THEORETICAL AND EXPERIMENTAL INVESTIGATIONS OF CONNECTIVITY IN THREE-DIMENSIONAL INTEGRATED CIRCUITS

A Dissertation
Presented to
The Academic Faculty

By

William Wahby

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2019

THEORETICAL AND EXPERIMENTAL INVESTIGATIONS OF CONNECTIVITY IN THREE-DIMENSIONAL INTEGRATED CIRCUITS

Approved by:

Dr. Muhannad Bakir, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Azad Naeemi
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Jeff Davis
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Tushar Krishna
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Yogendra Joshi
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: Nov 15, 2018

We expected something unexpected, but we did not expect this.

Gerry Skinner

To who I was.

ACKNOWLEDGEMENTS

This work would not have been possible had the initial conditions of the universe been slightly different.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

For decades, the pace of semiconductor development has been dictated by Moore's Law, the observation that transistor density roughly doubles every 18-24 months. The explosive growth of the semiconductor industry has been fueled by the metronomic pace of transistor scaling. In 1974, Dennard laid out the scaling principles that would serve to drive the industry forward for decades, enabling continual improvements in transistor performance, energy efficiency, and cost [1]. Dennard scaling broke down, however, in the early 2000s, as IC power density and transistor short channel effects reduced the efficiency benefits of scaling [2–4], and as the cost of lithography began to increase dramatically due to the need for ultra-fine feature sizes at advanced nodes [5]. Additionally, as on-chip interconnects scale down they begin to face significant performance and reliability challenges, as shown in Fig. 1.1 [6]. As copper wires decrease in size, the average crystal grain size decreases, increasing electron scattering from grain boundaries, and therefore increasing the resistivity of the copper [7, 8]. Line edge roughness also typically degrades at advanced process nodes, leading both to increases in surface scattering as well as reductions in minimum wire width [9, 10], and therefore further reducing the wire resisitivity. While news of the demise of Moore's Law has yet been premature, there is widespread agreement in the industry that conventional 2D CMOS device scaling is unlikely to continue indefinitely [11].

1.1 Motivation

Three-dimensional integration (3DI) has been proposed as an approach for increasing the density and performance of integrated circuits without requiring continued aggressive two-dimensional scaling [12]. Three-dimensional integrated circuits (3DICs) are composed of multiple active tiers, and interconnected vertically with through-layer vias. By stacking

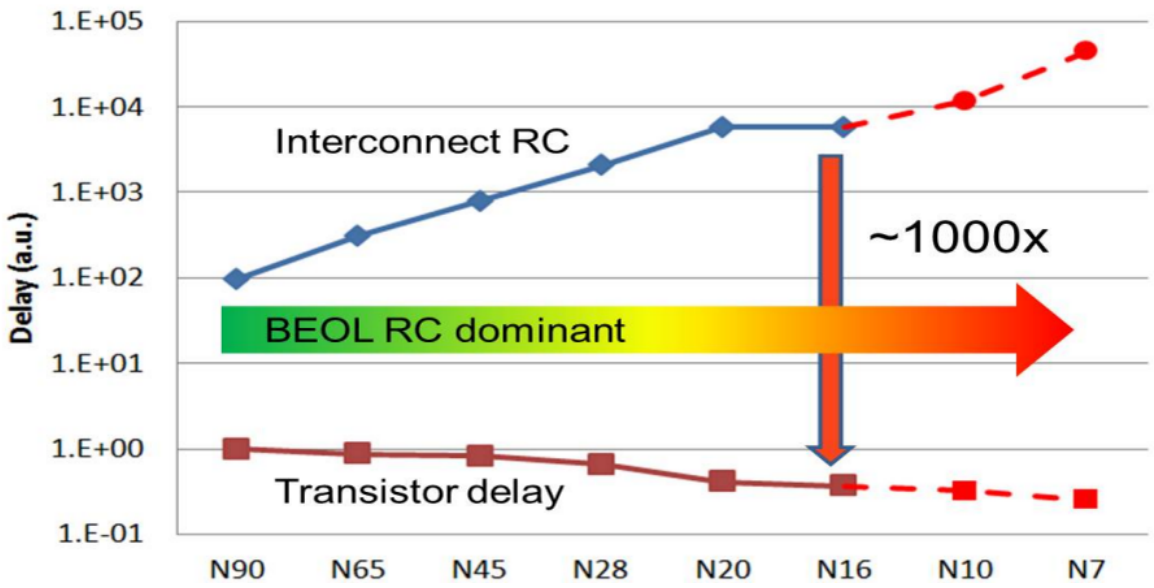

Figure 1.1: As process nodes continue to scale, logic and BEOL exhibit opposing delay trends. While logic cells typically operate faster and more efficiently at smaller nodes, the wire RC delay increases dramatically, primarily due to increased wire resistance. Figure from [6].

dice directly atop one another the inter-die interconnect distance can be greatly reduced, significantly reducing interconnect delay and power consumption. Furthermore, the inter-die interconnect density can be greatly enhanced compared to package-level integration, as through-layer vias are typically much smaller than package level solder bumps. When taken together, the combination of increased IO density and shorter interconnect distance offers potentially orders of magnitude improvement in inter-die signaling bandwidth, latency, and interconnect power consumption.

There are many different variants of 3D integration, as illustrated in Fig. 1.2, each with their own strengths and weaknesses. The most widely-studied form uses through-silicon vias (TSVs) to directly interconnect stacked dice by routing signals directly through the die substrate. Interposer-based solutions, in which dice are mounted on a (typically) silicon interposer are a stepping-stone between conventional 2D packaging and full 3DIC design [13–15]. The most aggressive form of 3D integration is monolithic 3D (M3D), in which logic layers are deposited or grown directly atop one another, yielding ultra-fine-

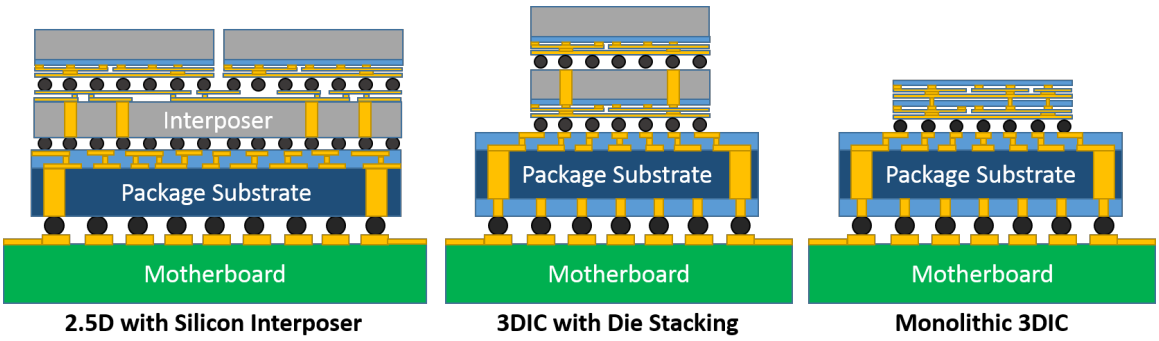

Figure 1.2: There are many potential configurations for 3DICs, each with their own costs and advantages. Designers must manage the complexity of the 3DIC design space in order to achieve higher performance and lower cost systems.

grained interconnection at the cost of significantly more complex fabrication [16–18].

Several significant challenges, however, stand in the way of widespread adoption of 3D integration techniques. Due to the increased complexity of 3DIC design, it is challenging to accurately compare the performance of a 3DIC to an equivalent 2D counterpart. Additionally, 3DICs suffer from reduced thermal performance due to the increased thermal resistance between the lower dice in the stack and the (top-mounted) heat sink. Additionally, despite the reduced interconnect power consumption, the overall 2D power density of a 3D stack could be higher than the power density of an equivalent 2DIC, placing further strain on the cooling solution. Due to the many possible combinations of material properties, transistor properties, integration methodologies, and operating conditions for a 3DIC, it is imperative to develop a tool for rapid 3DIC simulation and pathfinding.

Stochastic wirelength distributions are useful for quickly estimating the interconnect properties of hypothetical integrated circuits. In order to estimate the interconnect properties of large ICs, we can make use of Rent's rule, the observation that the number of pins/terminals leaving a logic block has a power-law relationship with the number of logic elements within that block. First observed in 1960 by E.F. Rent, and further explored in 1971 by Landman and Russo [19], Rent's rule forms the basis for interconnect length estimation techniques. In 1979, Donath expanded on Landman and Russo's work to estimate average connection lengths in ICs, [20], and further extended the methodology to

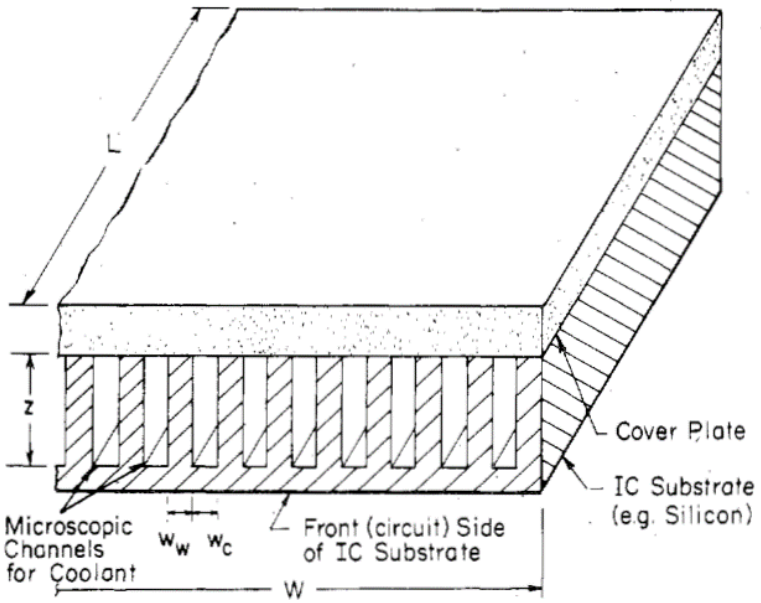

Figure 1.3: Schematic of a microfluidic system. Channels are etched into the reverse side of the silicon substrate and capped to create microscale cooling channels as close as possible to the active circuitry. Figure from [33].

estimate the overall wire length distribution in 1981 [21]. In 1987 Masaki and Yamada made the first extension towards estimating wire length in three dimensions [22]. Throughout the early 1990s the relationship between design type and Rent exponent was further explored [23, 24]. In 1998 Davis published an improved wirelength distribution [25, 26], which quickly became the basis for future wirelength estimation efforts. Several detailed 3D wirelength distributions were developed by extending the 2D framework [27,28]. These wirelength estimation methods have been used extensively to explore the impact of different technologies on 2D and 3D IC design. By modeling the delay and power consumption of interconnect links of various lengths, and by combining those models with models of wire size and pitch in various metal levels, it is possible to estimate the clock frequency and power consumption of a wide range of 2D and 3D systems [27, 29–32].

Thermal management poses a special challenge for 3DIC performance, as the very act of stacking dice increases the thermal resistance between the lowest dice and the heat sink. Making matters worse, the equivalent areal power density of the 3D stack will be

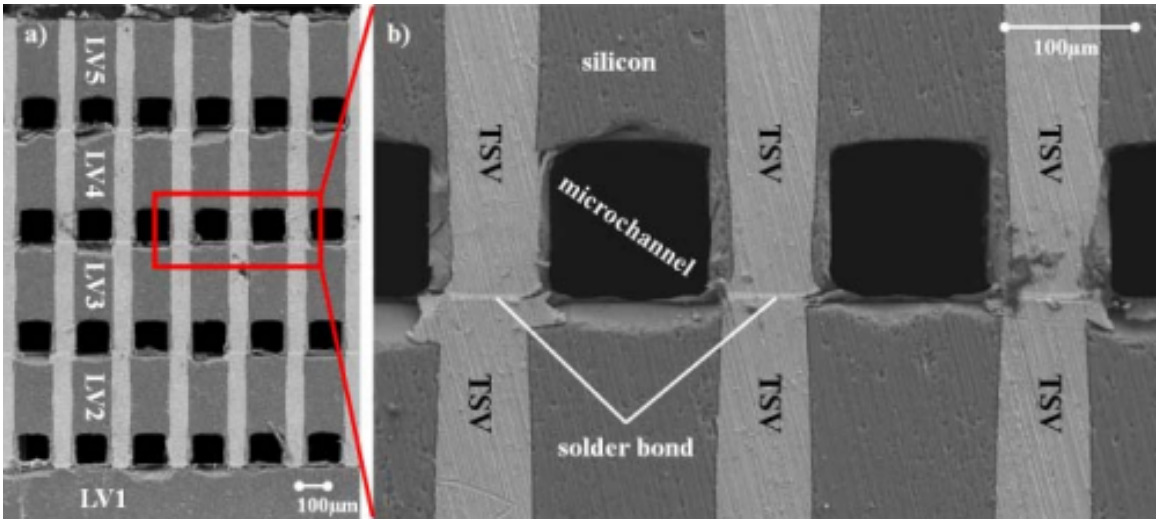

Figure 1.4: Example of a multi-die stack with integrated microfluidic cooling channels and through-silicon vias for inter-tier connectivity. Figure from [41].

significantly higher than the power density of either die in isolation, as the area available for heat removal is drastically reduced. Possibly the most ambitious suggestion is the use of microfluidic cooling, in which channels are etched into the silicon dice to enable coolant distribution as closely as possible to the active logic, as shown in Fig. 1.3 [33]. In order to enable 3D stacking of dice, through silicon vias can be embedded within the residual silicon, as shown in Fig. 1.4. To further increase both cooling capability and IO density, high aspect ratio TSVs embedded in micro pin-fin structures have been explored [34, 35], as shown in Fig. 1.5, and have been utilized to actively cool single-tier ICs [36, 37]. Less aggressive thermal solutions, such as improved thermal interface materials (TIMs), lateral heat conduction layers [38], vertical thermal conduction vias [39], and thermal-aware IC design [40] have also been investigated as potential solutions, and 3DICs will likely require some combination of all possible approaches to maximize performance and reliability.

1.2 Research objective and contribution

The objective of this research is to better explore the thermal and electrical tradeoffs inherent in the design of 3D integrated circuits. The main topics of research detailed in this

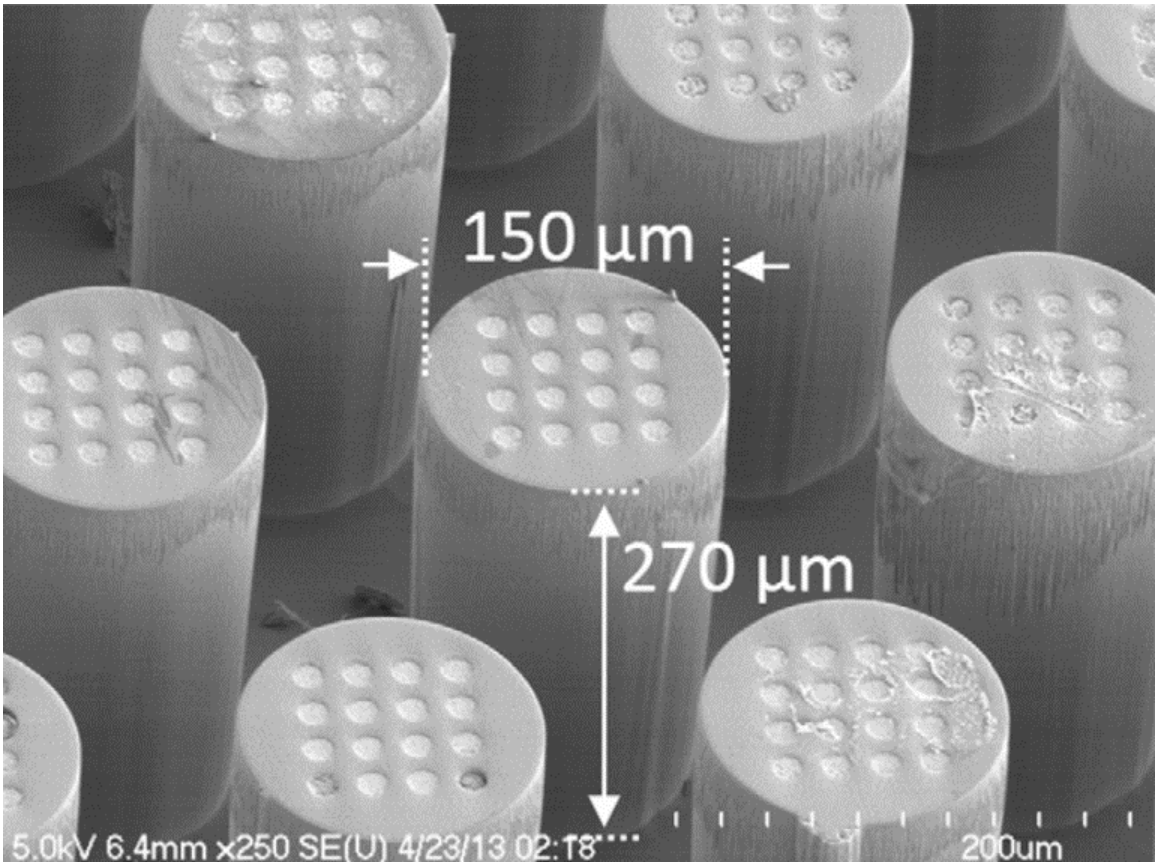

Figure 1.5: Silicon micro pin-fin heat sink with embedded high aspect ratio TSVs. Figure from [35].

thesis are as follows:

1. **Development of improved wire length models for 3D integrated circuits.** Stochastic wire length models have been used extensively to predict and analyze the performance of 2D and 3D ICs, but existing 3DIC wire length models do not fully account for the finite size of through silicon vias. In this thesis we derive a correction factor which can be applied to existing stochastic 3D IC wire length models to improve their accuracy.

2. **Development of improved TSV estimation methods for 3D integrated circuits.** As through silicon vias are critical components for 3D IC design, estimating the total number of vias required for a 3D implementation of a given design is a key requirement for yield and cost forecasting. Current TSV estimation models are adequate for

shallow 3D stacks of two to four tiers, but they neglect the impact of signals which must traverse several tiers. We adapt existing stochastic wire length techniques to improve the state of the art in TSV estimation, and benchmark our results against several test cases.

3. **Integrated thermal and electrical analysis for rapid 3DIC pathfinding.** The 3DIC design space is large and complex, and better tools are needed in order to rapidly explore the costs and benefits of the many possible 3D configurations for any given design. To accomplish this goal, the improved stochastic wire length and TSV estimation models are combined with wire sizing and repeater insertion algorithms, in order to get a complete picture of the interconnect stack in a 2D or 3D IC. These models are combined with a finite difference thermal solver, capable of rapidly determining the spatial thermal profile of an arbitrary 3D stack, as well as a 3D IC power delivery model, enabling consideration of the design of the power network and prediction of the power delivery TSV requirements. These models are then used to investigate the potential performance implications of a range of 3D integration scenarios, and to highlight the key challenges posed by heat removal in 3D systems.

4. **Experimental thermal benchmarking of a functional 3D testbed.** Our theoretical work suggests that thermal limitations will pose a key limiter for high performance 3D systems. In order to confirm these predictions, we develop a computationally-functional 3D thermal testbed, composed of a high performance compute-class GPU and a top-tier thermal die. Using this testbed, we show the direct performance impact of the reduced thermal headroom in 3D designs, and discuss potential avenues for thermal mitigation in 3D ICs.

5. **Stochastic interconnect length prediction in higher dimensions.** High-dimensional network topologies are routinely encountered in high performance computing systems, as well as in biological and bio-inspired computing systems. While these sys-

tems can only be physically implemented in three dimensions, understanding how interconnect length changes in higher dimensions could help set bounds on the performance of advanced computing systems. To that end, we adapt and extend existing stochastic wire length modeling frameworks to handle hypercubic systems of arbitrary dimension, and investigate some of the broader implications for the design of highly-interconnected networks.

1.3 Organization of the thesis

The remainder of this thesis is organized as follows:

1. In chapter 2 we develop a stochastic wire length model for 3DICs with large-diameter TSVs, as well as a general stochastic model for the number of TSVs required in a 3DIC of an arbitrary number of tiers.

2. In chapter 3 we combine the stochastic wire length and TSV estimation models developed in chapter 2 with wire and repeater sizing algorithms, a finite-difference thermal simulation module, and an analytic 3DIC power delivery model to develop a complete 3DIC virtual integration platform. The combined models are validated against published data, and predictions are made for various 3DIC design cases.

3. In chapter 4 we describe the development of a computationally-functional 3D thermal testbed. The performance of the thermal testbed is measured for a variety of machine learning workloads, and the thermal implications for high performance 3D stacks is discussed.

4. In chapter 5 we derive stochastic models for wire length estimation in cubic systems of arbitrary dimension. These models are then used to investigate overall interconnect parameters in highly-interconnected systems.

5. In chapter 6 the key conclusions of this thesis are summarized, and potential avenues for future work are discussed.

CHAPTER 2

INTERCONNECT MODELING FOR 3D SYSTEMS WITH LARGE TSVS

2.1 Methodology

Joyner *et al.* developed an interconnect length distribution (ILD) by employing a stochastic formalism [31]. Joyner defined the probability distribution function (PDF) for a gate existing at position x_1 to be

$$f_x[x] = \frac{1}{N_x}\left(u[x] - u[x - N_x]\right) \tag{2.1}$$

We extend Joyner's approach by adding a correction which accounts for the forbidden zones associated with real TSVs. Here we assume a periodic array of TSVs. We further assume that each TSV is symmetric, and that each gate has the same length and width, and that the chip is square. The assumption about gate length and width must be modified to apply the results of this analysis to real designs, but these assumptions allow us to develop a qualitative understanding of the impact of finite TSV area without bogging down the analysis in algebraic bookkeeping.

We define $r(x, a, w)$ to be rectangle function in variable x, starting at $x = a$, with width w.

$$r(x, a, w) = u[x - a)] - u[x - (a + w)] \tag{2.2}$$

then

$$f[x,y] = \frac{1}{N}\left(u[x]-u[x-N_x]\right)\left(u[y]-u[x-N_y]\right) - \frac{1}{N}\sum_n r(x,nT+t,w)\sum_m r(y,mT+t,w) \tag{2.3}$$

$$f[x,y] = \frac{1}{N} r(x,0,N_x) r(y,0,N_y) - \frac{1}{N}\sum_n r(x,nT+t,w)\sum_m r(y,mT+t,w) \tag{2.4}$$

where $u[x]$ is the unit step function, T is the TSV pitch, t is the TSV offset (distance from edge of the chip to the first TSV), w is the TSV width, N_x is the length of the chip in the x direction, N_y is the length of the chip in the y direction, n is the TSV index in the x direction, and m is the TSV index in the y direction. T, t, w, N_x, and N_y are all measured in gate lengths. N is a normalization factor which will be treated later; if TSV forbidden zones are ignored, N simply becomes N_x or N_y. For brevity, we will let q_{xy} designate the TSV correction term, i.e.,

$$q_{xy} = \sum_n r(x,nT+t,w)\sum_m r(y,mT+t,w) \tag{2.5}$$

We will also define f_o to be the two-dimensional PDF in the absence of TSVs,

$$f_o[x,y] = \left(u[x]-u[x-N_x]\right)\left(u[y]-u[y-N_y]\right) \tag{2.6}$$

Under these simplifications, Eq. (2.4) becomes

$$f[x,y] = \frac{1}{N}\left(f_o - q_{xy}\right) \tag{2.7}$$

Note that Eq. (2.7) gives the probability of finding a gate at position x, but it now depends on both x and y. In order to account for the impact of TSVs we must slightly break the symmetry of the problem.

In order to determine the total number of interconnects of length l, we must first determine the number of gate pairs separated by l gate lengths. In order for an interconnect to

be valid, it must start and end on a valid gate. In other words, if the starting point of the interconnect is (x_1, y_1), and the ending point is (x_2, y_2), then both $f[x_1, y_1]$ and $f[x_2, y_2]$ must be nonzero. In order to determine the number of gate pairs which satisfy this criterion we can simply sum up all possible combinations of x_1, x_2, y_1, and y_2.

$$\Delta[l] = \sum_{l_x=0}^{l} \sum_{x_1=0}^{N_x-1} \sum_{y_1=0}^{N_y-1} f[x_1, y_1] f[x_2, y_2] \tag{2.8}$$

We can constrain the various coordinates as follows

$$l_x = x_2 - x_1 \tag{2.9}$$

$$l_y = y_2 - y_1 \tag{2.10}$$

$$l = l_x + l_y \tag{2.11}$$

Expanding Eq. (2.8) we find

$$\Delta[l] = \sum_{l_x=0}^{l} \sum_{x_1=0}^{N_x-1} \sum_{y_1=0}^{N_y-1} \left(f_{o_1} - q_{xy_1}\right)\left(f_{o_2} - q_{xy_2}\right) \tag{2.12}$$

$$\Delta[l] = \sum_{l_x=0}^{l} \sum_{x_1=0}^{N_x-1} \sum_{y_1=0}^{N_y-1} \left\{f_{o_1} f_{o_2} - f_{o_1} q_{xy_2} - f_{o_2} q_{xy_1} + q_{xy_1} q_{xy_2}\right\} \tag{2.13}$$

The first term in Eq. (2.13) yields the number of gate pairs separated by distance l in the case where TSV width is not treated. The second term in Eq. (2.13) determines the number of interconnects which start on an allowed gate location, but end on a forbidden location. Expanding term 2 yields

$$\Delta_2 = \sum_{l_x=0}^{l} \sum_{x_1=0}^{N_x-1} \sum_{y_1=0}^{N_y-1} f_{o_1} q_{xy_2} \tag{2.14}$$

$$\Delta_2 = \sum_{l_x, x_1, y_1} r(x_1, 0, N_x) r(y_1, 0, N_y) \times \sum_{n,m} r(x_2, nT + t, w) r(y_2, mT + t, w) \tag{2.15}$$

Applying the constraints from Eqs. (2.9) to (2.11) removes x_2 and y_2 from the summation.

$$\Delta_2 = \sum_{l_x,x_1,y_1} r(x_1,0,N_x)r(y_1,0,N_y) \times \sum_{n,m} r(x_1-l_x,nT+t,w)r(y_1-l_y,mT+t,w) \tag{2.16}$$

$$\Delta_2 = \sum_{l_x,x_1,y_1} r(x_1,0,N_x)r(y_1,0,N_y) \times \sum_{n,m} r(x_1-l_x,nT+t,w)r(y_1-(l-l_x),mT+t,w) \tag{2.17}$$

Evaluating Eq. (2.17) directly is computationally expensive, but the expression can be simplified considerably by making several key observations.

1. The first portion of the summation is always 1 within the bounds of the chip
2. The second portion of the summation is only 1 when x_2 and y_2 are inside a forbidden zone
3. The x_2 and y_2 portions of the summation are completely independent from one another, and may be separated

Observations 1 and 2 indicate that the entire expression will only be nonzero when x_2 and y_2 are within a forbidden zone on the chip. Observation 3 allows us to make a significant simplification. With no loss of generality, Eq. (2.17) can be rewritten as

$$\Delta_2 = \sum_{l_x,x_1,y_1} r(x_1,0,N_x)r(y_1,0,N_y) \times \sum_{n} r(x_1-l_x,nT+t,w) \sum_{m} r(y_1-(l-l_x),mT+t,w) \tag{2.18}$$

Since x_2 and y_2 are independent we can separate Δ_2 into two independent summations.

$$\begin{aligned}\Delta_2 = &\sum_{l_x,x_1,y_1} r(x_1,0,N_x)r(y_1,0,N_y) \\ &\times \left(\sum_n r(x_1-l_x,nT+t,w)\right) \\ &\times \left(\sum_m r(y_1-(l-l_x),mT+t,w)\right)\end{aligned} \tag{2.19}$$

Interchanging the order of the n, m, x_1, and y_1 summations gives us an expression in the form of $f(x,y) = g(x)h(y)$.

$$\begin{aligned}\Delta_2 = &\sum_{l_x}\left(\sum_{n,x_1} r(x_1,0,N_x)r(x_1-l_x,nT+t,w)\right) \\ &\times \left(\sum_{m,y_1} r(y_1,0,N_y)r(y_1-(l-l_x),mT+t,w)\right)\end{aligned} \tag{2.20}$$

Or, more compactly

$$\Delta_2(l) = \sum_{l_x} \Delta_{2_x}\Delta_{2_y} \tag{2.21}$$

$$\Delta_{2_x}(l_x) = \sum_{n,x_1} r(x_1,0,N_x)r(x_1-l_x,nT+t,w) \tag{2.22}$$

$$\Delta_{2_y}(l_y) = \sum_{m,y_1} r(y_1,0,N_y)r(y_1-l_y,mT+t,w) \tag{2.23}$$

Inspecting Δ_{2_x} we can make additional observations to simplify the expression

1. Δ_{2_x} is only nonzero when $x_1 \in [nT+t+l_x, nT+t+w+l_x]$
2. x_1 may only range between 0 and N_x
   (a) In this range $r(x_1,0,N_x)$ is always 1
3. For each value of n, the x_1 summation counts the number of forbidden gate locations at the n-th TSV which are more than l_x gates away from the left end of the chip
   (a) If the gate location at $x_1 = l_x$ is not within a TSV, then the x_1 summation

will add w gates to the total for every value of n which satisfies the inequality $l_x < nT + t$

(b) If the gate location at x_1 = l_x is within a TSV, then the x_1 summation will add only a portion of the gates for that TSV, and w gates for all TSVs with larger n

Clearly Δ_{2_x} is strongly dependent upon l_x. Functionally, Δ_{2_x} adds up all the possible ways for the start of the *horizontal* portion of an interconnect to be located inside a (potential) forbidden zone, while the other end is located anywhere on the chip. Note that depending on the value of y_2 the starting end of the horizontal interconnect may not actually be in a forbidden zone; this situation is captured by the Δ_{2_y} term. Furthermore, note that x_1 could itself be inside a forbidden zone: this term does not account for that scenario, and will overestimate the number of forbidden interconnects. Term 4 of Eq. (2.13) corrects for the case where both ends of the interconnect are located in forbidden zones.

The value of l_x determines the value of Δ_{2_x}. If $l_x = 0$, then Δ_{2_x} is simply the number of potentially forbidden gate locations in the x directon, and is just equal to $N_{tsv}w$, or the number of TSVs multiplied by the width of each TSV (in gate lengths). Conversely, if $l_x = N_x$, there exists no valid value of x_1 large enough to cause $r(x_1 - l_x, nT + t, w)$ to be nonzero. As l_x increases from 0 to N_x, the lower bound of the x_1 summation increases as well.

Combining these observations, we can construct a function, $g(l_x)$, which reproduces the behavior of the complete summation.

$$g_x(l_x) = \begin{cases} n_t w & l'_x < T - t - w \\ (n_t - 1)w & l'_x > T - t \\ n_t w - l'_x + T - t - w & \text{else} \end{cases} \tag{2.24}$$

$$l_x \in [0, N_x] \tag{2.25}$$

$$l'_x = l_x \bmod T \tag{2.26}$$

$$n_t = (n_{max} + 1) - \left\lfloor \frac{l_x}{T} \right\rfloor \tag{2.27}$$

A similar analysis for Δ_{2_y} yields an equivalent function, $g(l_y)$, where $l_y = l - l_x$.

$$g_y(l_y) = \begin{cases} m_t w & l'_y < T - t - w \\ (m_t - 1)w & l'_y > T - t \\ m_t w - l'_y + T - t - w & \text{else} \end{cases} \tag{2.28}$$

$$l'_y = l_y \bmod T \tag{2.29}$$

$$m_t = (m_{max} + 1) - \left\lfloor \frac{l_m}{T} \right\rfloor \tag{2.30}$$

Note that for a square chip ($N_x = N_y$), $g_x = g_y = g$, and Δ_2 becomes

$$\Delta_2(l) = \sum_{l_x} g(l_x) g(l - l_x) \tag{2.31}$$

But this is simply the formula for the discrete convolution of g with respect to itself! We can therefore write

$$\Delta_2(l) = g_x * g_y \tag{2.32}$$

The third term in Eq. (2.13) counts the number of possible interconnects which start in a forbidden zone, but end in an allowed zone. We can take advantage of its similarity to term 2 to simplify the solution.

$$\Delta_3 = f_{o_2} q_{xy_1} \tag{2.33}$$

$$\Delta_3 = \sum_{l_x,x_1,y_1} r(x_2,0,N_x)r(y_2,0,N_y) \times \sum_n r(x_1,nT+t,w)\sum_m r(y_1,mT+t,w) \tag{2.34}$$

$$\Delta_3 = \sum_{l_x,x_1,y_1} r(x_1-l_x,0,N_x)r(y_1-(l-l_x),0,N_y) \times \sum_n r(x_1,nT+t,w)\sum_m r(y_1,mT+t,w) \tag{2.35}$$

$$\Delta_3 = \sum_{l_x}\left(\sum_{n,x_1} r(x_1-l_x,0,N_x)r(x_1,nT+t,w)\right) \times \sum_{m,y_1} r(y_1-(l-l_x),0,N_y)r(y_1,mT+t,w) \tag{2.36}$$

$$\Delta_3(l) = \sum_{l_x} \Delta_{3_x}\Delta_{3_y} \tag{2.37}$$

$$\Delta_{3_x}(l_x) = \sum_{n,x_1} r(x_1-l_x,,0,N_x)r(x_1nT+t,w) \tag{2.38}$$

$$\Delta_{3_y}(l_y) = \sum_{m,y_1} r(y_1-l_y,0,N_y)r(y_1,mT+t,w) \tag{2.39}$$

Following the same reasoning used in term 2, we can define a function $h(x)$ which captures the behavior of Δ_{3_x}.

$$h_x(l_x) = \begin{cases} n_t w & l'_x < t \\ (n_t-1)w & l'_x > t+w \\ n_t w - (l'_x - [T-t-w]) & \text{else} \end{cases} \tag{2.40}$$

$$l_x \in [0, N_x] \tag{2.41}$$

$$l'_x = l_x \bmod T \tag{2.42}$$

$$n_t = (n_{max}+1) - \left\lfloor \frac{l_x}{T} \right\rfloor \tag{2.43}$$

As with term 2 an equivalent function can be defined for $h_y(l_y)$, but for a square gate array

both functions are functionally identical, and Δ_3 becomes

$$\Delta_3(l) = \sum_{l_x} h(l_x)h(l - l_x) \tag{2.44}$$

Again, we note that this result is in the form of an autocorrelation. The final form is then given by

$$\Delta_3(l) = h_x * h_y \tag{2.45}$$

The final term in the summation corrects for the fact that we're double-counting forbidden interconnections in terms 2 and 3. Term 2 counts interconnects which end in forbidden zones, whereas term 3 counts interconnects which start at forbidden zones. Potential interconnects which both start and end in forbidden locations are counted in both terms 2 and 3, and the final term corrects for this issue. This term is not as analytically tractable, but it can be easily (if relatively slowly) computed numerically. It can be safely ignored in many cases, as the number of doubly-forbidden interconnections is generally much smaller than the number of singly-forbidden paths (relatively few paths start and end in forbidden zones, compared to paths which only have one end in a forbidden zone).

Now we can rewrite Eq. (2.13) in terms of known quantities

$$\Delta[l] = M_t^o - g * g - h * h + \sum_{l_x=0}^{l} \sum_{x_1=0}^{N_x-1} \sum_{y_1=0}^{N_y-1} q_{xy_1} q_{xy_2} \tag{2.46}$$

where M_t^o is Joyner's 2D gate distribution [31]. The final summation is difficult to treat analytically, but is generally a small correction, and can be ignored in many cases.

The discrepancy between the predicted wire length for a typical system with and without the TSV area correction is shown in Fig. 2.1. For systems with very small TSVs, a small error is expected, whereas systems with TSVs with diameters of 5 micron and greater are predicted to exhibit significant error in total wirelength. The performance of the model was also compared against fully routed monolithic 3DIC designs [42], as shown in Figs. 2.2 and 2.3, with good agreement between the predictions and the routed designs. Notably, for

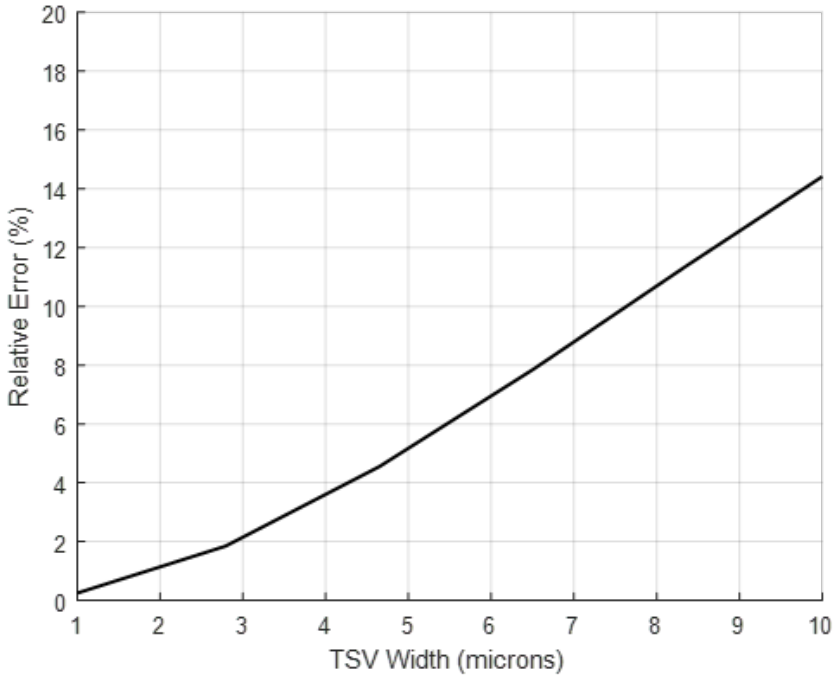

Figure 2.1: Error in total wirelength for a hypothetical computing system with 86M gates, implemented across 4 logic tiers, with 40k TSVs per logic tier. Error increases significantly as the TSV diameter is increased.

the smallest and largest designs, the model agrees very well with the fully routed results, whereas for the RCA and FFT designs, the model underestimates the total wirelength. This could be due to routing congestion in these particular designs; in the future, detailed investigation of the fully-routed designs could yield further insight into the root causes of this deviation. Despite this discrepancy, however, it is clear that stochastic methods are adequate for estimating overall wire length in 3D ICs, further bolstering their use as tools for rapid design space exploration and pathfinding.

2.2 Via number estimation for monolithic 3D systems

Accurately estimating the number of TSVs or Monolithic Interlayer Vias (MIVs) in a 3DIC is critical in order to get an accurate understanding of the overall system cost, yield, and performance, but current methods for via estimation are restricted to two-tier designs. We present an extension of the current method to devices with arbitrary numbers of tiers. The

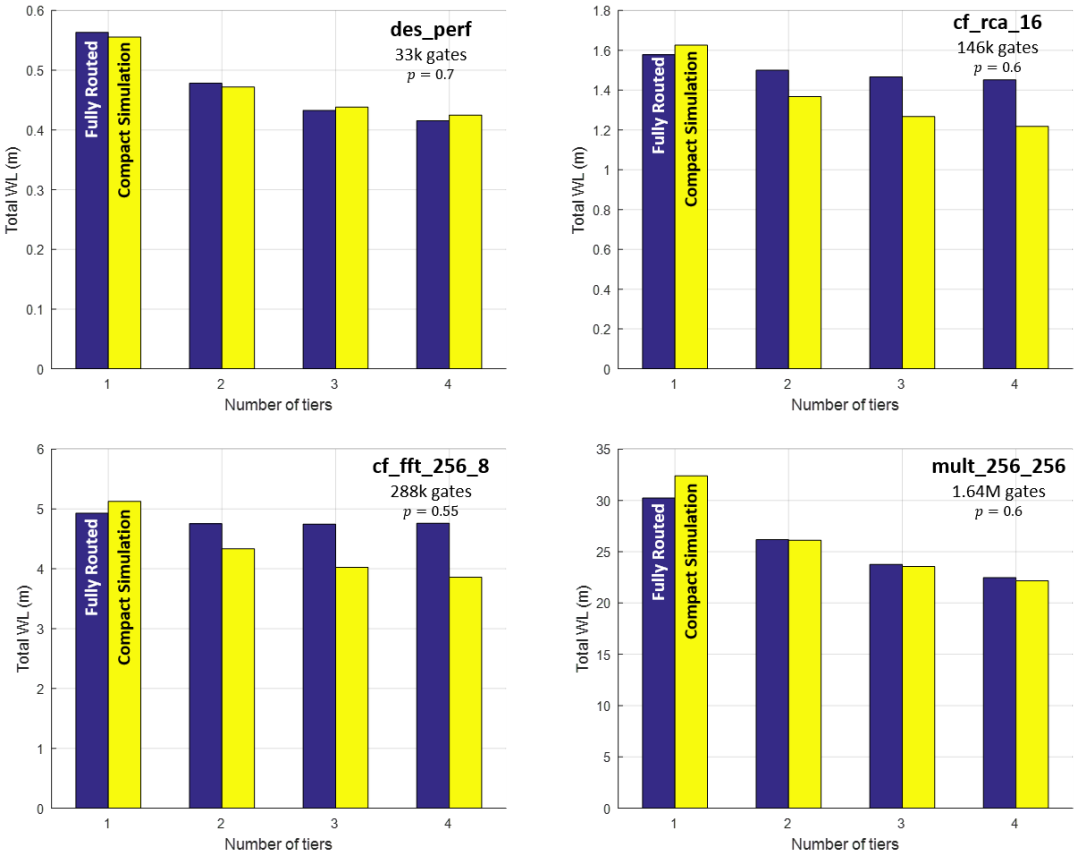

Figure 2.2: Comparison of the wire length distribution predictions to routed monolithic 3D designs presented in [42]. Good agreement is seen between the total wirelength predicted by the model and the fully routed designs.

new method is validated against recursively-partitioned netlists, to simulate the partitioning of a 2D design into a multi-tier 3DIC.

