ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Impact of Temperature and Power Supply Voltage Variations on Performance Prediction in 3D ICs during Early Stage Design Exploration on 45nm

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Φίλιππου Αποστόλου Τουφεξή

Επιβλέπων: Δημήτριος Σούντρης
Επ. Καθηγητής

Αθήνα, Ιούλιος 2011

Εθνικο Μετσοβιο Πολυτεχνειο
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Impact of Temperature and Power Supply Voltage Variations on Performance Prediction in 3D ICs during Early Stage Design Exploration on 45nm

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Φίλιππου Αποστόλου Τουφεξή

Επιβλέπων: Δημήτριος Σούντρης
Επ. Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την $4^{η}$ Ιουλίου 2011.

.....	.....	.....
Δημήτριος Σούντρης	Κιαμάλ Πεκμεστζή	Γεώργιος Σταμούλης
Επ. Καθηγητής	Καθηγητής	Καθηγητής

Αθήνα, Ιούλιος 2011.

.....

Φίλιππος Αποστόλου Τουφεξής
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών

Abstract

In this work, a novel design flow, to perform design exploration in 3D ICs is proposed. In this framework, power consumption, timing and fabrication cost are the design goals. Power supply voltage and thermal variations are taken into account, to allow accurate performance predictions. An extensive pre–characterization step, aids fast and accurate whole–system performance predictions. Extreme Value Theory is used to speed–up worst case power estimation. 3 levels of hierarchy are assumed: Modules comprising gates, Common Power Supply Hyper Modules comprising modules under the same power supply, the whole Chip, comprising Common Power Supply Hyper Modules. Floorplanning is a two–step procedure. Initially, Common Power Supply Hyper Modules are treated. Finally, the whole chip is designed. Most parts of the proposed framework were implemented, and used to investigate the impact of temperature and power supply voltage variations, on performance prediction in 3D ICs on 45nm. The ITC99 benchmark circuits were used, along with standard cells from Nangate45, a 45nm standard cell library. The behavior of standard cells in 45nm, and of bigger circuit modules up to 220000 gates, as a function of voltage and temperature is extracted and reported. Strong dependencies on supply voltage and temperature are observed. In a big system, on the order of hundreds million gates, the variability of power supply voltage and temperature is shown to cause on average 40% increase in timing and 53% increase in power consumption, compared to the assumption of nominal conditions, rendering the traditional 5 corner–based design flow inappropriate, for large designs. Finally, a multi–objective genetic algorithm, based on a previously published 3D floorplan representation, along with custom mutation and crossover operators, is shown to optimize for power consumption and timing, yielding clear and useful tradeoffs.

Keywords: 3D ICs, Thermal Simulation, Compact Thermal Model, IR–Drop, 3D Floorplanning, Metaheuristics, Genetic Algorithms, Extreme Value Theory, Maximum Power Estimation, Static Timing Analysis, Design Space Exploration

Acknowledgements

I would like to acknowledge Prof. Dimitrios Soudris for assigning me this topic and supervising me in this work. Also, I would like to thank doctorate student Dimitrios Bekiaris and post-doctorate researcher Antonis Papanikolaou for supporting me in the technical part. I would also like to thank Prof. George Stamoulis and Lect. Nestoras Eumorfopoulos for their inputs in extreme value theory and power grid analysis.

Based on the concept of the thesis, I participated at Intel Challenge Business Plan competition for students and achieved to be in the finalists of Eastern Europe. I would like to thank the management team of Helic, Dr. George Koutsoyannopoulos, Mr. Nikolas Provatas and Mrs Selini Valiaka for their support and guidance in this effort.

Finally, I deeply thank my family and friends for their long-term love and support. They all contributed indirectly to this work, either financially or psychologically.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 The drivers behind 3D Integration

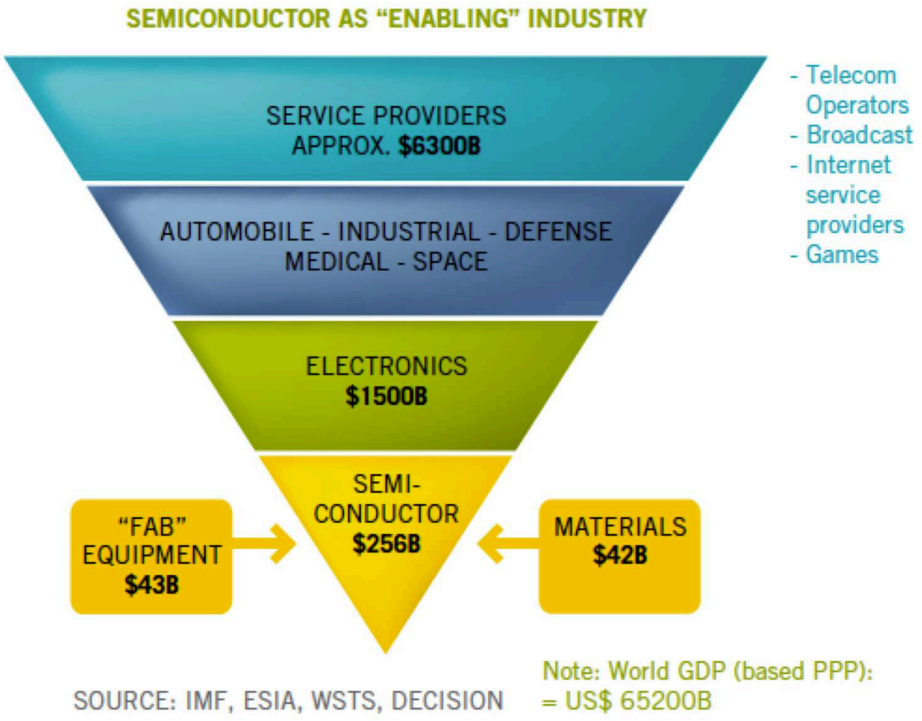
Semiconductors have become indispensable of modern life. They are everywhere, from personal computers to cars, microwave ovens etc, as seen in Figure 1.1.

Figure 1.1: Semiconductor Application Areas

Semiconductors enable 90% of the key technologies and innovations required

for advancing a sustainable information and communication economy, and directly contribute to generating approximately 10% of worldwide GDP as shown in Figure 1.2. The industry is focused on constantly delivering products that provide increased functionality, alongside improved energy performance. The semiconductor industry is a key enabler for energy reduction across other industrial sectors of production.

Figure 1.2: Impact of semiconductors on key downstream sectors worldwide 2007



Increasing complexity results in exponential increases in capital spending and critical know-how. Figure 1.3 clearly shows the exploding cost associated with complex designs, and the high proportion on these costs represented by software and design verification. This contributes to the high R&D intensity of the semiconductor industry.

Since the early 60's, the semiconductor industry has been doubling the number of transistors they put in a single chip every 2 years, as shown in Figure 1.4. This in turn created a need from all the industries that are built upon microchips, that the functionality doubles as well. As is shown in figure 1.4, in the near future, modern silicon processes will not be able to keep up with this rhythm.

To be more precise, integration of more transistor is physically possible,

Figure 1.3: The escalating cost of design

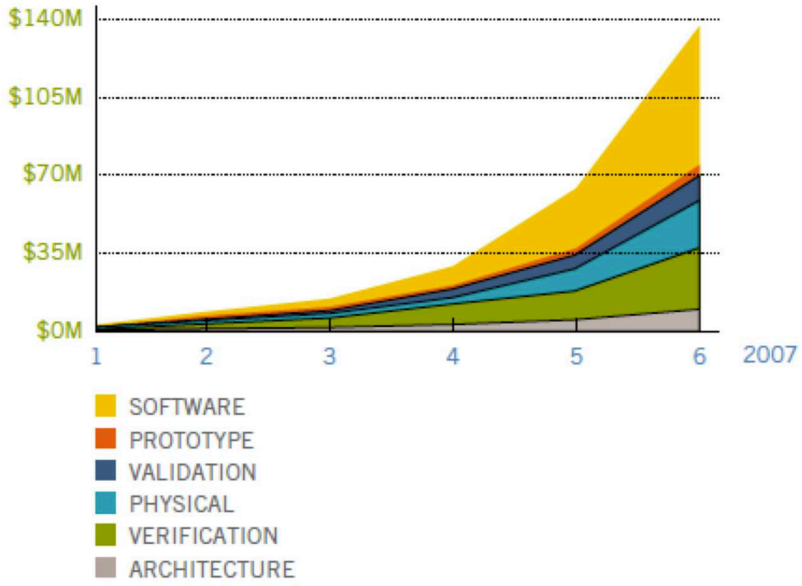
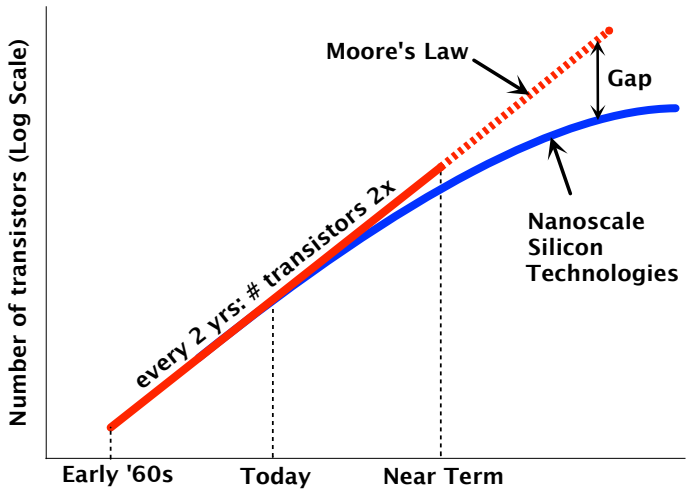


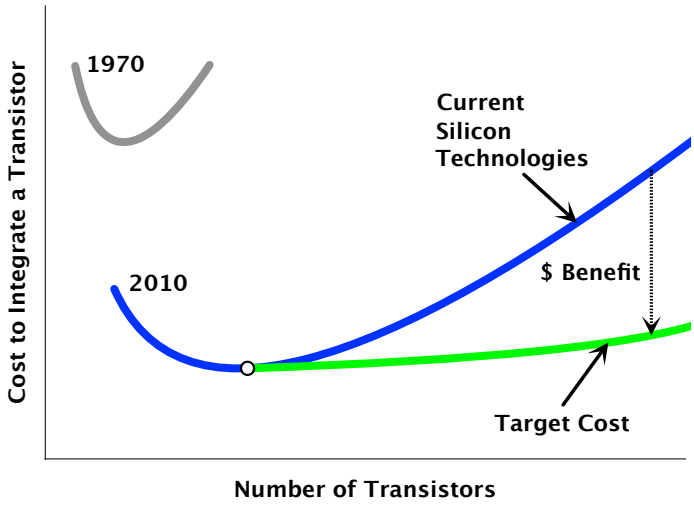
Figure 1.4: Moore's Law



however the cost becomes prohibiting. Figure 1.5 shows the cost function per transistor. The cost has been significantly reduced since 1970. However, there is a number of transistors that minimizes this cost function. After this number, cost grows exponentially.

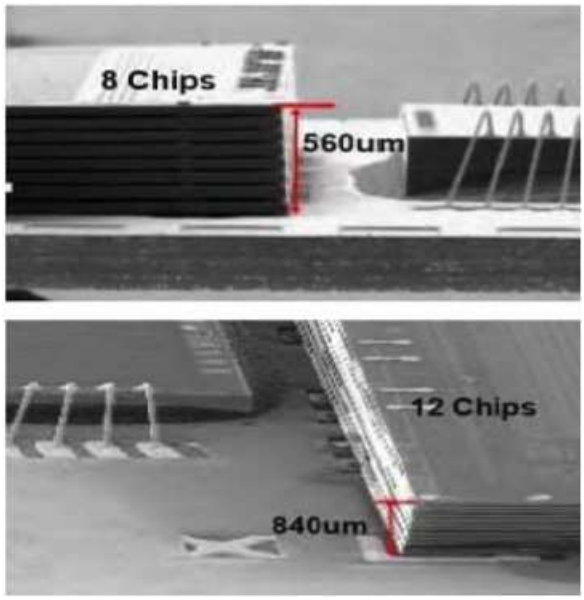
To keep this cost function flat, vertically stacking dies, with vertical through-silicon connections has been proposed as a viable solution, to keep the pace of

Figure 1.5: Per Transistor Cost Function, source [35]



Moore's Law. Figure 1.6 shows fabricated examples of such systems. This way the road a signal has to travel has been reduced from several mm to tens of um. Perhaps more importantly, it is expected that power consumption on the wires will significantly decrease due to the 3D architecture of the power distribution network.

Figure 1.6: Examples of fabricated 3D Systems, Source IEEE Santa Clara CPMT Society Chapter



1.2 Motivations

As shown in [14], 3D Systems are cost effective, when the circuit exceeds $200M$ gates. Vertically stacking so many transistors creates an excessive power to be dissipated. Several papers report temperature increases of more than 100^oC. Little is reported for IR Drop, but is expected that a system that requires so much power will cause great IR drop as well. Power consumption and timing highly depend on the temperature and voltage operating conditions. It is therefore expected that these effects will be amplified in such huge systems.

Although fabrication technology for TSV–based 3D ICs is being developed for more than a decade, lack of commercial EDA tools to target such designs slows down their adoption. Commercial EDA technology can only handle traditional 2D designs, with many constraints on design complexity. Traditional design flows are based on 5 operating condition corner models, $\pm 10\%$ from typical. It is expected that due to high temperature and voltage variations, traditional design tools will not be able to make accurate performance predictions.

On the other hand, academic research has not yet provided a complete solution to the design of 3D ICs. Efficient solutions have been shown to several subproblems, however, combining this knowledge to a design flow is not obvious. Most researchers focus on minimizing temperature and wire–length. Although minimizing these objectives may improve the performance of the system, there is no way to accurately predict the performance from them. Their impact on performance is also not studied. Little work has also been done concerning the impact of temperature and voltage variation on circuit performance. Finally, the only optimization algorithm used across the literature in physical design of 3D ICs is simulated annealing which is inherently a mono–objective optimization technique. This is not adequate since 3D IC design is inherently a multi–objective optimization problem, where a trade–of between fabrication cost, power consumption and timing must be made.

Designing a high performance chip is a project that involves tens or hundreds of engineers, costs more than $40M\$$ and takes one or two years, to build a product that won't last more than two years in the market. The project manager needs to have as soon as possible the high level design alternatives to give a direction for the project. Once this decision is taken, going back costs too much.

To ease adoption of 3D ICs as a viable alternative to process scaling, a design exploration tool is needed that can efficiently tackle the design complexity yet with maximum possible accuracy. It will provide the manager with all the information to make critical decisions, yielding solutions that trade–off cost, power consumption and timing. Additionally, the produced solution, must be as complete as possible, in order to shorten design time.

1.3 Research Questions

The initial questions in the beginning of this work were:

- *How maximum temperature and minimum power supply voltage, are linked with overall system performance?*
- *What series of steps must be taken, in order to be in position, to estimate overall system power consumption and timing?*
- *How can existing 3D floorplan representations, be used effectively, within a true multi–objective optimization algorithm?*

After doing some literature review, the following questions were added as well, and set the theme for this work:

- *How do gates behave, as a function of operating conditions of power supply voltage and temperature?*
- *How do bigger modules behave, in terms of timing and power consumption, as a function of operating conditions of power supply voltage and temperature?*
- *What are the limiting factors in adding logic into a chip?*
- *What is the impact of the operating condition–dependent power consumption, on overall system performance?*

1.4 Organization

In chapter 2, a quick literature review is performed, on the related fields of this work. In chapter 3, the proposed framework is explained in detail. In chapter 4, experimental results are presented, to answer the questions set in the previous section.

Chapter 2

State of the Art

2.1 Metaheuristics

Metaheuristics comprise a set of generic optimization algorithms, to handle problems where the complexity or available search time do not allow the use of exact optimization algorithms. [46] provides a detailed treatment of metaheuristics. For a given problem, an encoding for the solutions must be defined, along with an objective function, that will guide search toward better solutions. Often it is important to define constraints on the problem. Each metaheuristic algorithm defines certain parameters, that must be tuned on the specific problem, in order to improve performance and robustness of the algorithm. Metaheuristics are classified into two broad categories: single solution–based and population–based.

Single solution–based metaheuristics evolve one solution. To use this kind of algorithms on a specific problem, a method to generate an initial solution and the neighborhood of a given solution must be further defined. Popular algorithms in this category are: Local search, Simulated annealing, Tabu search, Iterated local search, and Guided local search. The most widely used algorithm in EDA is Simulated annealing. Simulated annealing is based on the principles of statistical mechanics whereby the annealing process requires heating and then slowly cooling a substance to obtain a strong crystalline structure. The main idea behind simulated annealing is to delay the convergence and improve the probability of escaping local optima to find the global optimum solution.

