum activity density.

$$d_{norm}(m,w) = \left(\frac{1-p}{p}\right)^{w^{0.1585(p-1)}} \cdot m^{-0.7781 p^{-0.0896}} + 1$$

3.3 Model Validation

We compare our model with actual data collected. Figure 11 shows the model accuracy with respect to different w values for a given p (= 0.05) and various m values. We can see that our model can correctly capture the impact of multiple timeframes w. Figure 12 shows the model accuracy with respect to p for a given timeframe (e.g., $w = 1$) and various m values. We again observe that our model can account for the impact of p with sufficient accuracy.

We also validate our model on a real design with the estimated power values using a typical power estimation flow. Figure 13 summarizes the following (typical) power estimation flow.

1. Run a cycle-accurate architectural simulator with benchmarks and periodically take a snapshot of the entire circuit state, i.e., values of all registers and input/output traces.
2. Generate a Verilog testbench that initializes the circuit states, drives inputs and check outputs.
3. Synthesize, place and route RTL designs.
4. Run circuit (gate-level) simulation with the generated Verilog testbench and with *standard delay format* (SDF) back-annotated, then generate a *value change dump* (VCD) file.
5. Feed the VCD file into a power estimator (*Synopsys PrimeTime-PX* [18]) to report power.

We study a submodule *sparc_exu_alu* from *OpenSparcT1* [17], which is implemented in 65GP technology with ∼3,000 standard

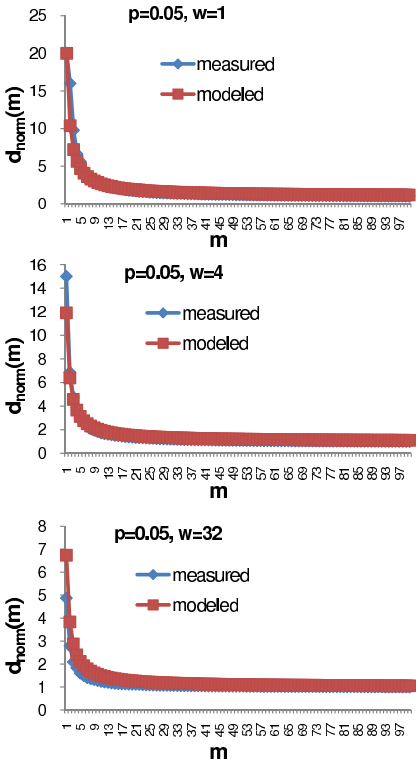


Figure 11: Estimated maximum activity density from our model versus measured maximum activity density from experiments, for a given average activity factor p = 0.05.

cells placed in a 130μm ×130μm region. We use *Synopsys Design Compiler* [14] for synthesis, and *Cadence SoC Encounter* [20] for placement and routing. We use *VirtuTech SIMICS* [19] for system-level simulation and *Cadence NC Verilog* [16] for RTL and gate-level simulations for a high-activity trace of 1 million cycles. Finally, we use *Synopsys PrimeTime-PX* [18] to analyze and report power for the top-1000 highest power cycles out of the 1 million cycles.

To analyze power density trends, we split the design into 100×100 grids, and assign power values of cell instances to corresponding grids where the cell instances are placed. Then, we calculate power density changes with respect to the area (m). Figure 14 shows normalized maximum power density of the *sparc_exu_alu* design for a single (maximum-power) cycle and multiple clock cycles. From the figure, we observe that power density trends with the real design are similar to the activity density trend with artificially generated data: (1) power density increases exponentially as area increases, and (2) activity density slowly decreases as the number of cycles considered increases. However, power density from the real design shows even larger discrepancy from our model than the artificial activity density data. This may be due to the switching activity of the real design being significantly lower than the assumed average activity for artificial data.

4. ANALYTIC MODEL BASED ON DISCREPANCY THEORY

Separately, we have attempted to analytically estimate maximum activity density based on Chernoff bounds [1].