2.2.1 Motivation and development

As technological and economic challenges to conventional 2D scaling intensify 3DICs are becoming attractive options for improving IC performance. Design in 3D is complicated by the relatively large number of unknowns, compared to conventional 2D designs. In order to better understand the 3D design space, extensive modeling of 3DIC performance, layout, yield, and manufacturing cost has been carried out. In order to determine the thermal profile of a 3DIC, the total power dissipation and thermal configuration must be known. Since

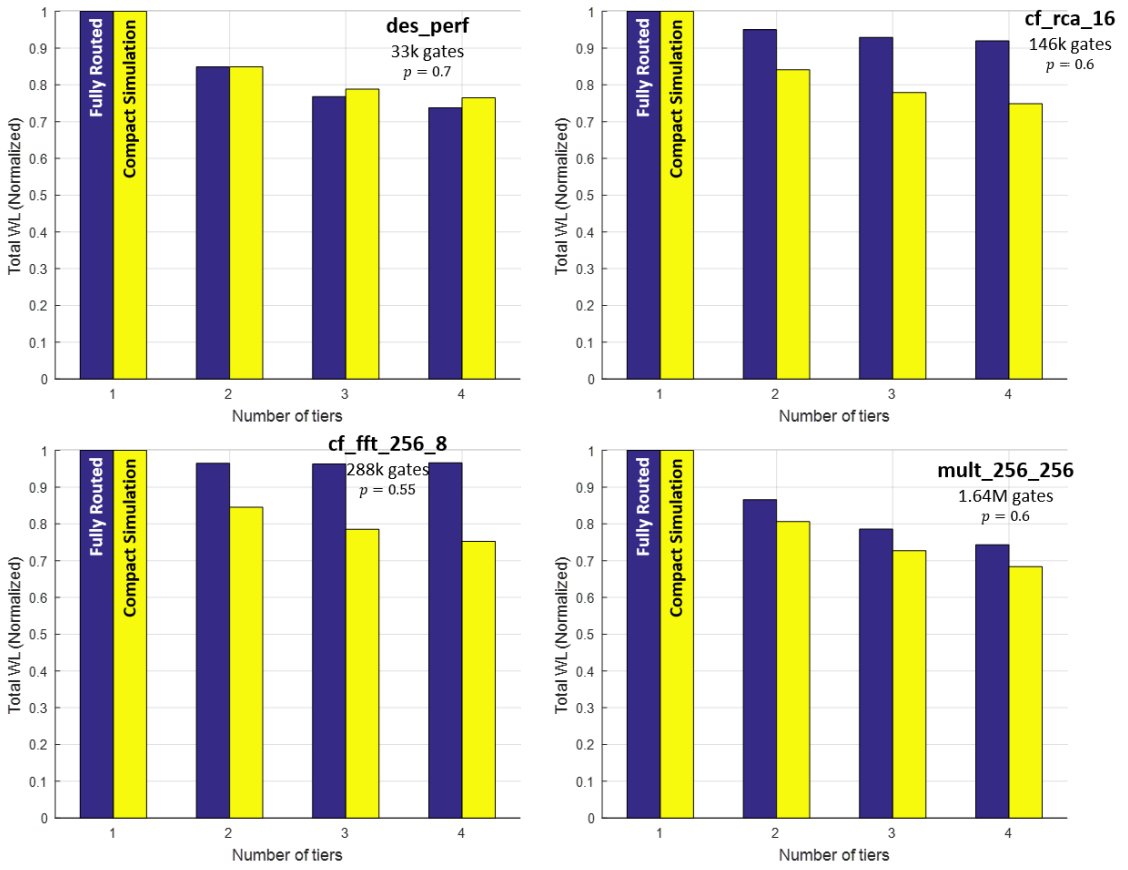

Figure 2.3: Comparison of the wire length distribution predictions to routed monolithic 3D designs presented in [42]. Good agreement is seen between the total wirelength predicted by the model and the fully routed designs. Total wirelength in all cases is normalized to the 2D case.

interlayer vias act as vertical heat pipes, their impact must be factored in to 3DIC thermal predictions. Additionally, the overall power dissipation of an IC is strongly influenced by the power consumed passively by wires, and actively by repeaters. In order to get an accurate picture of the salient wire and repeater properties in 3DICs, the number of vias must be accurately modeled [43, 44]. Additionally, the cost to manufacture and test 3DICs is strongly influenced by the number of interlayer vias, as is the overall system yield [45–53].

Current interlayer-via estimation techniques are direct applications of the stochastic wirelength distribution presented in [25, 26], which itself builds on Rent's Rule, the empirical observation that the number of terminals leaving a circuit is related through a simple

power law to the number of logic gates in the circuit [19]:

$$T = kN^p \tag{2.47}$$

where T is the number of terminals in the circuit, N is the number of logic gates in the circuit, and k and p are empirically-determined constants. An alternate formulation of Eq. (2.47) is presented in [20]:

$$T = \alpha k N \left(1 - N^{p-1}\right) \tag{2.48}$$

where α is a constant which corrects for the average fanout in the design. It has been shown that the Rent constants can be uniquely determined for a particular netlist (i.e., are insensitive to changes in fabrication methodology, such as lithographic shrinks), and similar types of designs tend to have similar values (e.g. high-performance logic devices tend to have $p = 0.6 - 0.7$, while memory devices tend to have $p = 0.4 - 0.5$) [54].

Eq. (2.47) can be used to determine the number of connections between two blocks of logic, A and C as well as the number of connections traversing another block, B, as shown in Fig. 2.4 [25]. In [25], the following formula is derived:

$$T_{A-C} = T_{BC} + T_{AB} - T_{ABC} - T_B \tag{2.49}$$

where T_{A-C} is the number of terminals connecting blocks A and C, T_B is the number of terminals leaving block B, and T_{BC} is the number of terminals leaving the super-block formed by the combination of blocks B and C. In [45,48,53,55] Eqs. (2.48) and (2.49) are used in conjunction to derive the following expression for the number of inter-tier connections in a two-tier 3DIC.

$$\begin{aligned} N_{tsv} = {} & \alpha k_{1,2}(N_1 + N_2)\left(1 - (N_1 + N_2)^{p_{1,2}-1}\right) \\ & - \alpha k_1 N_1 \left(1 - N_1^{p_1-1}\right) \alpha k_2 N_2 \left(1 - N_2^{p_2-1}\right) \end{aligned} \tag{2.50}$$

This result is useful for two-tier 3DICs, but ignores the through-tier connections required in multi-tier 3DICs, ultimately resulting in significant undercounting of via requirements

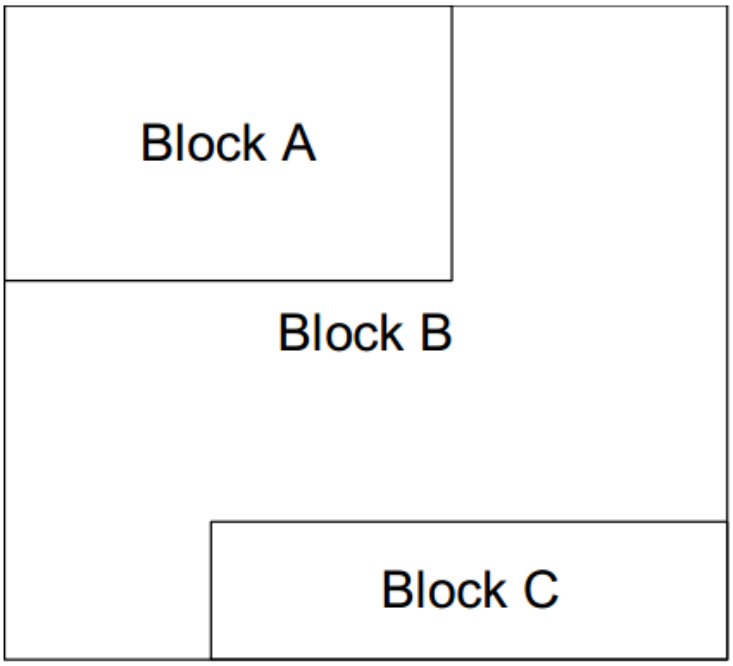

Figure 2.4: The scenario under consideration, in which two blocks of logic, A and C, are separated by a third logic block, B, and the number of connections between A and C must be determined. In a 3DIC, block B would consist of all logic planes between tiers A and C. Figure after [25]

as the number of tiers in a 3D stack increases.

2.2.2 Interlayer via estimation for multi-tier 3DICs

A more general expression can be derived by considering Eqs. (2.47) and (2.49). For simplicity, we will treat the case where a large 2D design is being partitioned into multiple layers for implementation as a 3DIC. Accordingly, we assume that the rent constant and exponent are identical for all layers in the 3DIC. This assumption can be relaxed by using the method of [56] to conglomerate multiple macrocells with distinct rent parameters into one homogeneous effective circuit with effective rent parameters, k_{eff}, p_{eff}. Additionally we assume that each layer has the same number of logic gates, N_g. Again, this assumption can be relaxed easily, but at the cost of a slightly more complex final expression. The system we are considering is shown in Fig. 2.5.

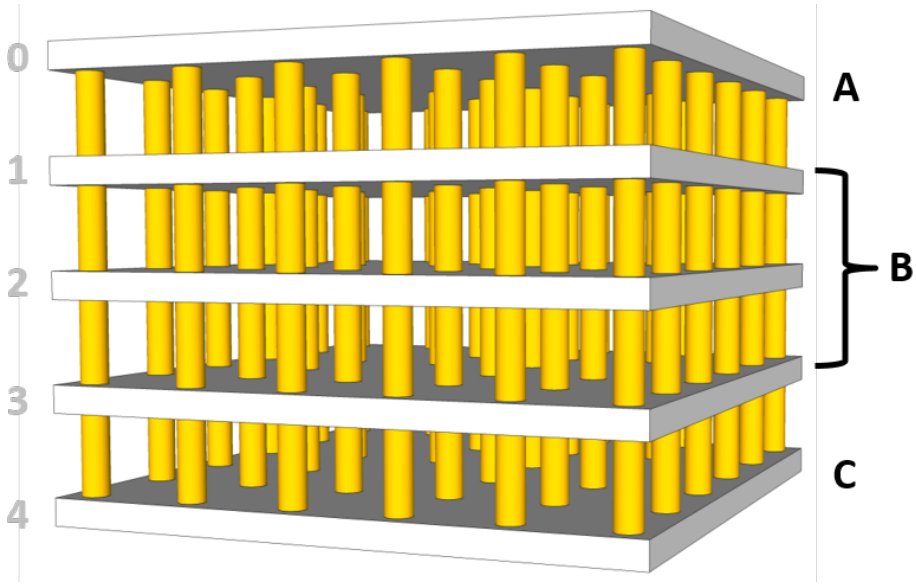

Figure 2.5: The scenario under consideration, in which two blocks of logic, A and C, are separated by a third logic block, B, and the number of connections between A and C must be determined. In a 3DIC, block B would consist of all logic planes between tiers A and C.

We can quickly adapt Eqs. (2.47) and (2.49) to determine the number of connections between any two tiers. First, we define N_g as the total number of logic gates in the design, N_{tiers} as the number of tiers in the 3DIC, and $N_s = N_g/N_{tiers}$ as the average number of logic gates on each tier. Then Eq. (2.49) becomes:

$$T_{A-C} = 2kN_s^p(b_{ac}+1)^p - kN_s^p(b_{ac}+2)^p - k(b_{ac}N_s)^p \tag{2.51}$$

where b_{ac} is the number of layers between tier A and tier C. Now that we have an expression for the number of connections between two tiers, A and C, we can determine the total number of vias in the ith layer, V_i, as

$$V_i = \sum_{j \neq i} T_{i-j} + \sum_{j<i<k} T_{j-k} \tag{2.52}$$

where the first term represents connections between tier i and all other tiers, and the second term represents all connections traversing tier i, i.e. connections between tiers below i and

tiers above i. The total number of vias required in the stack then becomes

$$V_{tot} = \sum_{i} V_i \tag{2.53}$$

This result can be quickly and easily calculated by setting up a matrix with each element i, j corresponding to T_{i-j},the number of connections between tiers i and j. For each i, the first term in Eq. (2.52) is calculated as the sum of the ith row, and the second term in Eq. (2.52) is calculated as the sum of the block formed by all elements to the right of the ith column and above the ith row, as shown in Fig. 2.6. In this example, the cells in blue are summed to calculate the total number of interconnects terminating on the second tier, and the cells in red are summed to calculate the total number of interconnects traversing this tier. By summing these two values, the total number of vias required on the second tier can be determined. Since the matrix is symmetric, Eq. (2.52) can also be calculated as the block formed by all elements to the right of the $i-1$th column and all elements above the $i-1$th row.

To illustrate the impact of the corrected TSV estimation algorithm, we evaluated Eq. (2.53) and compared it against the case where vias traversing layers are ignored. As can be seen in Fig. 2.7, the updated method predicts significantly higher via numbers in tall 3D stacks, and the old method begins to significantly undercount total via requirements for systems of 8 tiers and greater.

2.2.3 Validation

To validate Eq. (2.53) several netlists were partitoned into multiple tiers via spectral partitioning [57, 58] in order to simulate the process of breaking a 2D design into a multi-tier 3DIC. The rent parameters for each netlist were determined via repeated spectral bipartitioning in order to fit the ratio of terminals to gates to the relationship expected from Eq. (2.47) [54, 59]. Once the rent parameters are extracted, they are used in Eqs. (2.50) and (2.53) to calculate the expected number of vias required in each tier. These predic-

0	N_{01}	N_{02}	N_{03}	N_{04}
N_{01}	0	N_{12}	N_{13}	N_{14}
N_{02}	N_{12}	0	N_{23}	N_{24}
N_{03}	N_{13}	N_{23}	0	N_{34}
N_{04}	N_{14}	N_{24}	N_{34}	0

Figure 2.6: Graphical illustration of the calculations in Eqs. (2.52) and (2.53) for the determination of the total via number in the second layer of a 5-tier 3DIC.

Table 2.1: Comparison to actual data

Netlist	Cells	Nets	k	p
ibm01	12752	14111	21.9	0.45
industry2	12637	13419	16.6	0.59
industry3	12637	12637	11.2	0.59
cf_cordic	22023	30962	4.86	0.76

tions are compared to the actual number of connections between tiers, which is found by bipartitioning the netlist into an appropriate number of levels, and ordering the partitions vertically to minimize the total number of vias required for interlayer interconnection. In this analysis, off-chip IOs were neglected, and only intra-netlist connections were considered. The exclusion of off-chip connections may lead to a slight undercounting of the total via requirements in the design, as, were these netlists to be implemented as freestanding 3DICs, signals originating within the 3D stack which must be routed to the outside world must necessarily pass through every tier between their origin and the off-chip interconnects at the bottom tier of the design. The details of the netlists used for characterization are presented in Table 2.1.

The key improvement of the new method of TSV prediction over the previous meth-

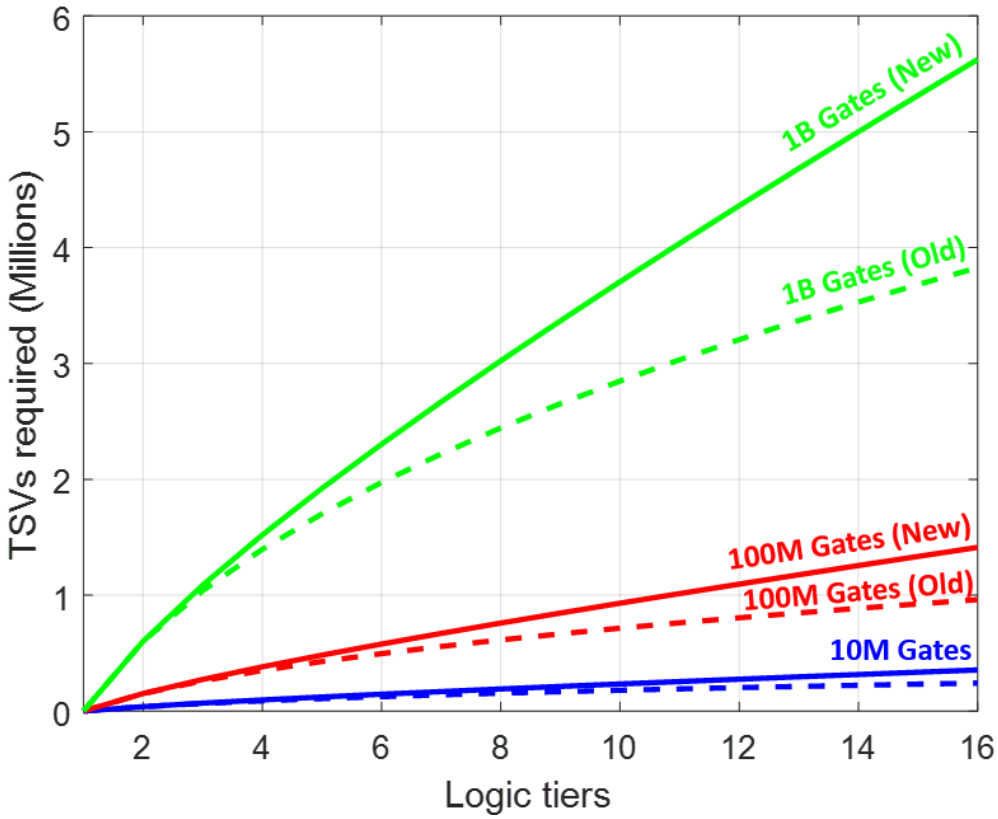

Figure 2.7: Comparison of the predicted total via number using the updated TSV prediction method (solid lines) and the previous method (dashed lines) for hypothetical systems with 10M (blue), 100M (red), and 1B (green) gates.

ods is the more complete accounting of through-tier vias, which have previously been neglected. The total TSV requirements under the old and new TSV methods are compared against the empirical TSV estimates for each netlist in Figs. 2.8 to 2.11. In all cases, the overall trend of the new method better matches the empirical trend as the number of logic tiers increases. The old method typically undercounts the overall TSV requirements significantly. The data are examined in greater detail in Figs. 2.12 to 2.15, in which the via requirements within *each tier* within a particular 3D implementation of each design are compared. In these cases we typically see higher intertier via requirements in the central tiers, which must support connections between the greatest number of adjacent tiers. The results for the largest netlist, cf_cordic, are further broken down in Fig. 2.15, in which this

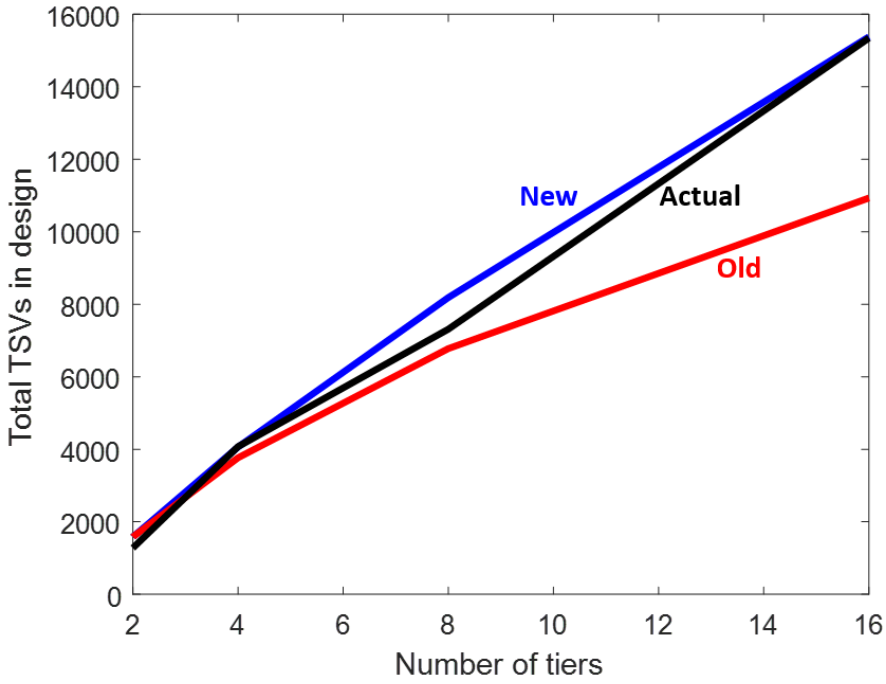

Figure 2.8: Comparison of the actual and expected total via requirements for the ibm01 netlist in hypothetical 3D configurations ranging from two to sixteen tiers.

netlist is examined in four-, eight-, sixteen-, and thirty-two-tier configurations. The impact of via undercounting clearly increases significantly as the degree of 3D integration increases; for shallow 3D stacks of up to four tiers, the impact of via undercounting does not appear to be significant, but it quickly becomes more relevant as the number of stacked tiers increases beyond eight.

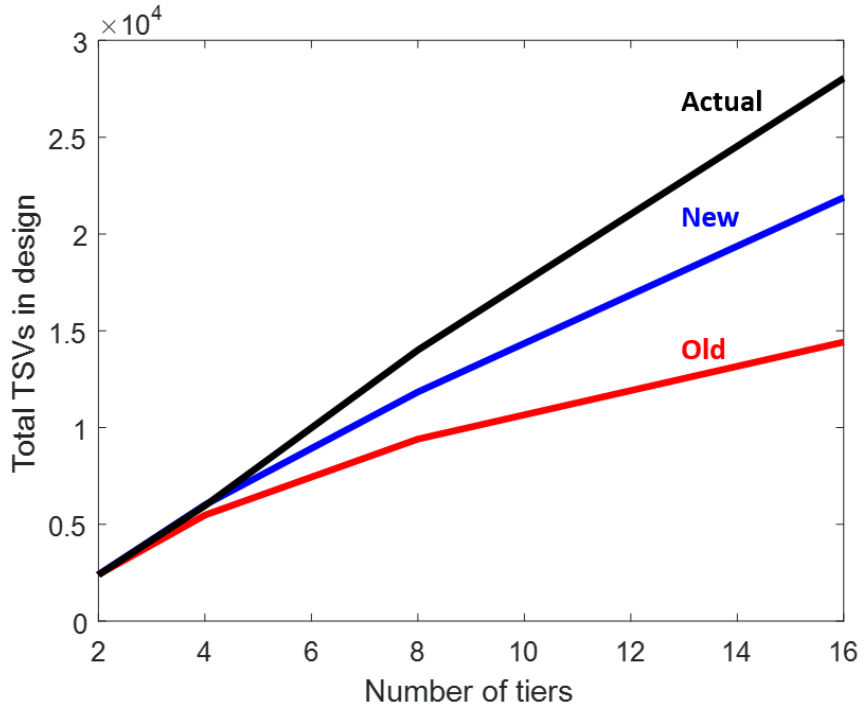

Figure 2.9: Comparison of the actual and expected total via requirements for the industry2 netlist in hypothetical 3D configurations ranging from two to sixteen tiers.

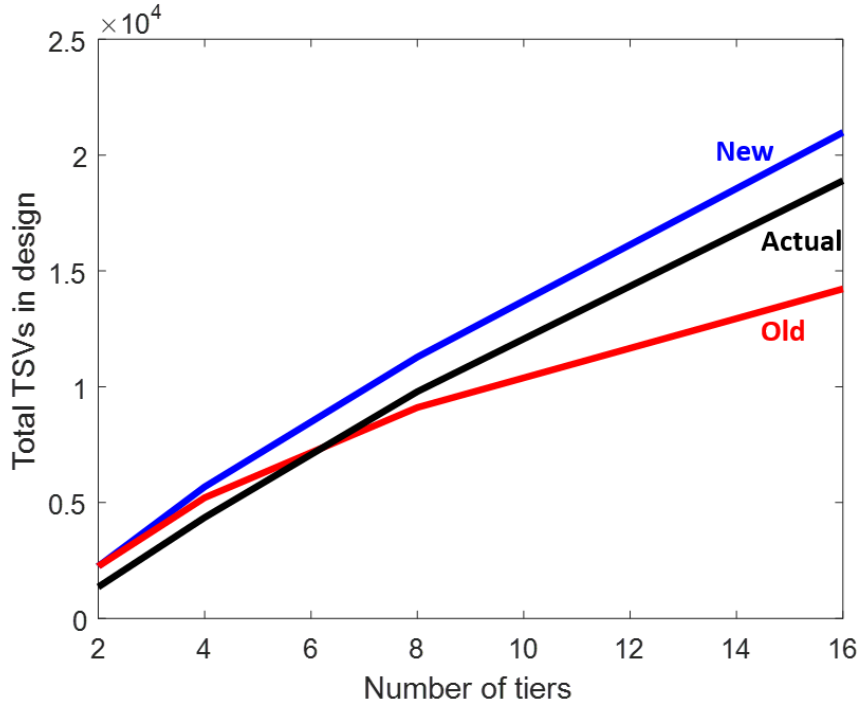

Figure 2.10: Comparison of the actual and expected total via requirements for the industry3 netlist in hypothetical 3D configurations ranging from two to sixteen tiers.

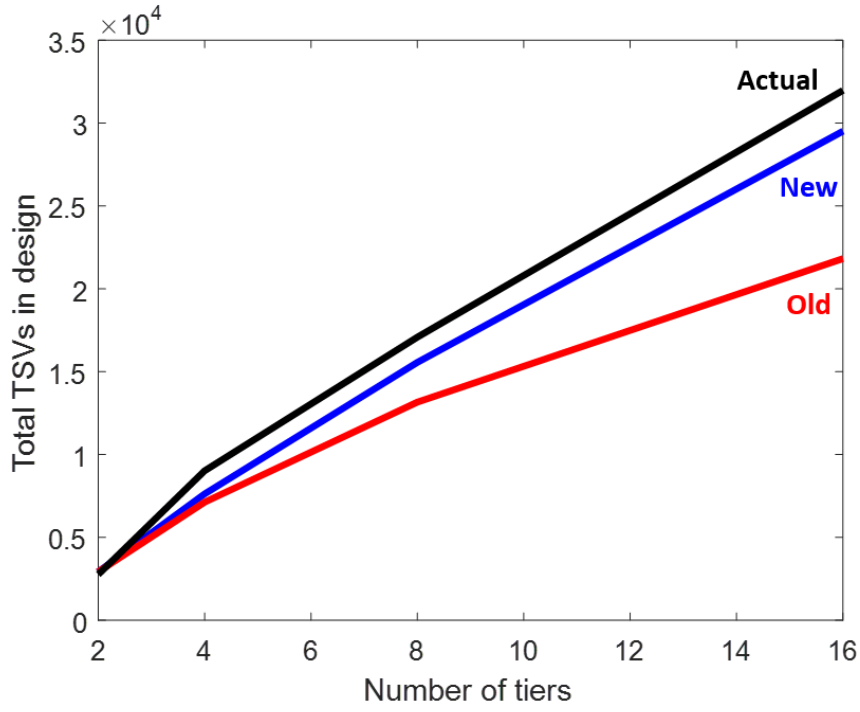

Figure 2.11: Comparison of the actual and expected total via requirements for the cf_cordic netlist in hypothetical 3D configurations ranging from two to sixteen tiers.

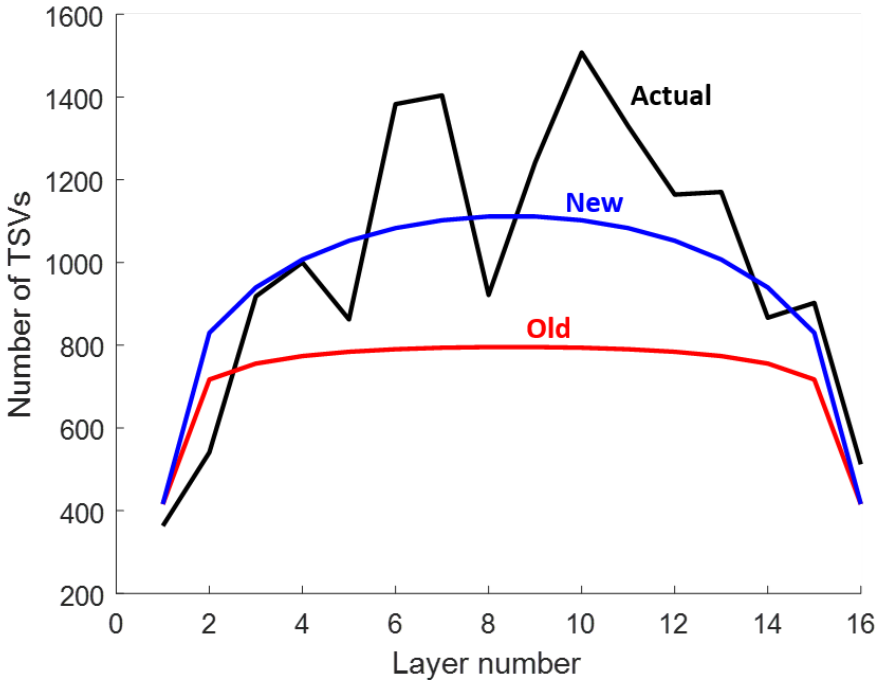

Figure 2.12: Comparison of the actual and expected via requirements on each logic tier for the ibm01 netlist in hypothetical 3D configurations ranging from two to sixteen tiers.

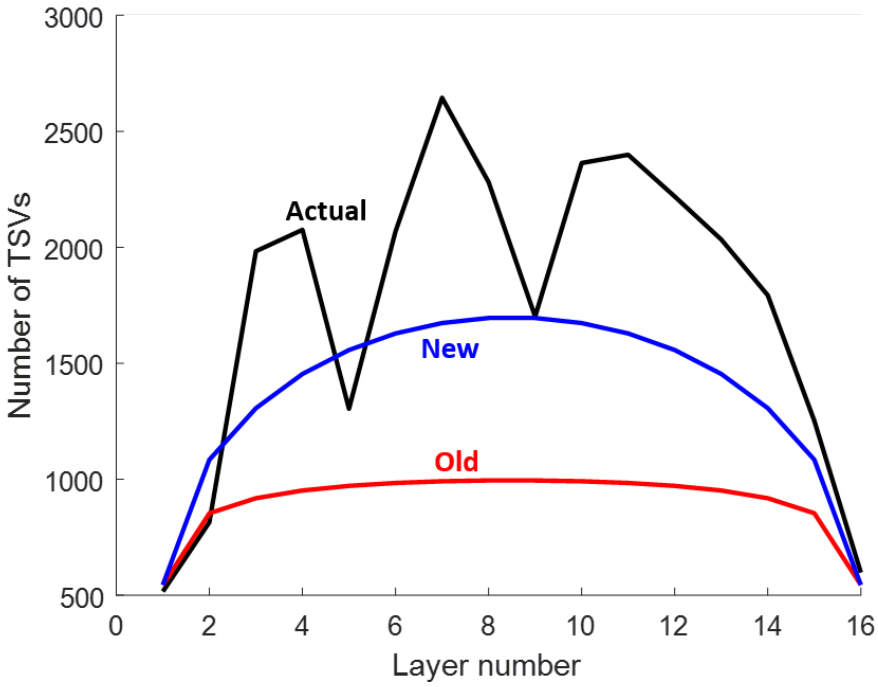

Figure 2.13: Comparison of the actual and expected via requirements on each logic tier for the industry2 netlist in a hypothetical 3D configuration with sixteen logic tiers.

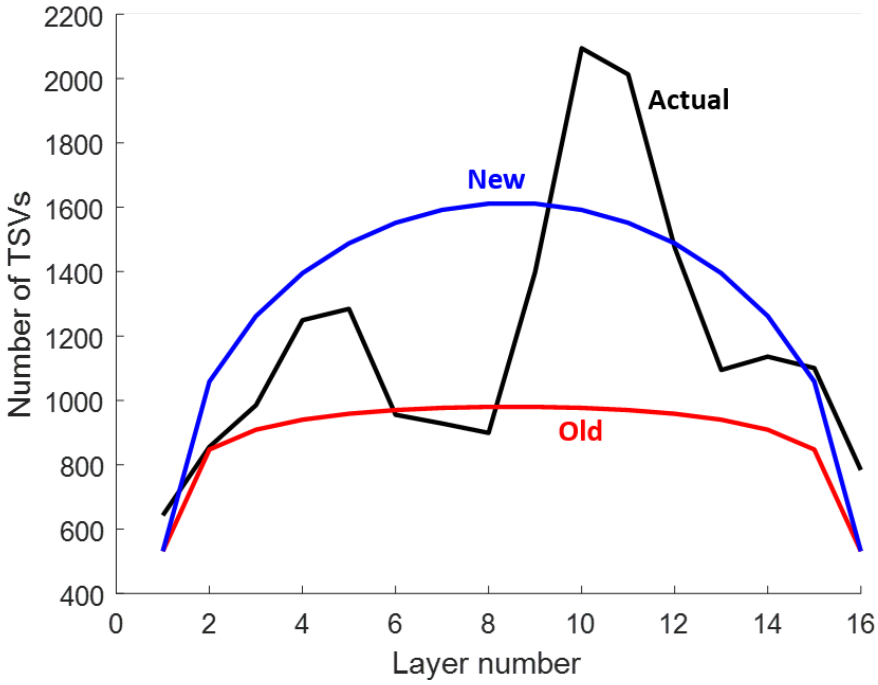

Figure 2.14: Comparison of the actual and expected via requirements on each logic tier for the industry3 netlist in a hypothetical 3D configuration with sixteen logic tiers.

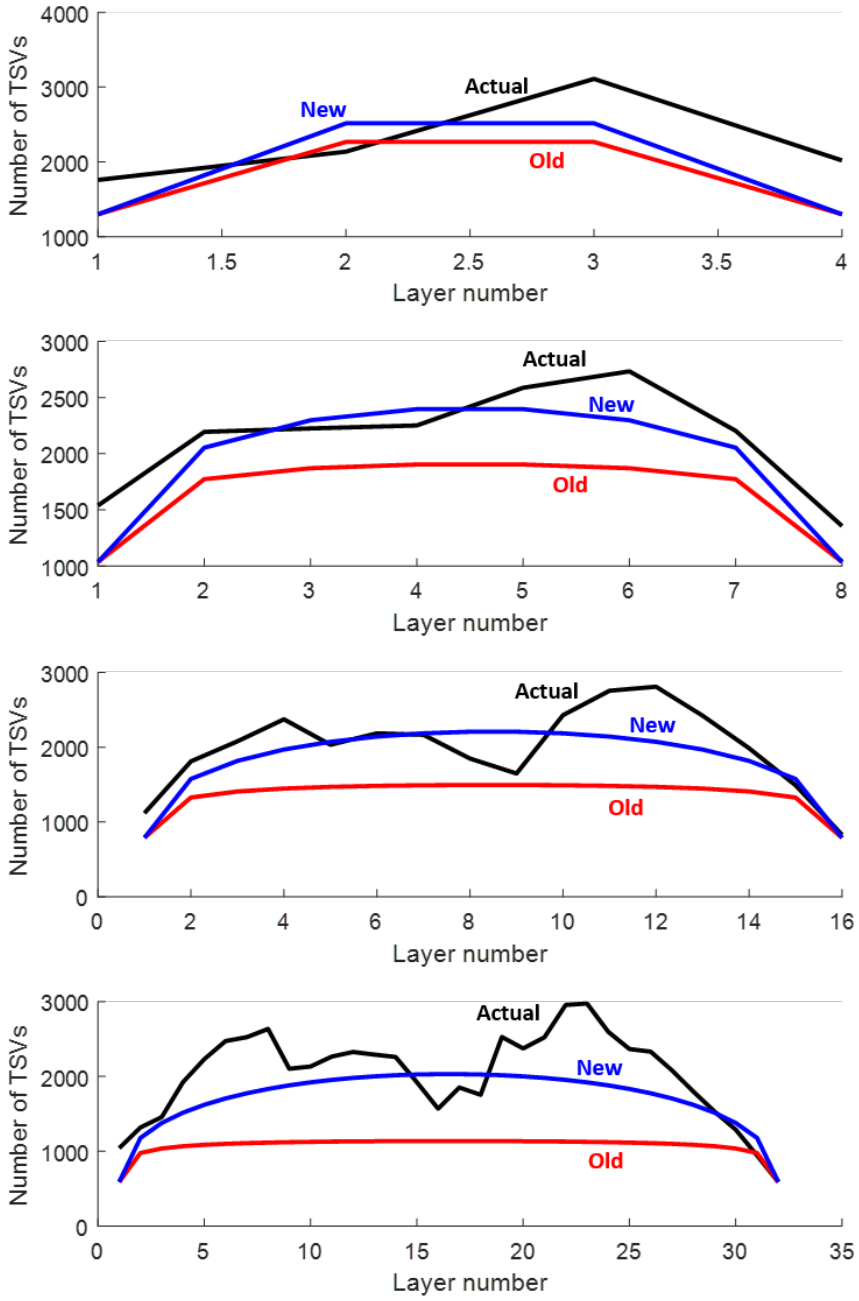

Figure 2.15: Comparison of the actual and expected via requirements on each logic tier for the cf_cordic netlist in hypothetical 3D configurations of 4, 8, 16, and 32 tiers. In all cases the old TSV estimation method significantly undercounts the TSV requirements throughout the design, especially in the central tiers.

CHAPTER 3

A VIRTUAL PLATFORM FOR 3D AND 2D IC DESIGN SPACE EXPLORATION

3.1 Introduction

Economic and physical challenges to transistor scaling are driving interest in 3D integration, but uncertainty regarding the fabrication costs and system-level tradeoffs of 3D integration complicate the design of three-dimensional integrated circuits (3DICs). Projections of 3DIC cost and performance are further impacted by the strongly-coupled nature of communication, power delivery, and thermal management in 3DICs. The 3DIC design space is complex, as 3DIC design encompasses a broad spectrum of possible design choices and integration methodologies, ranging from 2.5D interposer-based integration all the way to finely-grained monolithic 3DICs, as shown in Fig. 3.1, each with unique costs and strengths.

Additionally, different technologies must be evaluated for use in both 2D and 3DICs. Low-k dielectrics can be used to reduce the parasitic capacitances in the wiring stack, simultaneously improving RC delay and reducing the power consumption of the wiring network. Alternate wiring materials are also being considered to improve the RC delay

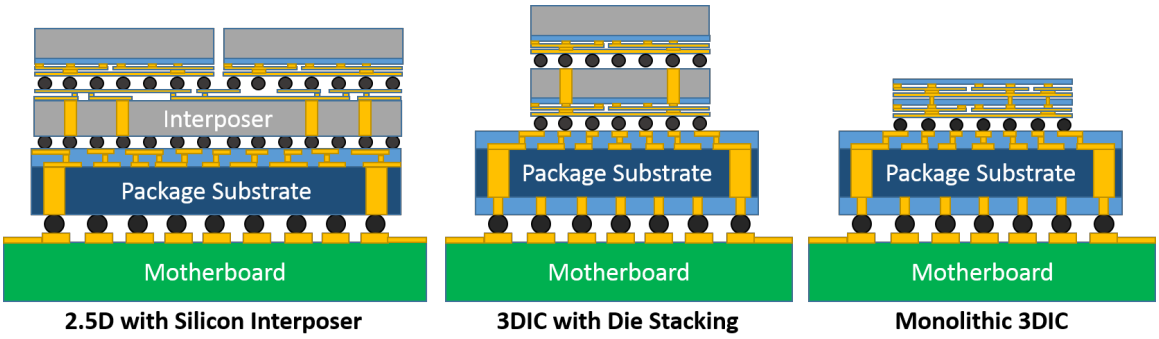

Figure 3.1: There are many potential configurations for 3DICs, each with their own costs and advantages. Designers must manage the complexity of the 3DIC design space in order to achieve higher performance and lower cost systems.

of on-chip interconnects, as well as to reduce the impact of electromigration [60]. Liquid cooling can be used to mitigate the thermal challenges in high performance 3DICs [33,34]. In order to understand when a 3D system might have advantages over a 2D system, all of these factors must be modeled simultaneously. Performing these coupled simulations with high fidelity is computationally intensive, however, rendering thorough exploration of the 3DIC design space challenging. Compact tools have been developed to estimate 2DIC performance [32,61], but no analog exists for 3DICs. Additionally, the existing 2D tools do not model power delivery or heat extraction, which are likely to be challenges for 3DICs.

The novel contribution of this work is the development of a compact tool capable of rapidly simulating the properties of 3DIC on-chip signal and power delivery networks. The simulator estimates the power consumption, simultaneous switching noise, and thermal profile of 2DICs and 3DICs, and enables the investigation of trends in performance and power consumption as materials, devices, and integration methodologies are varied. The details of the on-chip wires, TSVs, power delivery network, and steady-state thermal profile are all modeled in order to develop a self-consistent picture of the overall 3D system performance. The simulator is intended to help answer what-if questions and to guide more precise investigations into the details of 3DIC physical design.

The organization of this paper is as follows: in Section 3.2 the details of the models used and their interactions are presented; in Section 3.3 the simulator is benchmarked against wire pitch and TDP data from several commercial processors; in Sections 3.4 and 3.5 the simulator is used to investigate the impacts of advanced technologies on a 32nm CPU core. Specifically, the impacts of the interlayer dielectric, wire material, and 3D stacking on the power consumption, power supply noise, and metal layers required for routing are investigated.

3.2 Simulation framework

The simulation platform consists of the following:

1. 3D wire length distribution which accounts for TSV area
2. Metal layer pitch determination algorithms capable of handling alternate wiring materials
3. An optimal repeater insertion scheme
4. A power supply noise model for 3DICs
5. A finite difference thermal module for analyzing the thermal impacts of 3D integration

The simulation flow is shown in Fig. 3.2. First, the distribution of wire lengths in the system is estimated, which is in turn used to determine the number of wiring tiers required for signal routing, the wire pitch on each tier, and the number of repeaters needed to meet the delay constraint. In order to model the on-chip interconnects in a compact manner, the system is modeled as a homogeneous block of randomly interconnected logic. This assumption is commonly used with stochastic wire length models for rapid estimation of on-chip interconnect properties, at the cost of reduced insight into fine-grained design details [25,31,32]. In heterogeneous systems each block is modeled separately, and the results are assembled into an overall power density map of each tier in the design to determine the thermal profile throughout the stack. In order to better predict the performance of such systems, the method of [56] can be used to homogenize a heterogeneous system.