Population–based metaheuristics iteratively improve a population of solutions. To use this kind of algorithms on a specific problem, a method to generate an initial population, combination and mutation operators, that create new solutions from existing, and selection operators, that select solutions as parents of new ones must be defined. Most algorithms in this category are nature–inspired, such as genetic algorithms, particle swarm optimization, ant colony optimization, bee colony optimization and artificial immune systems.

To solve real–life, multi–objective problems, population based metaheuristics are most suitable, since their output is generally a set of Pareto optimal solutions rather than a single solution as in mono–objective optimization. Often, single solution–based metaheuristics are used in multi–objective optimization by combining the different fitness functions into one, through a weighted sum. However, this approach is considered inefficient. Multi–objective metaheuristics must also handle diversity preservation and elitism. Diversity preservation refers to maintaining a diverse set of Pareto solutions in the objective and/or the decision space. Elitism refer to the preservation and use of elite solutions to achieve robust, fast, and monotonically improving performance of the metaheuristic. One such algorithm is NSGA–II, [13], that will be used in this work.

2.2 Floorplanning of 3D ICs

From [49], the 3D Floorplanning Problem can be defined as: A set of n cuboidal blocks $B = \{B_1, B_2, \ldots, B_n\}$ lying parallel to the coordinate axis. Each block B_i has 3 dimensional sizes (w_i, h_i, t_i). A placement $P = \{(x_i, y_i, z_i) : 1 \leq i \leq n\}$ is an assignment of coordinates at the lower-left-front point of each block such that no two blocks overlap each other. A representation of a placement is the encoding E from which the placement can be retrieved together with $\{(w_i, h_i, t_i)\}$.

[51] contains a review of papers on 3D floorplanning. 3 major directions are identified: Analytical approach, floorplanning with 2D blocks and floorplanning with 3D blocks. In the analytical approach, the set of cubes is modeled as a mechanical system. Springs connect the blocks in order to bring them closer. Other forces are added, to spread the modules in the chip space, or alleviate high temperature. This approach results in a system of linear equations to be solved and is expected to lead to lower run times compared to the other two approaches that utilize stochastic optimization. However, it often cannot result in legal 3D solutions, and is not flexible enough to optimize for other objectives or constraints such as IR–drop.

Floorplanning with 2D blocks involves a list of 2D floorplan representations, one for each device layer. Solutions are generated using simulated annealing. Except for moves within the 2D representation, new moves are added, in order to move a cube to another layer or swap two cubes between different layers. Floorplanning with 3D blocks is considered the best way, since vertical dependencies between modules work better when incorporated directly into the data structure. [20] reviews current 3D floorplan representations. A summary is provided in table 2.1.

Data Structure	Year	Solution Space	Charcteristics
Sequence Triple/Quintuple	2000 [53]	$O((n!)^3)/O((n!)^5)$	Sequence: Three or five sequences.
3D Sub–Transitive Closure Graph	2004 [56]	$O((n!)^3)$	Graph: Three transitive graphs.
T-Tree	2004 [55]	$O(n!2^{3n}/2^{2n}n^{1.5})$	Graph: Ternary tree, nodes: modules, branches: neighbor information.
3D Bounded–Sliceplane Grid	2005 [52]	depends on grid	Grid: 3D grid.
3D Corner Block List	2005 [33]	$O(n!3^{n-1}2^{4n-4})$	Sequence/List: Sequence of modules, list of orientations, list of tribranches.
3D Slicing Tree	2005 [10]	$O(n!2^{3n}/n^{1.5})$	Graph: Binary Tree, inner nodes slices, leaves: modules.
O–Sequence	2006 [38]	n.g.	Sequence: sequence of modules and symbols.
Double Tree and Sequence	2007 [21]	$O(n!n^{2(n-1)})$.	Sequence/Graph: Two rooted trees (x–tree,y–tree), sequence (z–order).
Labeled Tree and Dual Sequences	2008 [49]	$O((n!)^2n^{n-1})$.	Sequence/Graph: Sequence of modules, number sequence and tree.

Table 2.1: State of the Art 3D Floorplan Representations, source [20]

2.3 Multi–Objective Floorplanning

[8] propose a thermal aware floorplanning algorithm supporting voltage islands for low power 2D SoC design. The B*–Tree 2D floorplan representation is used along with simulated annealing. Multi–objective optimization is achieved through a weighted sum of the various design objectives. Most importantly, the dependence of power on both temperature and voltage is taken into account via thermal and IR–Drop simulation and iteratively adjusting power. Experimental results, on a set of modified MCNC benchmarks, show that introduction of voltage islands can reduce the total power by 15% to 30%, and thermal aware voltage island optimization can further reduce the total power by 4% to 15%, while promoting even temperature distribution.

[30] integrate dynamic thermal via planning into 3D floorplanning process. Compact resistive thermal model is used for thermal simulation, 3D Corner Block List for 3D floorplan representation and simulated annealing for optimization. Before floorplanning, the temperature-constrained vertical thermal via planning is formulated as a convex programming problem. Based on the analytical solution, blocks are assigned into different layers by solving a se-

quence of knapsack problems. Then simulated annealing is used to generate floorplans of all these layers simultaneously. During floorplanning, thermal vias are distributed horizontally in each layer, along with white space redistribution to optimize thermal via insertion. Experimental results show that the proposed method can reduce thermal vias by 15% with 38% runtime overhead.

[31] integrate the power supply network co–synthesis into a commercial design flow to develop an effective power integrity driven design methodology for 2D designs. The B*–Tree 2D floorplan representation is used along with simulated annealing. Multi–objective optimization is achieved through a weighted sum of the various design objectives.

[28] propose an algorithm for voltage island aware floorplanning for power and timing optimization for 2D designs. An effective voltage assignment technique based on dynamic programming is developed. The B*–Tree 2D floorplan representation is used along with simulated annealing. Multi–objective optimization is achieved through a weighted sum of the various design objectives. The experimental results show up to 19.75% reduction in power and up to 15.81% reduction in power resources, with a reasonable area overhead of 4%.

[24] present a thermal–aware floorplanner for 3D ICs that accounts for the effects of the interconnect power consumption. It is reported that peak temperatures can be underestimated by as much as 15^oC in 90nm technology node without including interconnect power.

[54] proposes a thermal–aware floorplanning high-level synthesis system that makes use of integrated high–level and physical–level thermal optimization techniques. Voltage islands are automatically generated via slack distribution and voltage partitioning algorithms, in order to reduce the design's power consumption and peak temperature. The Adjacent Constraint Graph 2D floorplan representation is used along with simulated annealing. Multi–objective optimization is achieved through a weighted sum of the various design objectives. The proposed techniques reduces peak temperature by 12.5^oC on average. When used to minimize peak temperature with a fixed area, peak temperature reductions are common. Under a constraint on peak temperature, area is reduced by 9.9% on average.

[29] propose a thermal-driven 3D incremental floorplanning algorithm that uses the mixed integer linear programming formulation. With the analytical approach, chip-area, wirelength and maximal on-chip temperature could be optimized simultaneously. Experimental results show that compared to the original floorplans, these incremental floorplans could reduce max on-chip temperature by about 27% while chip area and total wirelength are enlarged just 1% and 2%, respectively.

[58] propose a force-directed solution to 3D thermal–aware floorplanning problem. By integrating layer assignment with the global optimization process, the proposed flow can smooth transition from continuous vertical space to discrete IC layers. The proposed legalization methods optimize the orientations of

blocks during the legalization process. The closed feedback loop between temperature and leakage power consumption is modeled via a closed form equation.

[23] address the issue of simultaneous buffer and interlayer via planning which is new for 3D. Experimental results show that the proposed algorithm can significantly improve the interconnect delay than 2D ICs. The 3D Corner Block List floorplan representation is used along with simulated annealing. Multi–objective optimization is achieved through a weighted sum of the various design objectives.

[19] propose a 3D Floorplan and P/G Co-synthesis tool that optimizes the floorplan in terms of wirelength, area and P/G routing area and IR drops. The tool integrates a 3D B*-tree floorplan representation, a resistive P/G mesh, and a Simulated Annealing engine to explore the 3D floorplan and P/G network. Multi–objective optimization is achieved through a weighted sum of the various design objectives.

2.4 3D IC Physical Design and Design Exploration Frameworks

[32] propose a 3D design exploration framework for microprocessors. Since the target frequency is fixed, the performance optimization goal is to minimize the IPC degradation, caused by the extra latency introduced by the interconnects in the layout. A modified version of the 3D Corner Block List is used for floorplan and the core optimization algorithm is simulated annealing. Multi–objective optimization is achieved through a weighted sum of performance, area, maximum temperature and wire length. 36% performance improvement and 30% power dissipation reduction over 2D is reported.

[47] explores the architectural design of cache memories using 3–D technologies. A cache delay and energy estimator called 3D-Cacti is proposed to explore different options to partition a cache across different active device layers.

[43] propose a two–step framework for system prototyping on 3D ICs. In the first step, an optimization loop guided by a high level performance estimator, partitions the design into separate dies and chooses the process for each die and bonding scheme between the dies and package. In the second step, traditional 2D tools are employed to design each die. Experimental results on a MPEG4 decoder show 21% increase in frequency, 19% reduction in power consumption, 8% reduction in area, and 16% reduction in wire length.

[14], [15] perform a cost analysis on 3D ICs and propose a cost–driven design exploration framework that is complementary to existing methodologies. Except for the various design guidelines to lower cost, it is reported that 3D ICs are cost effective above 100M gates, based on 65nm processes.

[11] worked on the database for 3D IC physical design. Several extensions to handle 3D designs are proposed on top of OpenAccess, a standardized design

database used by major EDA software vendors.

2.5 Thermal Model

From [51], conventional heat transfer in a chip is described by Fourier's law of conduction, which states that the heat flux, q (in W/m^2), is proportional to the negative gradient of the temperature, T (in K), with the constant of proportionality corresponding to the thermal conductivity of the material, k_t (in $W/(mK)$), i.e.,

$$q = -k_t \nabla T \tag{2.1}$$

The divergence of q in a region is the difference between the power generated and the time rate of change of heat energy in the region. In other words,

$$\nabla \cdot q = -k_t \nabla \cdot \nabla T = -k_t \nabla^2 T = g(\boldsymbol{r}, t) - \rho c_p \frac{\partial T(\boldsymbol{r}, t)}{\partial t} \tag{2.2}$$

Here, $\boldsymbol{r}$ is the spatial coordinate of the point at which the temperature is being determined, t represents time (in s), g is the power density per unit volume (in W/m^3), c_p is the heat capacity of the chip material (in $J/(kgK)$), ρ is the density of the material (in kg/m^3). This may be rewritten as the following heat equation, which is a parabolic PDE:

$$\rho c_p \frac{\partial T(\boldsymbol{r}, t)}{\partial t} = k_t \nabla^2 T + g(\boldsymbol{r}, t) \tag{2.3}$$

To obtain a well-defined solution to (2.3), a set of boundary conditions must be imposed. Typically, at the chip level, this involves building a package macromodel and assuming that this macromodel interacts with a constant ambient temperature.

(2.3) can be solved using numerical methods such as Finite Elements Method or Finite Differences Method. However, these are considered inefficient, especially when thermal simulation is part of an optimization loop.

[44] propose a compact RC model to capture thermal effects on a chip. There exists a well-known duality [27], between heat transfer and electrical phenomena. Table 2.2 summarizes this duality. Heat flow can be described as a "current" passing through a thermal resistance and leading to a temperature difference analogous to a "voltage". Thermal capacitance is also necessary for modeling transient behavior. The thermal Rs and Cs lead to exponential rise and fall times characterized by thermal RC time constants analogous to the electrical RC constants. The rationale behind this duality is that current and heat flow are described by exactly the same differential equations for a potential difference.

It is stated in [51], that time constants of heat transfer are of the order of milliseconds and are much longer than the subnanosecond clock periods in

Thermal Quantity	Unit	Electrical Quantity	Unit
P, Heat flow, power	W	I, Current Flow	A
T, Temperature difference	K	V, Voltage	V
R_{th}, Thermal Resistance	K/W	R, Electrical resistance	$\Omega = V/A$
C_{th}, Thermal mass, capacitance	J/K	C, Electrical capacitance	$F = As/V$
$\tau_{th} = R_{th}C_{th}$, Thermal RC constant	s	$\tau = RC$, Electrical RC constant	s

Table 2.2: Duality between thermal and electrical quantities, source [44]

today's VLSI circuits. Therefore, if a circuit remains within the same power mode for an extended period of time and its power density distribution remains relatively constant, steady-state analysis can capture the thermal behavior of the circuit accurately.

[57] proposes a method that is a combination of several computational techniques including the Green function method, the discrete cosine transform, and the table look-up technique. The fully analytical nature of the Green function method results in high accuracy, while, the fast Fourier transform technique along with the pre-calculated look-up table result in high efficiency.

2.6 Standard Cell Library

Standard cell libraries are the cornerstone of modern digital integrated circuit design flows. Circuit primitives, such as gates or flip flops, are characterized via analog Spice simulations for different excitations and operating conditions, and results such as delay, output slew, leakage power, dynamic power, output current waveforms are stored as lookup tables in the library. Tools, such as timing or power analysis tools, use the standard cell library to calculate performance of a big system, comprising millions of gates, in reasonable time, and accuracy within 2% of Spice. A library cell can be described using various standard formats. In this work, the Liberty format is used. [4] provides a detailed description on the format. [7] describes how the standard cell library is used to make timing and power calculations.

2.7 Extreme Value Theory

Classical Univariate Extreme Value Theory is a statistical theory to predict extreme cases of random variables. [9], [26], [22], [41] and [6] provide a detailed treatment of extreme value theory. There are two models within this theory, distributions of Order Statistics and distributions of Exceedances over a threshold.

In this work, the first model is used, and is further described.

Let X be a random variable with pdf $f(x)$ and cdf $F(x)$. Let $(x_1, x2, \ldots, x_n)$ be an independent and identically distributed sample drawn from $F(x)$. If these samples are partitioned into groups of constant size, and the maximum of each group is obtained, then according to a theorem, these maxima follow a specific family of distributions, called *maximal generalized extreme value distribution* (GEVD). This distribution can be estimated from the sample of maxima and the point where $CDF \to 1$, is the worst case estimate of the maximum value that X can obtain.

Classical Univariate Extreme Value Theory has been successfully applied into maximum power estimation in VLSI circuits. Instead of exhaustively simulating the circuit under all possible excitation vectors, or apply circuit–specific techniques, to identify the worst case excitation that draws the maximum current, extreme value theory presents a viable, circuit–independent, approach to identify the worst case power consumption, using only a small subset of random excitations. [40] proposed a method that can produce maximum power estimates to satisfy user-specified error and confidence levels. Experimental results show that this method provides maximum power estimates within 5% of the actual value and with a 90% confidence level, by simulating on average, about 2500 vector pairs. [18] further develop the method to make it a non–iterative single-loop procedure.

In contrast to univariate extremes, where there is a well developed theory, the field of multivariate extremes is relatively undeveloped, since several problems arise. While in the univariate case the output is a single value that represents the worst case estimate, in the multivariate case, the output is a list of worst case vectors. One obvious solution to the problem is to take the component–wise distributions of maxima and calculate one vector where each entry is the worst case estimate of the respective distribution as done in [17]. However, correlations between components are lost. The classical approach to the problem is a set of multivariate parametric distributions that can be fitted to yield the worst case vectors. However these models are often hard to fit, and cannot be applied in all cases, unlike the univariate case. [9] and [26] provide a detailed treatment of the classical approach to multivariate extremes.

A modern statistical theory with applications in multivariate extremes is Copula Theory. [37] and [25] provide a detailed treatment of copula theory. A copula is a multivariate joint distribution defined on the n-dimensional unit cube $[0, 1]^n$ such that every marginal distribution is uniform on the interval $[0, 1]$. To fit a multivariate distribution using copulas, the marginals are fitted first and then the copula is fitted based on the marginals. Copulas have received a lot of attention lately, due to mainly their ease of use. However, they have also received several criticism, as in [16] and [34]. [34] mentions though, that in case copulas are used for multivariate extremes, then indeed they can do the job. [42] and [25] provide a detailed treatment of multivariate extreme value theory

using copulas.