4.1 Model Construction

Let G_n denote an $n \times n$ grid of devices. We assume that each

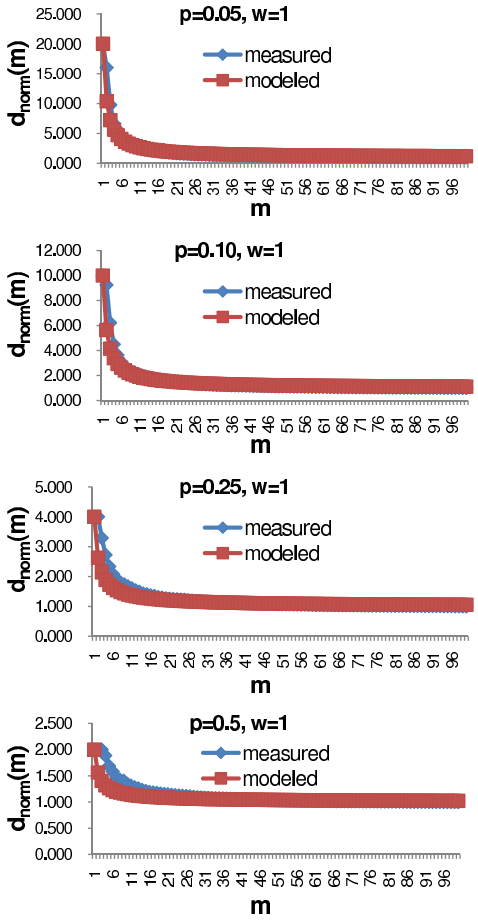


Figure 12: Estimated maximum activity density from our model versus measured maximum activity density from experiments, for a single timeframe (i.e., $w = 1$) with various p.

device is either quiescent (i.e., no switching) or active (i.e., switching) at each time step. For time step t and device d_{ij}, let $s_{ij}(t)$ denote the random variable that indicates whether d_{ij} is active at the time step. We find the number of active devices in geometric rectangles $R_m^{uv} = R_{m\times m}^{uv}$ of size $m \times m$ whose top-right corner is (u,v). The number of active devices in a rectangle at time t is $S_m^{uv}(t) = \sum_{(i,j)\in R_m^{uv}} s_{ij}(t)$. We define *the maximum activity count* $S_m(t)$ over rectangles R_m^{uv} as $S_m = \max_{(u,v)} S_m^{uv}(t)$. We seek to determine how $S_m(t)$ behaves as a function of m and t.

We first start with a simplified time-invariant model in which we assume that $s_{ij}(t) = 1$ with probability p independent of other devices and t. Since this model is time-invariant, we drop the dependence on t in our notation.

Fact 1 (Chernoff Bound). Let $x_1, \cdots, x_n$ be mutually independent 0/1 random variables, each equal to 1 with probability p. If $X = \sum_{i=1}^{n} x_i$, for any $0 < \delta < 1$, we have the following inequalities,

$$\mathbf{Pr}[X > (1+\delta)\mu] < e^{-\delta^2\mu/4} \quad (3)$$

$$\mathbf{Pr}[X < (1-\delta)\mu] < e^{-\delta^2\mu/2} \quad (4)$$

Equation (3) gives the probability of X being larger than its average value μ by $\delta\mu$. Without loss of generality, X can be replaced with S_m^{uv}. With the average activity factor p, the average count μ of

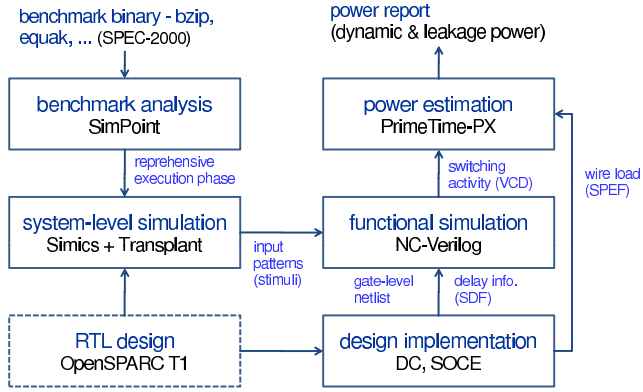


Figure 13: Used power estimation flow.