Once the parameters of the on-chip interconnects are known, their power consumption can be estimated. The transistor dynamic and leakage power is also calculated to determine the overall power requirements for the design. The total power consumption is used in conjunction with the TSV and package pin resistance and inductance to estimate the simultaneous switching noise in the power delivery network. Additional power pads and TSVs are inserted until the noise drops to acceptable levels. At this point the tool checks that the total TSV area does not exceed a user-specified limit; if TSV demand outstrips

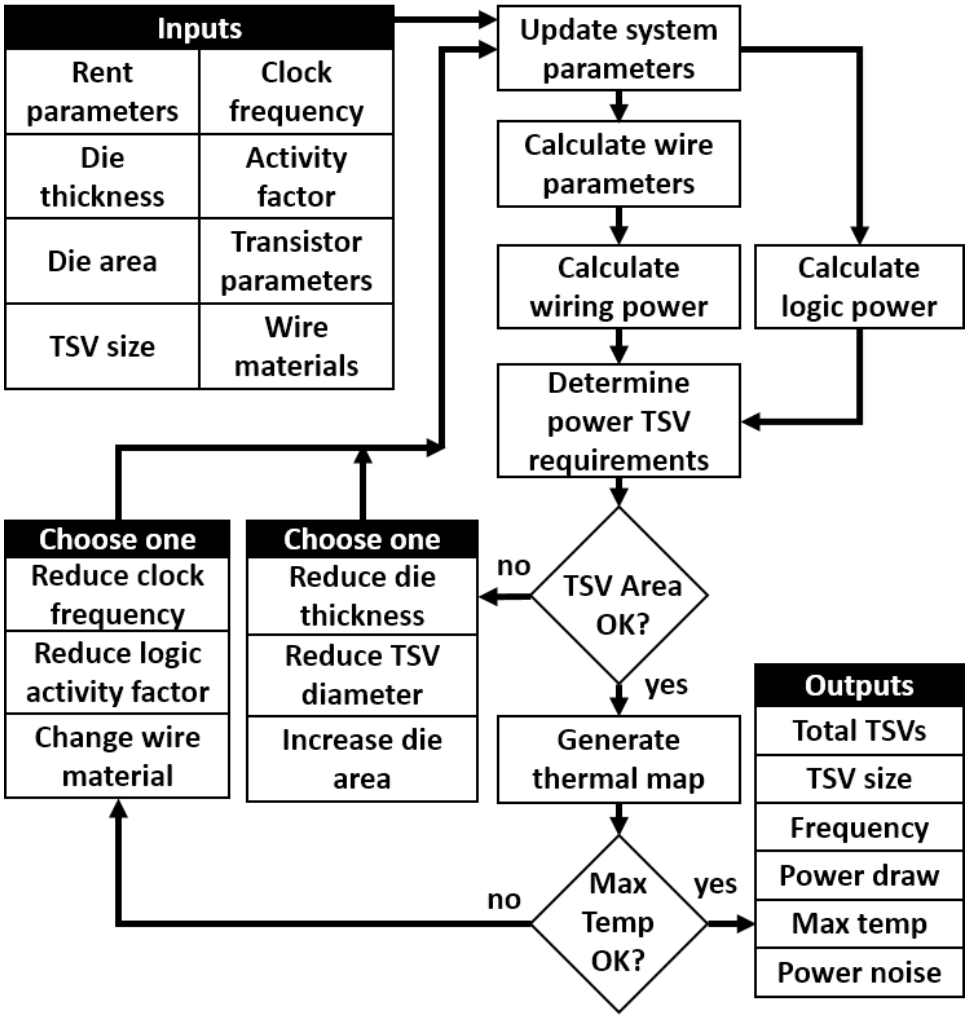

Figure 3.2: Block diagram of the simulation platform execution flow.

available area, the TSV diameter is reduced and the interconnect and power modules are rerun.

Once the design passes the TSV area check, the thermal module is used to estimate the maximum temperature in the 2D or 3D design. In order to do this, the material parameters of the die, wiring tiers (including wires and interlayer dielectric), TSVs, and interstitial layers are input into a finite difference thermal simulator, along with a heat transfer boundary condition representing the heatsink. If the maximum temperature in the stack exceeds a user-defined limit, the clock frequency is reduced and the previous modules are rerun until the maximum temperature is sufficiently reduced. As alternate means of power reduction,

the logic activity factor can also be reduced or the wire or interlayer dielectric materials be modified. The final design parameters are reported once the constraints on temperature and TSV area are satisfied.

The key limitations of this simulation platform are linked to the assumptions used to model the on-chip interconnects. Treating a design as a homogeneous block of randomly-interconnected logic allows rapid simulation of the aggregate properties of the on-chip wires, but limits visibility into the inner workings of the design. Off-chip IO and interconnections between blocks in heterogeneous systems are also not modeled, as design-specific information would be needed for accurate modeling. These omissions lead to underestimation of the system power, and limit the range of validity of the simulator, but they also enable the consideration of a wide range of system types without detailed design information.

3.2.1 Interconnect modeling

The on-chip interconnect network accounts for a significant fraction of the overall power consumption in many logic designs [32]; accordingly, accurate determination of the parameters of the on-chip interconnect network is crucial for developing reasonable estimates of the overall power consumption and operation frequency of an integrated circuit. Specifically, the number of metal layers used for wiring must be estimated, as well as the pitch of the wires on each level, in order to determine the overall wiring capacitance, which is a key factor in the maximum wire delay and power consumption.

Stochastic wire length models have been shown to be effective tools for the rapid prediction of interconnect properties in 2DICs [25, 32] and 3DICs [31], but the impact of TSV-induced gate-blockage in 3DICs was not considered until [62] introduced a correction to account for finite TSV size. The method of [62] requires brute-force calculation, however, which is too computationally intensive for rapid simulation. To address this issue, a compact correction to the wire length distribution was introduced in [44], which is used in

this work. Once the wire length distribution is known, the wire pitch and number of metal layers are determined using a bottom-up wire scaling technique [61], and a delay-optimal repeater insertion scheme is used to determine the size and number of repeaters required to meet timing constraints [63].

The impact of surface scattering and grain boundary scattering on wire resistivity are incorporated into the wire sizing algorithm with a combined Mayadas-Shatzke and Fuchs-Sondheimer (MS+FS) model, with specularity of 0.55 and backscattering probability of 0.43 [10]. The metal grain size is approximated as the smallest dimension of each wire.

3.2.2 Power supply noise modeling

Power supply noise must be suppressed to ensure reliable system operation, but power delivery in 3DICs is complicated by the limited area available for routing power interconnects between tiers and by the additional parasitic resistance and inductance of the TSVs used for power delivery. In order to determine the maximum allowable TSV diameter, the number and size of the power delivery TSVs must be estimated. The analytic 3DIC power supply model developed in [64] is used to estimate the simultaneous switching noise (SSN) in the 3D stack as a function of the number and size of the power delivery TSVs. The model uses the periodicity of the power grid to extrapolate the worst-case SSN in the system from the detailed frequency-domain behavior of a single power delivery unit cell. Power is assumed to be delivered via a regular rectangular array of power and ground pads or TSVs connected by planar power delivery wires. An inverse Laplace transform is used to convert the frequency response of the power delivery network into the temporal response, from which the worst-case noise can be easily extracted [65].

3.2.3 Thermal modeling

Thermal issues are one of the greatest challenges in 3DIC design. In order to design a thermally robust 3D system, the relationships between device technology, system perfor-

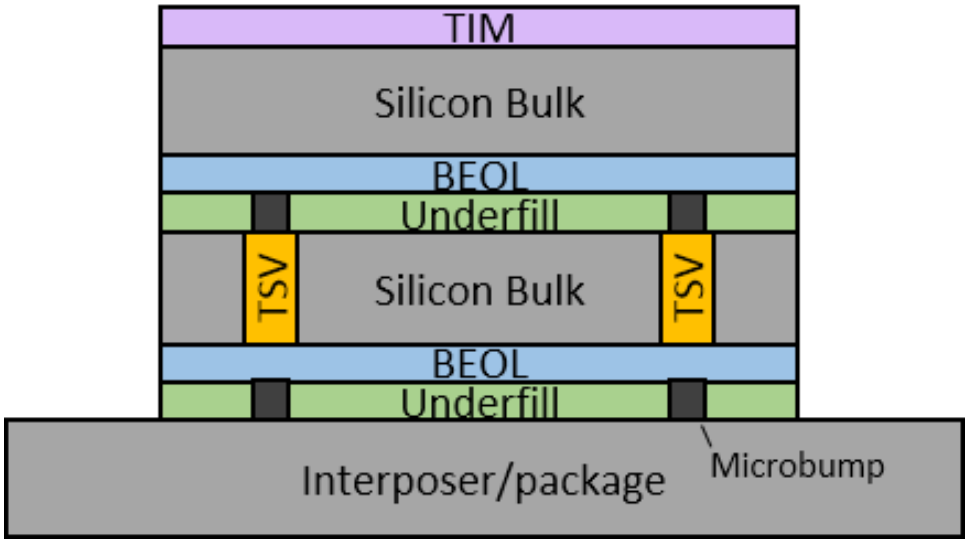

Figure 3.3: The geometry used in the thermal module. TSVs are individually meshed, in order to better capture the impact of 3D heat transport.

mance area constraints, and packaging materials and technology must be explored. To that end, we utilize a fast and accurate finite difference thermal model, described in [66], with a non-conformal meshing strategy described in [67].

The thermal configuration considered in the simulation tool is shown in Fig. 3.3. One or more dice are assumed to be stacked vertically atop an interposer (which may be either a conventional organic package substrate, or a silicon or glass interposer). Each die is separated into three regions: 1) the bulk silicon, 2) the BEOL, which is modeled as the volume-weighted average of the thermal conductivities of the wiring material and the interlayer dielectric, and 3) the underfill material between each die. The power dissipated by each die is applied as an excitation at the boundary between the bulk silicon and the BEOL. TSVs are modeled within the bulk of any dice below the top die in the stack. Each external boundary is modeled with a convective boundary condition; within-package boundaries are typically given a low heat transfer coefficient of 5 mW/m^{2}K, while the top and bottom surfaces are given higher values to reflect the cooling method used.

The accuracy of this finite difference module was assessed in [67], in which the performance of the finite difference scheme was compared against finite element ANSYS models

Table 3.1: Comparison to actual data

Processor	Node	Signal Wire Tiers Actual	Signal Wire Tiers Predicted	TDP (W)	Predicted Power (W)
E6850	65nm	8	8	65	60.27 (-7.3%)
E8600	45nm	8[2]	8	65	63.62 (-2.1%)
i7 880	45nm	8[2]	7	95	105.56 (11.1%)
i7 680[1]	32nm	8[2]	6	73	52.74 (-27.8%)
i7 2700k	32nm	8[2]	8	95	91.80 (-3.3%)

[1] Multi-chip package with 32nm CPU die and 45nm gpu/support die.

[2] Design has one additional global metal layer for power distribution.

of the same structure. The finite difference model was found to match the ANSYS results with a maximum error of 2.7%.

3.3 Validation

The simulator was validated by comparing its predictions against published data for Intel processors ranging from the 65nm node to the 32nm node [68–75]. For each test case, the chip area, number of logic transistors, number of memory transistors, and size and shape of the cores and memory blocks were gathered from published data. Logic cores were simulated with a Rent exponent of 0.6, while memory cores used a value of 0.4, and GPUs were simulated with a Rent exponent of 0.5 [54]. Each block was simulated separately to determine the number and pitch of metal levels required for routing and the total power consumption of each block. The pitch and number of metal layers used for the overall design were then set by the block which required the greatest number of wiring tiers (typically the CPU core). This information, along with the geometry and power requirements of each block was then used by the thermal module to determine the maximum temperature in the system.

The expected wire pitch, number of metal layers, power consumption, and maximum junction temperature generated by the simulator have been compared in Table 3.1. With

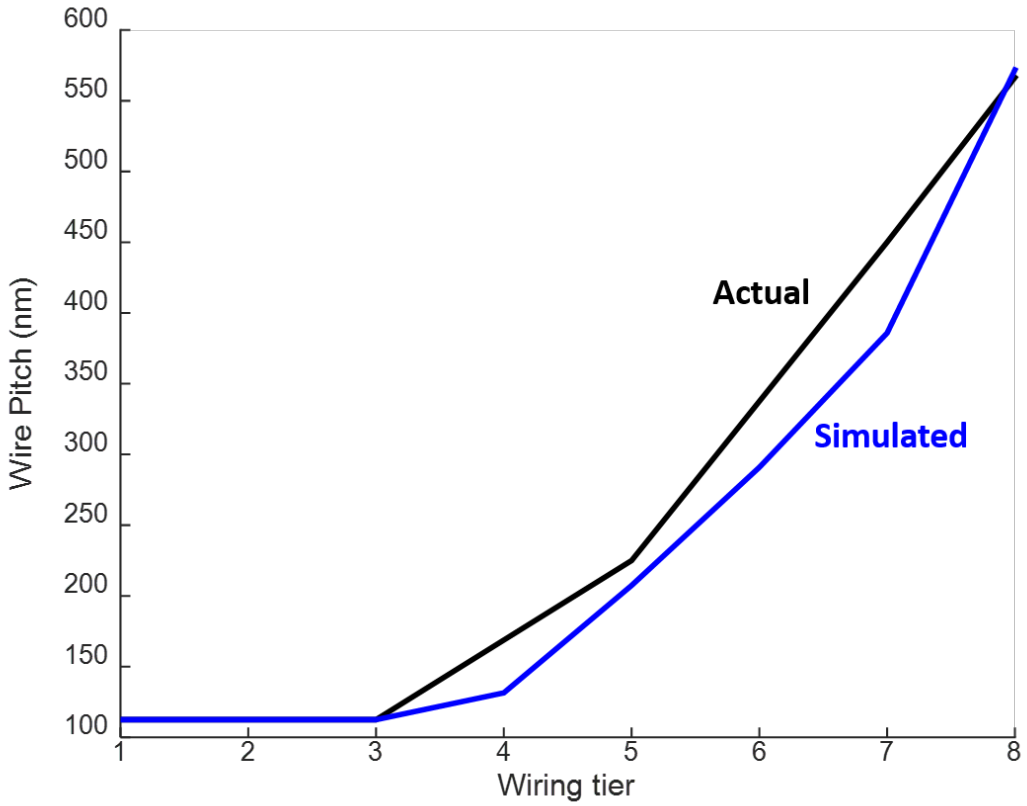

Figure 3.4: Actual and expected wire pitch in a Core i7 2700k processor.

the exception of the Core i7 680 test case, the tool shows reasonable agreement with the published data for these processors. The 45nm and 32nm test cases all have a total of 9 metal layers, but in all cases the wires on the top layer are sized very large and used for power and clock delivery, rather than signal routing. Since the interconnect estimation module is used estimate the size and pitch of signal wires only, the top metal layer in these designs is not counted for the purpose of signal wire pitch validation. Instead, the upper wire tier is modeled in the power delivery simulation module of the simulator, but without data on the power noise margin and the number of power and ground pads used in these designs, detailed validation of the power delivery module is not possible for these test cases. In addition to the number of metal levels, the simulator accurately predicts the wire pitch in each routing tier, as shown in Fig. 3.4.

The simulator underestimates both the number of wiring tiers and the overall power

consumption for the Core i7 680 test case; this error could be due to the fact that the Core i7 680 is the only design implemented as a multi-chip module (MCM), with a 32nm CPU die integrated with a 45nm GPU die in the same package. The simulator currently does not estimate the power required for inter-block communication in heterogeneous systems. While this will affect the power estimates for all test cases, the power required for communication between the two separate dice in the Core i7 680 is likely much larger than the power requirements for communication between GPUs and logic cores integrated on the same die.

The value of this simulation platform lies in its ability to consider many different effects very rapidly, enabling investigation of trends in the performance and power consumption of a reference design over a wide range of technologies and configurations. While the worst-case error in these benchmarks is relatively high (27.8%), the typical error is much lower, and this level of accuracy is sufficient for the investigation of power and performance trends in 2D and 3DICs. In the subsequent sections the simulator is used in this manner to investigate the impacts of material, technology, and packaging innovations on the power consumption and performance of 2D and 3DICs.

3.4 2D: the impact of materials innovation

One path towards increasing system performance is to achieve improvements in the wiring materials. The permittivity of the interlayer dielectric (ILD) directly impacts the parasitic capacitance of the on-chip wires, which in turn impacts the wire RC delay and power consumption. Additionally, decreasing the RC delay reduces the need for power-hungry repeaters.

In order to investigate the potential of ultra low-k (ULK) ILD materials, a 32nm Sandy Bridge Core i7 was simulated with a range of different ILD permittivities, ranging from 3.9 (silicon dioxide), all the way down to 1 (vacuum). Figure 3.5 shows that the number of metal layers required to fully route the Sandy Bridge processing core can be reduced

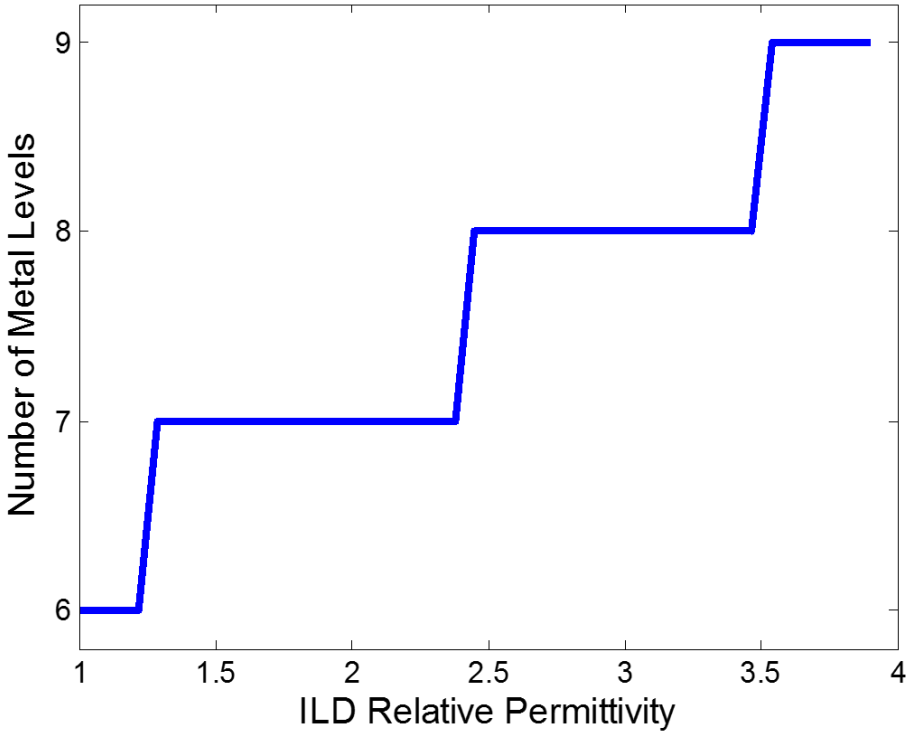

Figure 3.5: Impact of interlayer dielectric (ILD) permittivity on the number of metal layers required to route the wires in a Sandy Bridge Core i7 2700k.

from 8 to 6 if the relative dielectric constant of the ILD material can be brought below 1.3. Overall power consumption increases with ILD permittivity, as shown in Fig. 3.6, and significant power reductions are possible with ULK materials. The power reduction comes from both a reduction in wire power, as well as a reduction in the number and size of the repeaters needed to meet timing constraints.

As the critical dimensions of the smallest on-chip wires have decreased, electromigration has become a reliability concern at advanced process nodes [76, 77]. In order to address electromigration challenges at advanced process nodes, alternate materials may be required, potentially impacting signal performance, power consumption, and the number of metal layers required to fully route a design [60]. It is likely that only the lowest metal levels would use alternate materials [78].

To investigate the impact of alternate metals on routing, a 7nm Sandy Bridge CPU test

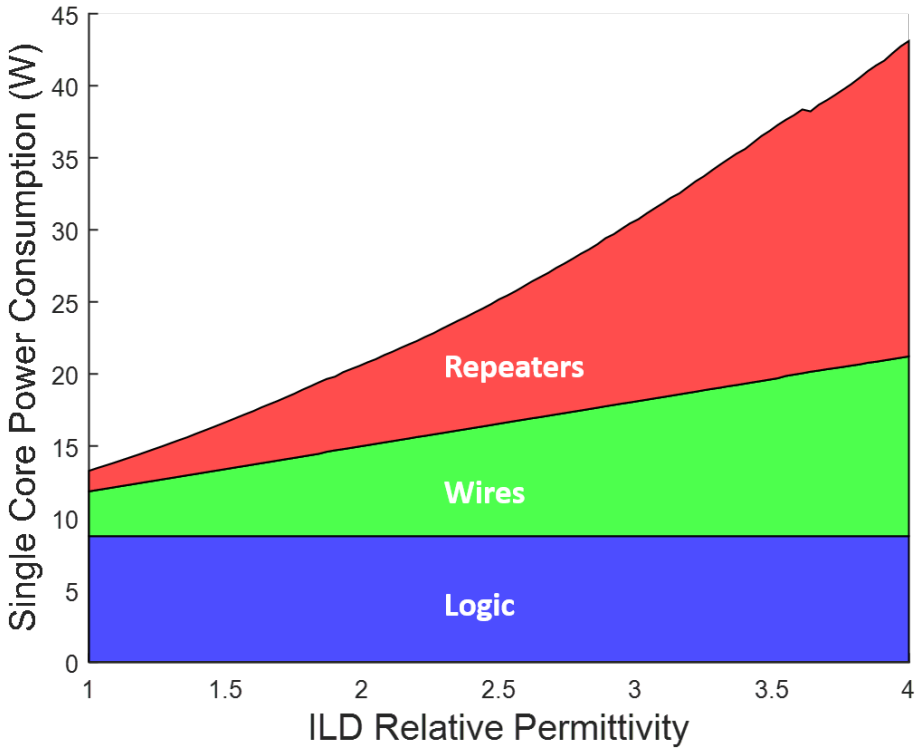

Figure 3.6: Impact of interlayer dielectric (ILD) permittivity on the power consumed by wires and repeaters in a Sandy Bridge Core i7 2700k core.

case was constructed by scaling down the gate pitch, minimum wire pitch, transistor size, and all other lengths in the 32nm Sandy Bridge by a factor of 4.57X (32/7). Two 7nm test cases were considered: *7nm A*, in which all wires are composed of an alternate material, and *7nm B*, in which only wires thinner than 25nm are replaced by the alternate material. The bulk resistivity of the alternate wiring material in both the 32nm and 7nm test cases was swept from 10 Ωnm (slightly lower than bulk Ag), to 60 Ωnm (slightly higher than bulk W). As can be seen in Fig. 3.7, higher resistivity metals can significantly increase the number of metal levels required for signal routing, but this effect can be mitigated by restricting the use of alternate metals to the lowest wiring tiers. Since the greatest numbers of wires are routed in the lowest tiers, small changes to their dimensions can have large impacts on the wiring stack.

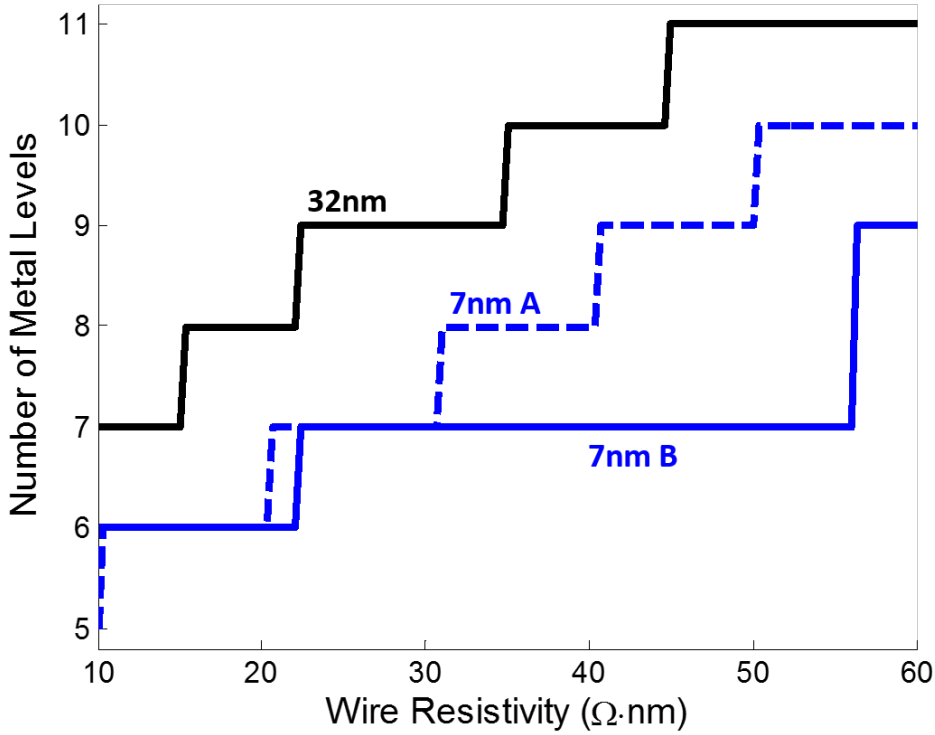

Figure 3.7: Impact of wire resistivity on the number of metal layers required to route the wires in a Sandy Bridge CPU. Three cases are considered: a) a 32nm Sandy Bridge core; b) 7nm A, a hypothetical 7nm Sandy Bridge core; and c) 7nm B, in which only wires with width below 25nm are modified.

3.5 3D: power reduction without exotic materials

3.5.1 Reducing power consumption

Implementing a design in 3D can greatly reduce the average length of the on-chip interconnects, leading to reductions in the average delay and power consumption of the signaling network [31]. In order to examine the impacts of 3D integration, a single 18.5 mm^2 CPU core from a 32nm Sandy Bridge Core i7 2700k is examined throughout this section. Unless otherwise noted, we assume a TSV aspect ratio of 20:1, and require that the TSVs use less than 10% of the total die area. Typically, 3DIC designs limit the TSV area to 1% or less to minimize the cannibalization of active area, but we have relaxed that limit here for illustrative purposes. In order to examine the impacts of 3D integration, a single CPU core from

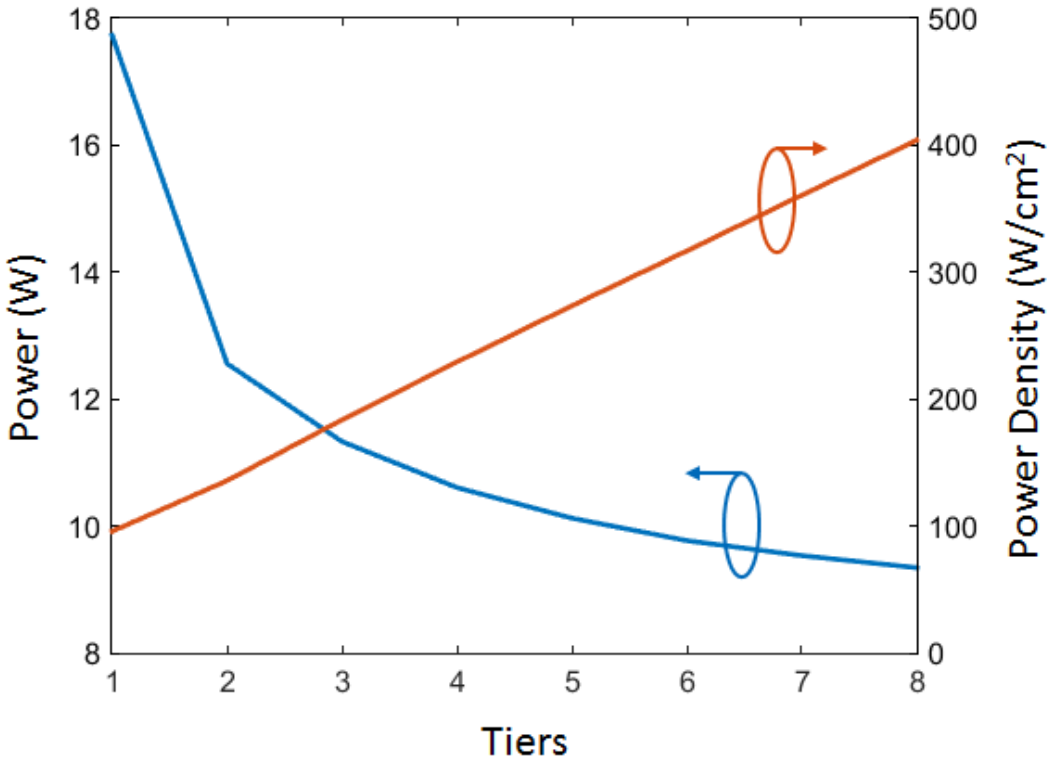

Figure 3.8: Impact of block folding on power consumption and power density of a single 32nm Sandy Bridge core.

a 32nm Sandy Bridge Core i7 2700k is examined throughout this section. We consider a scenario in which logic gates and blocks can be placed on any tier, and in which TSVs are used as point to point interconnects. The core is assumed to be partitioned into N equal pieces, which are then stacked vertically.

Significant power savings can be obtained by moving to a 3D design, as shown in Fig. 3.8, though the power reduction comes at the cost of increased areal power density, ultimately placing more stress on the heat sink. The design implications of the increased power density of 3DICs will be discussed further in Section 3.5.3. It is important to note that 3DICs reduce the on-chip communication power, fundamentally improving the energy efficiency of the system, as can be seen in Fig. 3.9.

In order to fully route a 3DIC, space must be allocated on each tier for TSVs. Since TSVs consume space that could be used for logic, it is desirable to minimize the fraction of

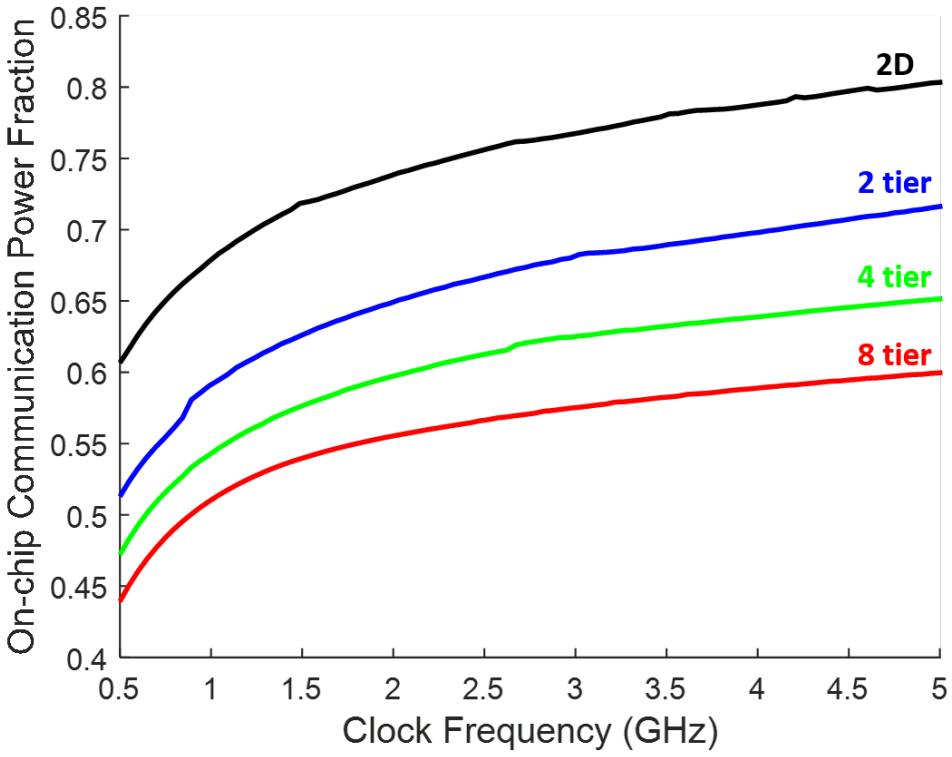

Figure 3.9: Fraction of power consumed for on-chip communication as a function of 3D configuration and operating frequency.

chip area consumed by TSVs. TSV diameter can be reduced by either increasing the TSV aspect ratio or by die thinning. Thicker logic tiers are attractive due to their higher mechanical stability, but they reduce the wire length advantages of 3DICs. TSVs are typically limited to diameters of 5-10μm and aspect ratios between 5-20:1 [34, 79].

The impact of die thickness on 3DIC power consumption is examined in Fig. 3.10. In this figure, we consider a single 32nm Sandy Bridge core folded over 2, 4, and 8 tiers. All connections between tiers are assumed to be point-to-point, and TSV number is determined using the method introduced in Section 2.2. In order to realize the greatest power reduction from 3D integration, the active layers should be as thin as possible, to maximize the number of long wires that can be shortened by block folding. When stacking large numbers of dice, the power gains from 3D integration can be entirely offset by the power consumption of the large TSVs required for 3D stacking. This highlights the need for aggressive scaling of

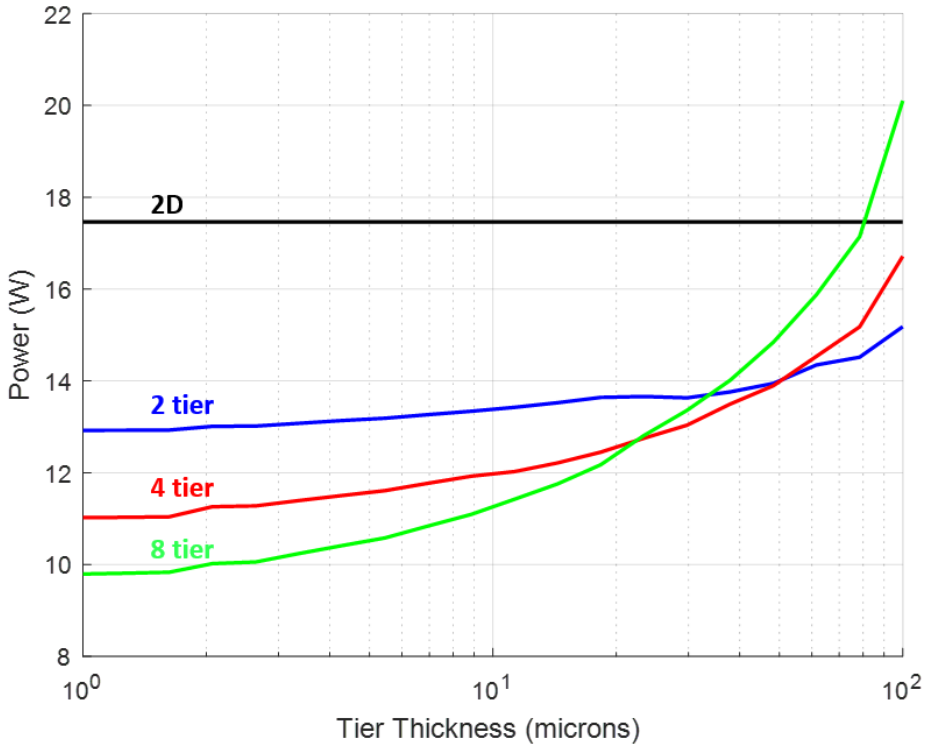

Figure 3.10: Impact of die thickness on power consumed by a single 32nm Sandy Bridge core implemented in 3D. Folding the logic core over additional tiers tends to reduce the power consumption of the core when small TSVs can be used.

TSV diameters, and suggests that monolithic-scale 3D integration may yield the greatest potential improvement in overall system power consumption. It is important to note that this is an extreme example, and that this configuration is not necessarily representative of heterogeneous 3D integration techniques, in which different devices (e.g. CPU and DRAM) are integrated into one 3D stack. In heterogeneous 3D stacking scenarios fewer intertier IOs would be required, and TSVs in such a stack would be replacing long, slow, and power-hungry board-level traces, rather than on-chip wires.

3.5.2 Power delivery

Power delivery in 3DICs is challenging, as a high performance 3DIC may have a significantly higher areal power density than an equivalent 2D chip, while simultaneously having less space available for routing power delivery resources. Additionally, power must be

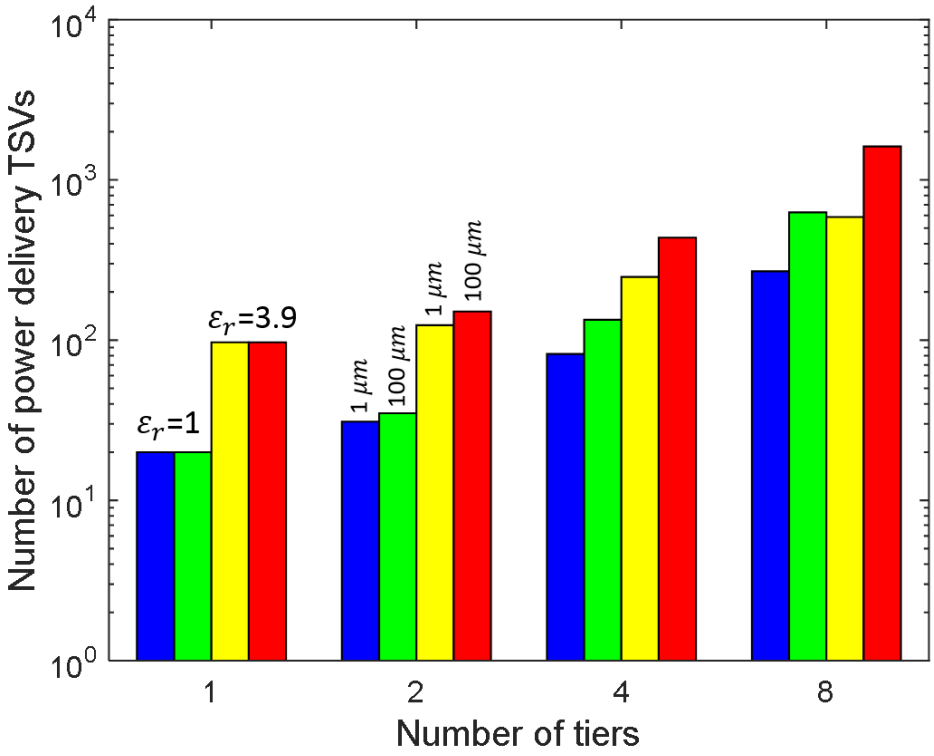

Figure 3.11: Impact of interlayer dielectric material, substrate thickness, and 3D integration on power pad/TSV requirements in a 3D Sandy Bridge CPU core.

delivered to each tier through TSVs, increasing the parasitic resistance and inductance of the power delivery network. The number of power connections required by each tier is determined by the power draw and power density of the system, which depends upon the dielectric properties and the substrate thickness (for 3DICs). Both 2D and 3D systems benefit from the use of ULK interlayer dielectrics, and the use of thin substrates in 3D configurations can further reduce the demands on the power supply network.

In order to explore these effects, the Sandy Bridge test case was simulated with several substrate thicknesses and dielectric permittivities, in order to determine the number of power delivery pads or TSVs needed to reduce the simultaneous switching noise to below 15% of the nominal supply voltage. The results are presented in Fig. 3.11. For two-tier stacks only a slight increase in power TSVs is observed over the 2D case, but the eight-tier implementation requires roughly an order of magnitude more power connections than the

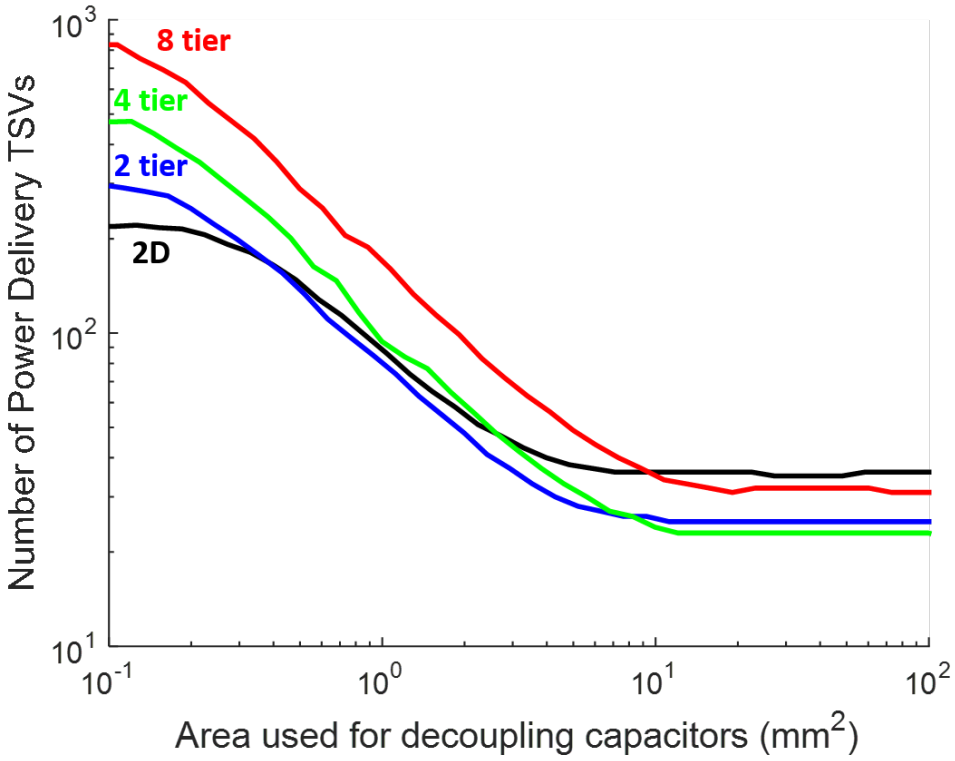

Figure 3.12: Power TSV requirements for a single Sandy Bridge core as a function of 3D configuration and area allocated for decoupling capacitors.

2D design. The ILD permittivity has a strong impact on the power delivery requirements, as it directly affects the power consumption of the on-chip communication network. The thickness of the 3DIC logic tiers is not a limiting factor for 2-tier designs, but 4- and 8-tier designs can realize nearly as much benefit from die thinning as from ULK dielectrics, due to the reduced power consumption of thin-tier 3DICs (Fig. 3.10).

Another method to reduce power supply noise is to integrate decoupling capacitors onto the die to compensate for the inductance of the power delivery network. While this practice can improve power quality, it also sets up a tradeoff between utilizing die area for logic and power delivery. To explore this tradeoff, the 32nm Sandy Bridge test case was simulated in 2D and 3D configurations with varying amounts of silicon area allocated for decoupling capacitors. The power TSV diameter is assumed to be 10 μm and the thickness of each die in the 3D stack is assumed to be 10 μm to investigate the potential of extreme die thinning.