Chapter 3

Proposed 3D IC Design Framework

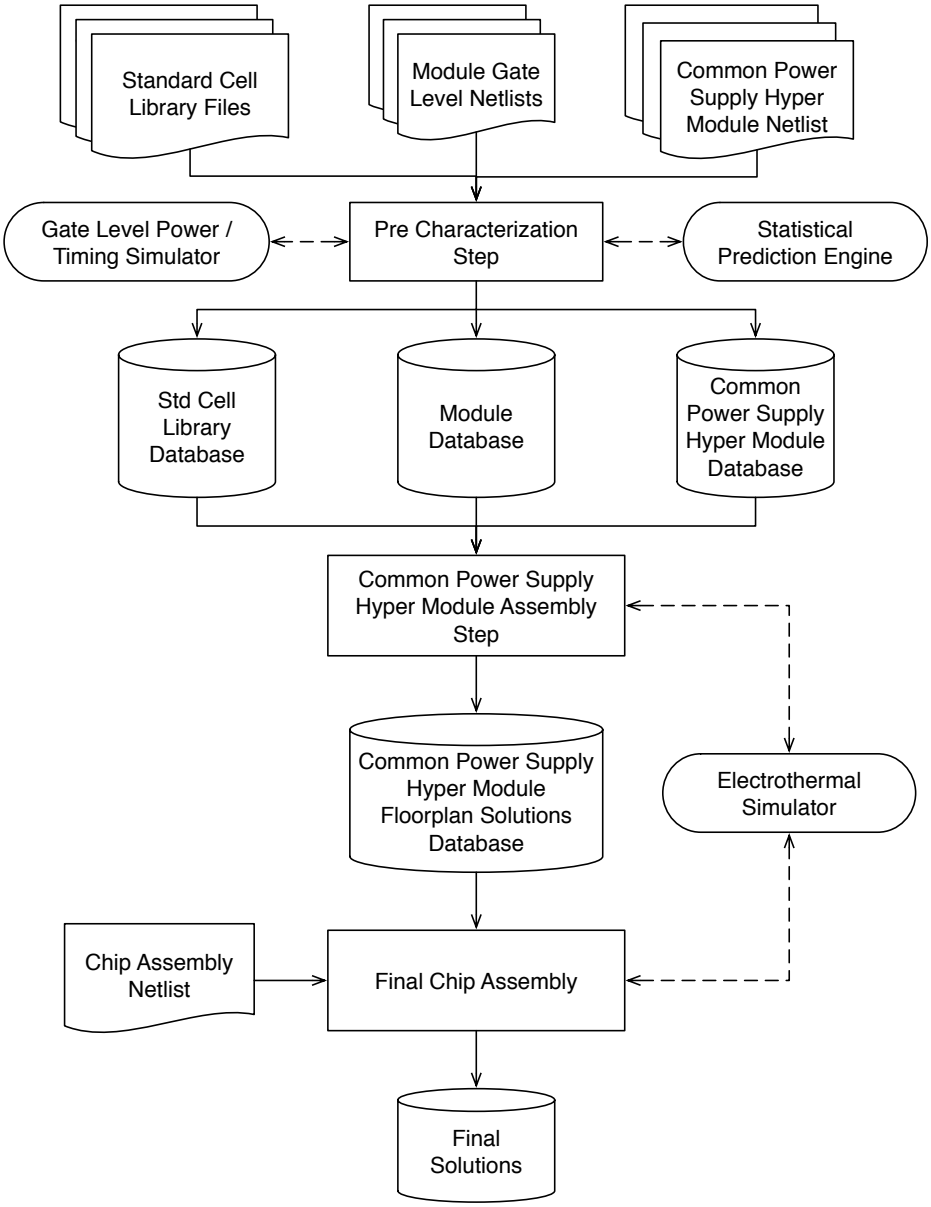
3.1 Overview

In this chapter, a novel design flow to perform design exploration in 3D ICs is proposed. Figure 3.1 shows the top level description of the proposed framework. The principles behind this framework are:

1. Power Consumption, Timing and Fabrication cost are the design goals.
2. Power supply voltage and thermal variations are taken into account to allow accurate performance predictions.
3. An extensive pre–characterization step that will aid fast and accurate whole–system performance predictions.
4. Extreme Value Theory is used to speed–up worst case power estimation.
5. 3 levels of hierarchy: Modules comprising gates, Common Power Supply Hyper Modules comprising modules under the same power supply, the whole Chip, comprising Common Power Supply Hyper Modules. Figures 3.2 and 3.3 show examples of this hierarchy.
6. A two–step assembly procedure. Initially, Common Power Supply Hyper Modules are treated. Finally, the whole chip is designed.

The aim of the design exploration phase in a high performance chip design project, is to assess the tradeoffs between the system's performance and fabrication cost. It is a critical step, since decisions are taken on which fabrication process or packaging technology will be used. Decisions are more complicated for 3D ICs: how many layers will be used, what fabrication process will be

Figure 3.1: Overview of Proposed Framework



used in each, what packaging or bonding technology will be used? Therefore, to make such decisions, the design goals must be power consumption, timing and fabrication cost. This requires both a method to estimate those parameters as precisely as possible, at the minimum computational cost, and an optimization method for these goals.

Increasing logic size, results in increased power consumption and causes high IR–drop. Vertically stacking dies results in increased power density and causes heating. It will be shown experimentally that these two factors are tightly interconnected and cause significant power supply voltage and temperature variations across chip, affecting both timing and power consumption. It is a chicken–and–egg loop. These across chip variations render the traditional corner model–based design flow useless. To make accurate predictions, one must take into account the operating conditions at each point of the chip.

To make performance prediction fast enough in order to be used in an optimization loop, a extensive pre–characterization step is proposed. Starting from the grounds up, gates are pre–characterized in a wide range of voltage and temperature operating conditions, instead of the traditional 5 corner $\pm 10\%$ approach. One level up, modules are pre–characterized in terms of worst case power consumption and timing in the same range of operating conditions. Traditional worst case power characterization techniques make an extensive simulation, to identify the excitation vector that causes the worst power consumption. This is computationally prohibiting, since it must be performed in wide range of operating conditions. To bypass this limitation, extreme value theory is used to make estimates of the worst case power consumption, using only a limited set of excitation vectors.

The most common IC design flows assume 2 levels of hierarchy: modules consisting of gates, and the whole chip consisting of modules. As it will be experimentally showed later, the limiting factors, as logic complexity increases, are yield and IR-drop. Yield severely affects production cost and can be treated by vertically stacking smaller dies instead of one huge single–die chip. On the other hand, IR–drop affects both power consumption and timing, and can even render the chip non operational, if power supply voltage falls under the transistor's threshold voltage. Vertically stacking dies improves IR-drop, but it still consists a limiting factor. For this reason, in the proposed approach, another level of hierarchy is placed between the modules and the chip. Hyper modules, that comprise modules under the same power supply, are designed separately. This significantly reduces optimization time as well, by creating two smaller sub–problems. The final chip, is a 2D assembly of such hyper modules. It will be shown experimentally later, that, since hyper modules are not connected to the same supply grid, the thermal interactions are negligible, because the dominant heat flow path is the vertical. Therefore, estimations on the system's performance based on the same power supply hyper modules are safe.

Figure 3.2: Example of Common Power Supply Hyper Modules Assemblies

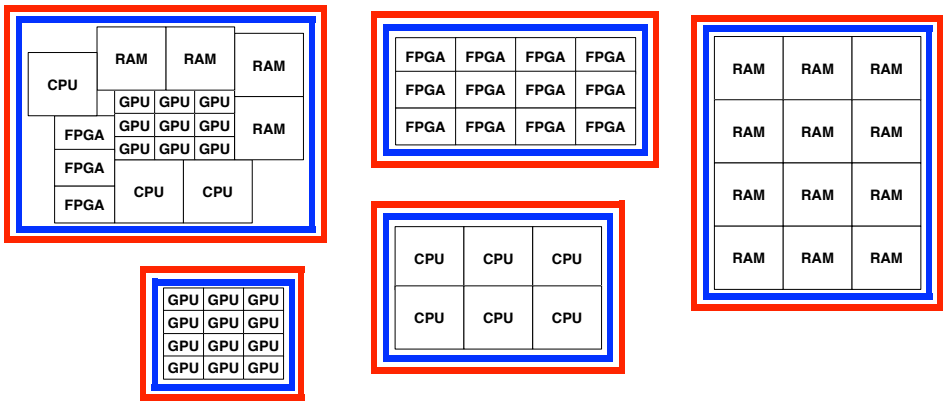
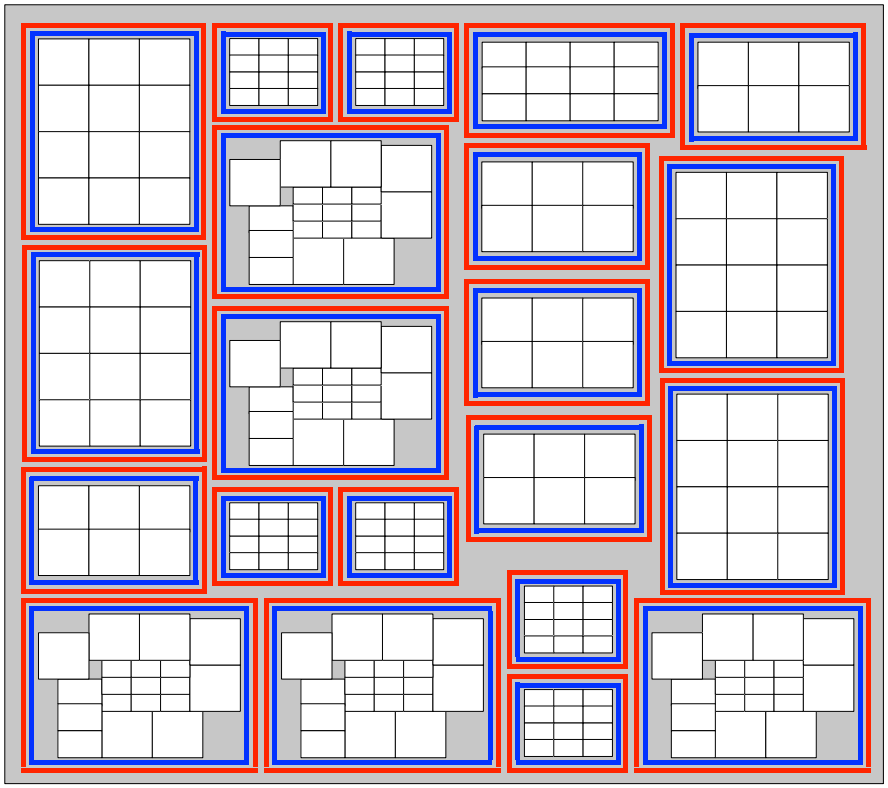


Figure 3.3: Example of Chip Assembly



3.2 Pre–Characterization Step

The Pre–Characterization Step is summarized in Figure 3.4. Note that tool configuration files are not shown in that figure. The aim of the Pre–Characterization Step is to precompute as much information possible in the 3 first levels of design hierarchy. Although this is a computationally-heavy step, the computational cost at later design steps is significantly decreased. The Pre–Characterization Step comprises three successive procedures:

1. Pre–Characterization of the Standard Cell Library. This is a generalization of a standard procedure in IC design, to include information in an abundance of operating conditions.

2. Pre–Characterization of the Module Library. Following the concept behind Standard Cell Library Pre–Characterization, the separate module types, that are instanced in the Common Power Supply Hyper Modules, are pre–characterized as well, in terms of timing, power and physical form.

3. Pre–Characterization of the Common Power Supply Hyper Modules. These circuits are characterized in terms of maximum power distribution across the modules they contain, under nominal operating condition.

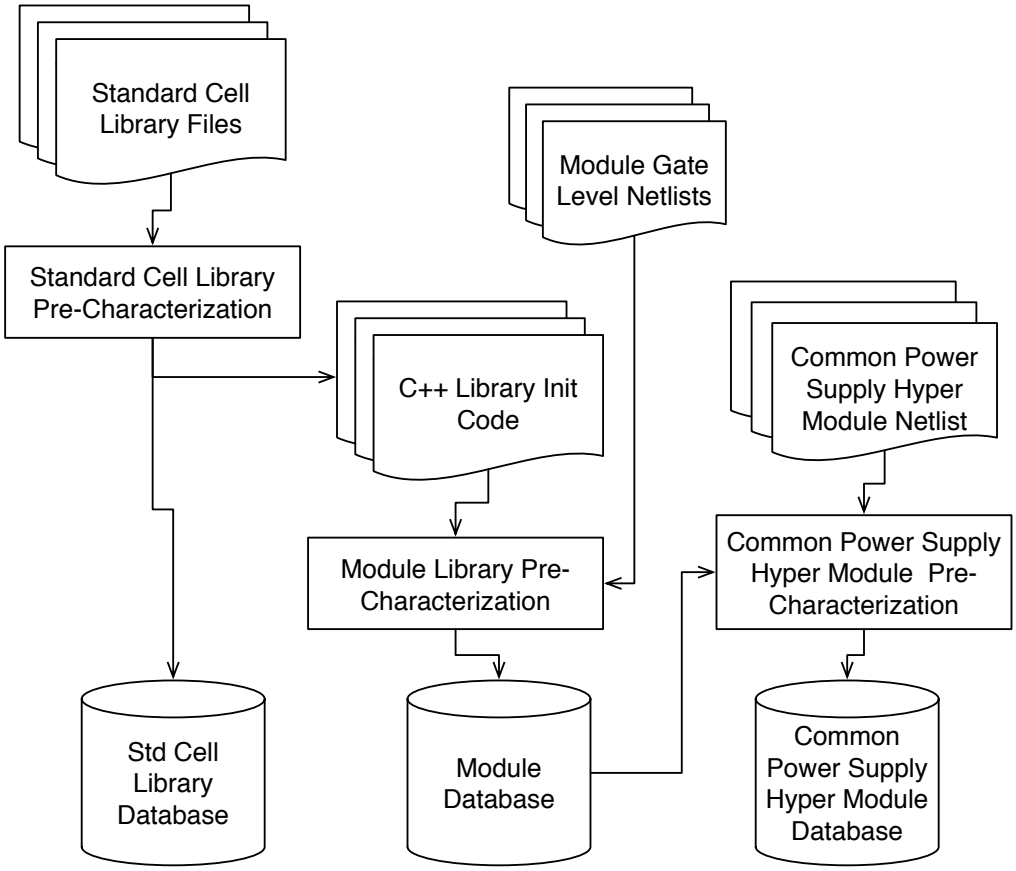
The following sub–sections describe in detail each of the aforementioned procedures. Note, that companies capable of designing and producing chips of the size targeted in this work have large computer clusters. All 3 pre–characterization steps are highly parallel, therefore the actual time required, before the design procedure starts, can be significantly reduced in an actual industrial environment.

3.2.1 Pre–Characterization of the Standard Cell Library

The Standard Cell Library Pre–Characterization step is summarized in Figure 3.5. The aim is to get the library of standard cells characterized under various operational condition of voltage, temperature, input slew and load capacitance. Traditional industrial tools, that perform standard cell library characterization, take as input the range of slew and load capacitance and perform the pre–characterization in a specific operational condition of voltage and temperature.

In order to overcome this, a separate Script Generator tool is developed. This tool takes as input, the transistor–level netlists of all standard cells, the device models, and the ranges of voltage, temperature, input slew and load capacitance. The output is a number of scripts to invoke the industrial standard cell library characterization tool, which in turn creates one .lib file per operational condition.