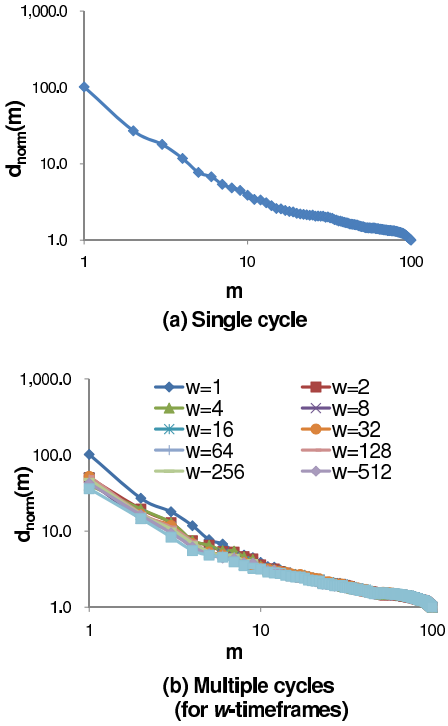


Figure 14: Power density changes in a real design.

S_m^{uv} is calculated as pm^2, and by probability theory, the following inequality must be satisfied.

$$\mathbf{Pr}[\exists_{uv}, S_m^{uv} > (1+\delta)\mu)] \le \sum_{uv} \mathbf{Pr}[S_m^{uv} > (1+\delta)\mu)] < \varepsilon.$$

The probability $\mathbf{Pr}[S_m^{uv} > (1+\delta)\mu)]$ is then bounded by $\varepsilon/(n-m+1)^2$, where $(n-m+1)^2$ is the number of locations of an $m \times m$ window in an $n \times n$ design. Equating the right-hand side of Equation (3) to $\varepsilon/(n-m+1)^2$ results in $\delta = \sqrt{4p \ln \frac{(n-m+1)^2}{\varepsilon}}/pm$. Substitution of δ in Equation (3) gives the following property.

Lemma. For any (u,v),

$$\mathbf{Pr}[S_m^{uv} > pm^2 + m\sqrt{4p \ln \tfrac{(n-m+1)^2}{\varepsilon}}] \le \varepsilon/(n-m+1)^2$$

From the Lemma, Fact 2 for the maximum activity count of an $m \times m$ window for a single time step is derived.

Fact 2.
$S_m \leq pm^2 + m\sqrt{4p \ln \frac{(n-m+1)^2}{\varepsilon}}$ with probability at least $1-\varepsilon$.

Similarly, for multiple time steps, maximum activity count S_m^T $(= \max_{(u,v)} \sum_{t=0}^{T} \sum_{(i,j)\in R_m^{uv}} s_{ij}(t))$ has the following property.

Fact 3.
$S_m^T \leq pm^2T + m\sqrt{4pT \ln \frac{(n-m+1)^2}{\varepsilon}}$ with probability at least $1-\varepsilon$.

4.2 Model Validation

We verify Fact 3 using the generated data set described in Section 2. We calculate S_m^T from all 1,843,200 combinations of the following parameters, using the algorithm given in Figure 6.

- p = {0.05, 0.10, 0.15, 0.25, 0.5, 0.9}
- n = {100, 200}
- m = {1,...,100} for n = 100, and m = {1,...,200} for n = 200
- T = {1,...,1024}

The inequality in Fact 3 is verified using data fitting. First, we transform the inequality into an equation with a model coefficient α as

$$S_m^T = \alpha \left(pm^2T + m\sqrt{4pT \ln \frac{(n-m+1)^2}{\varepsilon}} \right). \quad (5)$$

Second, we find coefficient α from curve fitting with the obtained S_m^T using the generated data set and evaluated values from Equation (5) with given parameters p, m, n and T. Finally, we check whether the value of α is reasonable; if Fact 3 is correct, α should approach one.

Table 1 shows the fitted α values for various ε values and errors of the proposed model in Equation (5) with each α. For each combination of p, n, m, and T, we calculate error as a ratio of the difference between the modeled value (S_m^T from Equation (5)) and the measured value (from our data set) to the measured value. We then take "average" and "maximum" errors from all p, n, m, and T combinations. From the table, we observe that all fitted α values are near one and are not larger than one. This suggests that Fact 3 gives tight estimates. We also observe that the model shows high accuracy; average model error is less than 2% for all ε values. However, as shown in the fourth column, we can see very high maximum error for a small portion of the data. Since S_m^T values are small when m is small, small discrepancy between the data and the estimation becomes significant in terms of percentage. (If we discard cases $m < 10$, the average error reduces to $0.67\% \sim 1.01\%$ and the maximum error reduces to $51.94\% \sim 74\%$.)