While handling thinned wafers can be challenging, alternate integration schemes in which wafers are bonded and subsequently thinned could enable the stacking of such thin layers without the need for modified wafer handling processes [80]. Alternately, monolithic 3DIC fabrication techniques could enable designs with extremely small intertier distances [18]. For simplicity, the impact of electromigration on power delivery TSVs is ignored. As can be seen in Fig. 3.12, increasing the decoupling capacitance can significantly reduce the number of power pads or TSVs, but achieving high decoupling capacitance densities could be challenging. Even with the use of decoupling capacitors, a more stringent lower bound on power TSV number may be set by the need to keep the current density carried by each TSV low enough to avoid electromigration.

3.5.3 Thermal management

Thermal management is a key challenge for 3DICs. While the total power dissipation of an IC is expected to decrease as the system is partitioned into increasing numbers of layers (as shown in Fig. 3.8), the areal power density will still increase as tiers are stacked atop one another, leading to increased stress on the cooling system. In order to demonstrate the thermal capabilities of the simulation tool, and to quantify the thermal impact of 3D stacking, the performance of a single 32nm Sandy Bridge CPU core is examined in both 2D and 3D configurations, and with both air-cooled and liquid-cooled heat sinks. In all cases the heat sinks are located on the back side of the top die. The air-cooled heat sink has a heat transfer coefficient of 1.83 W/cm^{2}K and the fluidic heat sink has a heat transfer coefficient of 4.63 W/cm^{2}K [81]. Boundaries internal to the package are assigned a heat transfer coefficient of 0.005 W/cm^{2}K. The thermal conductivities used for the materials in the stack are presented in Table 3.2.

The CPU core was simulated in each configuration to find the maximum operating frequency which could be maintained while keeping the stack below 70°C. In order to focus solely on the thermal aspects of 3D stacking we assumed that the system was thermally-

Table 3.2: Material parameters used for thermal simulation

Material	Thermal Cond. (W/mK)
Silicon	149
Copper	400
Underfill	0.3
Microbumps	60
Silicon Dioxide	1.38

limited, and ignored other factors which could limit the operating frequency of the chip, such as clock distribution. The power consumed by the cooling solution is ignored, in order to isolate intrinsic die-level effects from the details of the heat sink.

As can be seen in Fig. 3.13, the maximum frequency decreases steadily as the CPU is folded across more tiers, as the increased power density (Fig. 3.8) of the system increases the strain on the cooling system. In all cases, the microfluidically-cooled cores can run faster than the air-cooled cores. With a more aggressive cooling solution the thermally-limited logic core considered here can be folded over up to four tiers while still achieving performance parity with an air-cooled 2D implementation.

It is important to note that the configuration considered here represents the most thermally challenging 3D integration scenario: the stacking of high performance logic. Systems which are not thermally-limited will be able to take greater advantage of 3D integration to increase performance and decrease power consumption. Additionally, the performance of thermally-limited 3D stacks could be further improved by integrating microfluidic coolers into each die in the stack to thermally decouple each tier, allowing each cooler to efficiently extract heat dissipated in adjacent tiers. To examine the potential impact of intertier cooling, we simulated the test case again, but assumed that all heat from each logic tier was removed by a hypothetical cooling solution placed between each die. We further assumed that the system was solely thermally-limited – in a real system it is likely that power delivery and timing constraints would impose additional performance constraints, but the value of this exercise lies in identifying the full potential of thermally-unbound 3D integration. As can be seen in Fig. 3.14, significant performance increases could be attained

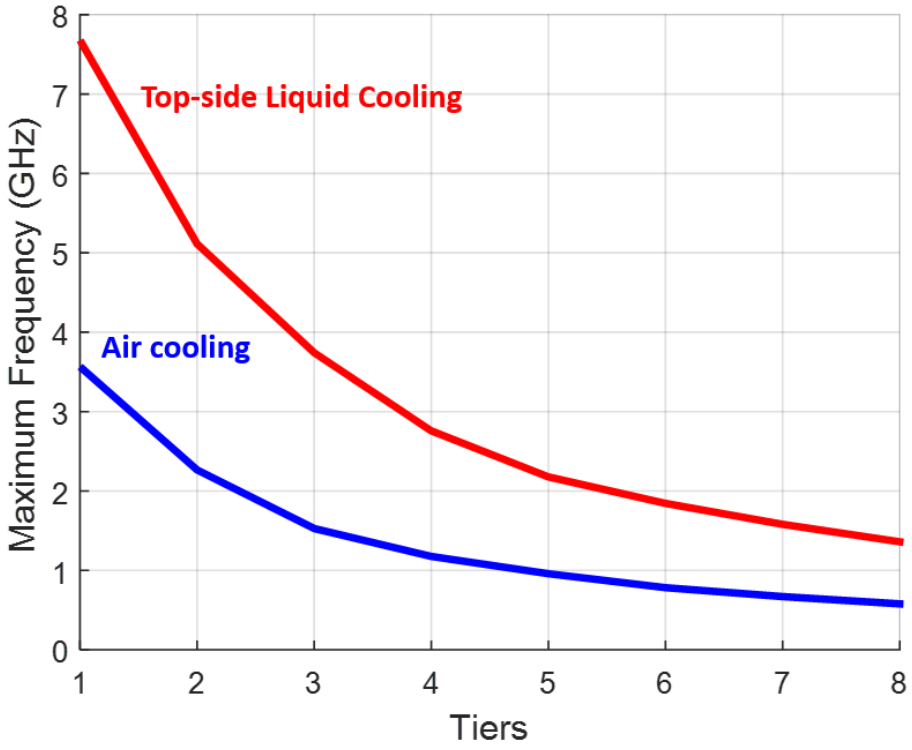

Figure 3.13: Maximum clock frequency of 2D and 3D 32nm Sandy Bridge CPU cores limited to 70°C under air cooling (blue) and liquid cooling (red).

by moving to a 3D configuration. The performance gains in this case come from significant reduction in on-chip communication power, as discussed in Section 3.5.

As intra-stack heat transfer is a critical concern in 3DICs, we also investigated the impact of varying the logic tier thickness on maximum performance. The same Sandy Bridge test case was simulated in 1, 2, 3, and 4-tier configurations, with substrate thicknesses of 100 nm, 1 μm, and 100 μm, representing 3D integration scenarios ranging from fine-grained monolithic 3D integration, to conventional TSV-based die stacking. As before, the design is assumed to be thermally constrained, and the maximum clock frequency which results in an operating temperature below 90°C is found. As shown in Fig. 3.15, increasing the substrate thickness degrades the thermal performance of the stack, resulting in a reduction in maximum attainable performance, due to the additional thermal resistance introduced by the thicker logic tiers, as well as the increased power consumption of the

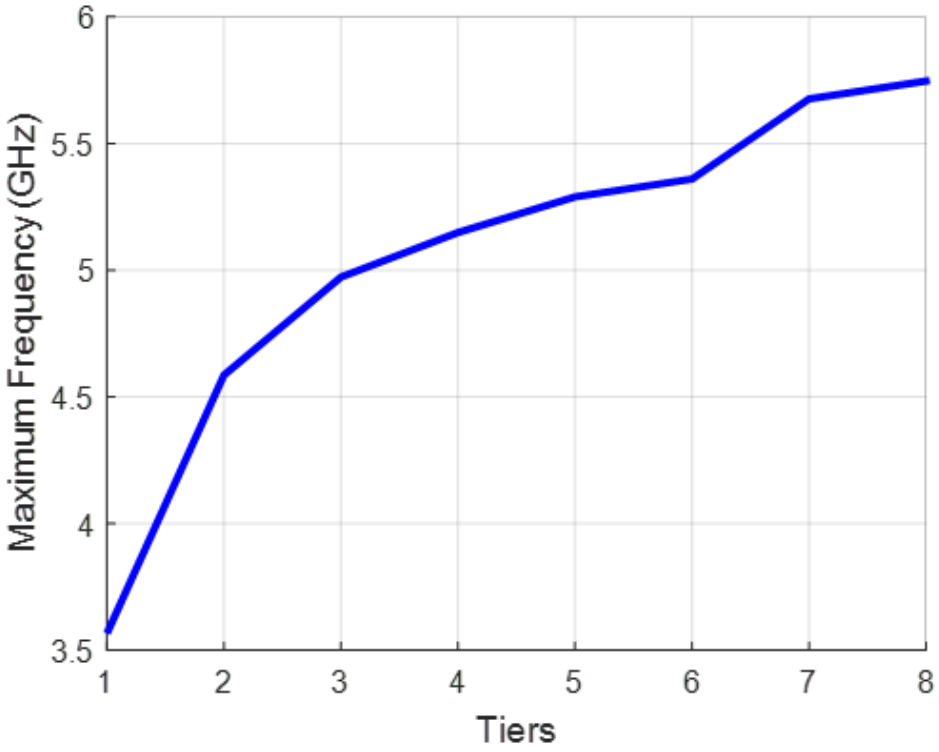

Figure 3.14: Maximum clock frequency of 2D and 3D 32nm Sandy Bridge CPU cores limited to 70°C, assuming heat can be removed from between the logic tiers.

overall device (as discussed in Section 3.5).

3.6 Impact of manufacturing constraints on monolithic 3DICs

Monolithic 3D integrated circuits (3DICs) are an attractive option for extending the density and performance gains demanded by Moore's Law without requiring additional 2D scaling [16–18]. Monolithic 3DICs achieve ultrahigh density vertical integration by dramatically reducing the distance between active tiers; in typical 3DICs, each die in the stack will be between 10-100 μm thick, whereas monolithic 3DICs are expected to have intertier distances of 0.1-1 μm. Reducing the intertier distance allows the size of the intertier vias to be dramatically reduced, enabling ultrahigh density vertical integration. Unfortunately, existing monolithic 3D (M3D) integration schemes require high-temperature processing to form the upper layer(s) of transistors, necessitating novel complex processing and materi-

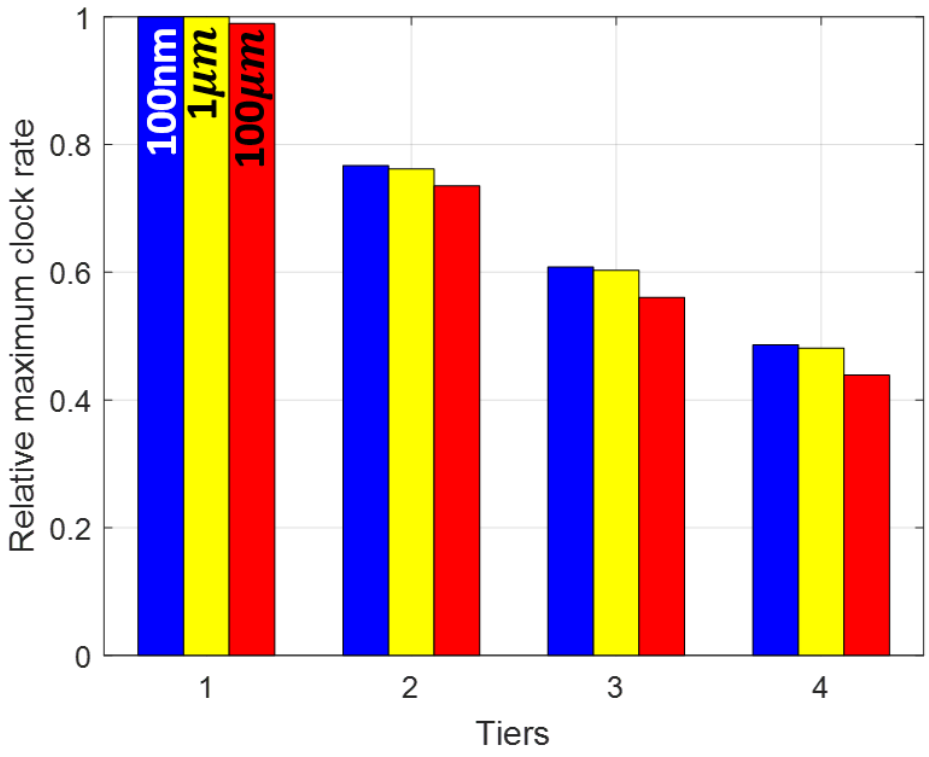

Figure 3.15: Maximum clock frequency of 2D and 3D 32nm Sandy Bridge CPU cores implemented with 100nm (blue), 1 μm (yellow) and 100μm (red) substrate/logic tier thicknesses. The thermal limit in each case is 90°C under air cooling. The maximum attainable clock frequencies are normalized to the 100nm single-tier case.

als to ensure interconnect and transistor reliability. To address this challenge, bottom-tier wires and interlayer vias (IVs) in M3D schemes are typically fabricated from tungsten or some other thermally-resistant material. As the resistivity of W is much higher than that of Cu, it is likely that the use of W wires and IVs will impact signal and power quality. No studies of the impact of the use of W wires on routability or power quality of monolithic 3DICs have been performed, however. The novel contribution of this work is the investigation of the impact of wire and power via resistivity on routing and power quality with a compact electrical and thermal 3DIC simulation tool [82]. The structure of this paper is as follows: in Section 3.6.1 the simulation methodology used for this work is discussed; in Section 3.6.2 the impact of wire resistivity on signal routing in monolithic 3DICs is investigated; in Section 3.6.3 the impact of intertier via resistivity on power delivery in monolithic

3DICs is discussed.

3.6.1 Methodology

We use a compact 3DIC simulation tool developed in [82] to investigate the impact of alternate wire resistivities on power delivery and signal routing in monolithic 3DICs. The simulation tool incorporates a 3DIC stochastic wirelength model [44], wire sizing algorithms [30], an optimal repeater insertion algorithm [63], a frequency domain 3DIC power delivery simulator [65] and a finite difference thermal module [67], and is described in greater detail in [82]. The simulator requires only high-level parameters describing the system of interest, such as the number of transistors in the design, the Rent parameters [54], the target clock frequency, the minimum wire pitch, the average gate pitch, the minimum inverter resistance and capacitance, and the wiring material resistivity and size parameters. From these relatively coarse descriptors, the simulator predicts the number of metal levels needed for routing, as well as the overall power consumption and simultaneous switching noise of the design.

In order to predict the impacts of alternate wiring materials on industrially-relevant designs, we use the 32nm Sandy Bridge CPU core test case benchmarked in [82] as a foundation for the modeling efforts presented here. To investigate the impacts of scaling, all physical dimensions in the design (transistor width, gate size, minimum wire pitch, and so on) are scaled down linearly. To investigate the impacts of wiring materials, the wire resistivity is swept from 10 Ωnm (slightly lower than bulk Ag) to 60 Ωnm (slightly higher than bulk W). These values are chosen to develop a picture of the overall wiring material solution space. In this work we consider only conventional metal wires; exotic interconnect structures such as graphene nanoribbon or carbon nanotube interconnects are not included in this analysis.

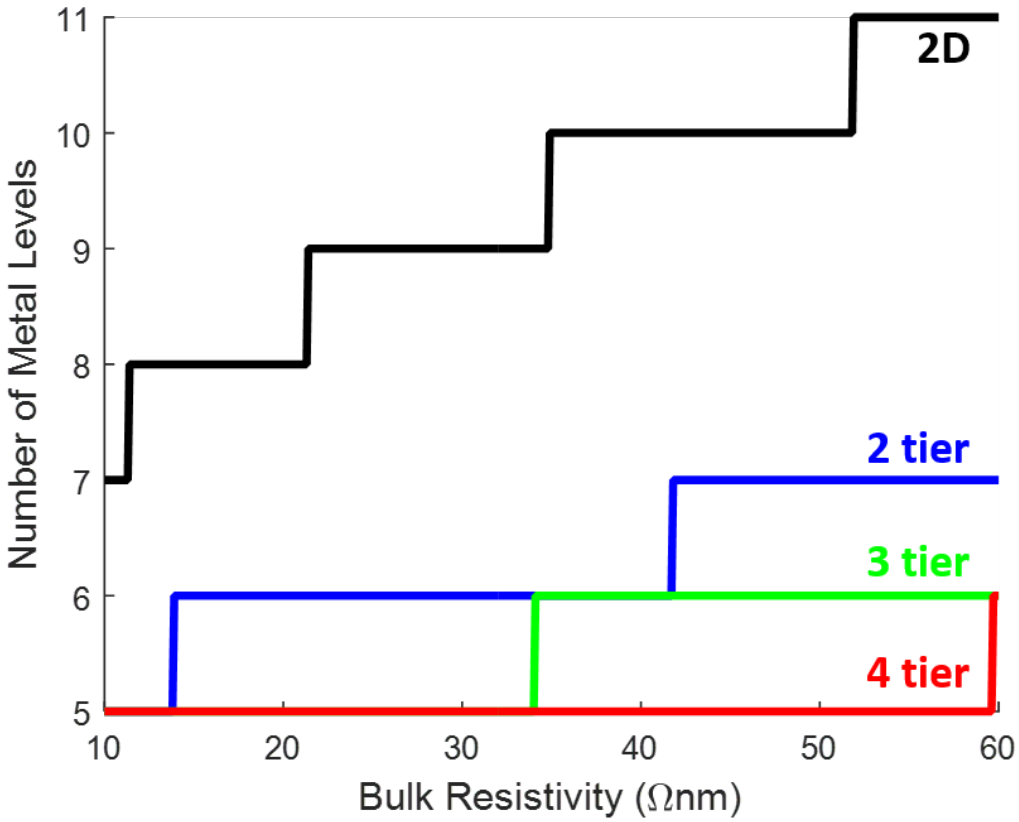

Figure 3.16: Number of interconnect tiers per die as a function of wire resistivity for a 32nm monolithic 3D Sandy Bridge CPU core.

3.6.2 Signal routing

Monolithic 3D concepts presented thus far typically utilize tungsten wires for lower-tier metallization [18]. In order to quantify the impact of this modification on a typical design we simulated a single CPU core from a 32nm Intel Sandy Bridge Core i7 using the simulation framework described in [82]. Conventional 2D designs were considered, as well as monolithic 3D designs with up to four logic tiers. In each case a modified optimal repeater insertion scheme is used: repeaters are inserted starting with the longest wires, and are continually inserted until either a) no wires remain which would benefit from repeater insertion, or b) the die area required for repeaters reaches 20%. As can be seen in Fig. 3.16, additional wiring tiers are necessary as the wire resistivity increases. A 2D design using

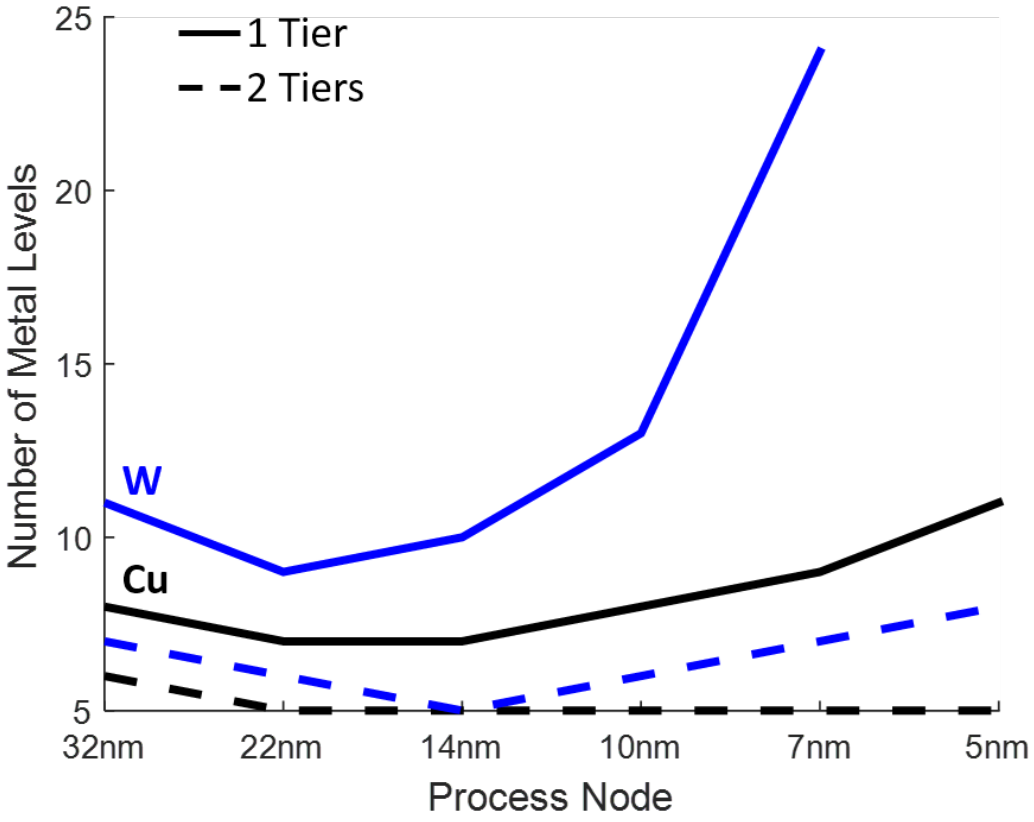

Figure 3.17: Number of interconnect tiers per die as a function of process node for a 32nm monolithic 3D Sandy Bridge CPU core implemented with different wiring materials. The 2D design using W wires becomes unroutable at the 5nm node.

copper wires is projected to require 8 signal routing tiers, which is in line with published data [73], whereas the same design using tungsten wires would require 11 signal tiers. Implementing the same design in 3D can significantly reduce the number of metal tiers per die, but the use of high-resistivity metals still requires additional signal routing tiers.

In order to examine the impacts of scaling, the 32nm Sandy Bridge test case was used as a template to create equivalent test cases at the 22nm, 14nm, 10nm, 7nm, and 5nm nodes by linearly scaling the physical dimensions of the chip, transistors, and wires in the design. In this case the two extremes of copper and tungsten were compared for both single-tier (2D) and two-tier (3D) designs. As can be seen in Fig. 3.17, the designs using tungsten wires always require more interconnect tiers than the designs using copper wires. The number

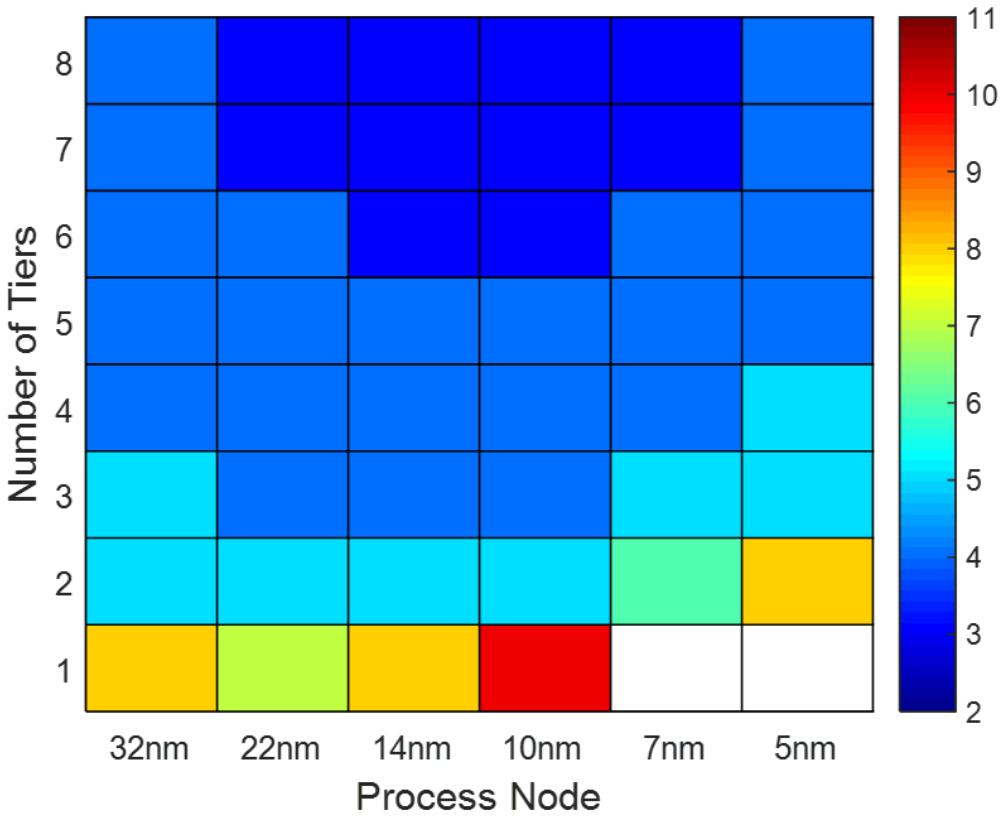

Figure 3.18: Number of interconnect tiers per die as a function of number of 3D tiers across different process nodes, assuming copper wires are used. White ares indicate designs which were considered unroutable.

of metal levels required for routing the tungsten designs diverges sharply at the 7nm node, and the 2D tungsten-based design becomes unroutable at the 5nm node.

The 3D designs both require significantly fewer metal levels for signal routing, due to the decrease in wire length achieved by moving from 2D to 3D. The increase in wiring tiers with scaling is attributed to the heightened consumption of repeaters as both the resistance and resistivity of the on-chip wires increase. The relationship between scaling, wire resistivity, and routing tiers is further examined in Figs. 3.18 and 3.19, which compare the number of metal levels required for signal routing in monolithic 3DICs using copper wires (Fig. 3.18) and tungsten wires (Fig. 3.19). Designs using tungsten wires require significantly higher numbers of signal routing tiers, and quickly become unroutable at advanced

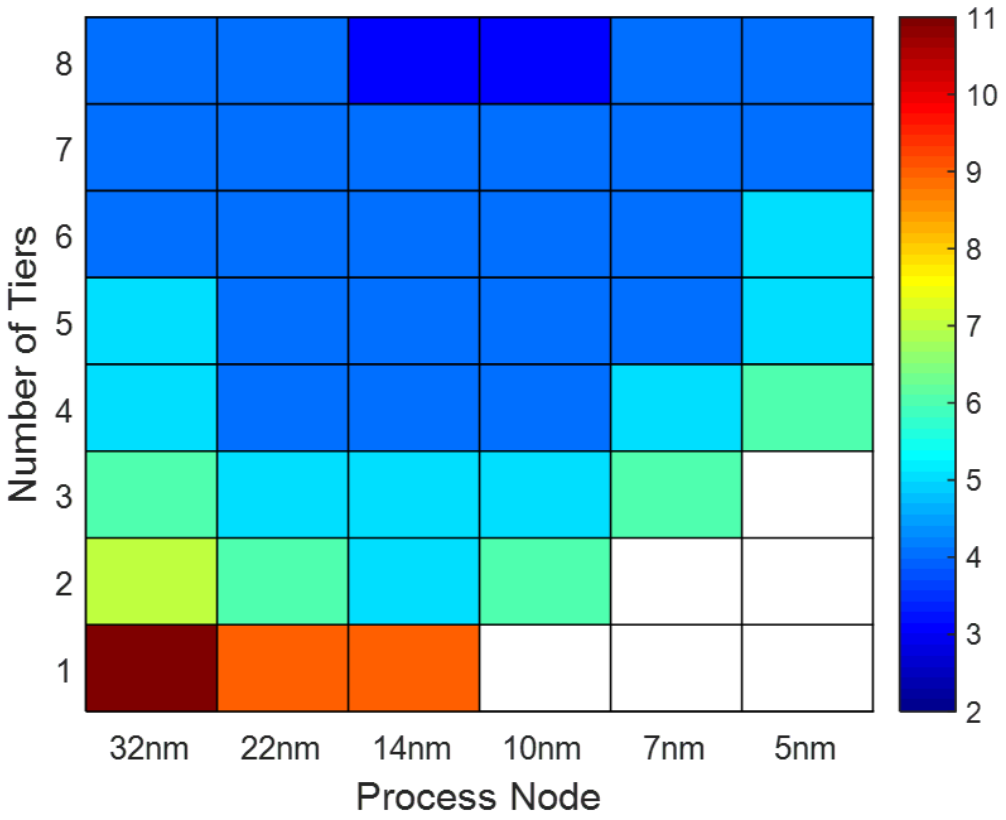

Figure 3.19: Number of interconnect tiers per die as a function of number of 3D tiers across different process nodes, assuming tungsten wires are used. White ares indicate designs which were considered unroutable.

process nodes, suggesting that monolithic 3DICs using tungsten wires may suffer from interconnect congestion issues.

3.6.3 Power delivery

Modern integrated circuits require highly stable power supplies for nominal operation, necessitating careful design of the on-chip power delivery network to minimize voltage noise and instability. The challenge of power delivery is exacerbated in 3DICs by the need to deliver power to multiple logic tiers through intertier vias, which introduce additional parasitic resistance and inductance into the power delivery network.

Further complicating 3DIC power delivery is the need to minimize the die area con-

sumed by intertier vias. While this problem is most keenly felt in conventional 3DICs, which have through-silicon via (TSV) diameters of roughly 5-10 μm, even monolithic 3D systems must trade potentially-useful silicon area for intertier vias. Since monolithic 3D systems are expected to use tungsten intertier interconnects rather than copper, power vias in monolithic 3DICs will introduce additional parasitic resistance to the power delivery network.

In order to gauge the impact of via resistivity on power delivery in monolithic 3DICs, we simulated the 32nm Sandy Bridge test case in 2D and 3D configurations with via resistivity ranging from 10 Ωnm to 60 Ωnm. The tier thickness was assumed to be 1 μm, and the intertier via diameter was assumed to be 50 nm. In this case we assumed that both signal and power vias would have the same dimensions, as this represents the simplest fabrication scenario, and that a maximum of 1% of the total die area could be allocated to intertier vias, to minimize the cannibalization of active silicon.

As can be seen in Fig. 3.20, power via requirements increase steadily as a function of via resistivity for all configurations considered. Switching from copper to tungsten vias has roughly the same impact to the power via requirements as partitioning the design into an additional tier. Importantly, even the 4-tier design using 60 Ωnm vias requires less than 1% of the total via area for power delivery, suggesting that monolithic 3DICs will not require careful via management.

3.7 Conclusions

A compact simulation tool for 2D and 3D IC pathfinding was developed and used to examine the impacts of advanced technologies on system performance. The tool incorporates models for signal delivery, power supply noise, and thermal performance in 2D and 3D ICs, and was validated against wire pitch and power consumption data for recent commercial microprocessors. The simulation tool is available upon request.

The simulation results suggest that high performance 3DICs may require large numbers

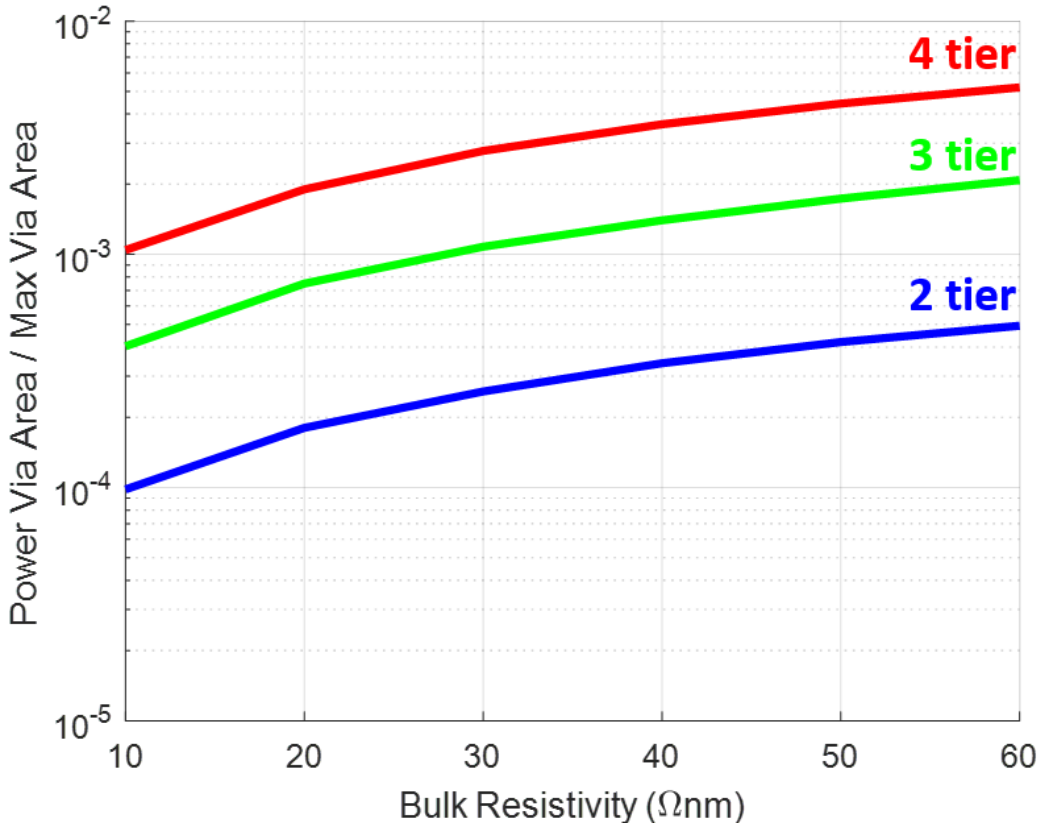

Figure 3.20: Fraction of maximum via area occupied by intertier power vias in a 32nm monolithic 3D Sandy Bridge CPU core, assuming that a maximum of 1% of the total die area is allocated for intertier signal and power vias.

of power delivery TSVs, due to the TSV parasitics introduced into the power delivery network, as well as the increased power density of the 3D cores. Die thinning and low-k dielectrics are expected to be effective tools for reducing the power consumption and power TSV requirements of high performance 3DICs. The results suggest that 3DICs should exhibit greater energy efficiency than their 2D counterparts, though thermally-limited 3D designs may require more aggressive cooling solutions.

Additionally, we have investigated the impact of wire resistivity on signal routing and power delivery in monolithic 3DICs by simulating the number of metal levels and number of power delivery vias required by a hypothetical monolithic 3D Sandy Bridge CPU core. The use of high-resistivity metals significantly increased the number of metal lev-

els required for signal routing, especially at advanced process nodes. While the number of power delivery vias required for stable operation increases steadily with via resistivity, even highly-resistive metals can be used without consuming undue silicon area for power delivery.

CHAPTER 4
INVESTIGATING 3DIC THERMAL IMPLICATIONS WITH A TWO-TIER FUNCTIONAL THERMAL TESTBED

4.1 Introduction

Three dimensional integrated circuits (3DICs) are becoming an increasingly attractive option for system interconnection due to their potential to unlock ultra-high bandwidth [83–85]. Applications which require high bandwidth, such as machine learning [86], stand to benefit significantly from the heterogeneous 3D integration of high performance computing elements coupled with large quantities of memory. Thermal constraints complicate the design of such 3D systems, however, as the areal power density of a 3DIC can be much higher than the power density of the equivalent 2D system, making heat removal and thermal coupling significant challenges in 3D systems [82,87,88]. In order to begin to quantify the impact of thermal coupling on the performance of functional systems, we have developed a two-tier air-cooled 3D thermal testbed, shown in Fig. 4.1, composed of an NVIDIA Tesla K40 GPU [89] and a top die with resistive heaters, which can emulate a variety of different workloads.

4.2 Design and assembly

The heater die (shown in Fig. 4.2) is composed of four serpentine platinum traces, each connected to two gold pads. Each quadrant of the die can be controlled and sensed separately, enabling the use of nonuniform power maps. The heater coils were fabricated via a lift-off process and are composed of a $0.2\mu m$-thick layer of platinum. The heater die was mounted back-to-back (B2B) with the GPU die. Since the heater die was stacked face up, the heater coils were covered with a layer of Kapton tape to electrically isolate them

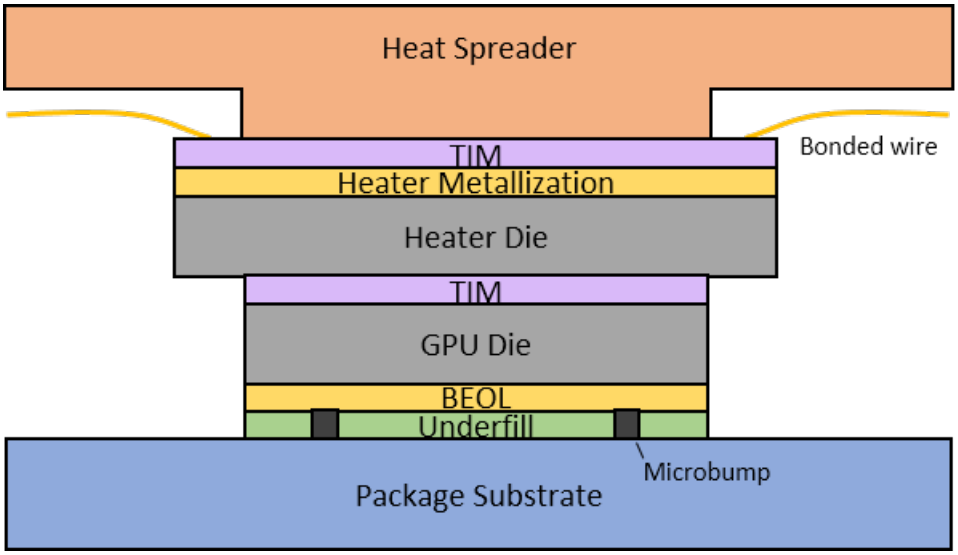

Figure 4.1: Schematic of the air-cooled 3DIC thermal testbed.

from the copper heat spreader, which slightly increased the thermal resistance of the stack. While a face-to-back (F2B) configuration would better reflect a typical 3D stacking scenario, the lack of clearance between the thermal die and the K40 board necessitated the B2B stacking approach. The heat spreader and heat sink were removed from the board, and the copper portion of the heat sink was milled down by approximately 0.5 mm to accommodate the heater die, and an additional 0.5 mm near the edges to accommodate the control/signal wires for the heaters, as can be seen in Figs. 4.3 to 4.5. Additionally, a portion of the aluminum board chassis was thinned down to allow the heater wires to exit the region immediately surrounding the GPU.

To improve the thermal contact between the GPU, the heater, and the copper heat spreader, we used a thin layer of Arctic Silver 5 thermal interface material (TIM) at each interface. The resistance of each heater was measured over a range of temperatures in a Baxter Scientific Products DP-22 oven. As can be seen in Fig. 4.6, the heaters show a linear relationship between resistance and temperature. During operation, the heaters are driven at a constant power, and their resistances are inferred from the driving voltages and currents. In order to validate the use of B2B stacking in the testbed, we simulated the thermal performance of a two-tier 3D stack, with a power density of 100 W/cm^2 dissipated on

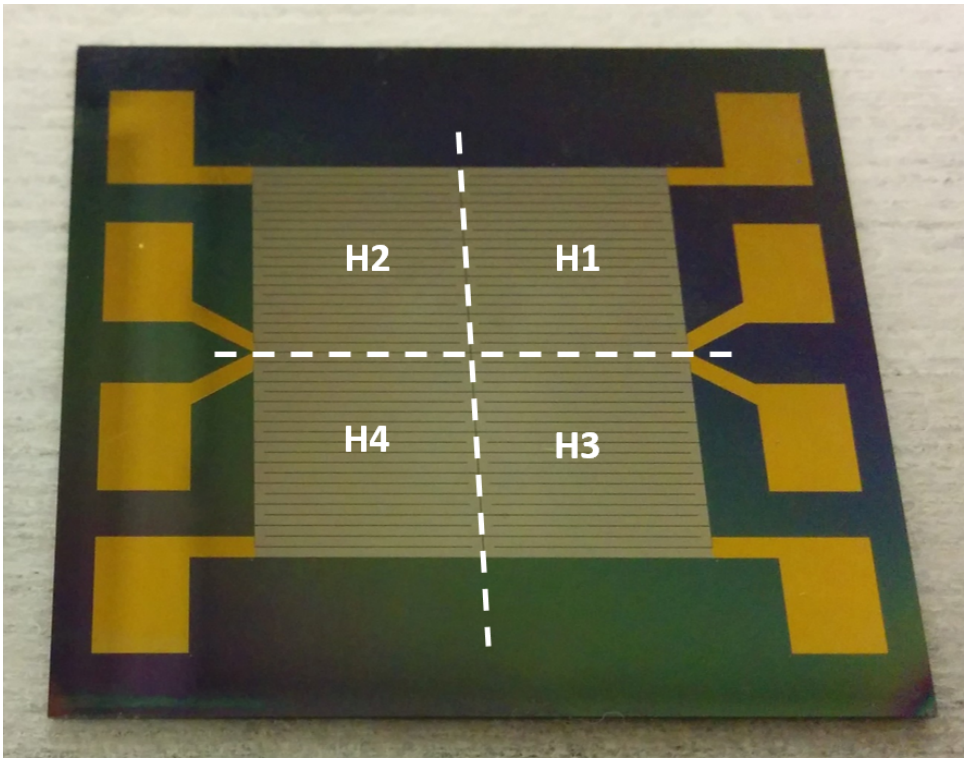

Figure 4.2: Heater/thermometer die used to emulate a second tier logic or memory device. Each quadrant can be independently controlled.

the bottom tier, and 10 W/cm^2 dissipated on the top tier. As can be seen in Fig. 4.7, the thermal difference between the two scenarios is very small, since the thermal conductivity of silicon is high. These results suggest that data from the B2B thermal testbed can be used to make reasonable inferences about F2B systems.