Figure 3.4: Overview of Pre–Characterization Step



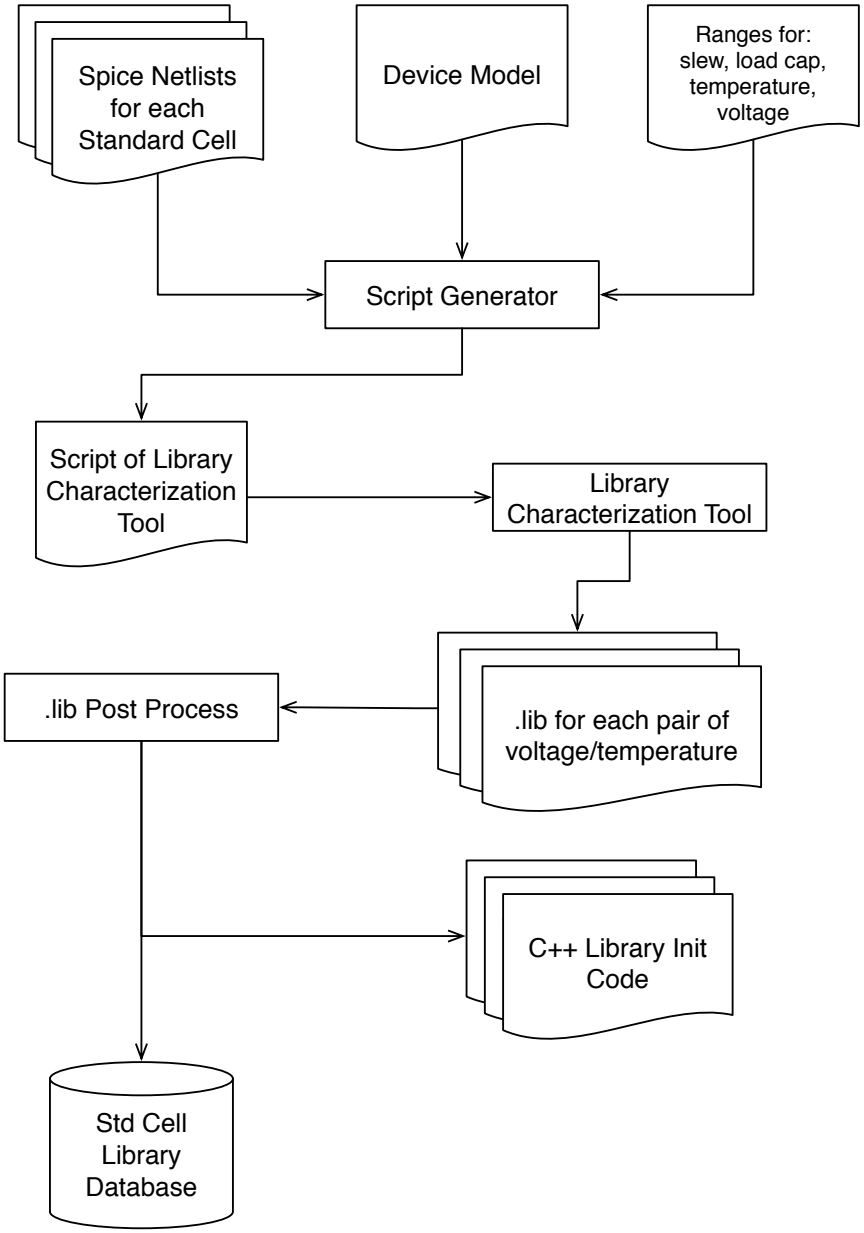
The collection of these .lib files, is fed into another tool, named .lib Post Process. This tool, merges for each cell the respective entries of the .lib file collection and produces C++ Library Init Code. The code contains the following:

1. Header files, one for each standard cell, that contain the timing and power information in the .lib files condensed for the entire range of voltage, temperature, load capacitance and input slew rate.

2. Code to fill a C++ class, for each standard cell, with the aforementioned information, to be used by the power / timing simulator for fast retrieval of information instead of using a database.

3. A C++ function for each standard cell, that calculates the logic function of the cell, with inputs the boolean state of cell inputs and outputs the boolean state of the cell outputs.

Figure 3.5: Standard Cell Library Pre–Characterization Procedure



4. A C++ function for each standard cell, that calculates the logic function of the cell, the arrival time and slew rate at the outputs, and power consumption per clock cycle, depending on state transition. This function is shown in algorithm 1.

Input: Vector of boolean inputs $\boldsymbol{I_b}$, Vector of arrival time of inputs $\boldsymbol{I_{time}}$, Vector of slew rate of inputs $\boldsymbol{I_{slew}}$,Vector of previous boolean outputs $\boldsymbol{O_b^{t-1}}$,Vector of output pin load capacitances $\boldsymbol{O_{loadcap}}$,Reference to overall power P, Voltage V, Temperature T, Pointer to gate structure G

Output: Vector of boolean outputs $\boldsymbol{O_b^t}$,Vector of arrival time of outputs $\boldsymbol{O_{time}}$, Vector of slew rate of inputs $\boldsymbol{O_{slew}}$

Calculate new outputs state $\boldsymbol{O_b^t} = f\left(\boldsymbol{I_b}, \boldsymbol{O_b^{t-1}}\right)$;
$P \mathrel{+}= G_{leakage}(V, T, \boldsymbol{I_b}, \boldsymbol{O_b^t}, \boldsymbol{O_b^{t-1}})$;
foreach *output* i **do**
  foreach *input* j **do**
    $risetime = G_{risetime}(V, T, \boldsymbol{O_{loadcap}}[i], \boldsymbol{I_{slew}}[j])$;
    $falltime = G_{falltime}(V, T, \boldsymbol{O_{loadcap}}[i], \boldsymbol{I_{slew}}[j])$;
    $\boldsymbol{O_{time}}[i] = max(\boldsymbol{I_{time}}[j] + max(risetime, falltime))$;
    $riseslew = G_{riseslew}(V, T, \boldsymbol{O_{loadcap}}[i], \boldsymbol{I_{slew}}[j])$;
    $fallslew = G_{fallslew}(V, T, \boldsymbol{O_{loadcap}}[i], \boldsymbol{I_{slew}}[j])$;
    $\boldsymbol{O_{slew}}[i] = max(riseslew, fallslew)$;
  end
end
foreach *input* i **do**
  if $\boldsymbol{O_b^t}[i] \neq \boldsymbol{O_b^{t-1}}[i]$ **then**
    if $\boldsymbol{O_b^t}[i] = true$ **then**
      $P \mathrel{+}= G_{risepower}(V, T, \boldsymbol{O_{loadcap}}, \boldsymbol{I_{slew}}[i])$;
    else
      $P \mathrel{+}= G_{fallpower}(V, T, \boldsymbol{O_{loadcap}}, \boldsymbol{I_{slew}}[i])$;
    end
  end
end

Algorithm 1: Single Standard Cell Simulation Function

Each of the aforementioned header files defines the following information:

1. A 1D array of voltage values, that the cell has been characterized into.
2. A 1D array of temperature values, that the cell has been characterized into.
3. A 1D array of slew values, that the cell has been characterized into.
4. A 1D array of load capacitance values, that the cell has been characterized into.

5. A map from a output pin state to a 2D lookup table of leakage power values, where, the first dimension corresponds to the voltage values, and the second to the temperature values.

6. For each input pin, a 2D lookup table of capacitance values, where, the first dimension corresponds to the voltage values, and the second to the temperature values.

7. For each input pin, a 4D lookup table of rise delay values, where, the first dimension corresponds to the voltage values, the second to the temperature values, the third to the slew values and the fourth to the load capacitance values.

8. For each input pin, a 4D lookup table of fall delay values, where, the first dimension corresponds to the voltage values, the second to the temperature values, the third to the slew values and the fourth to the load capacitance values.

9. For each input pin, a 4D lookup table of rise transition time values, where, the first dimension corresponds to the voltage values, the second to the temperature values, the third to the slew values and the fourth to the load capacitance values.

10. For each input pin, a 4D lookup table of fall transition time values, where, the first dimension corresponds to the voltage values, the second to the temperature values, the third to the slew values and the fourth to the load capacitance values.

11. For each input pin, a 4D lookup table of rise power time values, where, the first dimension corresponds to the voltage values, the second to the temperature values, the third to the slew values and the fourth to the load capacitance values.

12. For each output pin, a 4D lookup table of fall power time values, where, the first dimension corresponds to the voltage values, the second to the temperature values, the third to the slew values and the fourth to the load capacitance values.

This information in the aforementioned header files is used to fill a C++ class for each standard cell. This class provides the following functions:

1. Setter and getter functions for all the aforementioned information.

2. Function to retrieve static power as a function of power supply voltage, temperature and output state. 2D linear interpolation is used.

3. Function to retrieve input pin capacitance as a function of power supply voltage, temperature. 2D linear interpolation is used.

4. Function to retrieve rise and fall delay and transition time for each timing arc as a function of power supply voltage, temperature, load capacitance and input slew rate. 4D linear interpolation is used.

5. Function to retrieve rise and fall dynamic power for each timing arc as a function of power supply voltage, temperature, load capacitance and input slew rate. 4D linear interpolation is used.

Algorithm 1 shows how all the information for a standard cell, obtained during pre–characterization of the standard cell library, is used to calculate timing and power consumption for one gate. This is the basis of the power / timing simulator which is later described. It has as input, a vector of boolean inputs $\boldsymbol{I_b}$, a vector of arrival time of inputs $\boldsymbol{I_{time}}$, a vector of slew rate of inputs $\boldsymbol{I_{slew}}$, a vector of previous boolean outputs $\boldsymbol{O_b^{t-1}}$, a vector of output pin load capacitances $\boldsymbol{O_{loadcap}}$, a reference to overall power P, the power supply voltage V, the temperature T, and a pointer to the aforementioned standard cell C++ class G. Initially the new output pins state $\boldsymbol{O_b^t}$ is calculated from the logic function of the standard cell based on inputs $\boldsymbol{I_b}$ and previous output state $\boldsymbol{O_b^{t-1}}$. The leakage or static power is calculated as a function of voltage V, temperature T, new output state $\boldsymbol{O_b^t}$ and previous output state $\boldsymbol{O_b^{t-1}}$, and is added to the overall system power. The arrival time for each output, is calculated as the maximum of the sum of arrival time at the inputs, plus the delay of each timing arc. The slew rate for each output, is calculated as the maximum of slew rate caused by each timing arc. Both calculation are functions of voltage V, temperature T, load capacitance $\boldsymbol{O_{loadcap}}$ and input slew rate $\boldsymbol{I_{slew}}$. Finally, for each output, if a state transition is made, depending on the state transition the rise or fall dynamic power is calculated as a function of voltage V, temperature T, load capacitance $\boldsymbol{O_{loadcap}}$ and input slew rate $\boldsymbol{I_{slew}}$ and is added to the overall power P. Note, it is assumed that no intermediate state transitions take place.

3.2.2 Module Library Pre–Characterization

The Module Library Pre–Characterization step is summarized in Figure 3.6. Following the concept behind Standard Cell Library Pre–Characterization, the separate module types, that are instanced in Common Power Supply Hyper Modules, are pre–characterized as well, in terms of timing, power and physical form.

In order to acquire its physical form, each module is placed and routed, using a standard industrial place & route tool, for various aspect ratios, multiple times for each aspect ratio. Although current industrial place & route tools can only produce a 2D placement, the aspect ratio can also include the number of layers to use, once such tool is available. Performing place & route multiple times for the same aspect ratio, will result in different pin locations for the module.

Although this increases significantly the computational cost, the floorplanning procedure will be able to choose a solution that improves both area and overall timing.

In parallel with the place & route step, the C++ Standard Cell Library Initialization code, along with the module gate level netlist, and the ranges of voltage and temperature of the standard cell library, are fed into the Module Compiler, to produce the stimulus, the execution scripts and the executable that simulates the module and produces timing and power estimates. The Module Compiler is described in a later section.

When both the aformentioned steps are finished, each placed & routed entry is simulated. This comprises the following two steps. The entry is simulated for all pairs of voltage and temperature to get the power consumption and timing of the module, for the specific placement, under various operation conditions.

Finally, a post-process step gathers all information produced during the Module Library Pre–Characterization procedure and assembles it into the Module Database. The module database comprises entries for each module with the following information:

1. Input and output pins of the module.
2. Several placements.

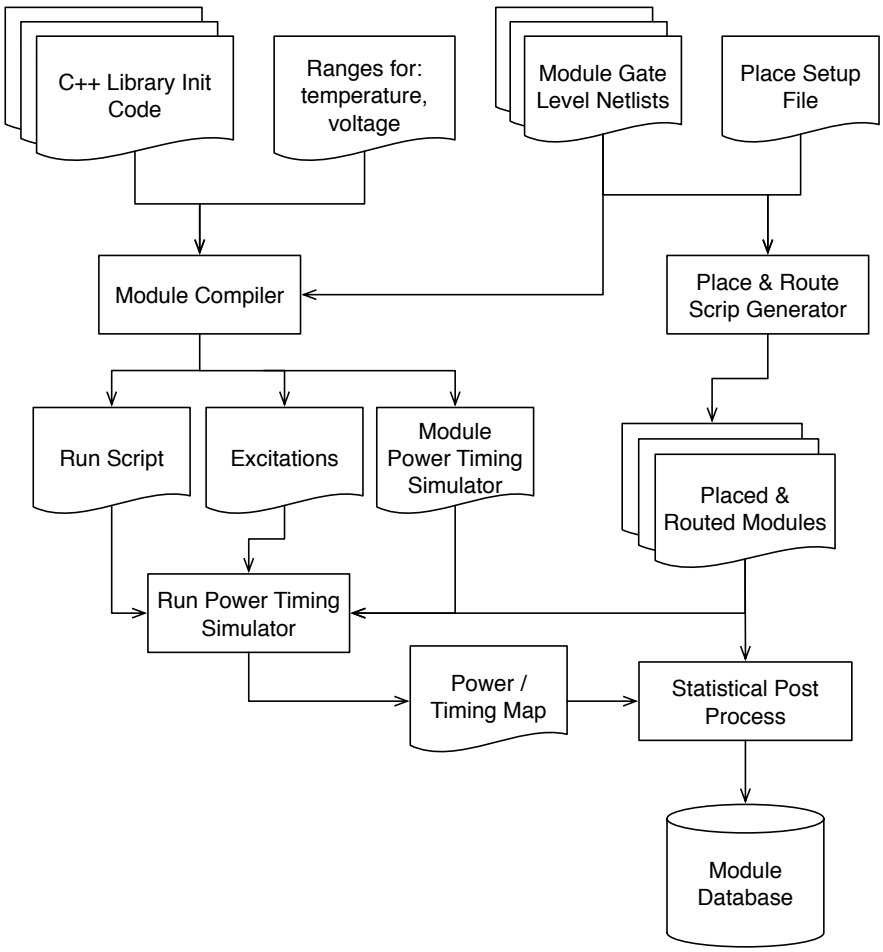
Each placement entry, within a module entry comprise the following information:

1. The width, length and number of layers of the placement.
2. Pin locations.
3. A 2D lookup table of maximum timing from inputs to outputs, where, the first dimension corresponds to the voltage values, and the second to the temperature values.
4. A 2D lookup table of maximum power consumption, where, the first dimension corresponds to the voltage values, and the second to the temperature values.

3.2.3 Pre–Characterization of Common Power Supply Hyper Modules

The C++ Standard Cell Library Initialization code, along with the flattened gate level netlist of the Common Power Supply Hyper Modules are fed into the Module Compiler, to produce the stimulus, the execution scripts and the executable that simulates the hyper module and produces timing and power

Figure 3.6: Module Library Pre–Characterization Procedure

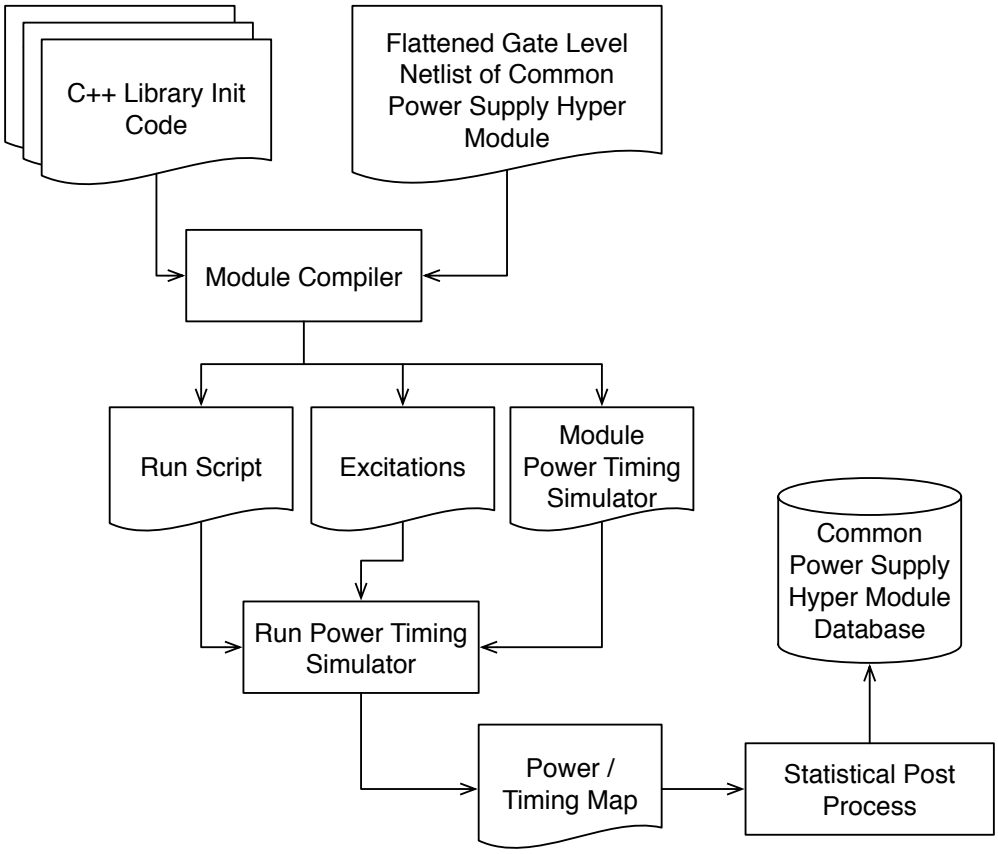


estimates. The Module Compiler is described in a later section. The circuit is simulated under nominal voltage and temperature in order to get the maximum power consumption distribution in each module. This process is summarized in figure 3.7.