Table 1: Average and maximum error of Eq. (3) with fitted α over 1,843,200 data points.

ε	α	Average Error (%)	Maximum Error (%)
0.001	0.9881	1.96	773.31
0.01	0.9893	1.74	720.90
0.1	0.9905	1.50	664.12
0.2	0.9909	1.42	646.04
0.5	0.9915	1.31	621.34
0.9	0.9918	1.24	604.87

5. CONCLUSION

We have presented a general law for power density which can potentially enable new estimates of power density and voltage noise, as well as of required power distribution network (PDN) resources in early design stages. From computational experiments as well as discrepancy analyses using random activity distributions, and from experiments using a real design and a production design flow, we empirically observe a power-law relationship between maximum switching activity density and area. We provide closed-form activity density models from empirical data analysis and probability theory; these can be used to improve the accuracy and efficiency of early-stage PDN resource prediction. Our ongoing work includes further simplification of models and validation of the proposed models against large industry designs. Our ultimate goal is to develop fast and accurate PDN design and optimization methodologies for early stages of IC design.

6. ACKNOWLEDGMENTS

We thank Mr. Thomas Toms and Dr. Riko Radojcic of Qualcomm, Inc. for the motivating suggestions behind this work, and Professor Ramamohan Paturi for valuable discussions on applying Chernoff bounds in analysis of maximum activity density.

7. REFERENCES

[1] H. Chernoff, "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations", *Annals of Mathematical Statistics* 23(4) (1952), pp. 493–507.
[2] P. Christie and D. Stroobandt, "The Interpretation and Application of Rent's Rule", *IEEE Transactions on Very Large Scale Integration Systems* 8(6) (2000), pp. 639–648.
[3] J. A. Davis, V. K. De and J. D. Meindl, "A Stochastic Wire-Length Distribution for Gigascale Integration (GSI) – Part I: Derivation and Validation", *IEEE Transactions on Electron Devices* 45(3) (1998), pp. 580–589.
[4] A. Ghosh, S. Devadas, K. Keutzer and J. White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits", *Proc. ACM/IEEE Design Automation Conference*, 1992, pp. 253–259.
[5] K. Jeong, A. B. Kahng and H. Yao, "On Modeling and Sensitivity of Via Count in SOC Physical Implementation", *Proc. International SoC Design Conference*, 2008, pp. 125–128.
[6] S. Landman and R. L. Russo, "On a Pin Versus Block Relationship for Partitions of Logic Paths", *IEEE Transactions on Computers* 20(12) (1971), pp. 1469–1479.
[7] F. N. Najm, "Transition Density: A New Measure of Activity in Digital Circuits", *IEEE Transactions on Computer-Aided Design* 12(2) (1993), pp. 310–323.
[8] F. N. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits", *IEEE Transactions on Very Large Scale Integration Systems* 2(4) (1994), 446–455.
[9] F. N. Najm, R. Burch, P. Yang and I. Hajj, "Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits", *IEEE Transactions on Computer-Aided Design* 9(4) (1990), pp. 439–450.
[10] R. Paturi, *personal communication*, 2010.
[11] T. Toms, *personal communication*, 2010.
[12] T. Uezono, K. Okada and K. Masu, "Via Distribution Model for Yield Estimation", *Proc. International Symposium on Quality Electronic Design*, 2006, pp. 479–484.
[13] P. Zarkesh-Ha, J. A. Davis, W. Loh and J. D. Meindl, "Stochastic Interconnect Network Fan-out Distribution Using Rent's Rule", *Proc. International Interconnect Technology Conference*, 1998, pp. 184–186.
[14] *Synopsys Design Compiler*, http://www.synopsys.com/.
[15] *MathWorks MATLAB*, http://www.mathworks.com/.
[16] *Cadence NC Verilog*, http://www.cadence.com/.
[17] *Sun OpenSPARC Project*, http://www.sun.com/processors/opensparc/.
[18] *Synopsys PrimeTime*, http://www.synopsys.com/.
[19] *VirtuTech SIMICS*, http://www.virtutech.com/.
[20] *Cadence SOC Encounter*, http://www.cadence.com/.