4.3 Results and discussion

We evaluated the two-tier thermal testbed with four deep neural networks (DNNs), detailed in Table 4.1, which represent the state-of-the-art in artificial intelligence, recognition, and classification. Each DNN benchmark was run 25 times back to back to allow the GPU time to reach a steady-state condition under load, and the average GPU temperature, power consumption, and computation time were recorded. After each set of 25 runs, the system was kept idle for 5 minutes to allow time for the GPU to return to a baseline temperature after which the next benchmark was run 25 times back to back. This process was repeated

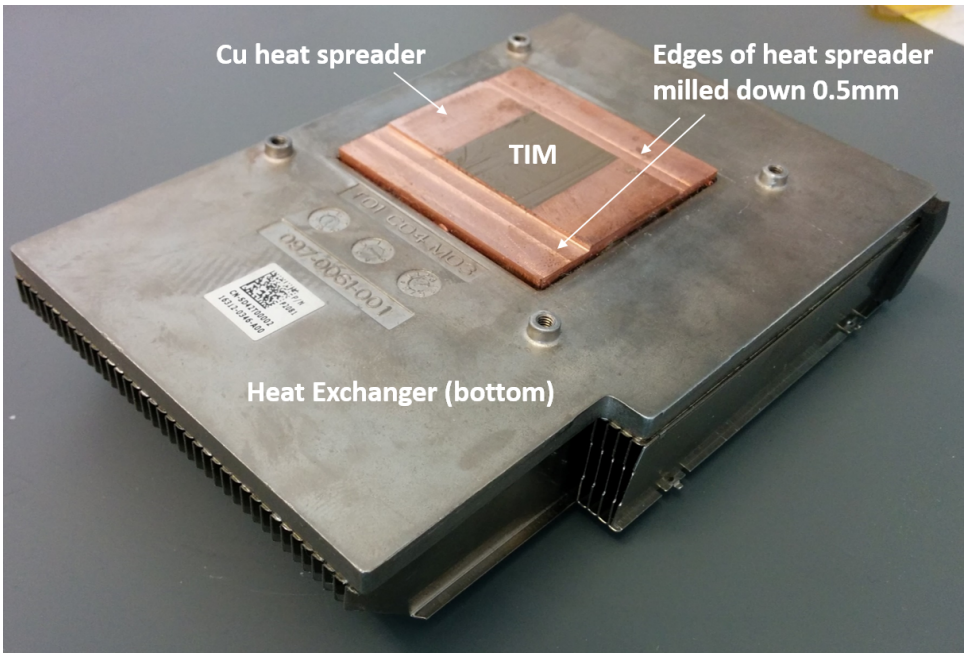

Figure 4.3: NVIDIA Tesla K40 heat spreader with edges milled to accommodate heater/thermometer wires, and with thermal interface material applied to ensure efficient heat transfer.

for each benchmark with top-die power dissipations of 0 W, 16 W, 24 W, 30 W, and 40 W. Each time the top die power dissipation was changed, the GPU was kept idle for 5 minutes to reach a steady-state temperature. After running the top die at 24 W, the resistance of heaters 2 and 4 dropped to zero, due to a short caused by a small gap in the electrical isolation. To approximately compensate for the loss of heaters 2 and 4, heaters 1 and 3 were run at twice the power density for the 30 W and 40 W runs. The resistance of each heater on the top die was sampled every 3.3 seconds to determine the dynamic top-die temperatures.

In Fig. 4.8, the temperature of each heater on the top die is shown as a function of time for one complete test run encompassing all four benchmarks, with the top-die power dissipation set to 0 W. The beginning and end of each workload are clearly visible as the die temperature rapidly increases to a steady state under load, then decays to its idle steady state. The variation in heater temperature is attributed to imperfect contact between the heat spreader and the heater/GPU stack.

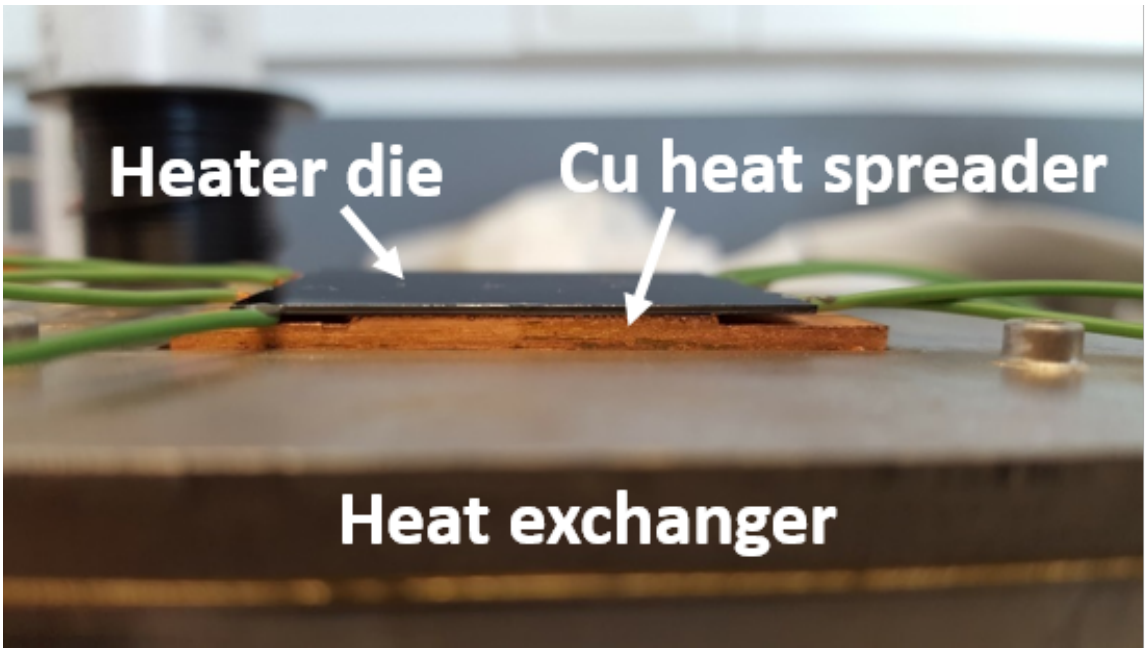

Figure 4.4: Profile view of the modified heat spreader with top-tier heater/thermometer die attached. A small portion of the copper heat spreader was milled down to make room for the heater wires.

Table 4.1: Benchmark overview

Network	Dataset	Domain	Model Size	MACCs
LeNet	MNIST	Digit recognition	0.8 MB	2M
AlexNet	ImageNet	Detect/classify	116.3 MB	736M
Overfeat	ImageNet	Detect/classify	278.3 MB	2,797M
VGG-16	ImageNet	Detect/classify	323.87 MB	16,361M

In Fig. 4.9, the average GPU temperature during each workload is shown as a function of top-die power dissipation. As the top-die power dissipation increases, the average GPU temperature measured during each workload tends to increase, and the temperature during the AlexNet, Overfeat, and VGG-16 workloads exceeds 85°C at a top-die power dissipation of approximately 30 W. The GPU remains relatively cool during the LeNet workload, as it has a much smaller computational footprint than the others, and does not fully stress the GPU. As shown in Fig. 4.10, the time required for each workload remains relatively flat until 30 W, at which point the larger workloads begin to overpower the heat sink, and the GPU begins limiting its performance to avoid exceeding its thermal limits.

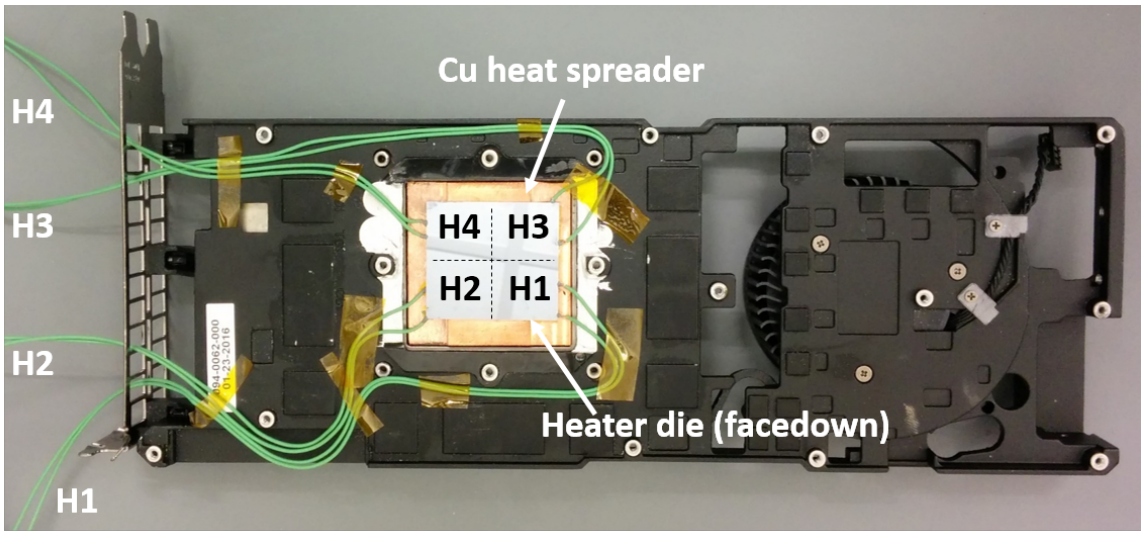

Figure 4.5: NVIDIA Tesla K40 heat spreader with heater die attached. The heat spreader sits within a cutout in the aluminum chassis (black).

In Fig. 4.11, the average GPU power consumption is shown for each workload as a function of top-die power dissipation. GPU power increases for each of the large workloads (AlexNet, Overfeat, and VGG-16) up to a top-die power dissipation of approximately 24 W, due in part to increased transistor leakage. Above 24 W, the GPU power consumption drops sharply for each of the large workloads. As can be seen in Fig. 4.10, the GPU appears to limit its performance in order to remain within its thermal envelope, as the average computation time for each benchmark stays roughly constant until the average GPU temperature approaches 90°C, at which point the computation time dramatically increases. While the average GPU power decreases at high top-die power dissipations, the computation energy increases significantly, due to the increase in computation time, as seen in Fig. 4.12.

The temperature measured at heater 1 on the top die for each experimental condition is shown in Fig. 4.13. During the 30 W and 40 W tests the maximum temperature of heater 1 increases to 110°C. This high temperature can be attributed to the poor thermal transfer between heater 1 and the heat sink, as shown in Fig. 4.8, and to the asymmetric power maps used for the 30 W and 40 W tests, during which only heaters 1 and 3 were used due to the failure of heaters 2 and 4.

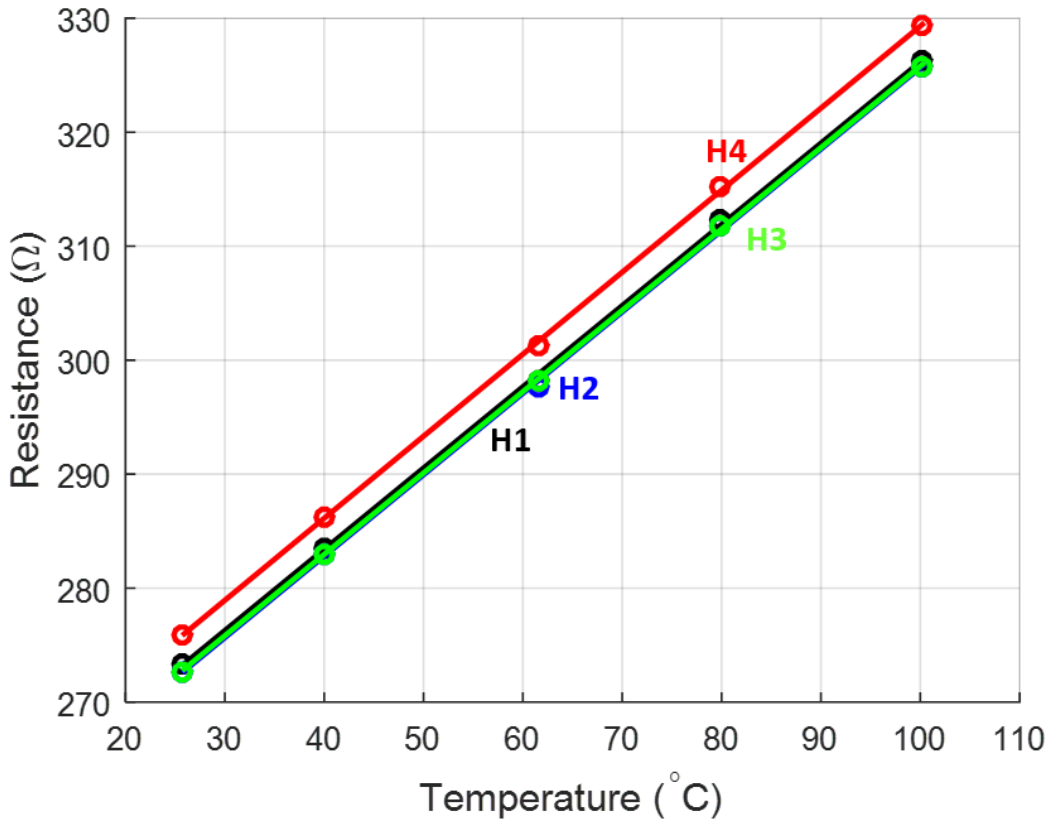

Figure 4.6: Calibration measurements (circles) and best fits (lines) for the heater/thermometer structures on the top-tier heater die.

4.4 Conclusions

GPU-accelerated deep neural networks could benefit greatly from the high bandwidth and low latency enabled by 3D integration, as DNNs require large sets of model parameters to be fed to the cores of the GPU, but thermal limits could offset the benefits of such integration. In order to explore the impact of 3D stacking on DNN computational performance, we have developed and characterized an air-cooled thermal testbed for the investigation of the impact of thermal crosstalk and cooling limits on the performance of high performance 3DICs. The thermal testbed was used to emulate a two tier GPU-based 3D stack with a thermal die on the top tier to emulate stacked memory or logic. The GPU operating temperature increased steadily with top-die power dissipation, and once the average GPU temperature approached 90°C (at 30 W top-die power dissipation), the GPU appears to

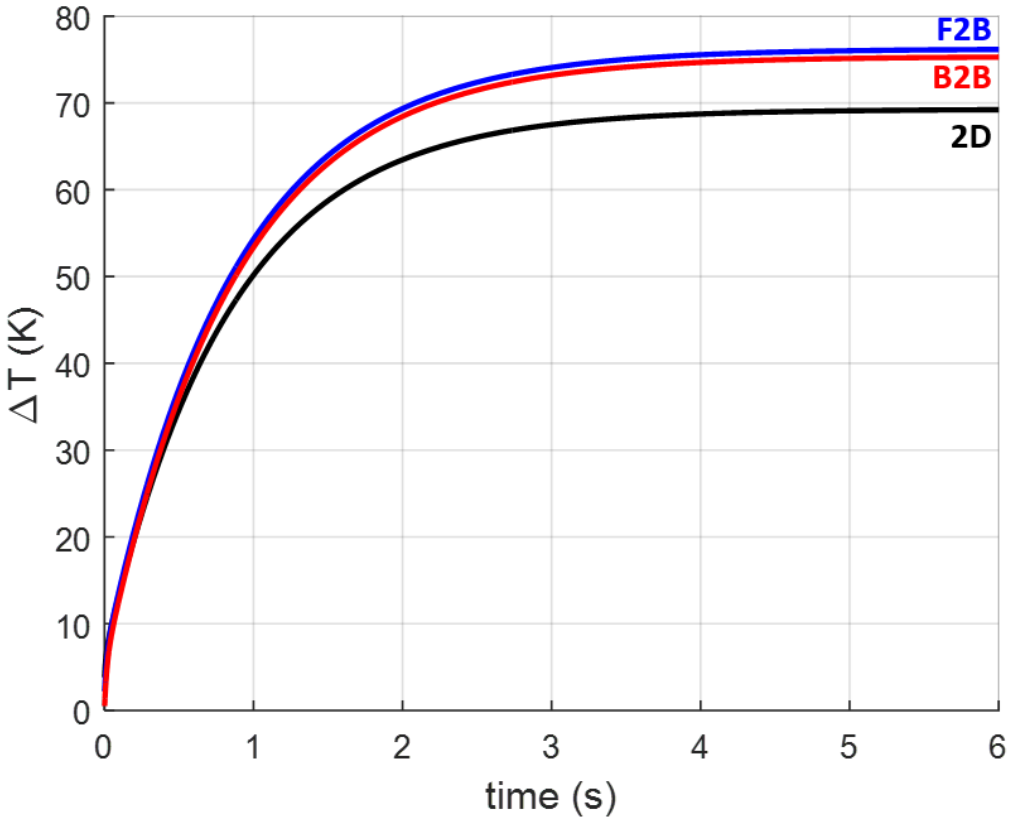

Figure 4.7: Simulated thermal impact of face to back (F2B) vs. back to back (B2B) bonding of a two-tier 3D stack. The B2B thermal response very closely mirrors the F2B response, justifying the B2B approach used in the testbed.

limit its performance to avoid exceeding its thermal limits. In the worst case, we observed a 2.6X increase in computation time, and a 2.2X increase in computation energy, and we expect higher top-tier power dissipations to yield worse performance/efficiency degradation. These results suggest that aggressive cooling techniques may have a significant impact on the viability of high performance 3DICs, especially for DNN workloads.

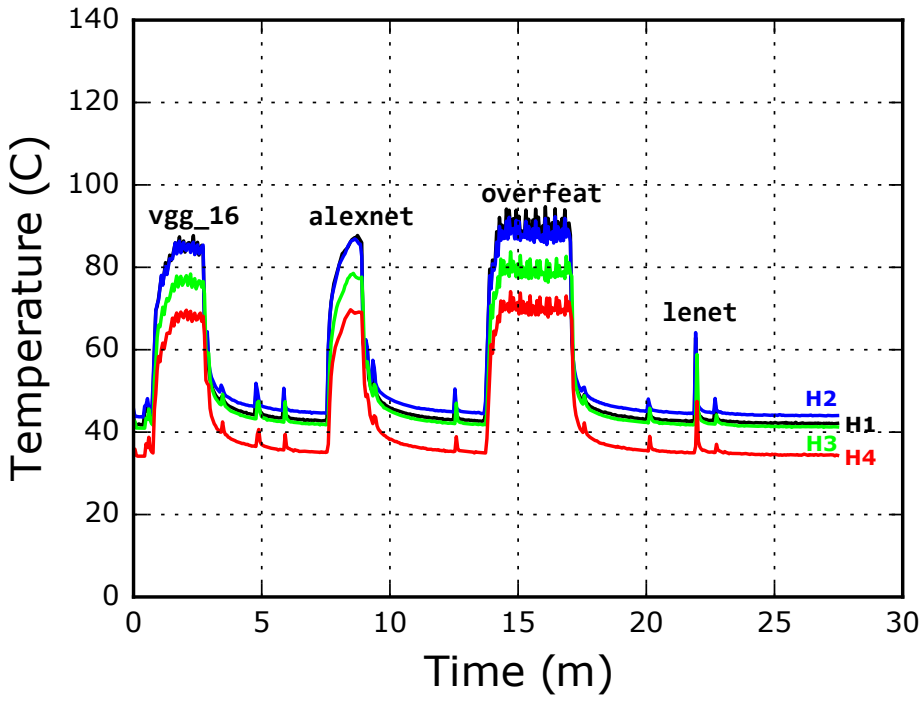

Figure 4.8: Measured temperature of the top-tier die with the GPU running various machine learning workloads, and with the top die dissipating 0 W.

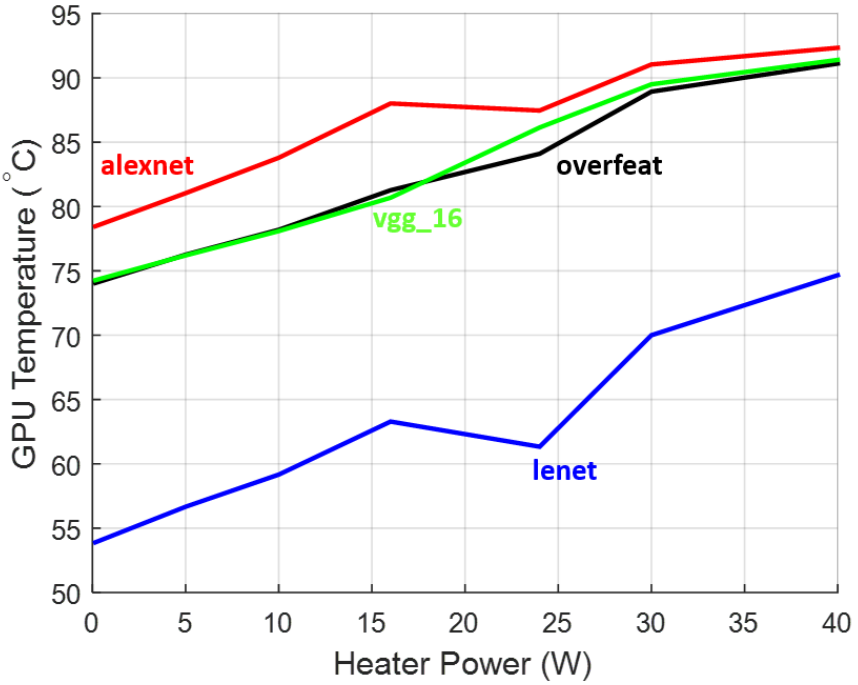

Figure 4.9: Impact on GPU temperature as top-die power is increased.

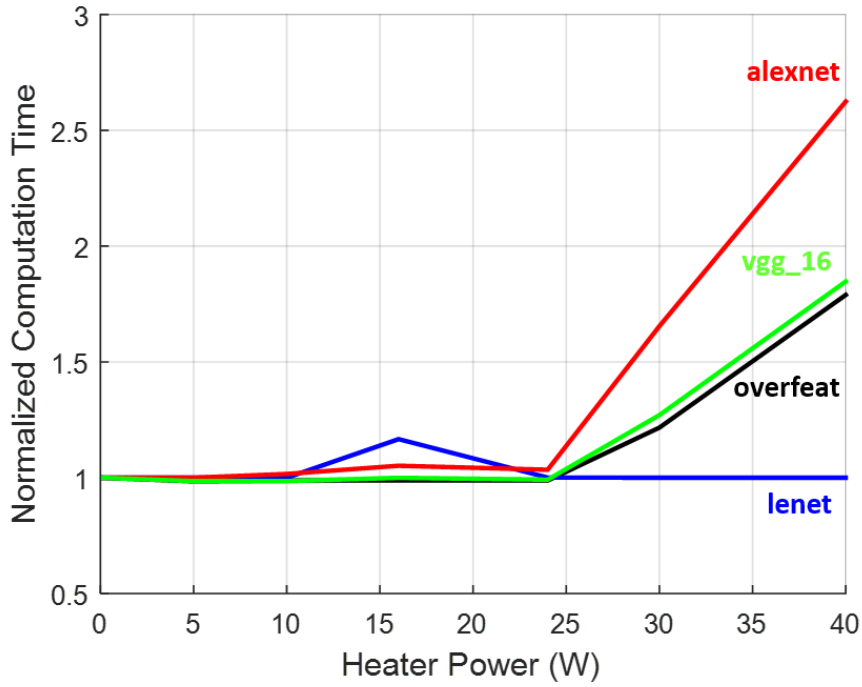

Figure 4.10: Impact of heater power on GPU computation time. Each curve is normalized to its value at a top-die power dissipation of 0 W.

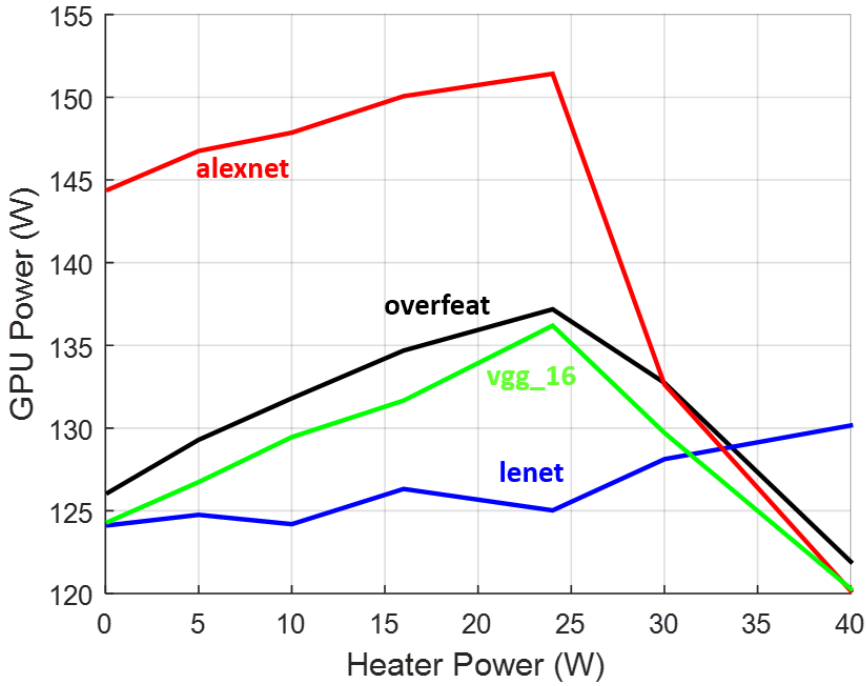

Figure 4.11: Impact on GPU power dissipation as top-die power is increased.

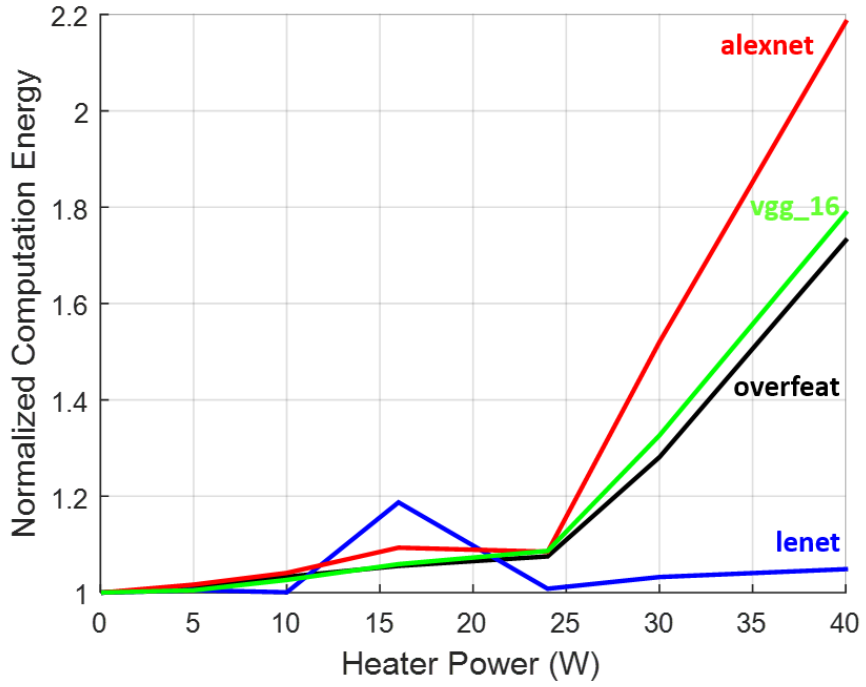

Figure 4.12: Impact of heater power on GPU computation energy. Each curve is normalized to its value at a top-die power dissipation of 0 W.

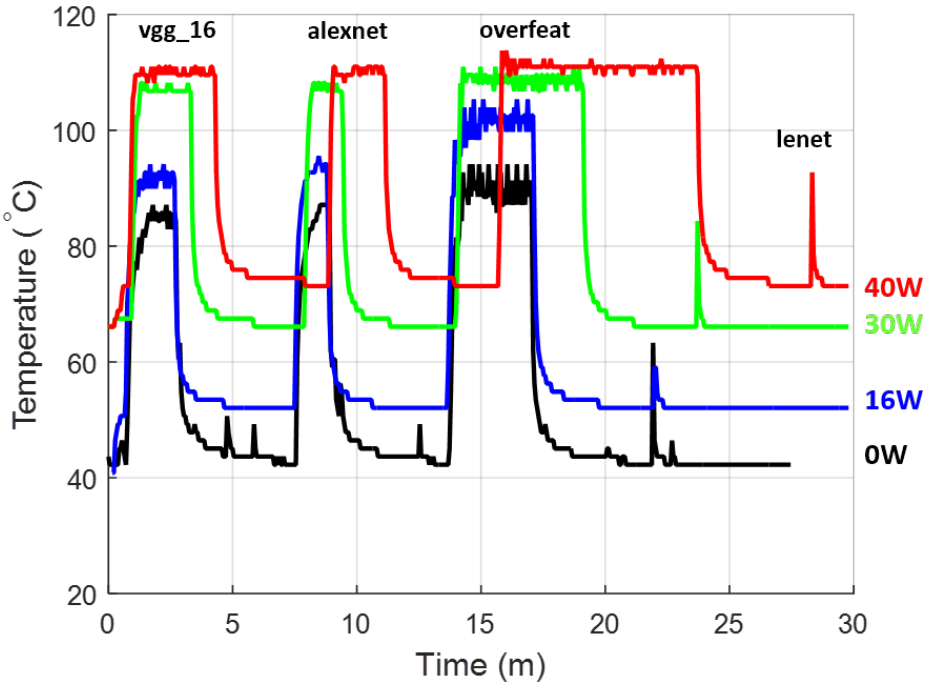

Figure 4.13: Measured temperature of heater 1 during the benchmark suite, for a range of top-tier power dissipations.

CHAPTER 5
STOCHASTIC WIRE LENGTH MODELING IN HIGH-DIMENSIONAL SYSTEMS

5.1 Introduction

Much work has been done to develop modeling capabilities for 2D and 3D ICs, in order to better predict the performance and energy requirements for these systems. To date, no work has been done in the literature on networks with higher-degree dimensionality, for the simple reason that we live in a three-dimensional universe. Network topologies, however, can take any dimensionality, and there are many well-known examples of systems with high-dimensional network topology. If we look at the simpler case of a system designed natively as a 3DIC, and then reimplemented as a monolithic 2DIC, we can clearly expect a degradation in performance due to the need to map an inherently 3D system onto a 2D plane. The same is true of systems with higher-dimensional connectivity – even implementing them as 3D ICs will still incur a performance/energy penalty with respect to the "native" space in which the system belongs. As of yet, we have no method to quantify this effect, but we can extend the models developed previously to treat the case of higher dimensional systems.

Obviously, we cannot truly implement a four- or five-dimensional system in three-dimensional space. We can, however, implement networks with higher-dimensional topologies in lower-dimensional spaces, albeit with reductions in implementation efficiency, as illustrated in Fig. 5.1. As these network topologies arise frequently in the design of high performance computing and communications networks, extensive work has been done on characterizing the performance of high-dimensional network topologies [90–97]. Additionally, high-dimensional interconnect topologies have been observed in natural intercon-

nect systems, such as the human brain [98]. Extending existing wire length models to higher-dimensional spaces, then, provides an avenue towards understanding the potential performance tradeoffs involved in the implementation of densely-interconnected systems.

As we are extending existing stochastic wirelength methodologies, we begin with some of their key assumptions. Chiefly, we will assume that all systems are composed of regular periodic arrays of computing elements, and that interconnections between them are routed in a manhattan grid, in which all interconections are broken into purely lateral, x-directed and y-directed components. Similar to how, when walking the streets of a large city, a pedestrian cannot walk directly to their destination, but must rather follow the natural grid formed by the city blocks, we assume that interconnects must likewise be routed in discrete, purely x-directed or y-directed components. While this assumption may hold well for ICs, in other kinds of computing systems (such as biological neural networks) interconnects are free to be routed in arbitrary directions. In systems such as these, our methodology will tend to overpredict the length of all wires in the system, as the true wirelength will be closer to the hypotenuse of the triangle formed by the total in-plane x-directed and y-directed wire length. Despite this discrepancy, these methods provide a method to rapidly investigate the behavior of the overall wire length trends in such systems in a manner which has not yet been possible. Relaxation of some or all of these assumptions is possible, but introduces significant additional complexity to the derivation and the result.

5.2 Derivation

In this work we will use a similar methodology as that laid out in [31] and extended in Chapter 2. We will continue to treat the system as a homogeneous block of randomly-placed logic, and for simplicity we will consider the case of a regular hypercube. This assumption can easily be relaxed, but it leads to a significantly more unwieldy result, without providing significant gains in the utility and clarity of the results. First, we must recast the original problem in a slightly different way, in order to simplify the approach.

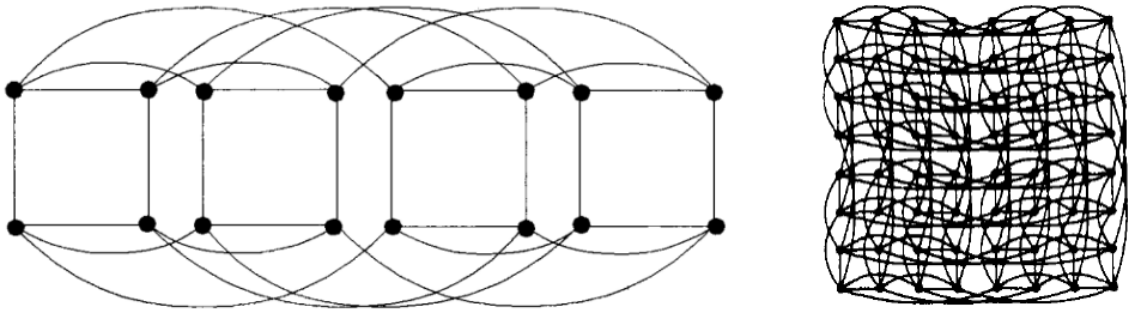

Figure 5.1: Four-dimensional (left, [99]) and six-dimensional (right, [90]) cubic networks, each with two nodes in each dimension, implemented in a two-dimensional plane. As the number of dimensions of the system increases, the details of the interconnection networks between nodes become significantly more complex.

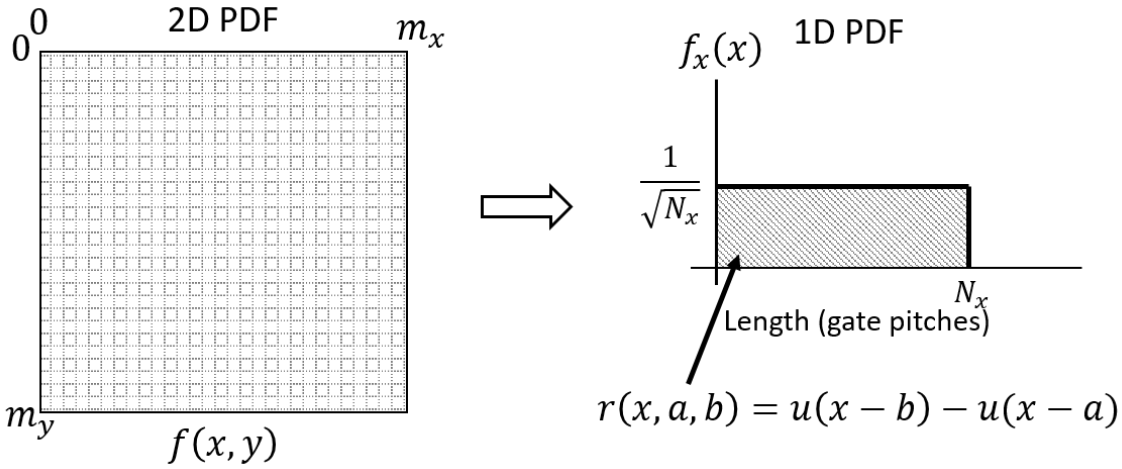

Figure 5.2: (left) Representation of a 2D system with uniformly-distributed homogeneous logic. The 2D representation can be broken into two independent 1D functions (right).

5.2.1 The gate pair function

$$f_l(l) = \sum_{l_x=0}^{l} \sum_{x_1=0}^{m} \sum_{y_1=0}^{m} r(x_1, 0, m_x) r(x_2, 0, m_x) r(y_1, 0, m_y) r(y_2, 0, m_y) \tag{5.1}$$

$$f_l(l) = \sum_{l_x=0}^{l} \sum_{x_1=0}^{m} \sum_{y_1=0}^{m} r(x_1, 0, m_x) r(x_1 - l_x, 0, m_x) r(y_1, 0, m_y) r(y_1 - l_y, 0, m_y) \tag{5.2}$$

$$f_l(l) = \sum_{l_x=0}^{l} \sum_{x_1=0}^{m} \sum_{y_1=0}^{m} r(x_1, 0, m_x) r(x_1 - l_x, 0, m_x) \times r(y_1, 0, m_y) r(y_1 - (l - l_x), 0, m_y) \tag{5.3}$$

The operand of the summation can take only two values: if the points described by x_1, y_1, x_2, and y_2 lie within the boundaries of the system, then the operand becomes 1, otherwise it is 0. Essentially we are simply looking at all possible combinations of x and y coordinates for two gates, and adding up all the possible combinations that are valid. Additionally, by starting the summation at one corner of the chip and rastering across the rest of the chip, we can avoid double-counting gate pairs. Additionally, we must note that the l_x summation must be handled differently when $l < m_x$ and when $l > m_x$. By keeping these points in mind we can dramatically simplify Eq. (5.3) by focusing on the limits of the summations.

For the case of $0 < l <= m$:

$$f_l(l) = \sum_{l_x=0}^{l} \sum_{x_1=l_x}^{m} \sum_{y_1=l_y}^{m} r(x_1, 0, m_x) r(x_1 - l_x, 0, m_x) r(y_1, 0, m_y) r(y_1 - l_y, 0, m_y) \tag{5.4}$$

$$f_l(l) = \sum_{l_x=0}^{l} \sum_{x_1=l_x}^{m} \sum_{y_1=l_y}^{m} 1 \tag{5.5}$$

$$f_l(l) = \sum_{l_x=0}^{l} \sum_{x_1=l_x}^{m} (n - l_y) \tag{5.6}$$

$$f_l(l) = \sum_{l_x=0}^{l} (n - l_x)(n - l_y) \tag{5.7}$$

$$f_l(l) = \sum_{l_x=0}^{l} (n - l_x)(n - l + l_x) \tag{5.8}$$

$$f_l(l) = \frac{1}{6}(l + 1)(6n^2 - 1 - 6ln + l^2) \tag{5.9}$$

And similarly, for the case of $m < l <= 2m$

$$f_l(l) = \sum_{l_x=l-m}^{m} \sum_{x_1=l_x}^{m} \sum_{y_1=l_y}^{m} r(x_1, 0, m_x) r(x_1 - l_x, 0, m_x) r(y_1, 0, m_y) r(y_1 - l_y, 0, m_y) \tag{5.10}$$

$$f_l(l) = \sum_{l_x=l-m}^{m} \sum_{x_1=l_x}^{m} \sum_{y_1=l_y}^{m} 1 \tag{5.11}$$

$$f_l(l) = \sum_{l_x=l-m}^{m} \sum_{x_1=l_x}^{m} (n - l_y) \tag{5.12}$$

$$f_l(l) = \sum_{l_x=l-m}^{m} (n - l_x)(n - l + l_x) \tag{5.13}$$

$$f_l(l) = \frac{1}{6}(2n - l + 1)(l - 2n)(1 + l - 2n) \tag{5.14}$$

So then our 2D gate pair function becomes:

$$f_l(l) = \begin{cases} \frac{1}{6}(l+1)(6n^2 - 1 - 6ln + l^2) & 0 < l \leq m \\ \frac{1}{6}(2n - l + 1)(l - 2n)(1 + l - 2n) & m < l \leq 2m \end{cases} \tag{5.15}$$

To extend to higher dimensions we can simply define a new variable, $l_{2D} = l_x + l_y$, and modify our summation accordingly.

$$\begin{aligned} f_l(l) = \sum_{l_{2D}=0}^{l} \sum_{l_x=0}^{l_{2D}} \sum_{x_1=l_x}^{m} \sum_{y_1=l_y}^{m} \sum_{z_1=l_z}^{m} & r(x_1, 0, m_x) r(x_1 - l_x, 0, m_x) \\ & \times r(y_1, 0, m_y) r(y_1 - l_y, 0, m_y) \\ & \times r(z_1, 0, m_z) r(z_1 - l_z, 0, m_z) \end{aligned} \tag{5.16}$$

$$f_l(l) = \sum_{l_{2D}=0}^{l} \sum_{z_1=l_z}^{m} f_{2D}(l_{2D}) \tag{5.17}$$

It is critical, however, to properly handle the cases of l_{2D} and l_x, which each are affected by the particular value of l (in the same manner as in the 2D case). In general we can define the gate pair function in N dimensions recursively as a function of the gate pair function in $N - 1$ dimensions. We must first make some simple observations about $f_N(l)$, the gate pair function in N dimensions:

1. $f_N(l)$ will be a piecewise function with N nontrivial regions.

2. There will be N position variables $(x_1, x_2, ..., x_N)$ to consider

3. There will be $N-1$ length variables $(l_1, l_2, ...l_{N-1})$ appearing as summation variables

4. l_N will apear in the summation only in the limits of the l_{N-1} summation

5. The summations in Eq. (5.17) will span at most two regions of f_{N-1}

If we define the variable r as the region number (i.e. which section of the domain of f_N we are interested in), then we can write the gate pair function in any dimension as:

$$f_N(l_N \in [(r-1)m, rm]) = ...$$

$$\sum_{\substack{0 \\ (l_{N-1})}}^{l_N} \sum_{\substack{l_N - l_{N-1} \\ (x_N)}}^{m} f_{N-1}(l_{N-1} \in [0, m]) \qquad r = 1 \tag{5.18}$$

$$\sum_{\substack{l_N - m \\ (l_{N-1})}}^{(r-1)m} \sum_{\substack{l_N - l_{N-1} \\ (x_N)}}^{(r-1)m} f_{N-1}(l_{N-1} \in [(r-2)m, (r-1)m])$$

$$+ \sum_{\substack{(r-1)m+1 \\ (l_{N-1})}}^{l_N} \sum_{\substack{l_N - l_{N-1} \\ (x_N)}}^{(r-1)m} f_{N-1}(l_{N-1} \in [(r-1)m, rm]) \qquad r \in [2, N-1] \tag{5.19}$$

$$\sum_{\substack{(l_N - m \\ (l_{N-1})}}^{rm} \sum_{\substack{l_N - l_{N-1} \\ (x_N)}}^{(r-1)m} f_{N-1}(l_{N-1} \in [(r-2)m, (r-1)m]) \qquad r = N \tag{5.20}$$

5.2.2 The connection function

Here we will again use the method of [31] as a guide, and again we will rework the two-dimensional problem with a slightly different methodology in order to make it more amenable to extension to higher dimensions. The crux of the problem is the derivation of the nonstarting gate function, which counts the number of gates which cannot possibly

participate in any connections of a particular length l. In order to determine the number of nonstarting gates, we can draw lines, A, and B, described by the following equations:

$$y_A = (m - x_A) + (m - l') \tag{5.21}$$

$$y_B = x_B + (m - l') \tag{5.22}$$

where $l' = l - 1$.