3.3 Assembly Steps

After the pre–characterization step, solutions to the 3D floorplan problem are generated. This is a two–step procedure. Initially, modules are grouped and floorplanned to form Common Power Supply Hyper Modules (CPSHM). It will be experimentally shown that power supply voltage is the main limiting factor in systems exceeding 10 million gates. It is impossible to design a system at

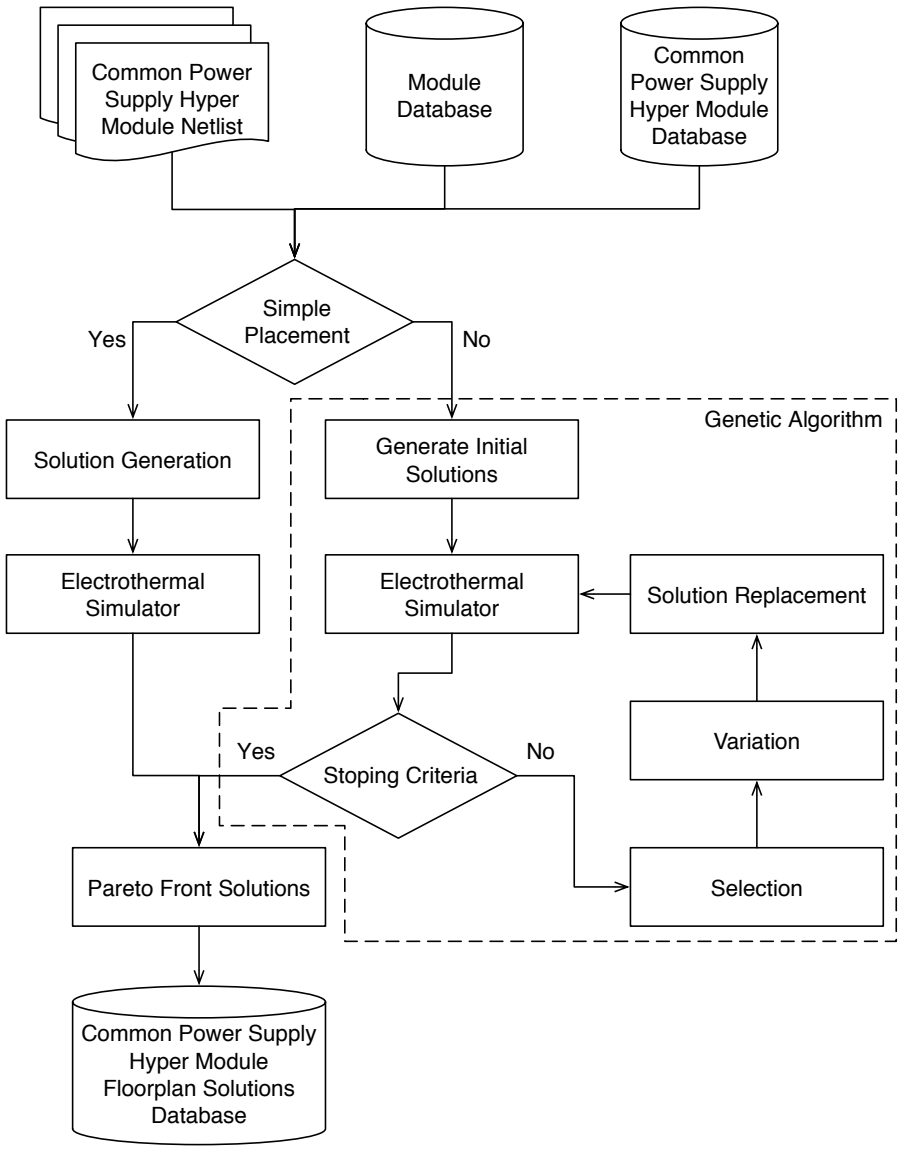
Figure 3.7: Common Power Supply Hyper Modules Pre–Characterization Procedure



the order of hundreds of million gates under one power supply. The final chip comprises several CPSHMs.

It is expected, that a system at the order of hundreds of million gates will be characterized by high redundancy. For example, many CPU cores, even more GPU cores, memory banks, video decoders, networking devices. Therefore, either of the two assembly step, may be a simple placement of the same block type with only variable the number of layers, or, it may be a placement of highly dissimilar blocks and require a sophisticated optimization algorithm.

Figure 3.8: Common Power Supply Hyper Module Assembly Step



3.3.1 Common Power Supply Hyper Module Assembly Step

Figure 3.8 shows the Common Power Supply Hyper Module Assembly Step. The inputs to this step are the CPSHM netlist, the Module Database and CPSHM database. It is reminded that the CPSHM database keeps the worst case power vectors between the modules of the CPSHM. Therefore, characterization of a floorplan solution with the Electrothermal Simulator, means characteriza-

tion for all worst case power vectors.

If the CPSHM comprises the same module types instanced multiple times, or a very small number of dissimilar modules, say 2 or 3, there is no need for an optimization procedure. In case of only one module type, the only variable is the number of layers. In case of two, for example a processor core and a memory block, solutions can be generated with one on top of the other, or to the side. In this case all possible solutions are generated and evaluated, and the pareto front solutions are stored to the Common Power Supply Hyper Module Floorplan Solutions Database.

If the CPSHM comprises a number of dissimilar modules, then a multi–objective optimization procedure is required. In this work, the use of a genetic algorithm is proposed along with a 3D floorplan representation. The proposed 3D floorplan representation is later described, while the evolution algorithm NSGA–II is described in [13]. In figure 3.8, within the dashed area, only the main steps of the evolution algorithm are shown.

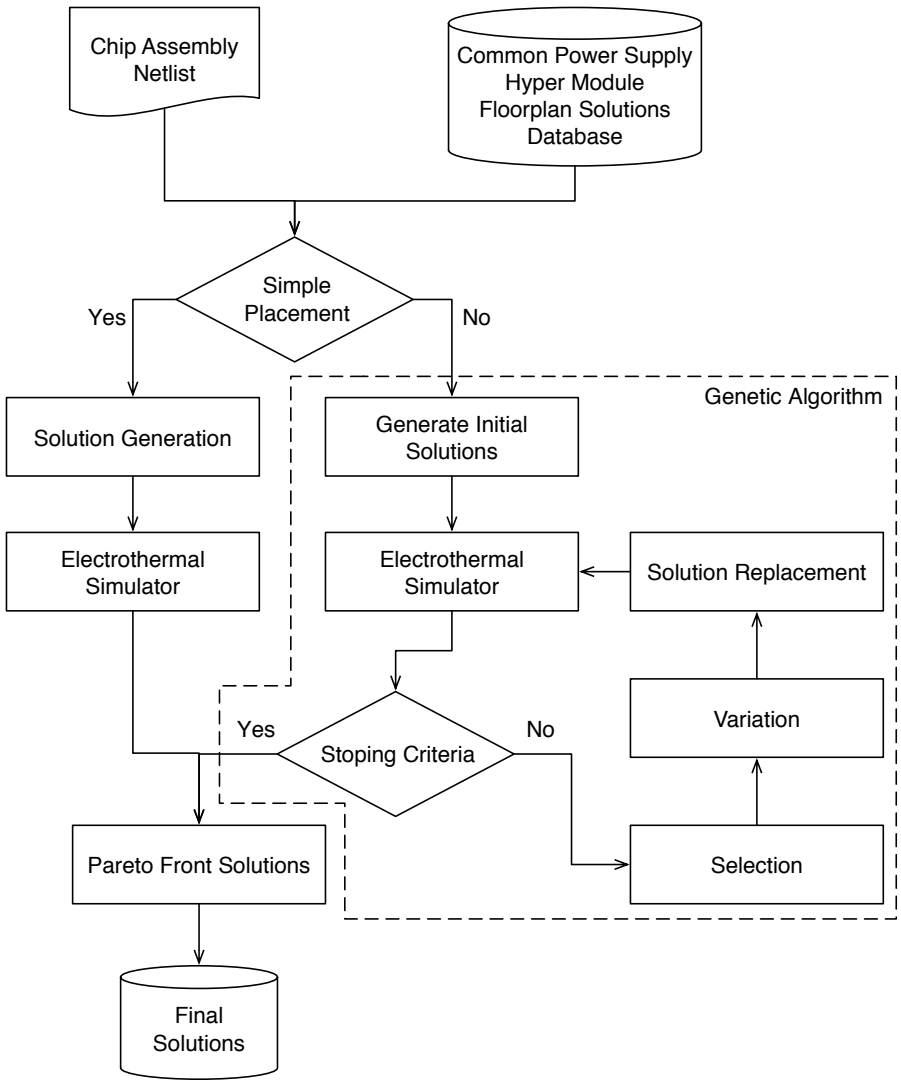
3.3.2 Chip Assembly Step

Figure 3.9 shows the final Chip Assembly Step. The inputs to this step are the Chip netlist and CPSHM Floorplan Solutions Database. In chip assembly, it is assumed that all CPSHMs may have peak power consumption at the same time. Also it is generally assumed that all CPSHMs will have the same number of layers. If not, and therefore some may be stacked, it is expected that system timing will worsen due to temperature increase, but power consumption will not seriously affected, since each CPSHMs is under its own power supply.

If the chip comprises the same module types instanced multiple times, or a very small number of dissimilar modules, say 2 or 3, there is no need for an optimization procedure. In case of only one module type, the only variable is the number of vertically stacked CPSHMs. In case of two, for example a multi–core processor core and a RAM block, solutions can be generated with one on top of the other, or to the side. In this case all possible solutions are generated and evaluated and the pareto front solutions are returned to the user.

If the chip comprises a number of dissimilar modules, then a multi–objective optimization procedure is required. In this work, the use of a genetic algorithm is proposed along with a 3D floorplan representation. If the CPSHMs comprise the same number of layers and further stacking is not required, then traditional 2D floorplan representations may be used. The proposed 3D floorplan representation is later described, while the evolution algorithm NSGA–II is described in [13]. In figure 3.9, within the dashed area, only the main steps of the evolution algorithm are shown.

Figure 3.9: Chip Assembly Step



3.4 Statistical Prediction Engine

3.4.1 Univariate Statistical Prediction Engine

The univariate statistical prediction engine is a procedure to identify the worst case estimate of a random variable from a small list of observations. The procedure used in this work is described in [9]. This procedure fits the *maximal generalized extreme value distribution* (GEVD) that has two discrete cases, $k = 0$ and $k \neq 0$.

Algorithm 2 shows the process of obtaining the worst case estimate from a

sample of observations. Initially the samples are partitioned into groups of user defined size. The maxima of each group consist the new sample that is used to fit the GEVD. QLS is used to obtain a initial estimate of the distribution parameters with a user defined number of samples, as described in [9], for the two cases of $k = 0$ and $k \neq 0$. $\theta_0^{k\neq 0} = (\lambda_0, \delta_0, \kappa_0)$ and $\theta_0^{k=0} = (\lambda_0, \delta_0)$ are obtained and used to get the Maximum Likelihood Estimates $\hat{\theta}_{k\neq 0} = (\hat{\lambda}, \hat{\delta}, \hat{\kappa})$ and $\hat{\theta}_{k=0} = (\hat{\lambda}, \hat{\delta})$. The likelihood ratio test is applied to see whether $k = 0$ or $k \neq 0$. The worst case estimate $\hat{x}_p$ and variance $\sigma^2_{\hat{x}_p}$ are obtained from the respective equation. Finally, the $(1 - \alpha)100\%$ confidence interval for x_p is returned as

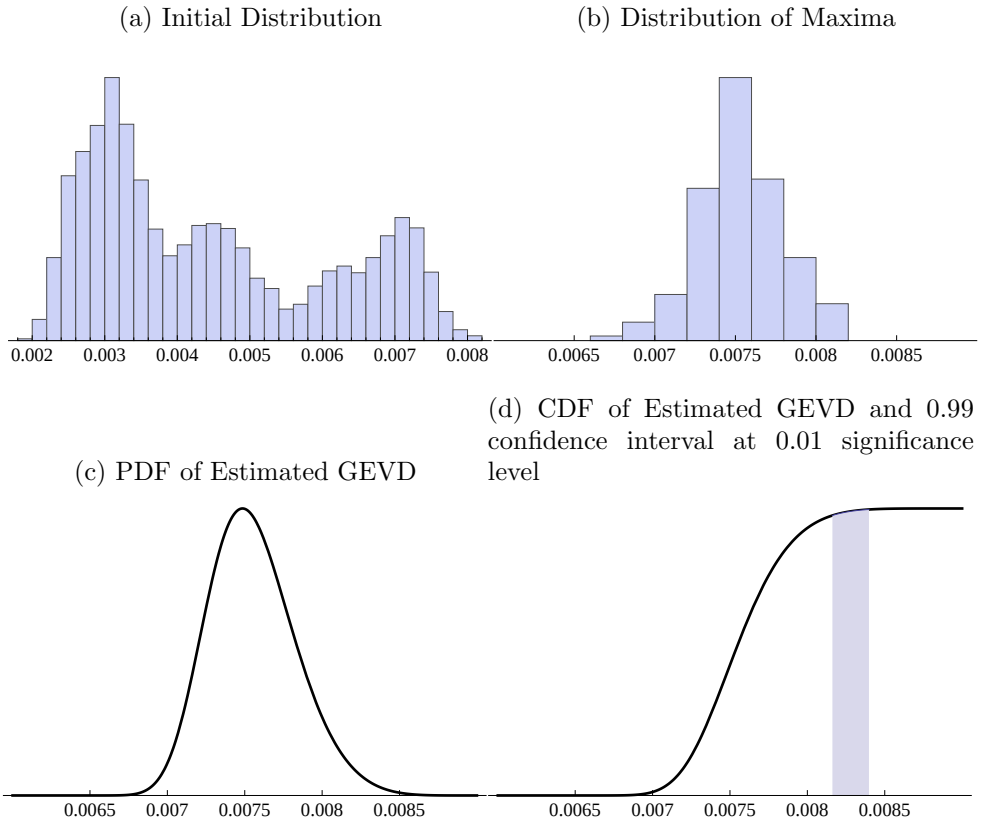
$$x_p \in \left(\hat{x}_p \pm z_{\alpha/2}\, \hat{\sigma}_{\hat{x}_p}\right) \tag{3.1}$$

Input: data, maximaSampleSize, pQuantile, significanceLevel, samples4QLS
Output: quantile, confidenceInterval
partitions = Partition[data,maximaSampleSize];
samplesOfMaxima = Apply[Max,partitions];
initEstimateKappaZero from (3.23);
initEstimateKappaNotZero from (3.6);
estimateKappaZero from (3.25);
estimateKappaNotZero from (3.9);
logLikelihoodKappaZero from (3.24);
logLikelihoodKappaNotZero from (3.7);
Apply Test (3.37);
if *passes test* **then**
    quantile from (3.18);
    confidenceInterval from (3.22);
else
    quantile from (3.30);
    confidenceInterval from (3.34);
end

Algorithm 2: Univariate Statistical Prediction Engine

Figure 3.10 shows an example of this procedure. Figure 3.10a shows the initial histogram. Figure 3.10b shows the histogram of maxima, obtained by partitioning the initial distribution into samples of 30 and taking the maximum of each. Figure 3.10c shows the PDF of the estimated GEVD from the sample of maxima. Figure 3.10d shows the CDF of the estimated GEVD and in grey, the 0.99 confidence interval at 0.01 significance level. The maximum estimate is the upper part of the confidence interval.

Figure 3.10: Example application of Univariate Statistical Prediction Engine



The rest of this paragraph describes the mathematical background of algortithm 2 from [9]. For a more detailed analysis and treatment of univariate extreme value theory see [9], [26], [22], [41] and [6].

Maximal Generalized Extreme value Distribution

The pdf of the *maximal generalized extreme value distribution* (GEVD) for $k \neq 0$ is given by

$$f(x;\lambda,\delta,\kappa) = \frac{1}{\delta}\exp\left[-\left[1-\kappa\left(\frac{x-\lambda}{\delta}\right)\right]^{1/\kappa}\right]\left[1-\kappa\left(\frac{x-\lambda}{\delta}\right)\right]^{1/\kappa-1} \tag{3.2}$$

where the support is $x \leq \lambda + \delta/\kappa$, if $\kappa > 0$, or $x \geq \lambda + \delta/\kappa$, if $\kappa < 0$. For $k = 0$ the pdf is

$$f(x;\lambda,\delta) = \frac{1}{\delta}\exp\left[-\exp\left(\frac{\lambda-x}{\delta}\right)\right]\exp\left(\frac{\lambda-x}{\delta}\right) \tag{3.3}$$

and the cdf by

$$F(x;\lambda,\delta,\kappa) = \begin{cases} \exp\left[-\left[1-\kappa\left(\frac{x-\lambda}{\delta}\right)\right]^{1/\kappa}\right] & 1-\kappa\left(\frac{x-\lambda}{\delta}\right) \geq 0, k \neq 0 \\ \exp\left[-\exp\left(\frac{\lambda-x}{\delta}\right)\right] & -\infty < x < +\infty, \kappa = 0 \end{cases} \tag{3.4}$$

The corresponding p-quantile is

$$x_p = \begin{cases} \lambda + \delta\left[1-(-\log p)^{\kappa}\right]/\kappa, & \kappa \neq 0 \\ \lambda - \delta\log\left(-\log p\right), & \kappa = 0 \end{cases} \tag{3.5}$$

The Case When $\kappa \neq 0$

The QLS method parameter estimates are the solutions of the minimization problem

$$Minimize(\lambda,\delta,\kappa) \quad \sum_{i=1}^{n}\{x_{i:n} - \lambda - \delta[1-(-\log p_{i:n})^{\kappa}]/\kappa\} \tag{3.6}$$

The log-likelihood functions becomes

$$\ell = -n\log\delta - (1-k)\sum_{i=1}^{n} z_i - \sum_{i=1}^{n}\exp(-z_i) \tag{3.7}$$

where

$$z_i = -\frac{1}{\kappa}\log\left[1-\kappa\left(x_i-\lambda\right)/\delta\right] \tag{3.8}$$

The maximum likelihood estimate of $\theta = (\lambda,\delta,\kappa)$ can be obtained by the iterative formula:

$$\theta_{j+1} = \theta_j + {\mathbf{I}_\theta}^{-1}\nabla_\theta\ell, \quad j = 0,1,\ldots \tag{3.9}$$

where θ_0 is an initial estimate and ${\mathbf{I}_\theta}^{-1}$ and $\nabla_\theta\ell$ are evaluated at θ_j. Here, $\nabla_\theta\ell$ is the gradient vector of ℓ with respect to θ and

$$\mathbf{I}_\theta = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \tag{3.10}$$

is the Fisher Information Matrix. The elements of $\mathbf{I}_\theta$ are given by:

$$m_{11} = E\left(-\frac{\partial^2 \ell}{\partial \lambda^2}\right) = \frac{n}{\delta^2} p$$

$$m_{22} = E\left(-\frac{\partial^2 \ell}{\partial \delta^2}\right) = \frac{n}{\delta^2 \kappa^2}\left[1 - 2\Gamma(2-\kappa) + p\right]$$

$$m_{33} = E\left(-\frac{\partial^2 \ell}{\partial \kappa^2}\right) = \frac{n}{\kappa^2}\left[\frac{\pi^2}{6} + \left(1 - \gamma - \frac{1}{\kappa}\right)^2 + \frac{2q}{\kappa} + \frac{p}{\kappa^2}\right]$$

$$m_{12} = m_{21} = E\left(-\frac{\partial^2 \ell}{\partial \lambda \partial \delta}\right) = \frac{n}{\delta^2 \kappa}\left[p - \Gamma(2-\kappa)\right]$$

$$m_{13} = m_{31} = E\left(-\frac{\partial^2 \ell}{\partial \lambda \partial \kappa}\right) = -\frac{n}{\delta \kappa}\left(q + \frac{p}{\kappa}\right)$$

$$m_{23} = m_{32} = E\left(-\frac{\partial^2 \ell}{\partial \delta \partial \kappa}\right) = \frac{n}{\delta \kappa^2}\left[1 - \gamma - \frac{1 - \Gamma(2-\kappa)}{\kappa} - q - \frac{p}{\kappa}\right]$$

where

$$\Gamma(u) = \int_0^\infty y^{u-1} e^{-y} dy \tag{3.11}$$

is the Gamma function,

$$\psi(u) = \frac{\partial \log \Gamma(u)}{\partial u} \tag{3.12}$$

is the Psi function,

$$p = (1-\kappa)^2 \Gamma(1-2\kappa) \tag{3.13}$$

$$q = \Gamma(2-\kappa)[\psi(1-k) - (1-\kappa)/\kappa] \tag{3.14}$$

and $\gamma = 0.5772157$ is the Euler's constant. The regularity conditions are satisfied only for $\kappa < 0.5$, and, in this case the asymptotic variances and covariances are given by

$$\boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}} = \boldsymbol{I}_{\boldsymbol{\theta}}{}^{-1} \tag{3.15}$$

When $\hat{\theta}$ is substituted for θ in (3.15), we obtain an estimate of the covariance matrix

$$\hat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\theta}}} = \boldsymbol{I}_{\boldsymbol{\theta}}{}^{-1}|_{\theta=\hat{\theta}} \tag{3.16}$$

Let $\theta = \{\lambda, \delta, \kappa\}$. Obtain the MLE of θ by maximizing the log-likelihood function in (3.7). The Fisher information matrix is the matrix $\boldsymbol{I}_\theta$ in (3.10),

the inverse of which is evaluated at $(\hat{\lambda}, \hat{\delta}, \hat{\kappa})$, is the estimated covariance matrix of $(\hat{\lambda}, \hat{\delta}, \hat{\kappa})$. The square root of the diagonal elements of this matrix are the standard errors, $(\hat{\sigma}_{\hat{\lambda}}, \hat{\sigma}_{\hat{\delta}}, \hat{\sigma}_{\hat{\kappa}})$, of the estimates $(\hat{\lambda}, \hat{\delta}, \hat{\kappa})$, respectively. Accordingly, the $(1-\alpha)100\%$ confidence intervals for the parameters are:

$$\begin{aligned} \lambda &\in \left(\hat{\lambda} \pm z_{\alpha/2}\, \hat{\sigma}_{\hat{\lambda}}\right) \\ \delta &\in \left(\hat{\delta} \pm z_{\alpha/2}\, \hat{\sigma}_{\hat{\delta}}\right) \\ \kappa &\in \left(\hat{\kappa} \pm z_{\alpha/2}\, \hat{\sigma}_{\hat{\kappa}}\right) \end{aligned} \tag{3.17}$$

The quantiles can be estimated by substituting the parameter estimates in (3.5) and obtaining

$$\hat{x}_p = \hat{\lambda} + \hat{\delta}\left[1 - (-\log p)^{\hat{\kappa}}\right]/\hat{\kappa} \tag{3.18}$$

The asymptotic variance of $\hat{x}_p$ using the Delta method is

$$\sigma^2_{\hat{x}_p} \approx \nabla^T_{\hat{\theta}}\, x_p \Sigma_{\hat{\theta}} \nabla_{\theta} x_p \tag{3.19}$$

where $\Sigma_{\hat{\theta}}$ is the asymptotic covariance matrix of $\hat{\theta} = (\hat{\lambda}, \hat{\delta}, \hat{\kappa})$ in (3.15) and

$$\nabla_{\theta} x_p = \begin{bmatrix} \frac{\partial x_p}{\partial \lambda} \\ \frac{\partial x_p}{\partial \delta} \\ \frac{\partial x_p}{\partial \kappa} \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1-(-\log p)^{\kappa}}{\kappa} \\ -\frac{\delta[1-(-\log p)^{\kappa}]}{\kappa^2} - \frac{\delta}{\kappa}(-\log p)^{\kappa}\log(-\log p) \end{bmatrix} \tag{3.20}$$

The estimated asymptotic variance of $\hat{x}_p$ is computed by evaluating $\Sigma_{\hat{\theta}}$ and $\nabla_{\theta} x_p$ at $\hat{\theta} = (\hat{\lambda}, \hat{\delta}, \hat{\kappa})$ and obtaining

$$\hat{\sigma}^2_{\hat{x}_p} \approx \nabla^T_{\hat{\theta}}\, x_p \Sigma_{\hat{\theta}} \nabla_{\hat{\theta}} x_p \tag{3.21}$$

Accordingly, a $(1-\alpha)100\%$ confidence interval for x_p is given by

$$x_p \in \left(\hat{x}_p \pm z_{\alpha/2}\, \hat{\sigma}_{\hat{x}_p}\right) \tag{3.22}$$

The Case When $\kappa = 0$

The QLS method parameter estimates are the solutions of the minimization problem

$$Minimize(\lambda, \delta, \kappa) \quad \sum_{i=1}^{n} [x_{i:n} - \lambda + \delta \log(-\log p_{i:n})] \tag{3.23}$$

The log-likelihood functions becomes

$$\ell = -n \log \delta - \sum_{i=1}^{n} \exp\left[-\left(\frac{x_i - \lambda}{\delta}\right)\right] - \sum_{i=1}^{n}\left(\frac{x_i - \lambda}{\delta}\right) \tag{3.24}$$

The maximum likelihood estimate of $\theta = (\lambda, \delta)$ can be obtained by the iterative formula:

$$\theta_{j+1} = \theta_j + \mathbf{I}_\theta{}^{-1} \nabla_\theta \ell, \quad j = 0, 1, \ldots \tag{3.25}$$

where θ_0 is an initial estimate and $\mathbf{I}_\theta{}^{-1}$ and $\nabla_\theta \ell$ are evaluated at θ_j. Here, $\nabla_\theta \ell$ is the gradient vector of ℓ with respect to θ and

$$\mathbf{I}_\theta = \begin{bmatrix} \frac{n(1-\kappa)^2\Gamma(1-2\kappa)}{\delta^2} & \frac{n(\gamma-1)}{\delta^2} \\ \frac{n(\gamma-1)}{\delta^2} & \frac{n(6-12\gamma+6\gamma^2+n^2)}{6\delta^2} \end{bmatrix} \tag{3.26}$$

is the Fisher Information Matrix.

In this case, the regularity conditions are satisfied, and, the asymptotic variances and covariances are given by

$$\boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}} = \boldsymbol{I}_{\boldsymbol{\theta}}{}^{-1} \tag{3.27}$$

When $\hat{\theta}$ is substituted for θ in (3.27), we obtain an estimate of the covariance matrix

$$\hat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\theta}}} = \boldsymbol{I}_{\boldsymbol{\theta}}{}^{-1}|_{\theta=\hat{\theta}} \tag{3.28}$$

Let $\theta = \{\lambda, \delta\}$. Obtain the MLE of θ by maximizing the log-likelihood function in (3.7). The Fisher information matrix is the matrix $\boldsymbol{I}_\theta$ in (3.10), the inverse of which is evaluated at $(\hat{\lambda}, \hat{\delta})$, is the estimated covariance matrix of $(\hat{\lambda}, \hat{\delta})$. The square root of the diagonal elements of this matrix are the standard errors, $(\hat{\sigma}_{\hat{\lambda}}, \hat{\sigma}_{\hat{\delta}})$, of the estimates $(\hat{\lambda}, \hat{\delta})$, respectively. Accordingly, the $(1-\alpha)100\%$ confidence intervals for the parameters are:

$$\begin{aligned} \lambda &\in \left(\hat{\lambda} \pm z_{\alpha/2}\, \hat{\sigma}_{\hat{\lambda}}\right) \\ \delta &\in \left(\hat{\delta} \pm z_{\alpha/2}\, \hat{\sigma}_{\hat{\delta}}\right) \end{aligned} \tag{3.29}$$

The quantiles can be estimated by substituting the parameter estimates in (3.5) and obtaining

$$\hat{x}_p = \hat{\lambda} - \hat{\delta} \log\left(-\log p\right) \tag{3.30}$$

The asymptotic variance of $\hat{x}_p$ using the Delta method is

$$\sigma^2_{\hat{x}_p} \approx \nabla_\theta^T x_p \Sigma_{\hat{\theta}} \nabla_\theta x_p \tag{3.31}$$

where $\Sigma_{\hat{\theta}}$ is the asymptotic covariance matrix of $\hat{\theta} = (\hat{\lambda}, \hat{\delta})$ in (3.15) and

$$\nabla_\theta x_p = \begin{bmatrix} \frac{\partial x_p}{\partial \lambda} \\ \frac{\partial x_p}{\partial \delta} \end{bmatrix} = \begin{bmatrix} 1 \\ -\log(-\log p) \end{bmatrix} \tag{3.32}$$

The estimated asymptotic variance of $\hat{x}_p$ is computed by evaluating $\Sigma_{\hat{\theta}}$ and $\nabla_\theta x_p$ at $\hat{\theta} = (\hat{\lambda}, \hat{\delta})$ and obtaining

$$\hat{\sigma}^2_{\hat{x}_p} \approx \nabla^T_{\hat{\theta}} x_p \Sigma_{\hat{\theta}} \nabla_{\hat{\theta}} x_p \tag{3.33}$$

Accordingly, a $(1-\alpha)100\%$ confidence interval for x_p is given by

$$x_p \in \left(\hat{x}_p \pm z_{\alpha/2}\, \hat{\sigma}_{\hat{x}_p}\right) \tag{3.34}$$

Likelihood Ratio Test

In the GEVD, we wish to test

$$H_0 : \kappa = 0 \quad versus \quad H_1 : \kappa \neq 0 \tag{3.35}$$

based on a data set $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ The log-likelihood, $\ell(\boldsymbol{x}, \boldsymbol{\theta})$, is a function of $\boldsymbol{\theta} = (\lambda, \delta, \kappa)$. Let $\hat{\boldsymbol{\theta}}_{\mathbf{0}} = (\hat{lambda}_0, \hat{\delta}_0, 0)$ and $\hat{\boldsymbol{\theta}}_{\mathbf{1}} = (\hat{lambda}_1, \hat{\delta}_1, \hat{\kappa}_1)$ be the maximum likelihood estimates of $\boldsymbol{\theta}$ under H_0 and H_1, respectively. The modified Likelihood Ratio Test compares the log-likelihood evaluated at $\hat{\boldsymbol{\theta}}_0$ with the log-likelihood evaluated at $\hat{\boldsymbol{\theta}}_1$, that is, it compares $\ell(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_0)$ with $\ell(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_1)$. Specifically the modified Likelihood Ratio Test is given by

$$LR^* = 2\left(1 - \frac{2.8}{n}\right)\left[\ell(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_1) - \ell(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_0)\right] \tag{3.36}$$

Under H_0, LR is a chi-square distribution with 1 degree of freedom. This H_0 is rejected at the significance level α if

$$LR^* > x^2_1(1-\alpha) \tag{3.37}$$

where $x^2_1(1-\alpha)$ is the $(1-\alpha)$ quantile of the chi-square distribution with 1 degree of freedom.

3.4.2 Multivariate Statistical Prediction Engine

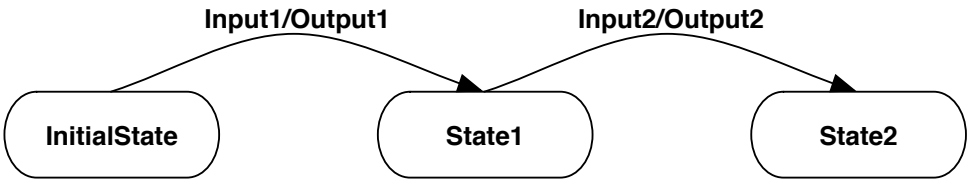
Multivariate Statistical Prediction Engine is a placeholder since only literature review was made in this area due to time limitations. Methods to accomplish this task are described in [42] and [25], where a detailed treatment of multivariate extreme value theory using copulas is provided.

3.5 Power / Timing Simulator

3.5.1 Simulator Description

The simulator described here, is used to calculate power consumption per clock cycle, and perform static timing analysis, under user defined operating conditions of power supply voltage and temperature. Note, that due to time limitations, the effect of interconnects is not taken into account in this work. However, it is straight forward to extend this methodology to account for interconnects. Given the initial state vector of all sequential elements, 2 input vectors, operating voltage and temperature, the circuit is simulated over 2 clock cycles, as shown in Figure 3.11. During the second cycle, along with logic simulation, the static and dynamic power consumption, and the time from inputs to outputs is calculated as a function of operating voltage, temperature, and previous and new states of all standard cells. The reason for simulating over two clock cycles, is to have the previous and new state of combinatorial elements, in order to calculate leakage power. All calculations are based on the pre–characterized standard cell library.

Figure 3.11: Simulation Cycle



The output of the circuit simulator is power consumption for each excitation, and max time from inputs to outputs. To get a realistic estimate of the circuit's power consumption, the circuit is simulated for a small number of excitations, compared to the number of all possible excitations. The power output is fed into the univariate statistical prediction engine, which predicts the worst case power consumption over all possible excitations.

Another mode of the simulator, takes an CPSHM netlist and reports power per module. The output of this mode is fed into the multivariate statistical prediction engine, to get worst case power distributions between modules. However, due to time limitations, this mode is not developed.

During the second cycle, algorithm 1 is used to calculate power consumption and timing for each standard cell. The static power is calculated as the max of the previous and new state and is added to the overall system power consumption. For each input pin, depending on the state transition of the cell, dynamic power consumption is calculated. The dynamic power is added to the

overall system power consumption. For each output pin, the arrival time at the outputs, is calculated as the maximum of arrival time at the inputs, plus maximum delay of each input pin.