By summing the gates contained below *both* y_A and y_B we can determine the number of nonstarting gates. In two dimensions, as l increases, we must deal with four distinct situations, as shown in Fig. 5.3. for $l < m/2$ we simply have the number of nonstarting gates as l. Once l exceeds $m/2$, we must sum up $x_A - x_B$ in the triangular central region, where $x_A > x_B$. Once l exceeds m we must keep track of the points at which lines A and B "clip" the sides of the chip – these points will separate a simple rectangular region from the triangular region, which must be summed as before. Finally, once $l > 3m/2$ we will also need to account for the fact that the top of the triangular region will also be clipped off.

Given this framework, the derivation of the nonstarting function is straightforward, if somewhat tedious. The key observation to note is that the summation can simply be computed as:

$$\sum_{\substack{y_{min} \\ (y)}}^{y_{clip}} (x_A(y) - x_B(y) + 1) + \sum_{y_{clip}+1}^{y_{max}} m \tag{5.23}$$

where $x_A(y)$ and $x_B(y)$ can be easily determined from Eq. (5.22), and where y_{clip} represents the vertical coordinate of the line separating the triangular and rectangular regions (where the lines described by $y_A(x)$ and $y_B(x)$ are "clipped" by the edges of the chip). Clearly then, the limits y_{min} and y_{clip} are the key quantities to be determined. For our purposes we will consider the origin to be the *top left* corner of the chip, therefore y_{min} corresponds to the top of the triangular region, and can be found by simply setting $x_A = x_B$

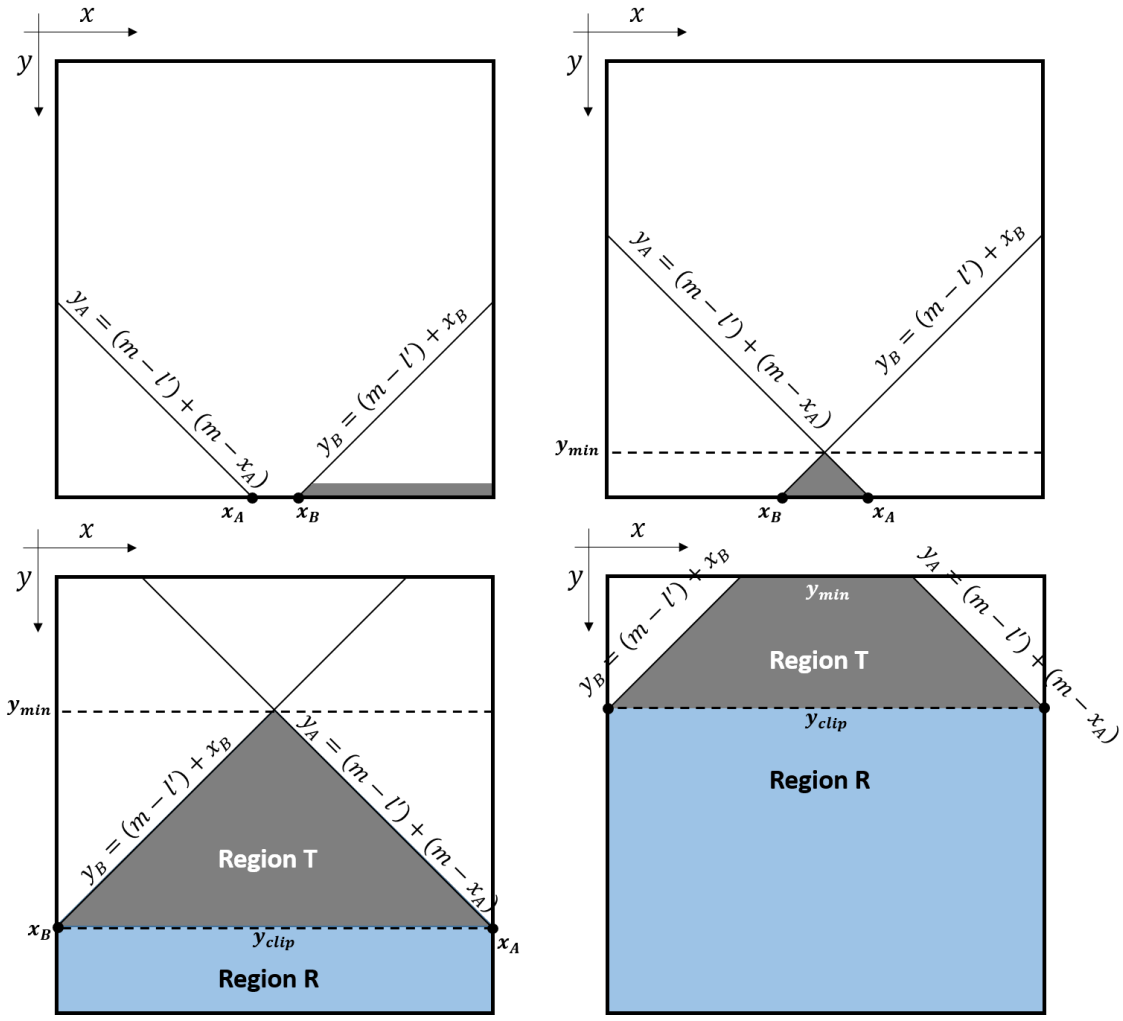

Figure 5.3: The four possible domains encountered during the determination of the number of nonstarting gates. As l increases, the number of nonstarting gates increases in a predictable, if unwieldy, manner.

and solving for y (while also noting that the minimum value of y_{min} is 0).

$$y_{min} = \begin{cases} \frac{3}{2}m - l' & l' \leq \frac{3}{2}m \\ 0 & l' > \frac{3}{2}m \end{cases} \tag{5.24}$$

To find y_{clip} we can simply set x to 0 in Eq. (5.22) and solve for either y_A or y_B. This gives us:

$$y_{clip} = 2m - l' \tag{5.25}$$

So then we end up with

$$G(l) = \begin{cases} l & l \in [0, m/2] \\ \sum\limits_{\substack{y_{min} \\ (y)}}^{y_{clip}} (x_A(y) - x_B(y) + 1) & l \in [m/2, m] \\ \sum\limits_{\substack{y_{min} \\ (y)}}^{y_{clip}} (x_A(y) - x_B(y) + 1) + \sum\limits_{y_{clip}+1}^{y_{max}} m & l \in [m, 3m/2] \\ \sum\limits_{\substack{y_{min} \\ (y)}}^{y_{clip}} (x_A(y) - x_B(y) + 1) + \sum\limits_{y_{clip}+1}^{y_{max}} m & l \in [3m/2, 2m] \end{cases} \tag{5.26}$$

$$G(l) = \begin{cases} l & l \in [0, m/2] \\ \frac{1}{4}(3 + 2l - 2 - n)^2 + n - l & l \in [m/2, m] \\ \frac{1}{4}(1 + n)^2 + n(l - n) & l \in [m, 3m/2] \\ (2n - l)(1 + l - n) + n(l - n) & l \in [3m/2, 2m] \end{cases} \tag{5.27}$$

5.2.3 The connection function in higher dimensions

We can make use of the 2D connection function to simplify the derivation of the connection function in higher dimensions. Consider, for instance, the connection problem in three dimensions. In this case we can break the three-dimensional connection problem into a series of two-dimensional problems, as shown in Fig. 5.4.

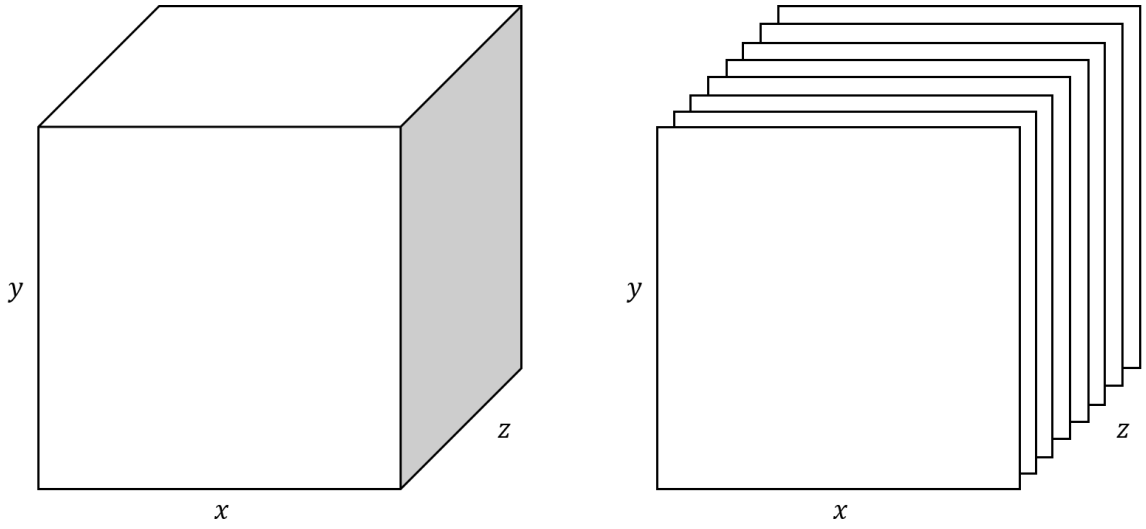

Figure 5.4: The connection problem in higher dimensions (left) can be broken into multiple lower-dimensional problems (right).

In order to derive the three-dimensional connection function we can first note that

$$l_{3D} = l_{2D} + l_z \tag{5.28}$$

where z and $l_z = z_1 - z_2$ are the position and length variables in the third dimension, respectively, and $l_{2D} = l_x + l_y$, as before. We know that $z \in [0, m]$, and that $l_z \in [0, \min(l_{3D}, m)]$. For each value of z we essentially have a pseudo-independent two-dimensional problem, as can be seen in Fig. 5.4b. In order to simplify the three-dimensional problem, we can make a few observations about how $N_{ns_{3D}}$ behaves as l_z varies:

1. For $l_z = 0$ we recover the original 2D problem

2. For $l_z = 1$ we recover the original 2D problem with $l_{2D} \rightarrow l_{2D} - 1$

3. For $l_z = i$ we recover the original 2D problem with $l_{2D} \rightarrow l_{2D} - i$

Based on these observations, we can postulate:

$$G_3(l_{3D}) = \sum_{\substack{0 \\ (l_z)}}^{m} G_2(l_{3D} - l_z) \tag{5.29}$$

where $l_{3D} = l_x + l_y + l_z$, and where G_2 is the two-dimensional nonstarting gate function, given by Eq. (5.27).

The same arguments can be extended to higher dimensions, and we can therefore write down a simple recursive expression for the N-dimensional nonstarting function, in terms of the (N-1)-dimensional nonstarting gate function.

$$G_d(l_d) = \sum_{\substack{0 \\ (l_{x_d})}}^{m} G_{d-1}(l_d - l_{x_d}) \tag{5.30}$$

where $l_d = \sum_{i=1}^{N} l_{x_i}$, with $x_1 = x$, $x_2 = y$, $x_3 = z$, and so on.

While Eq. (5.30) can be used to derive the functional form of the connectivity function in any dimension, further simplifications must be made to make the derivation process simpler and less unwieldy.

Note that Eq. (5.30) is an equation in $l_d - l_{x_d}$; we can clean up the equation a bit by defining a new variable of convenience, $p = l_d - l_{x_d}$. Then we have

$$G_d(p) = \sum_{\substack{l-m \\ (p)}}^{l} G_{d-1}(p) \tag{5.31}$$

First, we can note that the nonstarting gate function will ultimately reduce to a summed series of terms involving polynomials in n and powers of l. We can therefore rewrite the nonstarting gate function in terms of generic terms, as follows:

$$G_d(p) = \sum_{i=0}^{i_{\max}} a_i(n) p^i \tag{5.32}$$

Plugging Eq. (5.32) into Eq. (5.31), we get:

$$G_d(p) = \sum_{\substack{l-m \\ (p)}}^{l} \sum_{i=0}^{i_{\max}} a_i(n) p^i \tag{5.33}$$

For a particular value of i, we simply have:

$$G_d^{(i)}(p) = \sum_{\substack{l-m \\ (p)}}^{l} a_i(n) p^i \tag{5.34}$$

$$G_d^{(i)}(p) = a_i(n) \sum_{\substack{l-m \\ (p)}}^{l} p^i \tag{5.35}$$

$$G_d^{(i)}(p) = a_i(n) S_p(l_d - m, l_d) \tag{5.36}$$

In Eq. (5.36), the p^i summation has been folded into the new function $S_p(a, b)$, since it is a straightforward summation, and since no other part of the $G_d^{(i)}$ term depends upon it.

At this point we still must still account for the fact that $G_{d-1}(p)$ is a piecewise function, which will transition across multiple definition boundaries in the summation in Eq. (5.30). Each functional region in $G_{d-1}(p)$ is half the width of the relevant dimension; if we assume that the system is a symmetric hypercube in all dimensions, with side-length m, then each region will be $m/2$ units wide. In general, then, we must rewrite Eq. (5.31) as

$$G_d(p) = \sum_{\substack{l-m \\ (p)}}^{q_1} G_{d-1}(p) + \sum_{\substack{q_1+1 \\ (p)}}^{q_2} G_{d-1}(p) + \sum_{\substack{q_2+1 \\ (p)}}^{l} G_{d-1}(p) \tag{5.37}$$

where q_1 and q_2 are the boundaries between the functional domains of $G_{d-1}(p)$. For our symmetric hypercube we can define a simple function, $r(j)$, which returns the lower domain boundary, r, of region j, as:

$$r(j) = \lfloor \frac{jm}{2} \rfloor \tag{5.38}$$

We can then properly expand Eq. (5.31) as

$$
\begin{aligned}
G_d(p) &= \sum_{\substack{0 \\ (p)}}^{l} G_{d-1}(p) && j = 0 \\
G_d(p) &= \sum_{\substack{0 \\ (p)}}^{r(1)} G_{d-1}(p) + \sum_{\substack{r(1)+1 \\ (p)}}^{l} G_{d-1}(p) && j = 1 \\
G_d(p) &= \sum_{\substack{l-m \\ (p)}}^{r(j-1)} G_{d-1}(p) + \sum_{\substack{r(j-1)+1 \\ (p)}}^{r(j)} G_{d-1}(p) + \sum_{\substack{r(j)+1 \\ (p)}}^{l} G_{d-1}(p) && 2 \leq j < j_{max} - 1 \\
G_d(p) &= \sum_{\substack{l-m \\ (p)}}^{r(j-1)} G_{d-1}(p) + \sum_{\substack{r(j-1)+1 \\ (p)}}^{r(j)} G_{d-1}(p) && j = j_{max} - 1 \\
G_d(p) &= \sum_{\substack{l-m \\ (p)}}^{r(j)} G_{d-1}(p) && j = j_{max}
\end{aligned}
\tag{5.39}
$$

A Mathematica implementation of the expressions in Eq. (5.39) which can be used to iteratively derive the connection functions in arbitrarily high dimensions is presented in Chapter A.

5.3 Investigation of wirelength distributions in higher dimensions

Now that we have developed a method for modeling wire length distributions in higher dimensions, we can investigate the interplay of the physical and topological dimensionality of various systems. We will consider a large, homogeneous, and spatially-symmetric system, consisting of 10^{12} logic elements, implemented in physical spaces of 2, 3, 4, and 5 dimensions. This value for system size is chosen for the simple reason that a system with 10^{12} logic gates can be implemented as a two-dimensional array with 10^6 elements to a side, or a three-dimensional grid with 10^4 elements to a side, or a four-dimensional hypercube with 10^3 elements to a side, for ease of comparison.

First, we investigate the impact of physical topology on the overall interconnect distribution, by considering our hypothetical system with a rent exponent of 0.6. One key test

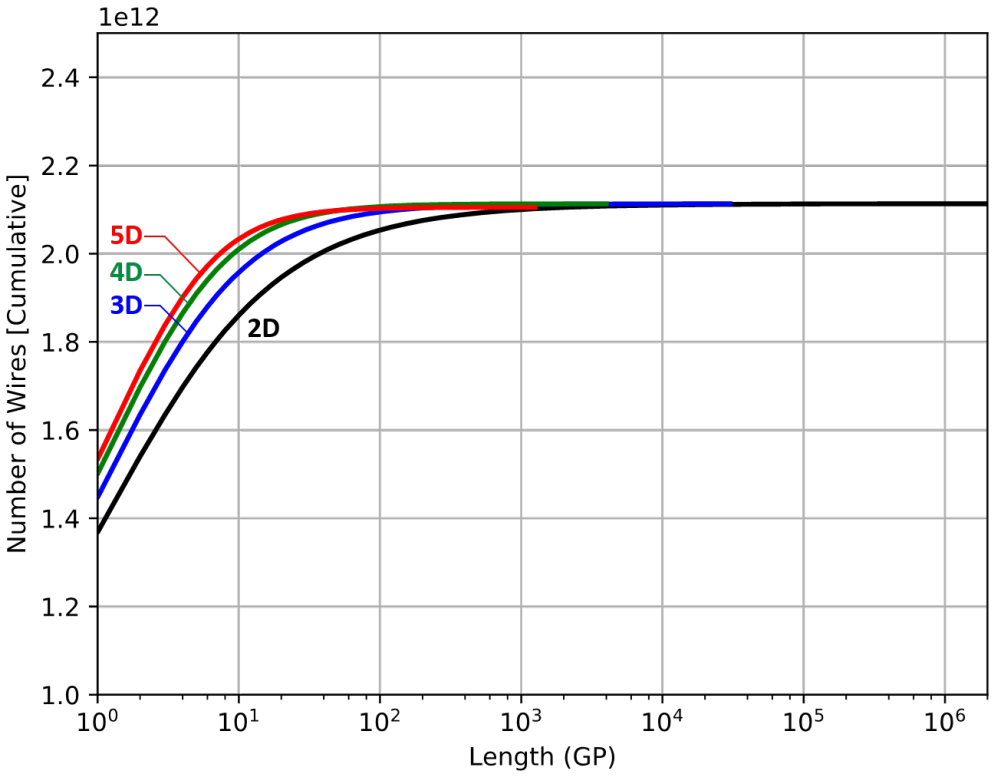

Figure 5.5: The total number of interconnects in the system is conserved, as expected.

of the new wire length distributions is the conservation of total interconnects; while the length of given interconnections may change, the number of inter-gate connections should ultimately be independent of the physical space in which the system is embedded, as it is determined by the system interconnect topology. In Fig. 5.5 we can see that the total number of inter-gate connections is indeed independent of the physical space in which the design is embedded.

As can be seen in Fig. 5.6, implementing the same system in higher-dimensional spaces significantly reduces both the maximum wirelength, as well as the expected number of long interconnects, with a maximum wire length reduction of roughly three orders of magnitude when moving a design from a two- to a five-dimensional implementation. Rather than considering the overall distribution of wire lengths, we can also consider the impact of physical topology on the total wire length in the system; in Fig. 5.7 we can see that higher-

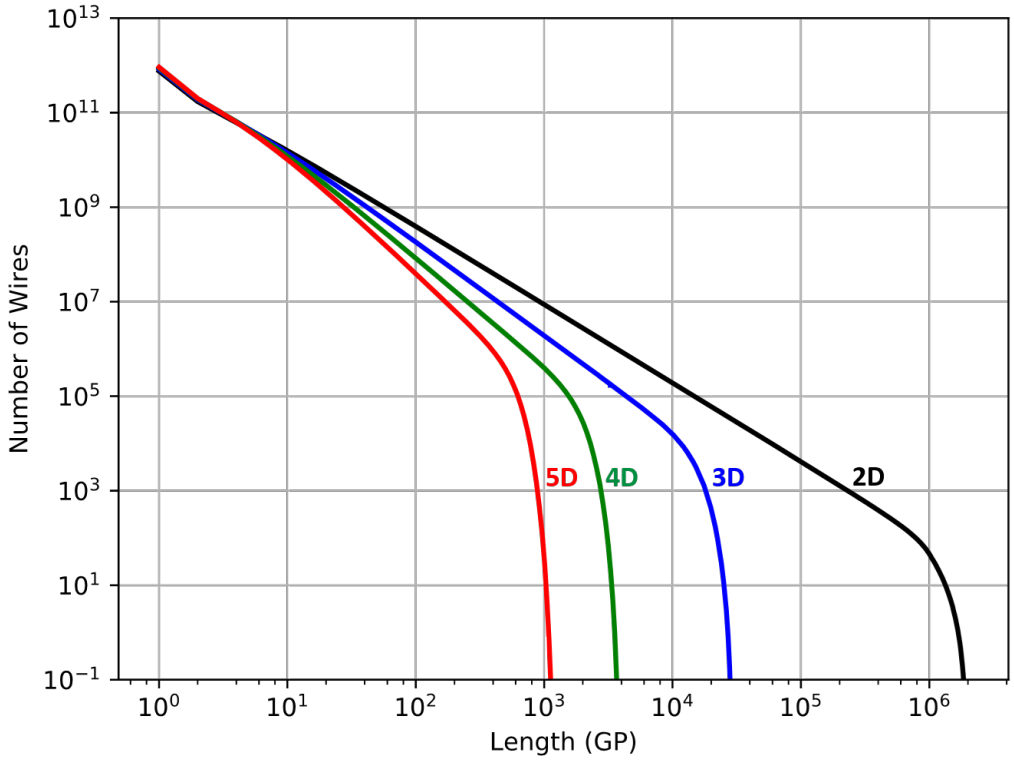

Figure 5.6: Implementing designs in higher-dimensional spaces dramatically reduces the maximum interconnection length.

dimensional systems have dramatically reduced total wire length, though they more shorter wires than their lower-dimensional counterparts.

In order to investigate the impact of interconnect topology on the overall interconnect complexity, we sweep the rent constant of the system from 0.1 to 0.9. The total wirelength of these hypothetical systems is shown in Fig. 5.8. In this figure, we visualize the total wirelength of the system, normalized to its value at a rent exponent of 0, and how that total wirelength changes as the interconnect complexity (as represented by the rent exponent) increases. Interestingly, we see the total wire length of the system fully transition from a low-growth to a high-growth regime at a rent exponent corresponding to the intrinsic physical dimensionality of the system.

For example, the 2D implementation achieves a wire length enhancement of 10X over

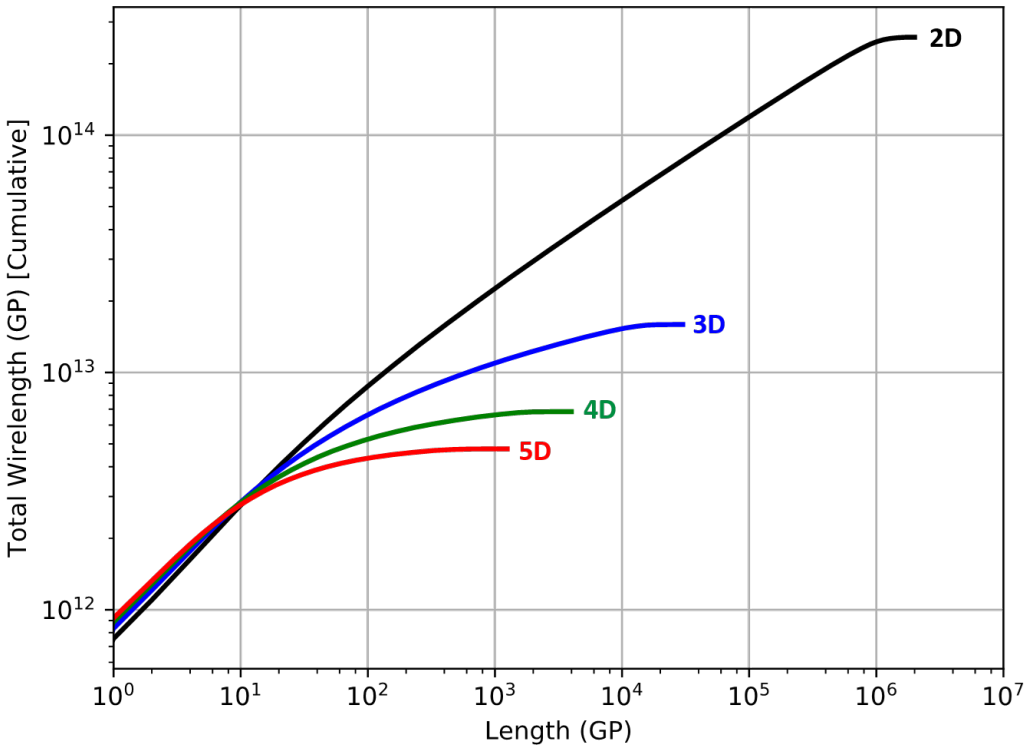

Figure 5.7: Higher-dimensional systems have dramatically-reduced total wire length.

the $p = 0$ case at a rent exponent of $p = 0.5$, which corresponds to a 2D mesh. The 3D system likewise achieves this same enhancement factor at a rent exponent of $p = 0.67$, which corresponds to a 3D mesh, and the 4D and 5D systems behave similarly at $p = 0.75$ and $p = 0.80$, corresponding to 4D and 5D meshes, respectively. By comparing the various curves at a fixed rent exponent we can see the wire length penalty imposed when implementing a system with high topological complexity into a lower spatial dimension; this penalty becomes extreme for rent exponents of 0.75 and greater which must be implemented in two-dimensional spaces.

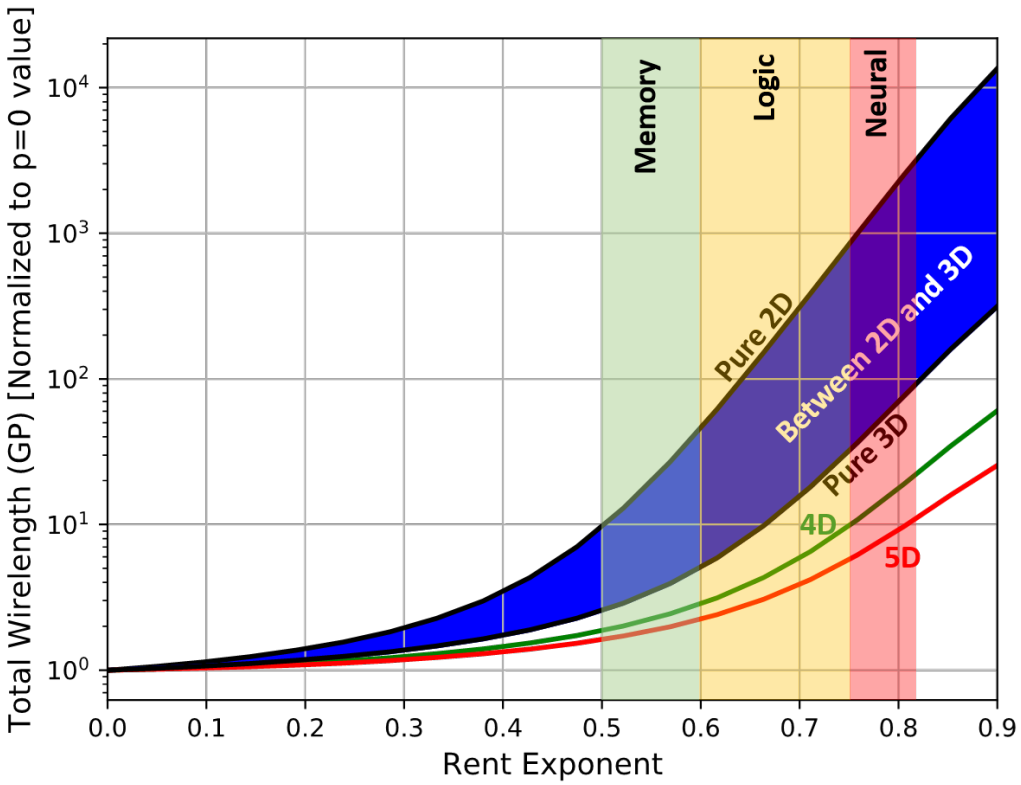

Figure 5.8: Interconnection complexity rapidly increases as the rent exponent exceeds the intrinsic spatial dimension of the design.

5.4 Application to Biological Neural Networks

With the slowing of Moore's law, there is significant interest in the investigation and development of alternative computational schemes. Neuromorphic computing, in which computational systems are designed to emulate the functionality and behavior of the human brain, is of particular interest as a potential enabler of higher performance computing. While Rent's rule was first observed in and formulated for silicon-based systems [19], rent-like scaling has also been observed in biological systems [100–103], with the human brain being given a rent exponent of between 0.75 and 0.8 [104]. While great advances have been made in the performance and power of neural networks and neuromorphic hardware, the determination of the necessary scale and complexity of artificial human-scale neural sys-

tems is still very much an open question. The rent exponents derived for the human brain put it squarely in the domain of a four- or five-dimensional network topology, implying significant interconnection overhead for two- and even three-dimensional implementations.

In order to investigate the potential for the application of stochastic wire length techniques to biological systems we applied the two- and three-dimensional stochastic wire length models presented earlier to estimate the wiring parameters of the human brain. In the brain, disparate neurons are connected by axons, which can very loosely be analogized to wires in a conventional IC. We estimated the average gate pitch of the system by determining the average neuronal pitch, as determined by the total number of neurons in the brain and the typical volume of a human brain. Rent parameters were taken from recent topological connectivity studies of the human brain. The parameters used in these simulations are summarized in Table 5.1.

As a first check to determine whether stochastic wire length techniques can be of use in neural modeling, we investigated the total wire length of a neural-scale system. The total length of axons in the human brain is estimated to be roughly 149,000-176,000 km [107]. Using the neural rent parameters reported in [104], we swept the rent exponent of the system and compared the results for fully two-dimensional and fully three-dimensional implementations. The simulations were repeated using the different Rent Constants reported in [104], yielding ranges of As can be seen in Fig. 5.9, this approach yields surprisingly good agreement with the reported biological data for the relevant parameter ranges, considering the limited nature of the input data.

Table 5.1: Neural Parameters

Number of neurons	10^{11}	[105]
Brain volume (cc)	1260	[106]
Rent Constant (DSI)	1.2	[104]
Rent Exponent (DSI)	0.78	[104]
Rent Constant (MRI)	1.6	[104]
Rent Exponent (MRI)	0.83	[104]
Total Axon Length (km)	149k - 176k	[107]

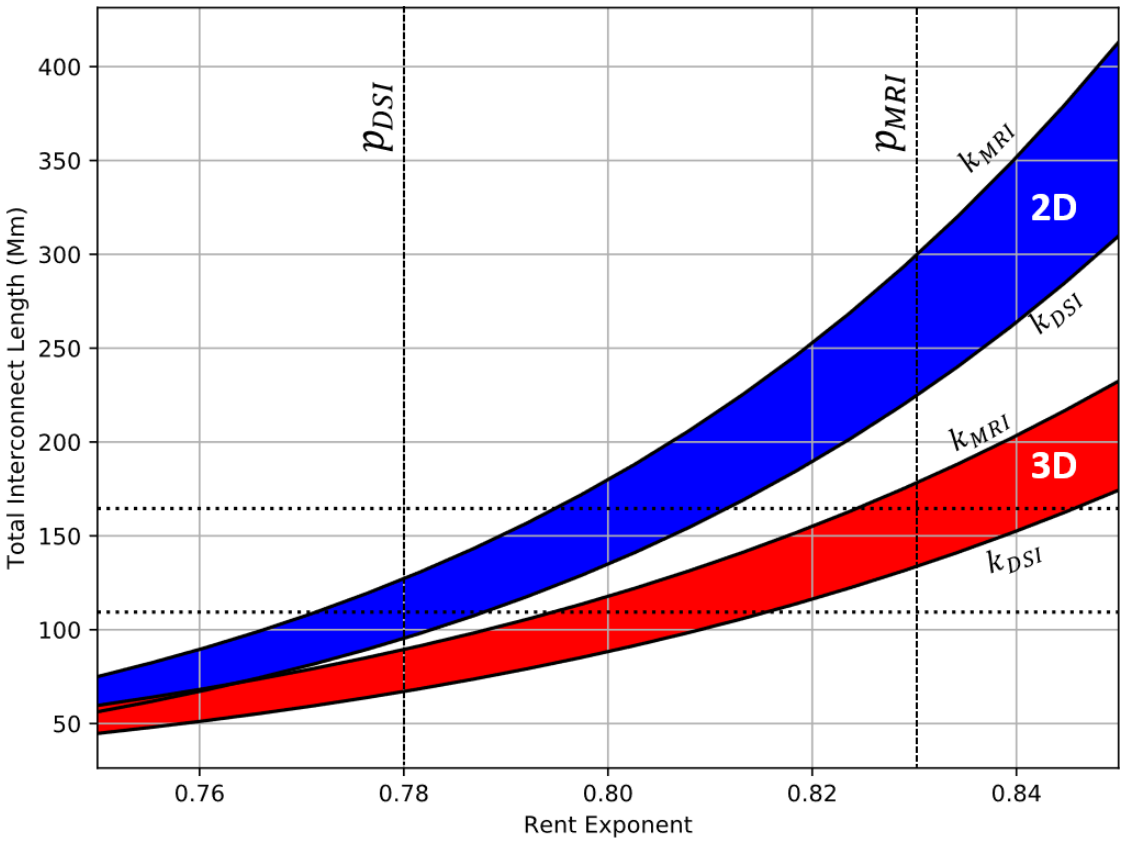

Figure 5.9: Wire length estimates for 2D (blue region) and 3D (red region) cubic networks of roughly the same size as a human brain. The horizontal dotted lines demarcate the first standard deviation of total neural wire lengths reported in [107]. The black curves are calculated using either the extracted DSI or MRI Rent constants, k_{DSI} and k_{MRI}, summarized in Table 5.1, while the colored areas between the black curves represent systems with Rent constants between the two values.

In order to begin to estimate what a brain-scale system might look like if implemented in silicon, we applied the wire sizing and repeater insertion algorithms described in Chapter 3 to arrive at estimates for the overall wiring demand in such a system. We assume a system surface area of 2500 cm^2, equivalent to that of the human brain, and a Rent exponent of 0.80, roughly between the values extracted by various methods in [104]. We consider a range of 2D (single tier) and 3D (two, four, eight, sixteen, and thirty-two tiers) systems, using the 3D stochastic wire length model developed in Chapter 2. We assume logic gate sizes and minimum wiring pitches equivalent to those at the 32nm node, as used for our Sandy Bridge test case in Chapter 3, and we consider a range of target clock frequencies,

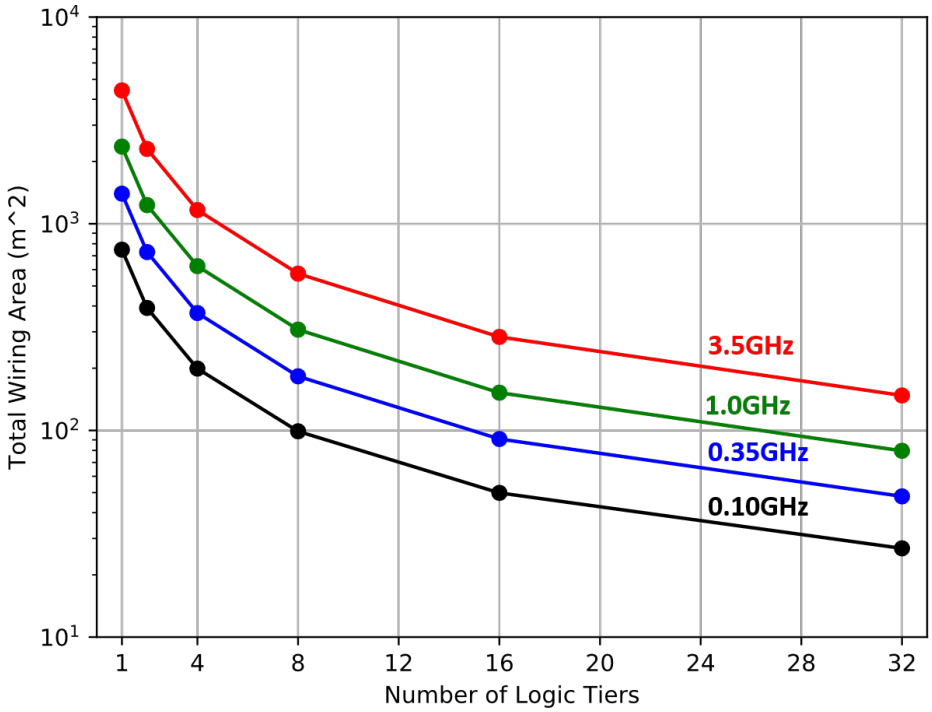

Figure 5.10: Total wiring area requirements for a brain-scale system implemented at the 32nm process node, with target clock frequencies of 100MHz (black), 350MHz (blue), 1GHz (green), and 3.5GHz (red).

from 100MHz to 3.5GHz. Additionally, we consider the same system implemented as a full 3D system, as well as a 4D and 5D system, using the hyperdimensional wire length model developed above.

The total wiring area requirements for a shallow 3DIC brain-scale system implemented at the 32nm node are shown in Fig. 5.10. Due to the high degree of intrasystem connectivity and the extremely high number of computational elements (100B "neurons"), the overall wiring demand in this case is extreme. Even for a 32 tier system running at only 100MHz, the total wiring area required is projected to exceed 3 m^2, which dramatically exceeds the 0.25 m^2 surface area of the system itself. This mismatch indicates that an extremely high number of routing layers would be needed to fully route the system.

In Fig. 5.11 we can clearly see the challenge posed by brain-scale interconnection: for a 2D system operating at 100Mhz, over 400 wiring tiers would be required. The wiring

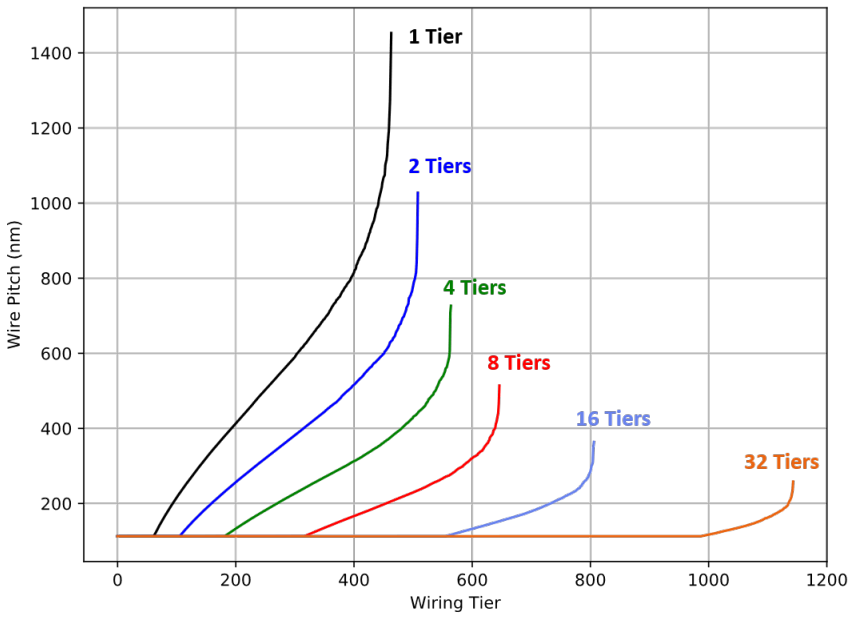

Figure 5.11: Wire pitch requirements for a 3DIC brain-scale system operating at 100MHz. The minimum patternable wire pitch is assumed to be 112nm, corresponding to the 32nm process node.

tier requirements increase as the number of logic tiers increases, due to the reduced per-tier area available for routing, but the reduction in overall interconnection length relaxes the timing constraints on the system, and allows the use of finer-pitch wires for more of the interconnect stack. While a 32-tier stack requires over 1100 total wiring tiers, the expected number of wiring tiers per logic tier is reduced to only 35.75; while this number is quite high compared to the 5-10-layer interconnect stacks common in microprocessors today, it is not entirely outside the bounds of feasibility.

Relaxing the target clock frequency in this case is unlikely to reduce the interconnect requirements for the 32-tier case, as most wires in that system are already pinned at the minimum wire pitch, but lower clock frequencies could significantly reduce wiring requirements for less-aggressive 3D stacks. Increasing the clock frequency from 100MHz to 3.5GHz changes the picture, however, as can be seen in Fig. 5.12. In this case, all configurations under consideration require over $17,500$ total wiring tiers; even in the 32-tier case, over 500 wiring tiers are required for each active logic tier.

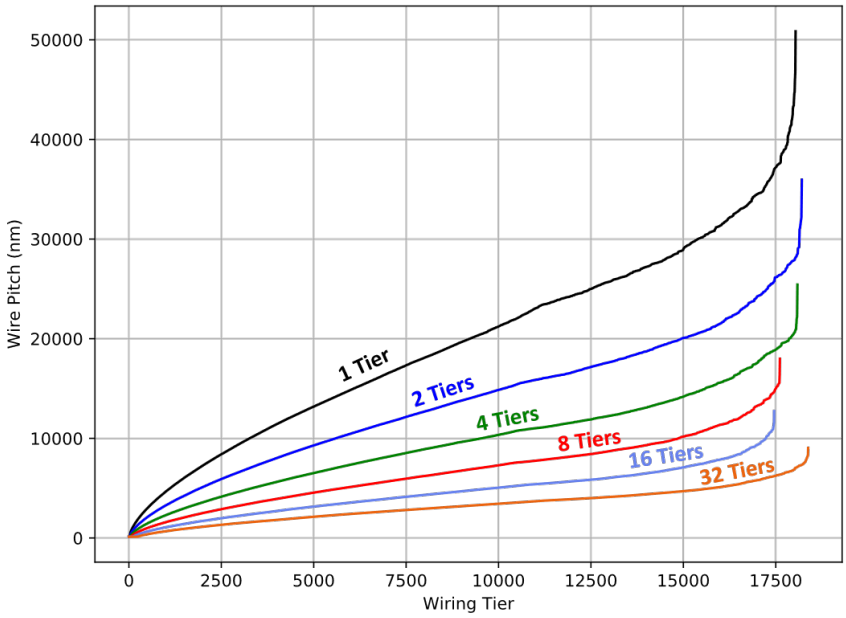

Figure 5.12: Wire pitch requirements for a 3DIC brain-scale system operating at 3500MHz. The minimum patternable wire pitch is assumed to be 112nm, corresponding to the 32nm process node.