Overall power consumption is calculated as the sum of power consumption of each circuit element. Maximum timing from inputs to outputs is the maximum time on all timing paths of the circuit. From [7], there are 4 types of timing paths:

1. From inputs to outputs, through combinatorial elements only. Timing in this type of paths is calculated as the sum of delays in each combinatorial element.

2. From inputs to sequential element inputs, through combinatorial elements. Timing in this type of paths is calculated as the sum of delays in each combinatorial element, plus the setup and hold time of the starting sequential elements.

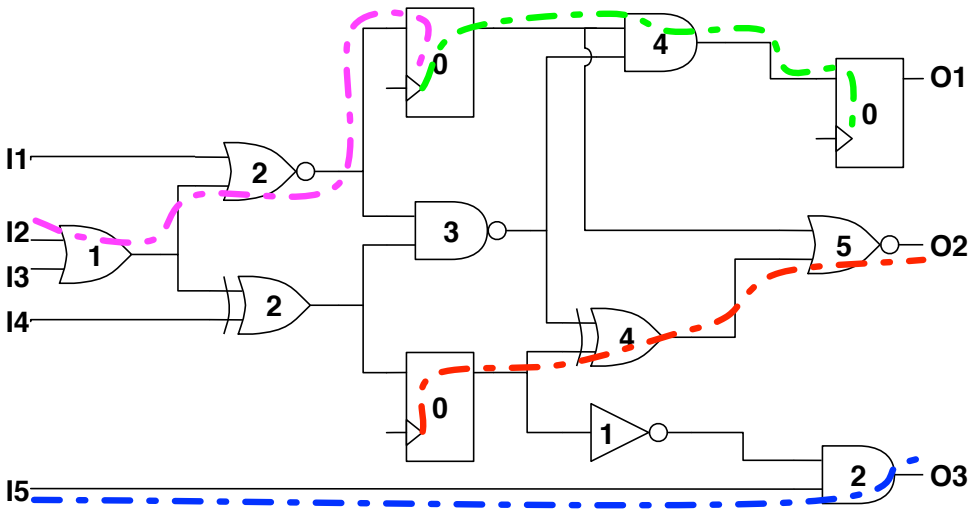
3. From sequential element outputs to outputs, through combinatorial elements. Timing in this type of paths is calculated as the sum of delays in each combinatorial element, plus the propagation delay of the arriving sequential elements.

4. From sequential element outputs to sequential element inputs, through combinatorial elements. Timing in this type of paths is calculated as the sum of delays in each combinatorial element, plus the setup and hold time of the arriving sequential elements and the propagation delay of the starting sequential elements.

Examples of these paths are shown in figure 3.12. To perform the logic simulation and calculate timing, standard cells must be evaluated in a specific order. Each sequential element is assigned order 0. Each combinatorial element is assigned order 1 if all its inputs are inputs of the circuit or the maximum order of its inputs plus 1. Figure 3.12 shows an example of this ordering. First, combinatorial elements are evaluated in ascending order. At the end, sequential elements are evaluated.

Algorithm 3 shows the core simulator function. Initially, the capacitances of all nets are calculated, since these are needed in order to calculate power, delay and slew. Then, the simulator goes into a loop over all given excitations. Circuit input signal and sequential elements state are assigned to the respective excitation. During the first cycle, a signal only carries its boolean state. Combinatorial elements are evaluated in ascending order. After this point, each signal carries, except for its boolean state, the boolean state at the previous clock cycle, the arrival time and slew rate. At the end of the first clock cycle, sequential elements are evaluated according to the algorithm 1. The second clock cycle begins and combinatorial elements are evaluated according to the algorithm 1

Figure 3.12: Gate Evaluation Order and Timing Arcs



in ascending order. At the end of the second clock cycle, sequential elements are evaluated according to the algorithm 1. The maximum arrival time of all output signals is taken and reported along the the power consumption during the second cycle.

3.5.2 Module Compiler

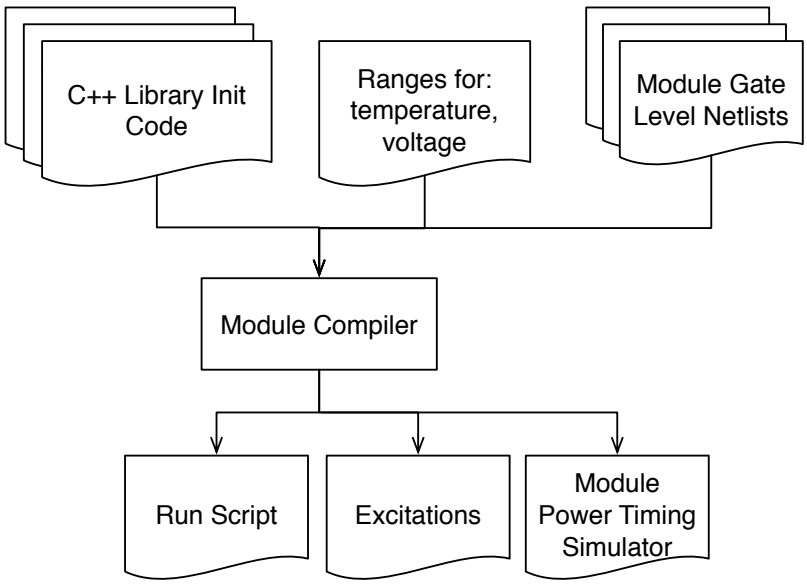
The module compiler is used to speed up the aforementioned simulation process, by compiling an executable for each module. The process is summarized in Figure 3.13. It takes as input the gate level netlist of the module, the C++ Library Init Code produced during Standard Cell Library Pre–Characterization Procedure, the ranges of temperature and voltage and the number of excitations to simulate the circuit. It produces C++ code that, when compiled to binary executable, simulates the specific circuit as described in the previous sub–section. It also produces the necessary scripts to run the Module Pre–Characterization Step in terms of power consumption and timing.

One important aspect of the Module Compiler is the ability to produce code from huge circuits that can compiled under current computing resources. Since the number of operations is proportional to the gate count of the circuit, the module compiler has a parameter to group operations and therefore control the function and source file size and number, which have a direct impact on the memory consumption of the compiler per source file, as well as the limitations on the number of files that can be linked in a final executable.

Input: Input Pair Vectors, Initial State Vectors, Voltage, Temperature
Output: Timing, Power Consumption per Excitation
Data: Net Capacitances, Net Logic States in Cycle 1, Net Logic States in Cycle 2, Arrival Time & Slew at each Input Pin
Calculate Net Capacitances;
forall the *excitations* **do**
    Clear Overal Power;
    Assign Inputs From First Input Vector;
    Assign Flip Flop State 1;
    Calculate Gate State 1;
    Assign Inputs From Second Input Vector;
    Calculate New Flip Flop State;
    Calculate New Gate State / Arrival Time & Slew;
    Calculate Final Flip Flop State;
    Calculate Max Timing of All Outputs;
    Report Timing / Power;
end

Algorithm 3: Core Power / Timing Simulator Function

Figure 3.13: Module Compiler



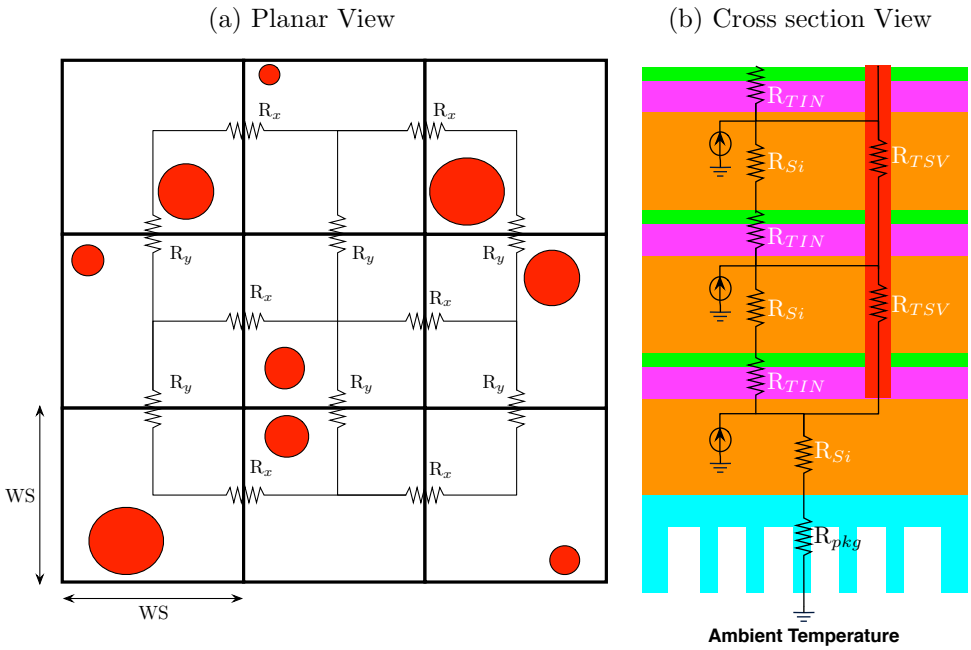
3.6 Adaptive Electrothermal Simulator

3.6.1 Thermal Model

The compact resistive thermal model of [44] is extended to capture the 3rd dimension of 3D ICs. The space of the chip is divided into cuboidal mesh. Mesh

cells have equal width and height WS, and thickness equal to the thickness of substrate plus the dielectrics and the glue layers. In each mesh cell, α TSV density exists. Figure 3.14a shows the planar view of the model. Red circles represent different TSV density within each mesh cell. Figure 3.14b shows the cross section of a mesh cell stack across all dies and package. Figure 3.15 shows the complete resistive thermal model. Note that parallel and series resistors have been reduced.

Figure 3.14: Planar and Cross section View of the Compact Thermal Model



From [44] the thermal resistance is calculated as

$$R = \frac{l}{kA} \tag{3.38}$$

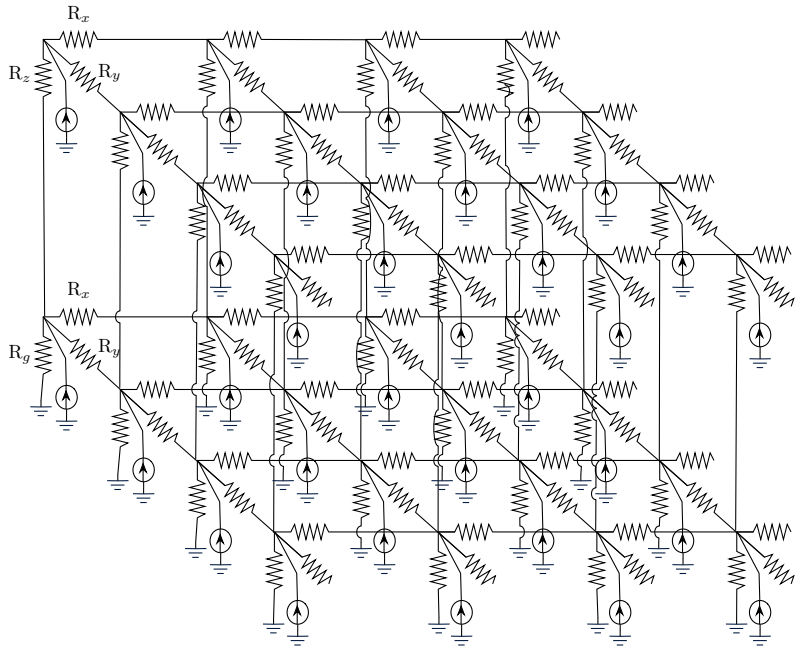
where l is the length of the mesh cell, k the thermal conductivity of the material and A the cross section area of the mesh cell.

The vertical thermal resistance through the TSV density is

$$R_{TSV} = \frac{t_{Si} + t_{glue} + t_{ILD}}{k_{Cu}\alpha WS^2} \tag{3.39}$$

where t_{Si} is the thickness of the silicon layer, t_{glue} the thickness of the glue layer, t_{ILD} the thickness of the dielectrics, k_{Cu} the thermal conductivity of Cu, α the TSV density, and WS the mesh cell width and height.

Figure 3.15: Compact Thermal Model



The vertical thermal resistance through the thermal interface material including the glue layer and dielectrics is

$$R^v_{TIN} = \frac{t_{glue}}{k_{glue}\,(1-\alpha)\,WS^2} + \frac{t_{ILD}}{t_{ILD}\,(1-\alpha)\,WS^2} \tag{3.40}$$

where t_{glue} is the thickness of the glue layer, k_{glue} the thermal conductivity of the glue layer, t_{ILD} the thickness of the dielectrics and t_{ILD} the thermal conductivity of the dielectrics.

The vertical thermal resistance through the silicon layer is

$$R^v_{Si} = \frac{t_{Si}}{k_{Si}\,(1-\alpha)\,WS^2} \tag{3.41}$$

where k_{Si} is thermal conductivity of the silicon layer.

Therefore, the vertical thermal resistance in figure 3.15 is

$$R_z = (R^v_{TIN} + R^v_{Si})\,//R_{TSV} \tag{3.42}$$

The thermal resistance to ground is

$$R_g = R_z + R_{pkg} \tag{3.43}$$

where R_{pkg} is the thermal resistance of the package.

The horizontal thermal resistance through the glue layer is

$$R^h_{glue} = \frac{1}{k_{glue} t_{glue}} \tag{3.44}$$

The horizontal thermal resistance through the dielectrics layer is

$$R^h_{ILD} = \frac{1}{k_{ILD} t_{ILD}} \tag{3.45}$$

The horizontal thermal resistance through the silicon layer is

$$R^h_{Si} = \frac{1}{k_{Si} t_{Si}} \tag{3.46}$$

Therefore, the horizontal thermal resistance in figure 3.15 is

$$R_x = R_y = R^h_{glue} // R^h_{ILD} // R^h_{Si} \tag{3.47}$$

From figure 3.15 and the aforementioned equations to calculate the several thermal resistances, the following matrix equation is formed:

$$\begin{pmatrix} T_1 - T_{amb} \\ T_2 - T_{amb} \\ \vdots \\ T_m - T_{amb} \end{pmatrix} = \begin{pmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,n} \\ R_{2,1} & R_{2,2} & \cdots & R_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ R_{m,1} & R_{m,2} & \cdots & R_{m,n} \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{pmatrix} \tag{3.48}$$

or

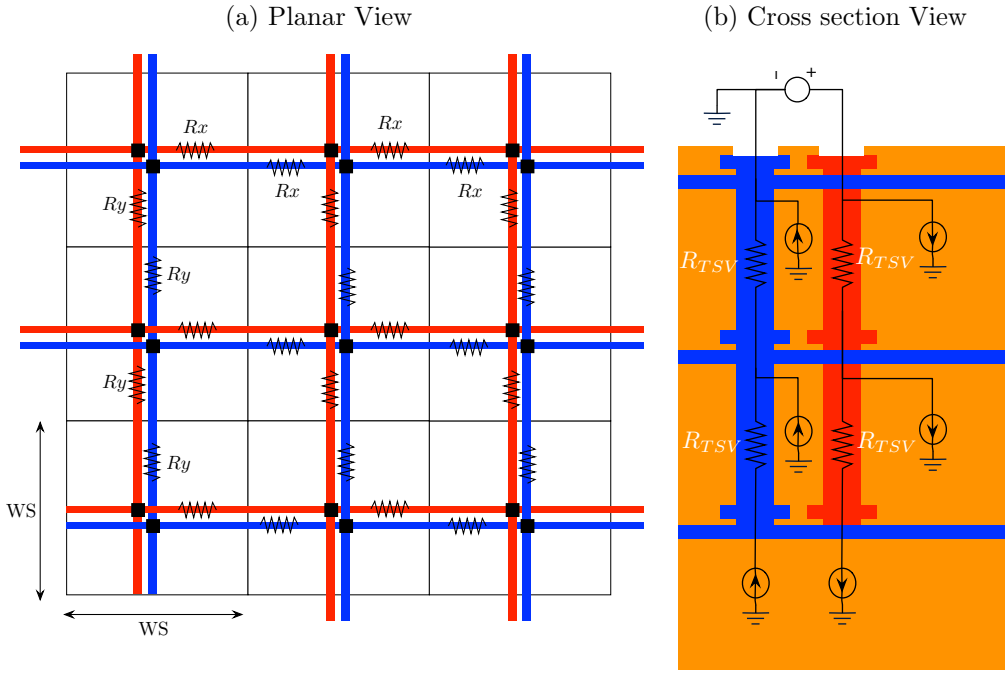
$$\boldsymbol{T}^t - \boldsymbol{T_{amb}} = \boldsymbol{R_{thermal}} \boldsymbol{P}^{t-1} \tag{3.49}$$

where $\boldsymbol{T}^t$ is the temperature in each mesh cell in the current iteration, $\boldsymbol{T_{amb}}$ is the ambient temperature of the package, $\boldsymbol{R_{thermal}}$ is the Inverse Modified Nodal Analysis Conductance Matrix for the circuit of figure 3.15 as described in [48] and $\boldsymbol{P}^{t-1}$ the power dissipation on each mesh cell, calculated at the operating conditions of the previous iteration.