In Fig. 5.13, we can clearly see the impact of dimensionality on the system interconnect properties. By moving to a full 3D system, the overall wiring area requirements can be reduced by roughly three orders of magnitude. If four- and five-dimensional systems were feasibly implementable, they could provide additional reductions in wiring requirements, though not as significant as the step from a two- to three-dimensional implementation. This effect is in line with our predictions in Fig. 5.8, as a Rent exponent of 0.8 corresponds roughly to a five-dimensional system, and the additional interconnection penalty is most severe for systems with the smallest physical dimension.

In Fig. 5.14 the wire pitch requirements are shown for 2D, 3D, 4D, and 5D systems operating at 3.5GHz. In this case, a full 3D implementation requires the total wiring tier requirements from over 10^4 to roughly 40. Again, further reductions in the dimensionality of the system provide additional reductions in wiring requirements, though not to the same degree.

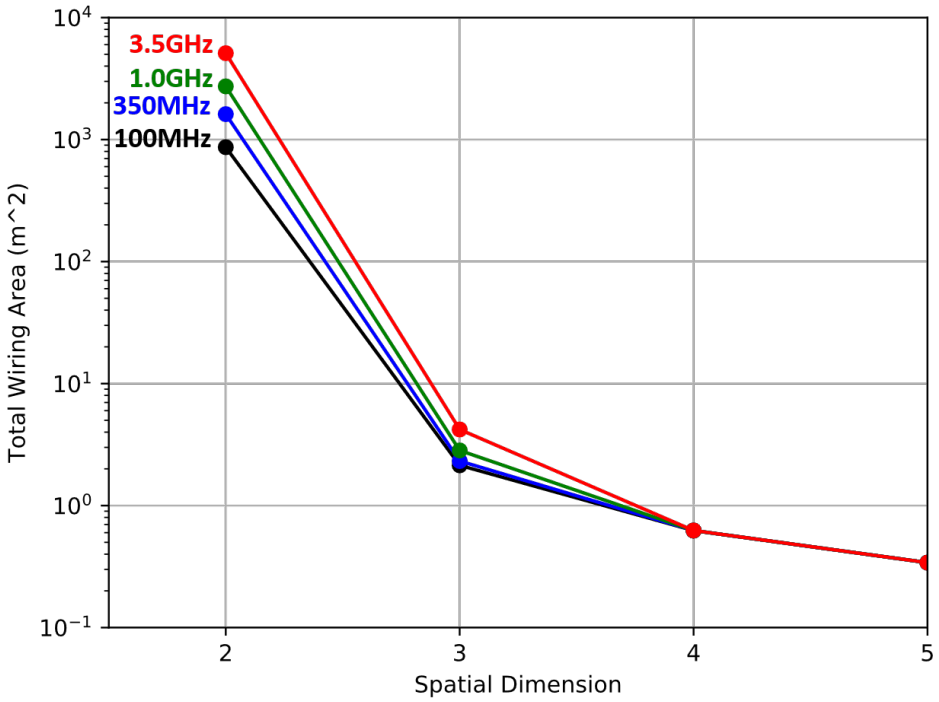

Figure 5.13: Total wiring area requirements for a brain-scale system implemented at the 32nm process node in 2, 3, 4, and 5 dimensions, with target clock frequencies of 100MHz (black), 350MHz (blue), 1GHz (green), and 3.5GHz (red).

5.5 Application to Network Modeling

Hypercubic networks are often considered in high performance network design due to their low network width [90, 94–97, 108]. The methods developed in this section can be directly applied to better understand the properties of information flow in these complex high-dimensional networks. Using the closed-form expression for the node-pair distribution from Eq. (5.20), a picture of the distribution of node distances in a typical hypercubic network of arbitrary dimensionality can be quickly generated. Despite the fact that these methods were initially introduced to model the distance between individual logic gates, the formalism is independent of the specific details of what kind of computing elements are being interconnected. In the case of a large high performance computing cluster, rather than seeking to determine the total length of wires between individual computing gates, we must chiefly worry about the total number of network ”hops” between computing nodes, as these

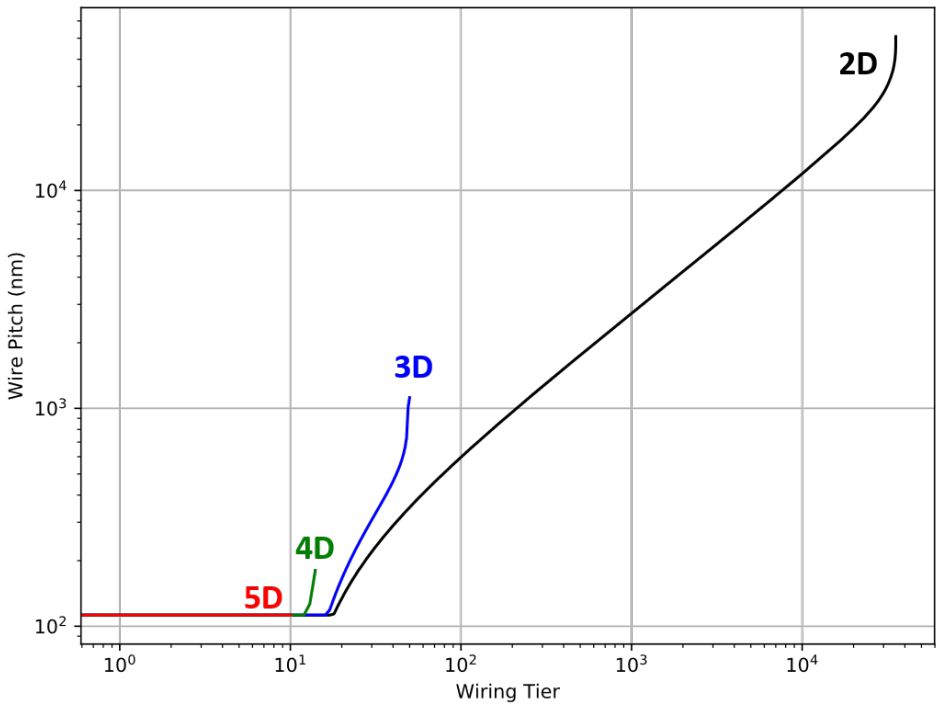

Figure 5.14: Wire pitch requirements for a 3DIC brain-scale system operating at 3500MHz. The minimum patternable wire pitch is assumed to be 112nm, corresponding to the 32nm process node.

dominate the total time required to route data from its origin to its destination. Ultimately, then, the mathematics of the problem are identical – we must still determine the distribution of lengths between pairs of computing nodes – allowing us to repurpose the gate-pair function, originally derived to describe the behavior of interconnects at the micro-scale, within an individual microprocessor, to better understand the distribution of possible communication lengths in these more macroscopic networks interconnecting many individual processors.

Hypercubic networks of this type have been examined for use in high-performance computing systems and supercomputers, due to their ability to dramatically reduce the maximum number of hops required to route information between cores or computing elements [109, 110]. When coupled with an appropriate probability distribution for the likelihood of inter-node communication, the same methodology can be used to predict the distribution of interconnection demand at various length scales in the network.

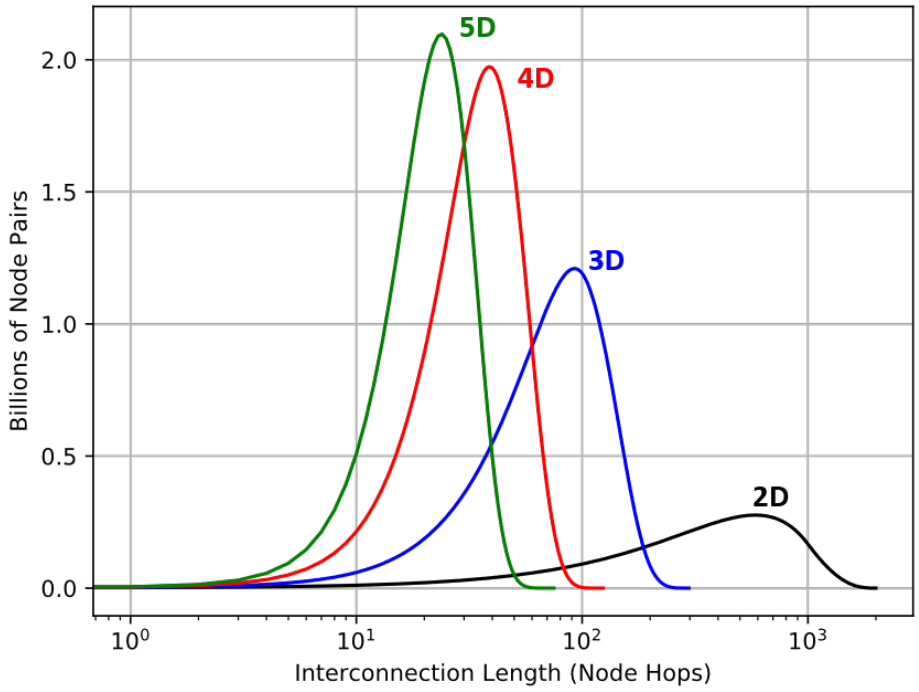

Figure 5.15: Node pair separation distribution in a hypothetical high-performance network with 1M total computing nodes, implemented as hypercubic network of increasing dimension. Implementing the network with higher interconnection dimensionality can significantly reduce the median distance between compute nodes.

For example, let us consider a hypothetical high performance n-cube network with 1M computing nodes, as can be seen in Fig. 5.15. By normalizing the cumulative node pair distribution, we can readily see the significant reduction in both median and maximum node separation, as shown in Fig. 5.16. By combining these node separation distributions with empirical or predicted internodal communication probability models, a more complete picture of network utilization and internodal bandwidth requirements can be determined for high density high performance networks.

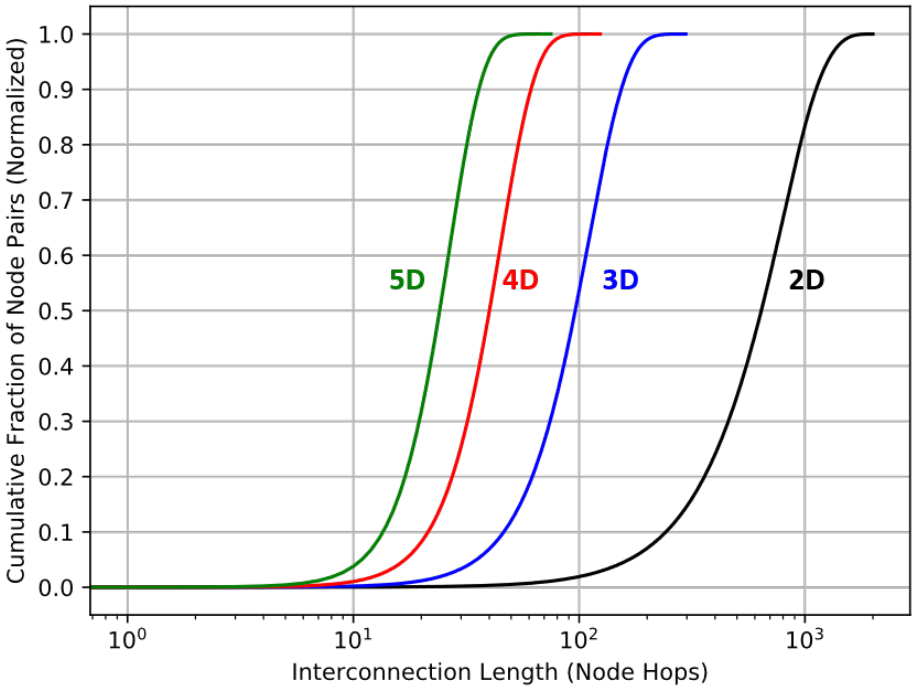

Figure 5.16: Cumulative fraction of node pair separations in a hypothetical high-performance network with 1M total computing nodes, implemented as hypercubic network of increasing dimension.

CHAPTER 6
CONCLUSIONS

3D integration offers an attractive avenue for maintaining the aggressive pace of system-level performance gains demanded by Moore's Law. By reducing the energetic and temporal costs of on-chip and off-chip communication, significant improvements in throughput, latency, and energy efficiency can be realized. These gains, however, come at the cost of increased thermal coupling and overall system complexity. In this thesis, key advancements have been presented to better understand the complexities of the 3DIC design space and to demonstrate and understand the challenges inherent in 3DIC design. The works presented include:

1. Improved stochastic wire length models which account for the impact of large through-silicon vias, without sacrificing solution speed.

2. Improved TSV demand estimation models, which fully account for the presence of signals which may need to be routed through tiers in deep 3D stacks.

3. An integrated 3DIC pathfinding tool, incorporating the improved 3DIC wire length and TSV demand models, wire and repeater sizing models, a finite difference thermal model, and an analytical 3DIC power delivery model.

4. Exploration of the 3D IC design space, and investigation of the tradeoffs inherent in monolithic 3DIC integration.

5. Experimental benchmarking of a computationally-functional 3D thermal testbed, and demonstration of thermal bottlenecking on a range of machine learning benchmarks.

6. Development of stochastic interconnect length models for higher-dimensional interconnect topologies.

In this final chapter, we will briefly summarize the key contributions of this work and discuss potential avenues for further research.

6.1 Summary of the presented work

This thesis has several major components, each of which are summarized as follows:

First, an improved stochastic wire length model for 3DICs was developed, which properly accounted for the impact of large through-silicon vias. While previous works expanded stochastic wire length modeling techniques to three dimensions, the lateral impact of TSVs was ignored. Since TSVs are typically orders of magnitude larger than logic, neglecting the impact of lateral TSV dimensions could lead to underestimating the overall wire length in the system, as TSVs displace logic, leading to a larger overall system footprint. Additionally, an improved TSV demand model was developed to more accurately estimate the TSV requirements in tall 3D stacks. Previous models extended stochastic wire length estimation techniques to estimate the number of interconnections between tiers in a 3D stack, but the existence of signals requiring routing through multiple tiers was neglected. The predictions of the improved TSV demand model were then validated against partition data for several benchmark netlists, and were compared against the predictions of models which did not account for through-tier signal routing.

The improved wire length and TSV demand models were then combined with wire sizing and repeater insertion algorithms, in order to develop a clear picture of the on-chip interconnect requirements in 3DICs. These models were further integrated with a finite difference thermal solver and an analytic 3DIC power delivery model, to create a virtual integration platform for 3DICs. The virtual integration platform enabled rapid investigation of the interplay between signaling, thermal, and power delivery requirements in 3DICs. The virtual platform was validated against published data for Intel processors at the 65, 45, and 32 nm process nodes, and the performance and power consumption of a Sandy Bridge 32nm CPU core was examined for a wide range of 3D configurations. Most notably, the

performance of a 3D implementation of the CPU core was projected to degrade with increasing levels of 3D integration, due to the challenge of adequately cooling such a device.

In order to experimentally investigate the thermal challenges predicted in the prior sections, a 3D thermal testbed was constructed to examine the impacts of 3D stacking on a functional device. The testbed was composed of a high performance NVIDIA Tesla K40 GPU die, atop which was mounted a silicon die equipped with resistive heaters and thermometers. The GPU was stressed with various machine learning workloads, and the power dissipation on the top die was varied to emulate a variety of operating conditions, and significant performance throttling was observed.

Finally, in order to better understand the impact of interconnect topology on overall interconnect performance, stochastic wire length models were developed for cubic systems of arbitrary spatial dimension. While higher-dimensional systems are not directly physically realizable, understanding how interconnect length changes in higher dimensions could help set bounds on the performance of advanced computing systems.

6.2 Avenues for future work

There are several key opportunities for extensions of the work presented in this thesis.

First, the virtual integration platform developed and presented throughout Chapters 2 and 3 could be extended to investigate 2.5D systems. While 3DICs show great potential for reducing system interconnect costs, 2.5D approaches such as interposer-based systems, fan-out wafer level packaging (FOWLP), and other forms of advanced packaging could potentially provide a significant fraction of the performance benefits of 3D integration, while enabling simpler and more cost-effective assembly and manufacturing. In order to fully understand the 3D and 2.5D design space, the virtual platform could be extended with models of off-chip interconnect latency and energy consumption, enabling direct comparison of interconnect performance and system-level power consumption for a range of 2D, 2.5D, and 3D integration methodologies. The integration of manufacturing yield and cost models

would further enhance the utility of such a tool, enabling a holistic evaluation and comparison of system performance, power consumption, and cost across a wide range of platforms and form factors.

Additionally, the thermal testbed presented in Chapter 4 can be further developed to support a wide range of thermal benchmarking for 3D systems. Additional workload classes, such as cryptography, image rendering, and image processing could be investigated, as differing workload profiles may exhibit lesser or greater thermal sensitivity. These investigations could be further enhanced with more detailed workflow emulation on the top die, as different applications might require either different devices (such as FPGAs, DRAM, or NAND Flash), or differing levels of performance and power consumption from the top die. Furthermore, additional dummy dice could be incorporated, in order to examine the thermal behavior of higher-order 3D stacks.

Finally, the high-dimensional interconnect length models presented in Chapter 5 offer several potential avenues of extension. Mechanistically, the model derivation can be revisited to relax the assumption of strict symmetry, enabling the consideration of a much wider range of system types and classes. The models can be further used to explore the interconnect requirements in systems with high-dimensional interconnect topology. The wire and repeater sizing algorithms used in Chapter 3 could be adapted to better investigate the implications for highly-interconnected neural or neuromorphic systems. Additionally, the networking properties of regular high-dimensional networks could potentially be investigated using the developed models.

BIBLIOGRAPHY

[1] R. H. Dennard, F. H. Gaensslen, V. L. Rideout *et al.*, "Design of ion-implanted mosfet's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, Oct 1974.

[2] R. H. Dennard, J. Cai, and A. Kumar, "A perspective on today's scaling challenges and possible future directions," *Solid-State Electronics*, vol. 51, no. 4, pp. 518 – 525, 2007, special Issue: Papers selected from the 2006 ULIS Conference.

[3] H. Esmaeilzadeh, E. Blem, R. S. Amant *et al.*, "Power challenges may end the multicore era," *Commun. ACM*, vol. 56, no. 2, pp. 93–102, Feb. 2013.

[4] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb 2014, pp. 10–14.

[5] A. Mallik, W. Vansumere, J. Ryckaert *et al.*, "The need for euv lithography at advanced technology for sustainable wafer cost," vol. 8679, 2013, pp. 8679 – 8679 – 10.

[6] G. Yeap, "Smart mobile socs driving the semiconductor industry: Technology trend, challenges and opportunities," in *2013 IEEE International Electron Devices Meeting*, Dec 2013, pp. 1.3.1–1.3.8.

[7] W. Steinhogl, G. Schindler, G. Steinlesberger *et al.*, "Comprehensive study of the resistivity of copper wires with lateral dimensions of 100 nm and smaller," vol. 97, no. 2, p. 023706.

[8] D. Josell, S. H. Brongersma, and Z. Tokei, "Size-dependent resistivity in nanoscale interconnects," vol. 39, no. 1, pp. 231–254.

[9] W. Steinhogl, G. Schindler, G. Steinlesberger *et al.*, "Impact of line edge roughness on the resistivity of nanometer-scale interconnects," *Microelectronic Engineering*, vol. 76, no. 1, pp. 126 – 130, 2004, materials for Advanced Metallization 2004.

[10] T. Sun, B. Yao, A. P. Warren *et al.*, "Surface and grain-boundary scattering in nanometric Cu films," *Phys. Rev. B*, vol. 81, p. 155454, Apr 2010.

[11] G. E. Moore, "No exponential is forever: but "forever" can be delayed!" in *2003 IEEE International Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC.*, Feb 2003, pp. 20–23 vol.1.

[12] M. Koyanagi, H. Kurino, K. W. Lee *et al.*, "Future system-on-silicon lsi chips," *IEEE Micro*, vol. 18, no. 4, pp. 17–22, Jul 1998.

[13] M. Matsuo, N. Hayasaka, K. Okumura *et al.*, "Silicon interposer technology for high-density package," in *2000 Proceedings. 50th Electronic Components and Technology Conference (Cat. No.00CH37070)*, 2000, pp. 1455–1459.

[14] M. Sunohara, T. Tokunaga, T. Kurihara *et al.*, "Silicon interposer with tsvs (through silicon vias) and fine multilayer wiring," in *2008 58th Electronic Components and Technology Conference*, May 2008, pp. 847–852.

[15] N. Kim, D. Wu, D. Kim *et al.*, "Interposer design optimization for high frequency signal transmission in passive and active interposer using through silicon via (tsv)," in *2011 IEEE 61st Electronic Components and Technology Conference (ECTC)*, May 2011, pp. 1160–1167.

[16] P. Batude, M. Vinet, A. Pouydebasque *et al.*, "Enabling 3D monolithic integration," *ECS Transactions*, vol. 16, no. 8, pp. 47–54, Oct. 2008.

[17] P. Batude, M. Vinet, A. Pouydebasque *et al.*, "GeOI and SOI 3D monolithic cell integrations for high density applications," in *2009 Symposium on VLSI Technology*, Jun. 2009, pp. 166–167.

[18] M. Vinet, P. Batude, C. Fenouillet-Beranger *et al.*, "Monolithic 3D integration: A powerful alternative to classical 2D scaling," in *SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S), 2014 IEEE*, Oct 2014, pp. 1–3.

[19] B. Landman and R. L. Russo, "On a pin versus block relationship for partitions of logic graphs," *Computers, IEEE Transactions on*, vol. C-20, no. 12, pp. 1469–1479, Dec 1971.

[20] W. Donath, "Placement and average interconnection lengths of computer logic," *IEEE Transactions on Circuits and Systems*, vol. 26, no. 4, pp. 272–277, 1979.

[21] W. Donath, "Wire length distribution for placements of computer logic," *IBM Journal of Research and Development*, vol. 25, no. 3, pp. 152–155, 1981.

[22] A. Masaki and M. Yamada, "Equations for estimating wire length in various types of 2-d and 3-d system packaging structures," *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, vol. 10, no. 2, pp. 190–198, Jun. 1987.

[23] P. Christie, "A fractal analysis of interconnection complexity," *Proceedings of the IEEE*, vol. 81, no. 10, pp. 1492–1499, Oct 1993.

[24] A. Y. Tetelbaum, "Generalizations of rent's rule," in *Proceedings of the Twenty-Seventh Southeastern Symposium on System Theory*, Mar 1995, pp. 370–374.

[25] J. Davis, V. De, and J. Meindl, "A stochastic wire-length distribution for gigascale integration (GSI). I. derivation and validation," *IEEE Transactions on Electron Devices*, vol. 45, no. 3, pp. 580–589, 1998.

[26] J. Davis, V. De, and J. Meindl, "A stochastic wire-length distribution for gigascale integration (GSI). II. applications to clock frequency, power dissipation, and chip size estimation," *IEEE Transactions on Electron Devices*, vol. 45, no. 3, pp. 590–597, 1998.

[27] A. Rahman and R. Reif, "System-level performance evaluation of three-dimensional integrated circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 6, pp. 671–678, 2000.

[28] J. Joyner, P. Zarkesh-Ha, and J. Meindl, "A stochastic global net-length distribution for a three-dimensional system-on-a-chip (3D-SoC)," in *ASIC/SOC Conference, 2001. Proceedings. 14th Annual IEEE International*, 2001, pp. 147–151.

[29] J. Davis and J. Meindl, "Compact distributed RLC interconnect models-part II: coupled line transient expressions and peak crosstalk in multilevel networks," *IEEE Transactions on Electron Devices*, vol. 47, no. 11, pp. 2078–2087, 2000.

[30] R. Venkatesan, J. Davis, and J. Meindl, "Performance enhancement through optimal n-tier multilevel interconnect architectures," in *ASIC/SOC Conference, 1999. Proceedings. Twelfth Annual IEEE International*, 1999, pp. 19–23.

[31] J. Joyner, R. Venkatesan, P. Zarkesh-Ha *et al.*, "Impact of three-dimensional architectures on interconnects in gigascale integration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 6, pp. 922–928, 2001.

[32] D. C. Sekar, A. Naeemi, R. Sarvari *et al.*, "IntSim: A CAD tool for optimization of multilevel interconnect networks," in *Proceedings of the 2007 IEEE/ACM International Conference on Computer-aided Design*, 2007, pp. 560–567.

[33] D. Tuckerman and R. Pease, "High-performance heat sinking for VLSI," *Electron Device Letters, IEEE*, vol. 2, no. 5, pp. 126–129, May 1981.

[34] Y. Zhang, H. Oh, and M. Bakir, "Within-tier cooling and thermal isolation technologies for heterogeneous 3D ICs," in *3D Systems Integration Conference (3DIC), 2013 IEEE International*, Oct 2013, pp. 1–6.

[35] H. Oh, J. M. Gu, S. J. Hong *et al.*, "High-aspect ratio through-silicon vias for the integration of microfluidic cooling with 3D microsystems," *Microelectronic Engineering*, vol. 142, no. Supplement C, pp. 30 – 35, 2015.

[36] T. E. Sarvey, Y. Zhang, L. Zheng *et al.*, "Embedded cooling technologies for densely integrated electronic systems," in *2015 IEEE Custom Integrated Circuits Conference (CICC)*, Sept 2015, pp. 1–8.

[37] H. Oh, Y. Zhang, T. E. Sarvey *et al.*, "TSVs embedded in a microfluidic heat sink: High-frequency characterization and thermal modeling," in *2016 IEEE 20th Workshop on Signal and Power Integrity (SPI)*, May 2016, pp. 1–4.

[38] C. Santos, P. Vivet, J. P. Colonna *et al.*, "Thermal performance of 3D ics: Analysis and alternatives," in *2014 International 3D Systems Integration Conference (3DIC)*, Dec 2014, pp. 1–7.

[39] B. Goplen and S. Sapatnekar, "Thermal via placement in 3d ics," in *Proceedings of the 2005 International Symposium on Physical Design*, ser. ISPD '05. New York, NY, USA: ACM, 2005, pp. 167–174.

[40] J. Cong and G. Luo, "Thermal-aware 3D placement," in *Three Dimensional Integrated Circuit Design*, ser. Integrated Circuits and Systems, Y. Xie, J. Cong, and S. Sapatnekar, Eds. Springer US, Jan. 2010, pp. 103–144.

[41] Y. Madhour, M. Zervas, G. Schlottig *et al.*, "Integration of intra chip stack fluidic cooling using thin-layer solder bonding," in *2013 IEEE International 3D Systems Integration Conference (3DIC)*, Oct 2013, pp. 1–8.

[42] S. Panth, K. Samadi, Y. Du *et al.*, "High-density integration of functional modules using monolithic 3d-IC technology," in *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, pp. 681–686.

[43] D. H. Kim, S. Mukhopadhyay, and S. K. Lim, "Through-silicon-via aware interconnect prediction and optimization for 3D stacked ICs," in *Proceedings of the 11th International Workshop on System Level Interconnect Prediction*, ser. SLIP '09. New York, NY, USA: ACM, 2009, p. 85–92.

[44] W. Wahby, A. Dembla, and M. Bakir, "Evaluation of 3DICs and fabrication of monolithic interlayer vias," in *3D Systems Integration Conference (3DIC), 2013 IEEE International*, Oct 2013, pp. 1–6.

[45] X. Dong and Y. Xie, "System-level cost analysis and design exploration for three-dimensional integrated circuits (3D ICs)," in *Design Automation Conference, 2009. ASP-DAC 2009. Asia and South Pacific*, 2009, pp. 234–241.

[46] D. Velenis, M. Stucchi, E. Marinissen *et al.*, "Impact of 3D design choices on manufacturing cost," in *IEEE International Conference on 3D System Integration, 2009. 3DIC 2009*, Sep. 2009, pp. 1–5.

[47] X. Dong and Y. Xie, "System-level 3D IC cost analysis and design exploration," in *Three Dimensional Integrated Circuit Design*, ser. Integrated Circuits and Systems, Y. Xie, J. Cong, and S. Sapatnekar, Eds. Springer US, Jan. 2010, pp. 261–280.

[48] X. Dong, J. Zhao, and Y. Xie, "Fabrication cost analysis and cost-aware design space exploration for 3-D ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 12, pp. 1959–1972, Dec. 2010.

[49] J. Lau, "TSV manufacturing yield and hidden costs for 3D IC integration," in *Electronic Components and Technology Conference (ECTC), 2010 Proceedings 60th*, Jun. 2010, pp. 1031–1042.

[50] D. Velenis, E. Marinissen, and E. Beyne, "Cost effectiveness of 3D integration options," in *3D Systems Integration Conference (3DIC), 2010 IEEE International*, Nov. 2010, pp. 1–6.

[51] C.-C. Chan, Y.-T. Yu, and I. Jiang, "3DICE: 3D IC cost evaluation based on fast tier number estimation," in *2011 12th International Symposium on Quality Electronic Design (ISQED)*, Mar. 2011, pp. 1–6.

[52] T.-Y. Hsueh, H.-H. Yang, W.-C. Wu *et al.*, "A layer prediction method for minimum cost three dimensional integrated circuits," in *2011 12th International Symposium on Quality Electronic Design (ISQED)*, Mar. 2011, pp. 1–5.

[53] Q. Zou, J. Xie, and Y. Xie, "Cost-driven 3D design optimization with metal layer reduction technique," in *2013 14th International Symposium on Quality Electronic Design (ISQED)*, Mar. 2013, pp. 294–299.

[54] P. Christie and D. Stroobandt, "The interpretation and application of Rent's rule," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 6, pp. 639–648, 2000.

[55] X. Dong and Y. Xie, "System-level 3D IC cost analysis and design exploration," in *Three Dimensional Integrated Circuit Design*, ser. Integrated Circuits and Systems, Y. Xie, J. Cong, and S. Sapatnekar, Eds. Springer US, Jan. 2010, pp. 261–280.

[56] P. Zarkesh-Ha, J. Davis, W. Loh *et al.*, "On a pin versus gate relationship for heterogeneous systems: heterogeneous Rent's rule," in *Custom Integrated Circuits Conference, 1998. Proceedings of the IEEE 1998*, 1998, pp. 93–96.

[57] L. Hagen and A. Kahng, "Fast spectral methods for ratio cut partitioning and clustering," in *, 1991 IEEE International Conference on Computer-Aided Design, 1991. ICCAD-91. Digest of Technical Papers*, 1991, pp. 10–13.

[58] L. Hagen and A. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 9, pp. 1074–1085, 1992.

[59] L. Hagen, A. Kahng, F. Kurdahi *et al.*, "On the intrinsic rent parameter and spectra-based partitioning methodologies," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 1, pp. 27–37, Jan. 1994.

[60] C. Adelmann, L. G. Wen, A. Peter *et al.*, "Alternative metals for advanced interconnects," in *Interconnect Technology Conference / Advanced Metallization Conference (IITC/AMC), 2014 IEEE International*, May 2014, pp. 173–176.

[61] R. Venkatesan, J. Davis, K. Bowman *et al.*, "Optimal n-tier multilevel interconnect architectures for gigascale integration (GSI)," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 9, no. 6, pp. 899–912, Dec 2001.

[62] D. H. Kim, S. Mukhopadhyay, and S. K. Lim, "Through-silicon-via aware interconnect prediction and optimization for 3D stacked ICs," in *Proceedings of the 11th International Workshop on System Level Interconnect Prediction*, ser. SLIP '09. New York, NY, USA: ACM, 2009, p. 85–92.

[63] H. B. Bakoglu and J. Meindl, "Optimal interconnection circuits for VLSI," *IEEE Transactions on Electron Devices*, vol. 32, no. 5, pp. 903–909, 1985.

[64] L. Zheng, Y. Zhang, G. Huang *et al.*, "Novel electrical and fluidic microbumps for silicon interposer and 3-D ICs," *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, vol. 4, no. 5, pp. 777–785, May 2014.

[65] G. Huang, M. Bakir, A. Naeemi *et al.*, "Power delivery for 3-D chip stacks: Physical modeling and design implication," *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, vol. 2, no. 5, pp. 852–859, May 2012.

[66] J. Xie and M. Swaminathan, "Electrical-thermal co-simulation of 3D integrated systems with micro-fluidic cooling and joule heating effects," *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, vol. 1, no. 2, pp. 234–246, Feb 2011.

[67] Y. Zhang, Y. Zhang, and M. Bakir, "Thermal design and constraints for heterogeneous integrated chip stacks and isolation technology using air gap and thermal bridge," *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, vol. 4, no. 12, pp. 1914–1924, Dec 2014.

[68] P. Bai, C. Auth, S. Balakrishnan *et al.*, "A 65nm logic technology featuring 35nm gate lengths, enhanced channel strain, 8 Cu interconnect layers, low-k ILD and 0.57 μm^2 SRAM cell," in *Electron Devices Meeting, 2004. IEDM Technical Digest. IEEE International*, Dec 2004, pp. 657–660.

[69] N. Sakran, M. Yuffe, M. Mehalel *et al.*, "The implementation of the 65nm dual-core 64b Merom processor," in *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, Feb 2007, pp. 106–590.

[70] K. Mistry, C. Allen, C. Auth *et al.*, "A 45nm logic technology with high-k+metal gate transistors, strained silicon, 9 Cu interconnect layers, 193nm dry patterning, and 100% Pb-free packaging," in *Electron Devices Meeting, 2007. IEDM 2007. IEEE International*, Dec 2007, pp. 247–250.

[71] V. George, S. Jahagirdar, C. Tong *et al.*, "Penryn: 45-nm next generation Intel Core2 processor," in *Solid-State Circuits Conference, 2007. ASSCC '07. IEEE Asian*, Nov 2007, pp. 14–17.

[72] R. Kumar and G. Hinton, "A family of 45nm IA processors," in *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, Feb 2009, pp. 58–59.

[73] P. Packan, S. Akbar, M. Armstrong *et al.*, "High performance 32nm logic technology featuring 2nd generation high-k + metal gate transistors," in *Electron Devices Meeting (IEDM), 2009 IEEE International*, Dec 2009, pp. 1–4.

[74] N. Kurd, S. Bhamidipati, C. Mozak *et al.*, "Westmere: a family of 32nm IA processors," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, Feb 2010, pp. 96–97.

[75] M. Yuffe, E. Knoll, M. Mehalel *et al.*, "A fully integrated multi-CPU, GPU and memory controller 32nm processor," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, Feb 2011, pp. 264–266.

[76] N. L. Michael, C.-U. Kim, P. Gillespie *et al.*, "Electromigration failure in ultra-fine copper interconnects," *Journal of Electronic Materials*, vol. 32, no. 10, pp. 988–993, Oct. 2003.

[77] Z. Tokei, K. Croes, and G. P. Beyer, "Reliability of copper low-k interconnects," *Microelectronic Engineering*, vol. 87, no. 3, pp. 348–354, Mar. 2010.

[78] A. S. Oates, "Strategies to ensure electromigration reliability of Cu/low-k interconnects at 10 nm," *ECS Journal of Solid State Science and Technology*, vol. 4, no. 1, pp. N3168–N3176, 2015.

[79] J. Lau, "Evolution, challenge, and outlook of TSV, 3D IC integration and 3D silicon integration," in *2011 International Symposium on Advanced Packaging Materials (APM)*, 2011, pp. 462–488.

[80] R. Patti, "Three-dimensional integrated circuits and the future of system-on-chip designs," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1214–1224, June 2006.

[81] Y. Zhang, A. Dembla, and M. Bakir, "Silicon micropin-fin heat sink with integrated TSVs for 3-D ICs: tradeoff analysis and experimental testing," *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, vol. 3, no. 11, pp. 1842–1850, Nov 2013.

[82] W. Wahby, L. Zheng, Y. Zhang *et al.*, "A simulation tool for rapid investigation of trends in 3-DIC performance and power consumption," *Components, Packaging and Manufacturing Technology, IEEE Transactions on*, vol. 6, no. 2, pp. 192–199, Feb 2016.

[83] D. H. Woo, N. H. Seong, D. L. Lewis *et al.*, "An optimized 3D-stacked memory architecture by exploiting excessive, high-density TSV bandwidth," in *HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, Jan 2010, pp. 1–12.

[84] G. H. Loh, "3D-Stacked Memory Architectures for Multi-core Processors," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 453–464.

[85] J. T. Pawlowski, "Hybrid Memory Cube (HMC)," in *Hot Chips: A Symposium on High Performance Chips*. Stanford, CA, USA: IEEE Technical Committee on Microprocessors and Microcomputers, in cooperation with ACM SIGARCH., 2011.

[86] B. Catanzaro, N. Sundaram, and K. Keutzer, "Fast support vector machine training and classification on graphics processors," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 104–111.

[87] J. H. Lau and T. G. Yue, "Thermal management of 3D IC integration with TSV (through silicon via)," in *2009 59th Electronic Components and Technology Conference*, May 2009, pp. 635–640.

[88] H. C. Chien, J. H. Lau, Y. L. Chao *et al.*, "Thermal evaluation and analyses of 3D IC integration SiP with TSVs for network system applications," in *2012 IEEE 62nd Electronic Components and Technology Conference*, May 2012, pp. 1866–1873.

[89] *Tesla K40 GPU Active Accelerator*, NVIDIA, November 2013.

[90] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Transactions on Computers*, vol. 39, no. 6, pp. 775–785, Jun. 1990.

[91] A. Agarwal, "Limits on interconnection network performance," *IEEE Transactions on Parallel and Distributed Systems*, vol. 2, no. 4, pp. 398–412, Oct. 1991.

[92] W. J. Dally, "Express cubes: improving the performance of k-ary n-cube interconnection networks," *IEEE Transactions on Computers*, vol. 40, no. 9, pp. 1016–1023, Sep. 1991.

[93] L. M. Ni and P. K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *Computer*, vol. 26, no. 2, pp. 62–76, 1993.

[94] F. Petrini and M. Vanneschi, "k-ary n-trees: high performance networks for massively parallel architectures," in *Proceedings 11th International Parallel Processing Symposium*, Apr. 1997, pp. 87–93.

[95] K. Day and A. E. Al-Ayyoub, "Fault diameter of k-ary n-cube networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 9, pp. 903–907, Sep. 1997.

[96] J. Kim, W. J. Dally, and D. Abts, "Flattened Butterfly: A Cost-efficient Topology for High-radix Networks," in *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ser. ISCA '07. New York, NY, USA: ACM, 2007, pp. 126–137.

[97] B. Grot, J. Hestness, S. W. Keckler *et al.*, "Express Cube Topologies for on-Chip Interconnects," in *2009 IEEE 15th International Symposium on High Performance Computer Architecture*, Feb. 2009, pp. 163–174.

[98] E. Bullmore and O. Sporns, "The economy of brain network organization," *Nature Reviews Neuroscience*, vol. 13, no. 5, pp. 336–349, May 2012.

[99] S. A. Ghozati and H. C. Wasserman, "The k-ary n-cube network: modeling, topological properties and routing strategies," *Computers & Electrical Engineering*, vol. 25, no. 3, pp. 155–168, May 1999.

[100] O. Sporns and J. D. Zwi, "The small world of the cerebral cortex," *Neuroinformatics*, vol. 2, no. 2, pp. 145–162, Jun. 2004.

[101] V. Beiu and W. Ibrahim, "Does the brain really outperform Rent's rule?" in *IEEE International Symposium on Circuits and Systems, 2008. ISCAS 2008*, May 2008, pp. 640–643.

[102] E. Bullmore and O. Sporns, "Complex brain networks: graph theoretical analysis of structural and functional systems," *Nature Reviews Neuroscience*, vol. 10, no. 3, pp. 186–198, Mar. 2009.

[103] Y. He and A. Evans, "Graph theoretical modeling of brain connectivity : Current Opinion in Neurology," 2010.

[104] D. S. Bassett, D. L. Greenfield, A. Meyer-Lindenberg *et al.*, "Efficient Physical Embedding of Topologically Complex Information Processing Networks in Brains and Computer Circuits," *PLoS Comput Biol*, vol. 6, no. 4, p. e1000748, Apr. 2010.

[105] S. Herculano-Houzel, "The Human Brain in Numbers: A Linearly Scaled-up Primate Brain," *Frontiers in Human Neuroscience*, vol. 3, Nov. 2009.

[106] K. P. Cosgrove, C. M. Mazure, and J. K. Staley, "Evolving Knowledge of Sex Differences in Brain Structure, Function and Chemistry," *Biological psychiatry*, vol. 62, no. 8, pp. 847–855, Oct. 2007.

[107] L. Marner and B. Pakkenberg, "Total length of nerve fibers in prefrontal and global white matter of chronic schizophrenics," *Journal of Psychiatric Research*, vol. 37, no. 6, pp. 539–547, Nov. 2003.

[108] A. K. Kodi, B. Neel, and W. C. Brantley, "Power and performance analysis of scalable photonic networks for exascale architecture," in *2015 Sixth International Green and Sustainable Computing Conference (IGSC)*, Dec. 2015, pp. 1–8.

[109] E. Anderson, J. Brooks, C. Grassl *et al.*, "Performance of the CRAY T3e Multiprocessor," in *Proceedings of the 1997 ACM/IEEE Conference on Supercomputing*, ser. SC '97. New York, NY, USA: ACM, 1997, pp. 1–17.

[110] X. Li, J. Fan, C.-K. Lin *et al.*, "The extra connectivity, extra conditional diagnosability and t/k-diagnosability of the data center network DCell," *Theoretical Computer Science*, Sep. 2018.

Appendices

APPENDIX A

AUTOMATIC DERIVATION OF HYPERDIMENSIONAL WIRELENGTH FUNCTIONS IN MATHEMATICA 10.3

```
1
2  ClearAll["Global`*"]
3
4  (* Setting up initial 2D and 3D pair functions. These will
       be used to automatically build higher-dimensional pair
       functions *)
5  pairFunc2dA[l_] = n*(n-l)*(l+1) + 1/2*l^2*(l+1) - 1/6*l*(l
       +1)*(2*l+1);
6  pairFunc2dB[l_] = n*(n-l)*(2*n-l-1) + 1/2*l^2*(2*n-l-1) -
       1/6*( n*(n-1)*(2*n-1) - (l-n)*(l-n+1)*(2*l-2*n+1) );
7  pairFunc3dA[l_] = Simplify[Sum[ (n-l+l2)*pairFunc2dA[l2], {
       l2, 0, l} ]];
8  pairFunc3dBA[l_] = Sum[ (n - l+l2) *pairFunc2dA[l2], {l2, l-
       n+1, n-1} ];
9  pairFunc3dcoeffs[l_] = Sum[(n-l+l2)*pairFunc2dB[l2], {l2, n,
        l}] ;
10
11 pairFunc3dB[l_] = Simplify[ pairFunc3dBA[l] +
       pairFunc3dcoeffs[l] ];
12 pairFunc3dB[L];
13
```

```
14 pairFunc3dC[l_] = Simplify[ Sum[ (n-l + l2)*pairFunc2dB[l2],
       {l2, l-(n-1), 2*(n-1)} ] ];
15 pairFunc3dC[L];
16
17 pairFunc4dD[l4_] = Simplify[ Sum[  (n-l4+l3)*pairFunc3dC[l3
   ], {l3, l4-(n-1), 3*(n-1)}] ];
18 pairFunc4dD[L];
19
20 pairFunc2d[l_] = Piecewise[ {{0,l<0},{pairFunc2dA[l], 0 < l
   <= m},{pairFunc2dB[l], m<= l<=2*m},{0, 2*m< l}}];
21 pairFunc3d[l_] = Piecewise[ {{0,l<0},{pairFunc3dA[l], 0 < l
   <= m},{pairFunc3dB[l], m<= l<=2*m},{pairFunc3dC[l], 2*m<=
    l<=3*m},{0, 3*m<l}}];
22
23
24 Element[m&&n, Integers && Positive];
25
26 xa[lp_,y_] = y + lp - m;
27 xb[lp_,y_] = 2*m-lp-y;
28 ymin1 = Ceiling[3/2*m-lp];
29 ymin1 = 3/2*m-lp;
30 yclip = 2*m-lp ;
31 m=n-1;
32
33
34 (* Setting up 2D nonstarting gate functions *)
35 nns2da[lp_] = Sum[1,{x, xb[lp,m], m}];
```

```
36 nns2db[lp_] = Sum[xa[lp,y]-xb[lp,y]+1, {y, ymin1,m}] + Sum
       [1, {x, xa[lp,m]+1,m}];
37 nns2dc[lp_] = Sum[xa[lp,y]-xb[lp,y]+1, {y, ymin1, yclip}] +
       Sum[n,{y,yclip+1,m}];
38 nns2dd[lp_] = Sum[xa[lp,y]-xb[lp,y]+1, {y, 0, yclip}] + Sum[
       n,{y,yclip+1,m}];
39
40 nns2d[l_] = Piecewise[ { {0,l<0},{nns2da[l-1], 0<=l<=1/2*(n
       +1)},{nns2db[l-1],1/2*(n+1)<= l<=m},{nns2dc[l-1],m <= l
       <=3/2*m},{nns2dd[l-1],3/2*m<= l<=2*m},{n^2,l>2*m} } ];
41 nnsGen[func_Symbol,l_] := Sum[func[l-z],{z,0,m}];
42
43
44 (* Expand to higher dimensions *)
45 Element[m&&n,Integers];
46
47 coeffsAndLims = Map[CoefficientList[#,l]&,nns2d[l]];
48 coeffs = Flatten[Map[Delete[#,2]&, coeffsAndLims[[1]] ], 1];
49 coeffLims = Flatten[Map[Delete[#,1]&, coeffsAndLims[[1]] ],
       1];
50
51 numSections = Dimensions[coeffLims][[1]];
52 bounds = Range[0,numSections-1]*m/2;
53 bounds[[-1]] = Infinity;
54
55 maxDim = Max[Flatten[Map[Dimensions[#] &, coeffs,1],1]];
56 coeffsPadded = Map[PadRight[#, maxDim]&, coeffs, 1];
```

```
57
58 SpSum[a_,b_,exp_] := If[exp == 0, Sum[ 1, {p, a, b}],Sum[p^
      exp,{p,a,b}]]; (* If exp is 0 we're just summing 1 from a
       to b, otherwise we need to actually worry about
      functional form of p *)
59
60 Result = ConstantArray[0, {numSections+2, Dimensions[
      coeffsPadded][[2]]} ]; (*numSections+2 because we're
      adding on another dimension, so two more m/2 sized
      sections*)
61
62 (* Version without Floor *)
63 For[ii=0, ii<Dimensions[coeffsPadded][[2]],ii++ ,
64 jjj = 1; (* start at 1 since first domain is actually domain
      -1 *)
65 Result[[jjj+1,ii+1]] =coeffsPadded[[(jjj+1), ii+1]]*SpSum[0,
      1, ii];
66 ]
67
68 For[ii=0, ii<Dimensions[coeffsPadded][[2]],ii++ ,
69 jjj = 2;
70 Result[[jjj+1,ii+1]] = coeffsPadded[[(jjj+1)-1, ii+1]]*SpSum
      [0,(jjj-1)*m/2, ii]
71 + coeffsPadded[[(jjj+1)-0, ii+1]]*SpSum[(jjj-1)*m/2+1,1, ii
      ];
72 ]
73
```

```
74 For[jj=3, jj< numSections, jj++,
75 For[ii=0, ii<Dimensions[coeffsPadded][[2]],ii++ ,
76 Result[[jj+1,ii+1]] = coeffsPadded[[(jj+1)-2,ii+1]]*SpSum[1-
      m,(jj-2)*m/2, ii]
77 + coeffsPadded[[(jj+1)-1, ii+1]]*SpSum[(jj-2)*m/2+1, (jj-1)*
      m/2, ii]
78 + coeffsPadded[[(jj+1)-0, ii+1]]*SpSum[(jj-1)*m/2+1, 1, ii];
79 ]
80 ]
81
82 jj = numSections;
83 For[ii=0, ii<Dimensions[coeffsPadded][[2]],ii++ ,
84 Result[[jj+1,ii+1]] = coeffsPadded[[(jj+1)-2,ii+1]]*SpSum[1-
      m,(jj-2)*m/2, ii]
85 + coeffsPadded[[(jj+1)-1, ii+1]]*SpSum[(jj-2)*m/2+1, (jj-1)*
      m/2, ii]
86 + coeffsPadded[[(jj+1)-1, ii+1]]*SpSum[(jj-1)*m/2+1, 1, ii];
87 ]
88
89
90 FunctionalForms = Map[Total[#]&, Result];
91 FunctionalForms[[-1]] = Total[coeffs[[-1]]]*n;
92 FunctionalForms = Map[Simplify[#]&, FunctionalForms];
93 lowerBounds = Range[-1,numSections + 1 -1]*m/2;
94 upperBounds = Range[0,numSections+1]*m/2;
95 lowerBounds[[1]] = -Infinity;
96 upperBounds[[-1]] = Infinity;
```

```
97
98  limits = MapThread[ #1 <=  l <=  #2 &, {lowerBounds,
        upperBounds}];
99
100 newFunc[l_] := Piecewise[Partition[ Riffle[FunctionalForms,
        limits], 2]];
101 nns3d[l_] = Piecewise[Partition[ Riffle[FunctionalForms,
        limits], 2]];
102 nns3d[L]
103
104 (* This portion extends the results from 3D to 4D. This core
        loop can be repeated again to expand the results to
        higher dimensions *)
105 Element[m&&n, Integers];
106 coeffsAndLims = Map[CoefficientList[#,l]&,newFunc[l]];
107 coeffs = Flatten[Map[Delete[#,2]&, coeffsAndLims[[1]] ], 1];
108 coeffLims = Flatten[Map[Delete[#,1]&, coeffsAndLims[[1]] ],
        1];
109
110 numSections  = Dimensions[coeffLims][[1]];
111 bounds = Range[0,numSections-1]*m/2;
112 bounds[[-1]] = Infinity;
113
114 maxDim = Max[Flatten[Map[Dimensions[#] &, coeffs,1],1]];
115 coeffsPadded = Map[PadRight[#, maxDim]&, coeffs, 1];
116
```

```
117 SpSum[a_,b_,exp_] := If[exp == 0, Sum[ 1, {p, a, b}],Sum[p^
    exp,{p,a,b}]]; (* If exp is 0 we're just summing 1 from a
     to b, otherwise we need to actually worry about
    functional form of p *)
118
119 Result = ConstantArray[0, {numSections+2, Dimensions[
    coeffsPadded][[2]]} ]; (*numSections+2 because we're
    adding on another dimension, so two more m/2 sized
    sections*)
120
121 (* Version without Floor *)
122 For[ii=0, ii<Dimensions[coeffsPadded][[2]],ii++ ,
123 jjj = 1; (* start at 1 since first domain is actually domain
     -1 *)
124 Result[[jjj+1,ii+1]] =coeffsPadded[[(jjj+1), ii+1]]*SpSum[0,
    1, ii];
125 ]
126
127 For[ii=0, ii<Dimensions[coeffsPadded][[2]],ii++ ,
128 jjj = 2;
129 Result[[jjj+1,ii+1]] = coeffsPadded[[(jjj+1)-1, ii+1]]*SpSum
    [0,(jjj-1)*m/2, ii]
130 + coeffsPadded[[(jjj+1)-0, ii+1]]*SpSum[(jjj-1)*m/2+1,1, ii
    ];
131 ]
132
133 For[jj=3, jj< numSections, jj++,
```

```
134 For[ii=0, ii<Dimensions[coeffsPadded][[2]], ii++ ,
135 Result[[jj+1,ii+1]] = coeffsPadded[[(jj+1)-2,ii+1]]*SpSum[1-
      m,(jj-2)*m/2, ii]
136 + coeffsPadded[[(jj+1)-1, ii+1]]*SpSum[(jj-2)*m/2+1, (jj-1)*
      m/2, ii]
137 + coeffsPadded[[(jj+1)-0, ii+1]]*SpSum[(jj-1)*m/2+1, 1, ii];
138 ]
139 ]
140
141 jj = numSections;
142 For[ii=0, ii<Dimensions[coeffsPadded][[2]], ii++ ,
143 Result[[jj+1,ii+1]] = coeffsPadded[[(jj+1)-2,ii+1]]*SpSum[1-
      m,(jj-2)*m/2, ii]
144 + coeffsPadded[[(jj+1)-1, ii+1]]*SpSum[(jj-2)*m/2+1, (jj-1)*
      m/2, ii]
145 + coeffsPadded[[(jj+1)-1, ii+1]]*SpSum[(jj-1)*m/2+1, 1, ii];
146 ]
147
148
149 FunctionalForms = Map[Total[#]&, Result];
150 FunctionalForms[[-1]] = Total[coeffs[[-1]]]*n;
151 FunctionalForms = Map[Simplify[#]&, FunctionalForms];
152 lowerBounds = Range[-1,numSections + 1 -1]*m/2;
153 upperBounds = Range[0,numSections+1]*m/2;
154 lowerBounds[[1]] = -Infinity;
155 upperBounds[[-1]] = Infinity;
156
```

```
limits = MapThread[ #1 <=  l <=  #2 &, {lowerBounds,
    upperBounds}];


newFunc2[l_] := Piecewise[Partition[ Riffle[FunctionalForms,
    limits], 2]];
nns4d[l_] := Piecewise[Partition[ Riffle[FunctionalForms,
    limits], 2]];
nns4d[L]


(* This portion extends the results from 4D to 5D. This core
    loop can be repeated again to expand the results to
    higher dimensions *)
Element[m&&n, Integers];
coeffsAndLims = Map[CoefficientList[#,l]&,newFunc2[l]];
coeffs = Flatten[Map[Delete[#,2]&, coeffsAndLims[[1]] ], 1];
coeffLims = Flatten[Map[Delete[#,1]&, coeffsAndLims[[1]] ],
    1];

numSections  = Dimensions[coeffLims][[1]];
bounds = Range[0,numSections-1]*m/2;
bounds[[-1]] = Infinity;

maxDim = Max[Flatten[Map[Dimensions[#] &, coeffs,1],1]];
coeffsPadded = Map[PadRight[#, maxDim]&, coeffs, 1];

```

```
178 SpSum[a_,b_,exp_] := If[exp == 0, Sum[ 1, {p, a, b}],Sum[p^
    exp,{p,a,b}]]; (* If exp is 0 we're just summing 1 from a
     to b, otherwise we need to actually worry about
    functional form of p *)
179
180 Result = ConstantArray[0, {numSections+2, Dimensions[
    coeffsPadded][[2]]} ]; (*numSections+2 because we're
    adding on another dimension, so two more m/2 sized
    sections*)
181
182 (* Version without Floor *)
183 For[ii=0, ii<Dimensions[coeffsPadded][[2]],ii++ ,
184 jjj = 1; (* start at 1 since first domain is actually domain
    -1 *)
185 Result[[jjj+1,ii+1]] =coeffsPadded[[(jjj+1), ii+1]]*SpSum[0,
    1, ii];
186 ]
187
188 For[ii=0, ii<Dimensions[coeffsPadded][[2]],ii++ ,
189 jjj = 2;
190 Result[[jjj+1,ii+1]] = coeffsPadded[[(jjj+1)-1, ii+1]]*SpSum
    [0,(jjj-1)*m/2, ii]
191 + coeffsPadded[[(jjj+1)-0, ii+1]]*SpSum[(jjj-1)*m/2+1,1, ii
    ];
192 ]
193
194 For[jj=3, jj< numSections, jj++,
```

```
195 For[ii=0, ii<Dimensions[coeffsPadded][[2]], ii++ ,
196 Result[[jj+1,ii+1]] = coeffsPadded[[(jj+1)-2,ii+1]]*SpSum[1-
      m,(jj-2)*m/2, ii]
197 + coeffsPadded[[(jj+1)-1, ii+1]]*SpSum[(jj-2)*m/2+1, (jj-1)*
      m/2, ii]
198 + coeffsPadded[[(jj+1)-0, ii+1]]*SpSum[(jj-1)*m/2+1, 1, ii];
199 ]
200 ]
201
202 jj = numSections;
203 For[ii=0, ii<Dimensions[coeffsPadded][[2]], ii++ ,
204 Result[[jj+1,ii+1]] = coeffsPadded[[(jj+1)-2,ii+1]]*SpSum[1-
      m,(jj-2)*m/2, ii]
205 + coeffsPadded[[(jj+1)-1, ii+1]]*SpSum[(jj-2)*m/2+1, (jj-1)*
      m/2, ii]
206 + coeffsPadded[[(jj+1)-1, ii+1]]*SpSum[(jj-1)*m/2+1, 1, ii];
207 ]
208
209
210 FunctionalForms = Map[Total[#]&, Result];
211 FunctionalForms[[-1]] = Total[coeffs[[-1]]]*n;
212 FunctionalForms = Map[Simplify[#]&, FunctionalForms];
213 lowerBounds = Range[-1,numSections + 1 -1]*m/2;
214 upperBounds = Range[0,numSections+1]*m/2;
215 lowerBounds[[1]] = -Infinity;
216 upperBounds[[-1]] = Infinity;
217
```

```
limits = MapThread[ #1 <=  l <=  #2 &, {lowerBounds,
    upperBounds}];


newFunc3[l_] := Piecewise[Partition[ Riffle[FunctionalForms,
     limits], 2]];
nns5d[l_] := Piecewise[Partition[ Riffle[FunctionalForms,
    limits], 2]];
nns5d[L]

(* The above operations can be repeated indefinitely to
    extend the connection/nonstarting gate results to
    arbitrary dimensions *)
```

APPENDIX B

WIRELENGTH MODELS FOR UP TO FIVE DIMENSIONS IN PYTHON 3.6

```
1
2  import numpy as np
3  import gmpy2 as gp
4
5  def create_length_vec(n,d, use_vpa=False):
6          if(use_vpa):
7                  L = np.array([gp.mpz(thing) for thing in
                       range(d*(n-1)+1)])
8          else:
9                  L = np.linspace(0, d*(n-1), d*(n-1)+1)
10
11         return L
12
13
14 def f2d(L, n):
15         m = n-1
16
17         f2d = ( 0*(L<0)
18         + (n*(n - L)*(L + 1) + 1/2*L**2*(L + 1) - 1/6*L*(L +
               1)*(2*L + 1))*((0<=L) & (L<=m))
19         + (-(1/6)*(-1 + L - 2*n)*(L - 2*n)*(1 + L - 2*n))*((
               m<L) & (L<=2*m))
20         + 0*(L>2*m)
```

```
21          )
22
23          return f2d
24
25
26  def f2d_alt(L,n):
27          m = n-1
28          f2d = ( 0*(L<0)
29          + ((0<=L) & (L<=m))*( 1/6*(1 + L)*(-L + L**2 - 6*L*n
                + 6*n**2))
30          + ((m<L) & (L<=2*m))*( -(1/6)*(-1 + L - 2*n)*(L - 2*
                n)*(1 + L - 2*n) )
31          )
32
33          return f2d
34
35
36  def nns2d(L, n):
37          nns2d = ( 0*(L<0)
38          + (-2+L-2*(-1+n)+2*n)*( (0<=L) & (L<=(1+n)/2) )
39          + (-L+1/4*(3+2*(-1+L)-n)**2+n) * ( ((1+n)/2<=L) & (L
                <=-1+n))
40          + (1/4*(1+n)**2+n*(-2+L-2*(-1+n)+n)) * ( (-1+n<=L) &
                (L<=3/2*(-1+n)) )
41          + (-(L-2*n)*(1+L-n)+n*(-2+L-2*(-1+n)+n)) * (
                (3/2*(-1+n)<=L) & (L<=2*(-1+n)) )
42          + n**2 * (L>2*(-1+n))
```

```
43          )
44
45          return nns2d
46
47
48  def ns2d(L, n):
49          ns2d = n**2 - nns2d(L, n)
50
51          return ns2d
52
53
54  def nns3d(L,n):
55          m = n-1
56
57          nns3d = ( 0*(L<=0)
58          + 1/2*L*(L+1)*((0<L) & (L <= m/2))
59          + 1/24*(8*L**3 - 12*L**2*m + n - n**3 + 2*L*(5 + 3*n
                **2))*((m/2<L) & (L<= m))
60          + 1/24*(12*L**2*m - 6*L*(1 - 8*n + 3*n**2) + n*(17 -
                 24*n + 7*n**2))*((m < L) & (L <= 3*m/2))
61          + 1/24*(-6 - 16*L**3 + 47*n - 78*n**2 + 61*n**3 +
                12*L**2*(-3 + 7*n) - 2*L*(13 - 60*n + 63*n**2))
                *((3*m/2< L) & (L <=2*m))
62          + 1/24*(-6 + 7*n - 12*L**2*n + 66*n**2 - 67*n**3 +
                6*L*(-1 - 4*n + 11*n**2))*((2*m < L) & (L <= 5*m
                /2))
```

```
63          + (L**3/3 + L**2*(1 - 3*n) + n*(-2 + 9*n - 8*n**2) +
                L*(2/3 - 6*n + 9*n**2))*((5*m/2 < L) & (L <= 3*m
               ))
64          + n**3*(3*m < L)
65          )
66
67          return nns3d
68
69
70  def ns3d(L,n):
71          ns3d = n**3 - nns3d(L,n)
72
73          return ns3d
74
75
76  def f3d(L,n):
77          m = n-1
78          f3d = ( 0*(L<0)
79                  - (1/120)*((1 + L)*(2 + L)*(L**3 - 60*n**3 -
                      3*L**2*(1 + 5*n) + L*(2 + 15*n + 60*n
                      **2)))*((0<=L) & (L<=m))
80                  + 1/120*(2*L**5 - 30*L**4*n - 30*L**2*n*(-2
                      - 6*n + 11*n**2) + 10*L**3*(-1 - 3*n +
                      15*n**2) - 3*n*(4 + 20*n - 15*n**2 - 80*n
                      **3 + 31*n**4) + L*(8 + 30*n - 105*n**2 -
                      360*n**3 + 315*n**4))*((m<L) & (L<=2*m))
```

```
81                  -(1/120)*((1 + L - 3*n)*(2 + L - 3*n)*(L**3
                        - 3*L**2*(1 + 3*n) - 3*n*(2 + 9*n + 9*n
                        **2) + L*(2 + 18*n + 27*n**2)))*((4*m<L)
                        & (L<=3*m) )
82                  -(1/120)*(1 + L - 3*n)*(2 + L - 3*n)*(L**3 -
                          3*L**2*(1 + 3*n) - 3*n*(2 + 9*n + 9*n
                          **2) + L*(2 + 18*n + 27*n**2))*((2*m<L) &
                          (L<=3*m) )
83                  + 0*(L>3*m)
84                  )
85
86          return f3d
87
88
89  def nns4d(L,n):
90          m = n-1
91          nns4d =( 0*(L<=0)
92                  + 1/6*L*(2 + 3*L + L**2) *((0<L) & (L<=m/2))
93                  + 1/192*(16*L**4 - 32*L**3*(-2 + n) + (-1 +
                        n)**2*(-3 - 2*n + n**2) + 8*L**2*(13 - 6*
                        n + 3*n**2) - 8*L*(-7 + n - 3*n**2 + n
                        **3))*( (m/2<L) & (L<=m) )
94                  + 1/192*(-3 + 32*L**3*(-2 + n) + 164*n -
                        270*n**2 + 124*n**3 - 15*n**4 - 24*L
                        **2*(7 - 14*n + 3*n**2) + 8*L*(-13 + 67*n
                         - 45*n**2 + 7*n**3))*( (m<L) & (L <=3*m
                        /2))
```

```
95              + 1/96*(-21 - 24*L**4 + 196*n - 378*n**2 +
                  332*n**3 - 129*n**4 + 16*L**3*(-7 + 10*n)
                   - 24*L**2*(8 - 22*n + 15*n**2) + 16*L
                  *(-8 + 37*n - 45*n**2 + 22*n**3))*( (3*m
                  /2<L) & (L <=2*m) )
96              + 1/96*(-21 + L**3*(16 - 32*n) - 28*n + 486*
                  n**2 - 692*n**3 + 255*n**4 + 24*L**2*(1 -
                   10*n + 9*n**2) - 16*L*(1 + 17*n - 51*n
                  **2 + 26*n**3))*( (2*m<L) & (L<= 5*m/2) )
97              + 1/192*(45 + 48*L**4 - 876*n + 3522*n**2 -
                  4884*n**3 + 2385*n**4 - 32*L**3*(-8 + 17*
                  n) + 24*L**2*(19 - 90*n + 93*n**2) - 8*L
                  *(-37 + 323*n - 729*n**2 + 479*n**3))
                  *((5*m/2<L) & (L<=3*m))
98              + 1/192*(32*L**3*n - 24*L**2*(-1 - 6*n + 15*
                  n**2) + 8*L*(9 + n - 135*n**2 + 169*n**3)
                   - 3*(-15 + 68*n + 122*n**2 - 676*n**3 +
                  501*n**4))*( (3*m<L) & (L<=7*m/2) )
99              + 1/12*(-L**4 + 2*L**3*(-3 + 8*n) + L
                  **2*(-11 + 72*n - 96*n**2) - 4*n*(-6 +
                  44*n - 96*n**2 + 61*n**3) + L*(-6 + 88*n
                  - 288*n**2 + 256*n**3))*( (7*m/2<L) & (L
                  <= 4*m) )
100             + n**4*(4*m < L)
101             )
102
103         return nns4d
```

```
104
105
106 def ns4d(L,n):
107         ns4d = n**4 - nns4d(L,n)
108         return ns4d
109
110
111 def nns5d(L,n):
112         m = n-1;
113         nns5d = (0*(L<=0)
114                 + 1/24*L*(6+11*L+6*L**2+L**3)    * ((0<L) & (
                    L<=1/2*m))
115                 + (-30+32*L**5-80*L**4*(-3+n)+31*n+20*n
                    **2-30*n**3+10*n**4-n**5+80*L**3*(9-4*n+n
                    **2)-40*L**2*(-24+9*n-6*n**2+n**3)+2*L
                    *(209-40*n+90*n**2-40*n**3+5*n**4))/1920
                    * ((1/2*m<L) & (L<=m))
116                 + (-30+80*L**4*(-3+n)+1759*n-3340*n**2+2050*
                    n**3-470*n**4+31*n**5-80*L**3*(17-20*n+3*
                    n**2)+40*L**2*(-60+147*n-66*n**2+7*n**3)
                    -10*L*(131-664*n+606*n**2-184*n**3+15*n
                    **4))/1920 * ((m<L) & (L<=3/2*m))
117                 + (-360-128*L**5+1040*L**4*m+3907*n-8560*n
                    **2+8530*n**3-4520*n**4+1003*n**5-80*L
                    **3*(41-80*n+39*n**2)+40*L**2*(-118+363*n
                    -336*n**2+115*n**3)-2*L*(1371-6800*n
                    +9510*n**2-6320*n**3+1695*n**4))/1920 *
```

```
      ((3/2*m<L) & (L<=2*m))
118   + (-360-240*L**4*m-2237*n+17040*n**2-27310*n
      **3+15960*n**4-3093*n**5+80*L**3*(15-48*n
      +25*n**2)-120*L**2*(-14+103*n-144*n
      **2+47*n**3)+10*L*(33-1200*n+3474*n
      **2-2832*n**3+685*n**4))/1920 * ((2*m<L)
      & (L<=5/2*m))
119   + (810+192*L**5-13907*n+57540*n**2-94810*n
      **3+72210*n**4-21843*n**5-240*L
      **4*(-7+11*n)+80*L**3*(69-228*n+175*n**2)
      -120*L**2*(-68+373*n-594*n**2+297*n**3)
      +2*L*(2499-22200*n+57870*n**2-59160*n
      **3+22175*n**4))/1920 * ((5/2*m<L) & (L
      <=3*m))
120   + (810-2963*n-13020*n**2+60710*n**3-70350*n
      **4+24813*n**5+80*L**4*(-1+3*n)-80*L
      **3*(3-36*n+41*n**2)+40*L**2*(8+177*n
      -594*n**2+405*n**3)-10*L*(-135-264*n
      +3978*n**2-7176*n**3+3341*n**4))/1920 *
      ((3*m<L) & (L<=7/2*m))
121   + (-420-128*L**5+13809*n-90440*n**2+225350*n
      **3-238420*n**4+92041*n**5+80*L
      **4*(-15+31*n)-80*L**3*(51-232*n+237*n
      **2)+40*L**2*(-150+1185*n-2652*n**2+1777*
      n**3)-2*L*(1721-23440*n+90450*n
      **2-131920*n**3+64725*n**4))/1920 *
      ((7/2*m<L) & (L<=4*m))
```

```
122                     + (-420+3441*n-80*L**4*n-2120*n**2-30650*n
                            **3+68780*n**4-39031*n**5+80*L**3*(-1-8*n
                            +19*n**2)-40*L**2*(12+15*n-228*n**2+271*n
                            **3)+10*L*(-85+272*n+1110*n**2-4336*n
                            **3+3439*n**4))/1920 * ((4*m<L) & (L
                            <=9/2*m))
123                     + 1/60*(L**5-5*L**4*(-2+5*n)+5*L**3*(7-40*n
                            +50*n**2)-25*L**2*(-2+21*n-60*n**2+50*n
                            **3)-5*n*(24-250*n+875*n**2-1250*n
                            **3+613*n**4)+L*(24-500*n+2625*n**2-5000*
                            n**3+3125*n**4)) * ((9/2*m<L) & (L<=5*m))
124                     + n**5 * (5*m<L)
125                     )
126
127             return nns5d
128
129
130     def ns5d(L, n):
131             ns5d = n**5 - nns5d(L, n)
132
133             return ns5d
134
135
136     def f4d(L,n):
137             m = n-1
138             f4d = ( 0*(L<0)
```

```
139         + ((0<=L) & (L<= m)) * ((1 + L)*(2 + L)*(3 +
                L)*(L**4 + 840*n**4 - 2*L**3*(3 + 14*n)
140         + L**2*(11 + 84*n + 252*n**2)
141         - 2*L*(3 + 28*n + 126*n**2 + 420*n**3)))
              /5040
142         + ((m < L) & (L <= 2*m)) * (1/5040)*(-3*L**7
                + 84*L**6*n + 280*L**4*n*(-2 - 9*n + 14*
              n**2) -
143         42*L**5*(-1 - 4*n + 20*n**2) +
144         28*L**2*n*(29 + 180*n + 160*n**2 - 990*n**3
              + 408*n**4) -
145         7*L**3*(21 + 120*n - 140*n**2 - 1800*n**3 +
              1340*n**4) +
146         4*n*(-36 - 252*n - 623*n**2 + 945*n**3 +
              2821*n**4 - 1953*n**5 +
147         358*n**6) -
148         4*L*(-27 - 168*n - 63*n**2 + 2205*n**3 +
              3710*n**4 - 6615*n**5 +
149         1603*n**6))
150         + ((2*m < L) & (L <=3*m)) * (1/5040)*(3*L**7
                - 84*L**6*n + 42*L**5*(-1 - 2*n + 22*n
              **2) -
151         140*L**4*n*(-5 - 9*n + 38*n**2) -
152         28*L**2*n*(46 + 135*n - 490*n**2 - 810*n**3
              + 1176*n**4) +
153         7*L**3*(21 + 60*n - 640*n**2 - 1080*n**3 +
              2500*n**4) -
```

```
154                     4*n*(-72 - 252*n + 791*n**2 + 2835*n**3 -
                            2653*n**4 - 5103*n**5 +
155                     3194*n**6) +
156                     4*L*(-27 - 84*n + 903*n**2 + 2835*n**3 -
                            4970*n**4 - 8505*n**5 +
157                     8141*n**6))
158                     + ((3*m < L) & (L<=4*m)) * ((-1/5040)*(2 + L
                            - 4*n)*(3 + L - 4*n)*(L**5 - 5*L**4*(1 +
                            4*n) +
159                     5*L**3*(1 + 16*n + 32*n**2) -
160                     5*L**2*(-1 + 12*n + 96*n**2 + 128*n**3) -
161                     8*n*(-3 - 10*n + 40*n**2 + 160*n**3 + 128*n
                            **4) +
162                     2*L*(-3 - 20*n + 120*n**2 + 640*n**3 + 640*n
                            **4)))
163 #                   + ((3*m< L) & (L<=4*m)) * (-1)*(((2+L-4*n)
    *(3+L-4*n)*(L**5-5*L**4*(1+4*n)+5*L**3*(1+16*n+32*n**2)
    -5*L**2*(-1+12*n+96*n**2+128*n**3)-8*n*(-3-10*n+40*n
    **2+160*n**3+128*n**4)+2*L*(-3-20*n+120*n**2+640*n
    **3+640*n**4)))/5040)
164                     )
165
166             return f4d
167
168
169 def f5d(L, n):
170             m = n-1;
```

```
171
172             f5d = (  0*(L<0)
173                    + ((0<=L) & (L<=1*m)) * (-1*(((1+L)*(2+L)
                       *(3+L)*(4+L)*(L**5-15120*n**5-5*L
                       **4*(2+9*n)+5*L**3*(7+54*n+144*n**2)-5*L
                       **2*(10+99*n+432*n**2+1008*n**3)+6*L
                       *(4+45*n+240*n**2+840*n**3+2520*n**4)))
                       /362880))
174                    + ((1*m<L) & (L<=2*m)) * (1/362880)*(4*L
                       **9-180*L**8*n+60*L**7*(-2-9*n+51*n**2)
                       -420*L**6*n*(-7-36*n+61*n**2)-210*L**4*n
                       *(64+480*n+755*n**2-3360*n**3+1431*n**4)
                       +42*L**5*(26+180*n-195*n**2-3600*n
                       **3+2775*n**4)-30*L**2*n*(-572-4872*n
                       -14245*n**2+3360*n**3+89355*n**4-68544*n
                       **5+11766*n**6)+10*L**3*(-328-2646*n
                       -3171*n**2+32760*n**3+110355*n**4-168840*
                       n**5+43890*n**6)-5*n*(576+5184*n+18620*n
                       **2+29232*n**3-74571*n**4-164304*n
                       **5+158730*n**6-51552*n**7+5509*n**8)+3*L
                       *(768+6480*n+14620*n**2-35280*n
                       **3-281645*n**4-285600*n**5+885010*n
                       **6-384720*n**7+50895*n**8))
175                    + ((2*m<L) & (L<=3*m)) * (1/120960)*(-2*L
                       **9+90*L**8*n-60*L**7*(-1-3*n+27*n**2)
                       +420*L**6*n*(-4-12*n+37*n**2)+210*L**4*n
                       *(43+200*n-305*n**2-1520*n**3+1507*n**4)
```

```
      -42*L**5*(13+60*n-375*n**2-1320*n
      **3+2115*n**4)+30*L**2*n*(-464-2576*n
      +1183*n**2+27440*n**3+4935*n**4-65856*n
      **5+31534*n**6)-10*L**3*(-164-882*n+4193*
      n**2+26880*n**3-8785*n**4-105000*n
      **5+70490*n**6)+L*(-1152-6480*n+25820*n
      **2+216720*n**3+162995*n**4-1192800*n
      **5-575470*n**6+1953840*n**7-694065*n**8)
      +5*n*(576+3456*n+652*n**2-37968*n
      **3-54663*n**4+127344*n**5+95058*n
      **6-153312*n**7+43049*n**8))
176   + ((3*m<L) & (L<=4*m)) * (1/362880)*(4*L
      **9-180*L**8*n-1260*L**6*n*(-3-4*n+29*n
      **2)+60*L**7*(-2-3*n+57*n**2)-630*L**4*n
      *(36+80*n-565*n**2-640*n**3+1679*n**4)
      +42*L**5*(26+60*n-1185*n**2-1440*n
      **3+5805*n**4)-90*L**2*n*(-428-1176*n
      +8267*n**2+17920*n**3-40845*n**4-43008*n
      **5+58862*n**6)+10*L**3*(-328-882*n
      +18543*n**2+40320*n**3-149415*n
      **4-161280*n**5+298830*n**6)-15*n
      *(576+1728*n-12388*n**2-37632*n**3+73941*
      n**4+172032*n**5-179622*n**6-196608*n
      **7+153781*n**8)+3*L*(768+2160*n-49420*n
      **2-141120*n**3+486185*n**4+1075200*n
      **5-1630090*n**6-1720320*n**7+1777245*n
      **8))
```

```
177         + ((6*m<L) & (L<=5*m)) * (-1)*(((3+L-5*n)
              *(4+L-5*n)*(L**7-7*L**6*(1+5*n)+7*L
              **5*(1+30*n+75*n**2)-35*L**4*(-1+5*n+75*n
              **2+125*n**3)+7*L**3*(-8-100*n+250*n
              **2+2500*n**3+3125*n**4)-7*L**2*(4-120*n
              -750*n**2+1250*n**3+9375*n**4+9375*n**5)
              -5*n*(48+140*n-1400*n**2-4375*n**3+4375*n
              **4+21875*n**5+15625*n**6)+L*(48+280*n
              -4200*n**2-17500*n**3+21875*n**4+131250*n
              **5+109375*n**6)))/362880)
178         + ((4*m<L) & (L<=5*m)) * (((3 + L - 5*n)*(4
              + L - 5*n)*(L**7 - 7*L**6*(1 + 5*n) + 7*L
              **5*(1 + 30*n + 75*n**2) - 35*L**4*(-1 +
              5*n + 75*n**2 + 125*n**3) + 7*L**3*(-8 -
              100*n + 250*n**2 + 2500*n**3 + 3125*n**4)
               - 7*L**2*(4 - 120*n - 750*n**2 + 1250*n
              **3 + 9375*n**4 + 9375*n**5) - 5*n*(48 +
              140*n - 1400*n**2 - 4375*n**3 + 4375*n**4
               + 21875*n**5 + 15625*n**6) + L*(48 +
              280*n - 4200*n**2 - 17500*n**3 + 21875*n
              **4 + 131250*n**5 + 109375*n**6)))
              /(-362880))
179         + ((8*m<L) & (L<=5*m)) * (-1)*( ((3 + L - 5*
              n)*(4 + L - 5*n)*( L**7 - ( 7*L**6*(1 +
              5*n) + 35*L**4*(-1 + 5*n + 75*n**2 + 125*
              n**3) + 7*L**2*(4 - 120*n - 750*n**2 +
              1250*n**3 + 9375*n**4 + 9375*n**5) + 5*n
```

```
                        *(48 + 140*n - 1400*n**2 - 4375*n**3 +
                        4375*n**4 + 21875*n**5 + 15625*n**6) ) +
                        7*L**5*(1 + 30*n + 75*n**2) + 7*L**3*(-8
                        - 100*n + 250*n**2 + 2500*n**3 + 3125*n
                        **4) + L*(48 + 280*n - 4200*n**2 - 17500*
                        n**3 + 21875*n**4 + 131250*n**5 + 109375*
                        n**6)))/362880)
180                     )
181
182             # Need to round due to floating-point/integer errors
                    in the tail. Even uint64 is not large enough to
                    deal with the numbers in the tail without running
                    into numerical issues
183             #f5dr = np.array([np.round(thing) for thing in f5d])
184
185             return f5d
186
187
188     def nc2d(L, n):
189             nc2d = f2d(L,n)/ns2d(L,n)
190
191             return nc2d
192
193
194     def nc3d(L,n):
195             nc3d = f3d(L,n)/ns3d(L,n)
196             return nc3d
```

```
197
198
199 def nc4d(L,n):
200         nc4d = f4d(L,n)/ns4d(L,n)
201         return nc4d
202
203
204 def nc5d(L, n):
205         nc5d = f5d(L, n)/ns5d(L, n)
206         return nc5d
207
208
209 def nb(nc):
210         nb = np.concatenate(([0, 0], np.cumsum(nc[1:-1])) )
211         return nb
212
213
214 def approx_tac_diff(x,b,p):
215         approx_tac_diff = ((-b**p+(1+b)**p)
216                 +(-b**(-1+p)+(1+b)**(-1+p))*p*(x-b)
217                 +1/2*(-b**(-2+p)+(1+b)**(-2+p))*(-1+p)*p*(x-
                        b)**2
218                 +1/6*(-b**(-3+p)+(1+b)**(-3+p))*(-2+p)*(-1+p
                        )*p*(x-b)**3
219                 +1/24*(-b**(-4+p)+(1+b)**(-4+p))*(-3+p)*(-2+
                        p)*(-1+p)*p*(x-b)**4
```

```
220                     +1/120*(-b**(-5+p)+(1+b)**(-5+p))*(-4+p)
                            *(-3+p)*(-2+p)*(-1+p)*p*(x-b)**5
221                     +1/720*(-b**(-6+p)+(1+b)**(-6+p))*(-5+p)
                            *(-4+p)*(-3+p)*(-2+p)*(-1+p)*p*(x-b)**6
222                     )
223
224             return approx_tac_diff
225
226
227     def tacv_tail_approx(p, Nb, Nc):
228             tacv_tail_approx = ( ((1 + Nb)**p -Nb**p) -
                        approx_tac_diff(Nb+Nc, Nb, p) )
229
230             return tacv_tail_approx
231
232
233     def tacv(p, Nb, Nc):
234             tacv = ( ((1 + Nb)**p -Nb**p) - ((1+Nb+Nc)**p - (Nb+
                        Nc)**p) )
235
236             return tacv
237
238
239     def probv_head(alpha, k, p, n, Nb, Nc):
240             probv_head = (alpha*k/(Nc))*( tacv(p,(Nb),(Nc)) )
241
242             return probv_head
```

```
243
244
245 def probv_tail(alpha, k, p, n, Nb, Nc):
246         probv_tail = (alpha*k/(Nc))*( tacv_tail_approx(p,(Nb
                ),(Nc)) )
247
248         return probv_tail
249
250
251 def probv(alpha, k, p, n, Nb, Nc, ndiv=3):
252         probv = (alpha*k/(Nc))*( np.concatenate( (tacv(p, Nb
                [:round(n/ndiv)], Nc[:round(n/ndiv)]),
                tacv_tail_approx(p, Nb[round(n/ndiv):], Nc[round(
                n/ndiv):] )) ) )
253
254         return probv
255
256
257 def prob2d(alpha, k, p, n, L):
258         nc = nc2d(L, n)
259         prob2d = probv(alpha, k, p, n, nb(nc), nc, ndiv=1e2)
260
261         return prob2d
262
263
264 def prob3d(alpha, k, p, n, L):
265         nc = nc3d(L, n)
```

```
266         prob3d = probv(alpha, k, p, n, nb(nc), nc)
267
268         return prob3d
269
270
271 def prob4d(alpha, k, p, n, L):
272         nc = nc4d(L, n)
273         prob4d = probv(alpha, k, p, n, nb(nc), nc)
274
275         return prob4d
276
277
278 def prob5d(alpha, k, p, n, L):
279         nc = nc5d(L, n)
280         prob5d = probv(alpha, k, p, n, nb(nc), nc)
281
282         return prob5d
283
284
285 def wld2d(alpha, k, p, n, L):
286         wld2d = f2d(L, n) * prob2d(alpha, k, p, n, L)
287
288         return wld2d
289
290
291 def wld3d(alpha, k, p, n, L):
292         wld3 = f3d(L, n) * prob3d(alpha, k, p, n, L)
```

```
293
294         return wld3
295
296
297 def wld4d(alpha, k, p, n, L):
298         wld4 = f4d(L, n) * prob4d(alpha, k, p, n, L)
299
300         return wld4
301
302
303 def wld5d(alpha, k, p, n, L):
304         wld5 = f5d(L, n) * prob5d(alpha, k, p, n, L)
305
306         return wld5
307
308
309 wldict = {
310                 2 : wld2d,
311                 3 : wld3d,
312                 4 : wld4d,
313                 5 : wld5d,
314                 }
315
316
317 def wld(dim, alpha, k, p, n, use_vpa=False):
318         L = create_length_vec(n, dim, use_vpa)
319         return (L, wldict[dim](alpha, k, p, n, L) )
```

```
320
321
322 def gen_system_statistics(dimension, num_gates, alpha, k, p,
        use_vpa=False , chop_zero=False):
323
324         n = round(num_gates**(1/dimension)) # number of
                gates along one edge
325         (L_gp, wl_dist_gp) = wld(dimension, alpha, k, p, n,
                use_vpa)
326
327         if chop_zero:
328                 L_gp = L_gp[1:]
329                 wl_dist_gp = wl_dist_gp[1:]
330
331         wl_total_gp = np.sum( L_gp * wl_dist_gp )
332         num_wires = np.sum(wl_dist_gp)
333         wl_avg_gp = wl_total_gp / num_wires
334
335         return ( n, L_gp, wl_dist_gp, wl_total_gp, num_wires
                , wl_avg_gp )
```