3.6.2 Power Grid Model

The global power grid model of [31] is extended to capture the 3rd dimension of 3D ICs. The space of the chip is divided into cuboidal mesh. Mesh cells have equal width and height WS and thickness equal to the thickness of substrate plus the dielectrics and the glue layers. Through each cell, one vertical and one horizontal power conductors, pass on different layers. At the crossing point there is one TSV to connect the cell, with the cell below, unless this is the first layer. The same applies for the ground conductors. Figure 3.16a shows the planar view of the model. Figure 3.16b shows the cross section of a mesh cell stack across all dies. In these figures, red denotes power conductors, and blue denotes

Figure 3.16: Planar and Cross section View of the Power Grid Model



ground conductors. Figure 3.17 shows the model for the power conductors and figure 3.18 for the ground conductors.

The horizontal resistances in figures 3.17 and 3.18 are calculated as:

$$R_x = R_y = R_{PG\_metal} WS \tag{3.50}$$

where R_{PG_metal} is the resistance per unit length of the metal conductors and WS is the mesh cell width and height.

The vertical resistance in figures 3.17 and 3.18 is calculated as:

$$R_z = R_{PG\_TSV} \tag{3.51}$$

where R_{PG_TSV} is the resistance of the power TSV.

From figure 3.17 and the aforementioned equations to calculate the resistances, the following matrix equation is formed for the power network:

$$\begin{pmatrix} V_{supply} - V_1 \\ V_{supply} - V_2 \\ \vdots \\ V_{supply} - V_n \end{pmatrix} = \begin{pmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,n} \\ R_{2,1} & R_{2,2} & \cdots & R_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ R_{m,1} & R_{m,2} & \cdots & R_{m,n} \end{pmatrix} \begin{pmatrix} -I_1 \\ -I_2 \\ \vdots \\ -I_n \end{pmatrix} \tag{3.52}$$

or

Figure 3.17: Power Grid Model

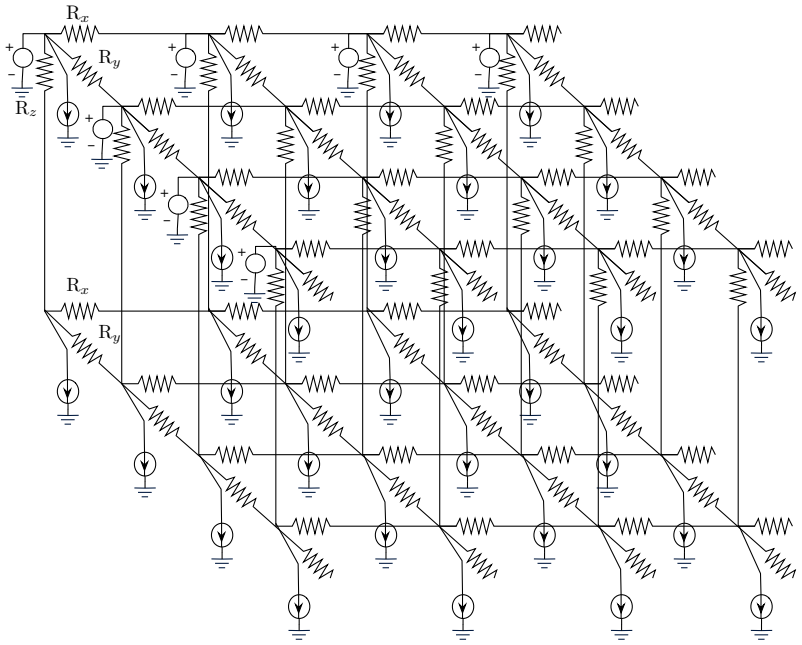
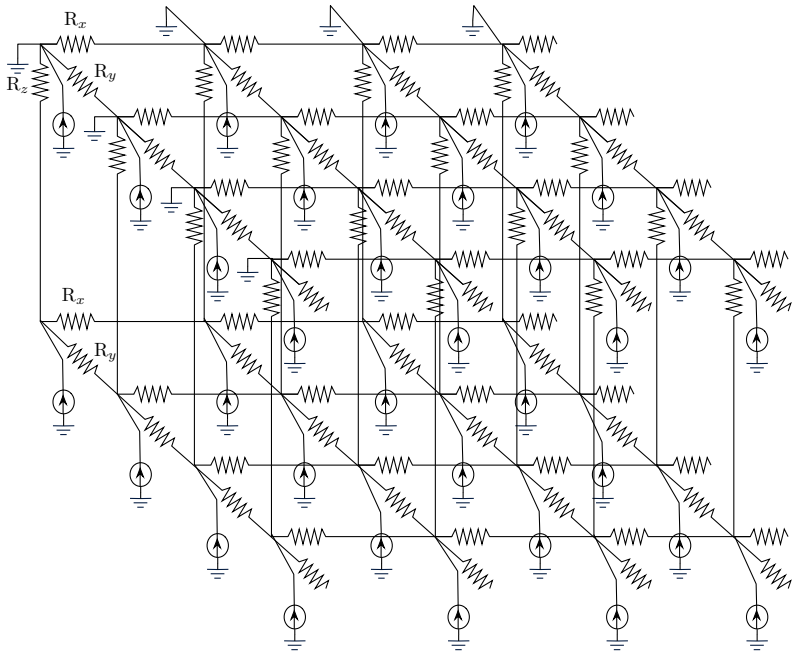


Figure 3.18: Ground Grid Model



$$\boldsymbol{V_{supply}} - \boldsymbol{V_p}^t = -\boldsymbol{R_{pg}}\boldsymbol{I}^{t-1} \tag{3.53}$$

From figure 3.18 and the aforementioned equations to calculate the resistances, the following matrix equation is formed for the ground network:

$$\begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{pmatrix} = \begin{pmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,n} \\ R_{2,1} & R_{2,2} & \cdots & R_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ R_{m,1} & R_{m,2} & \cdots & R_{m,n} \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{pmatrix} \tag{3.54}$$

or

$$\boldsymbol{V_g}^t = \boldsymbol{R_{pg}}\boldsymbol{I}^{t-1} \tag{3.55}$$

where $\boldsymbol{V_{supply}} - \boldsymbol{V_p}^t - \boldsymbol{V_g}^t$ is the supply voltage in each mesh cell in the current iteration, $\boldsymbol{R_{pg}}$ is the Inverse Modified Nodal Analysis Conductance Matrix for the circuit of figures 3.17 and 3.18 as described in [48] and $\boldsymbol{I}^{t-1}$ the current on each mesh cell, calculated at the operating conditions of the previous iteration as:

$$I_i^{t-1} = \frac{P_i^{t-1}}{V_i^{t-1}} \tag{3.56}$$

3.6.3 Adaptive Electrothermal Simulator

Algorithm 4 summarizes the Adaptive Electrothermal Simulator. In each mesh cell, supply voltage is initialized to the external supply voltage and temperature is initialized to the ambient temperature of the package. The inverse of the modified nodal analysis matrices are calculated as previously described. Then the simulation loop begins. Current and dissipated power in each mesh cell, overall power consumption and timing are calculated based on the supply voltage and temperature of the previous iteration, using algorithm 5. New supply voltage and temperatures are calculated solving the modified nodal analysis equations. The loop continues until overall power consumption and timing converge.

Algorithm 5 describes the process of calculating current and power dissipation in each cell, and overall power and timing. Power and current in each cell is initialized to 0. For each module, the mesh cells it overlaps are identified and the overlap percentage with each is calculated. The temperature and voltage of a module is the weighted sum of temperature and voltage of each mesh cell it overlaps. Based on these values, power consumption and timing for the module are retrieved from the module database. In each mesh cell the module overlaps, the current it draws is calculated as the power consumption over the supply voltage, and current and power dissipation are added to the cell. The overall

Input: CPSHM Netlist, Module Database D, Mesh Cells $\boldsymbol{M}$, External Supply Voltage V_0, Ambient Temperature T_0
Output: Overall Power Consumption PC, Timing $Timing$
Data: Power Vector $\boldsymbol{P}$, Current Vector $\boldsymbol{I}$, Temperature Vector $\boldsymbol{T}$, Supply Voltage Vector $\boldsymbol{V}$, Inverse Thermal MNA Matrix $\boldsymbol{R_{thermal}}$, Inverse Power Grid MNA Matrix $\boldsymbol{R_{pg}}$, Power Grid Resistances $\boldsymbol{R}$
Initialize Supply Voltage Vector $\boldsymbol{V}$ to V_0;
Initialize Temperature Vector $\boldsymbol{T}$ to T_0;
Calculate $\boldsymbol{R_{thermal}}$, $\boldsymbol{R_{pg}}$, $\boldsymbol{R}$;
$t = 1$;
repeat
  Calculate $\boldsymbol{P}^t$, $\boldsymbol{I}^t$, PC^t, $Timing^t$ from algorithm 5 using $\boldsymbol{V}^{t-1}$ and $\boldsymbol{T}^{t-1}$;
  $\boldsymbol{T}^t - \boldsymbol{T_{amb}} = \boldsymbol{R_{thermal}}\boldsymbol{P}^{t-1}$;
  $\boldsymbol{V}^t = \boldsymbol{V_0} - 2\boldsymbol{R_{pg}}\boldsymbol{I}^{t-1}$;
  $t = t + 1$;
until PC *and* $Timing$ *Converge*;
Report PC and $Timing$ from last iteration;

Algorithm 4: Adaptive Electrothermal Simulator

power consumption is the sum of power consumptions of each module plus the power dissipated on the supply network. Timing analysis is performed using the netlist and timing of each module.

Note that the above description is for simulation of CPSHMs. The only extension to simulating the whole chip is to consider separate power supply network for each CPSHMs. It is considered straightforward and is not further described.

3.7 Genetic Algorithm

In this work, the use of a multi–objective genetic algorithm is proposed to solve the 3D floorplanning problem. In the experiments, NSGA–II is used, which is described in [13]. The problem specific items that must be defined are the solution representation, and mutation and crossover operators.

Following the concept of genetic algorithms, a solution can be seen a DNA molecule. The separate information groups controlling different aspects of the solution can be seen as genes of the DNA molecule. Decoding each gene of a solutions yield the solution itself. During mutation, each gene is separately treated. During crossover, each gene type treated separately between the two

Input: CPSHM Netlist, Module Database D, Mesh Cells $\boldsymbol{M}$, Temperature Vector $\boldsymbol{T}$, Supply Voltage Vector $\boldsymbol{V}$, Power Grid Resistances $\boldsymbol{R}$
Output: Power Vector $\boldsymbol{P}$, Current Vector $\boldsymbol{I}$, Overall Power Consumption PC, Timing $Timing$
Data: Timing Per Module $\boldsymbol{TM}$
Initialize Power Vector $\boldsymbol{P}$ to 0.0;
Initialize Current Vector $\boldsymbol{I}$ to 0.0;
foreach *module* m **do**
  Find overlapping mesh cells $C \subset M$;
  Calculate Overlap Percentage with each mesh cell O;
  $w = \sum_{i \in C} O_i$;
  $t = \frac{\sum_{i \in C} T_i}{w}$;
  $v = \frac{\sum_{i \in C} V_i}{w}$;
  $p = D_{power}(m, v, t)$;
  $TM_m = D_{timing}(m, v, t)$;
  $PC \mathrel{+}= p$;
  foreach $i \in C$ **do**
    $i = \frac{p}{V_i}$
    $P_i \mathrel{+}= p$;
    $I_i \mathrel{+}= \frac{p}{V_i}$;
  end
end
foreach *branch* b *from mesh cell* i *to* j **do**
  $PC \mathrel{+}= \frac{(V_i - V_j)^2}{R_b}$;
end
Calculate $Timing$ doing timing analysis based on $\boldsymbol{TM}$ and CPSHM Netlist;

Algorithm 5: Mesh Power and Current Calculation for Current Iteration

parents. The mutation and crossover operators are based on [36].

3.7.1 Solution Encoding

The proposed solution representation is based on Double Tree and Sequence and Labeled Tree and Sequence 3D floorplan representation. In Double Tree and Sequence, two trees are used to define the X and Y coordinates of each module. After X and Y coordinates for each module are decoded, a permutation is used to push the modules along the Z axis and therefore obtain the Z coordinate.

On the other hand, Labeled Tree and Sequence uses one tree to decode the Y coordinate and a number sequence X and a permutation P to decode the other two coordinates. The permutation is used to decode the Z coordinate as in Double Tree and Sequence. To decode the X coordinate, $x_{P[i]} = x_{P[X[i]]} + w_{P[X[i]]}$, where $0 \leq X[i] < i$. Notice, that this basically defines a second tree, where module $P[i]$ has as parent $P[X[i]]$. If $X[i]$, then the parent is the root. Therefore the two representations are the same, except that Double Tree and Sequence uses a separate permutation for the Z coordinate.

Following the same principles and decode procedure, the proposed representation has 3 permutations P_x, P_y, P_z and 3 number sequences X, Y, S, such that $0 \leq X[i] < i$ and $0 \leq Y[i] < i$. Number sequence S is used to choose a placement for module i from the placement entries of the Module Database, and therefore choose a combination of width, height and thickness for the module. To decode X and Y coordinates, $x_{P[i]} = x_{P_x[X[i]]} + w_{P_x[X[i]]}$ and $y_{P_y[i]} = y_{P_y[X[i]]} + h_{P_y[X[i]]}$. To decode the Z coordinate, the same procedure with the aforementioned representations is applied, using permutation P_z. In the sequence of the permutation, for each module, the previously treated modules it overlaps on the horizontal plane, are identified. The height $z_{P_z[i]}$ of the module is the maximum height plus thickness of the modules it overlaps.

3.7.2 Mutation Operator

Algorithm 6 show the mutation operator for the number sequence. A random position is picked and the respective entry is assigned a random number in the allowed range.

Input: Number Sequence N
i = RandomInteger(0, Length(N));
j = RandomInteger(0, i);
$N[i] = j$;

Algorithm 6: Number Sequence Mutation

Algorithm 7 show the mutation operator for the placement number sequence. A random position is picked and the respective entry is assigned a random number in the allowed range.

Input: Placement Sequence P, Max Placement Index M
i = RandomInteger(0, Length(P));
j = RandomInteger(0, $M[i]$);
$P[i] = j$;

Algorithm 7: Placement Sequence Mutation

Algorithm 8 show the mutation operator for the permutation. Two random positions are picked and the respective entries are swapped.

Input: Permutation P
i = RandomInteger(0, Length(P));
j = RandomInteger(0, Length(P));
$t = P[i]$;
$P[i] = P[j]$;
$P[j] = t$;

Algorithm 8: Permutation Mutation

3.7.3 Crossover Operator

Algorithm 9 show the crossover operator for the number sequence. The same applies to the placement sequence as well. A random position is picked. Entries up to this position are inherited from the first parent and the rest from the second parent.

Input: Parent Number Sequence N_1, Parent Number Sequence N_2
Output: Offspring O_1,Offspring O_2
l = Length(N_1);
p = RandomInteger(0, l);
for $i \leftarrow 1$ **to** p **do**
  $O_1[i] = N_1[i]$;
  $O_2[i] = N_2[i]$;
end
for $i \leftarrow p + 1$ **to** l **do**
  $O_1[i] = N_2[i]$;
  $O_2[i] = N_1[i]$;
end

Algorithm 9: Number Sequence Crossover

Algorithm 10 show the crossover operator for the permutation. One parent is sorted towards the other, using bubble sort for example, and all intermediate steps are kept. The offspring is a randomly chosen step in the sorting sequence.

Input: Parent Number Sequence P_1, Parent Number Sequence P_2
Output: Offspring O_1,Offspring O_2
list = Sort(P_1, P_2);
$i1$ = RandomInteger(2, Length(list) - 1);
$i2$ = RandomInteger(2, Length(list) - 1);
O_1 = list[$i1$];
O_2 = list[$i2$];

Algorithm 10: Permutation Crossover

Chapter 4

Experimental Results

4.1 Overview

4.1.1 ITC99 Benchmark Circuits

Name	Original Functionality	Number of gates	Inputs	Outputs	Number of Flip Flops
b14	Viper processor (subset)	10098	32	54	245
b15	80386 processor (subset)	8922	36	70	449
b19	2 copies of b14 and 6 of b15	231320	21	30	6642

Table 4.1: ITC99 benchmark circuits used in this thesis

In order to experiment on the proposed framework, 3 circuits from the ITC99 benchmark circuits from Politecnico di Torino were used. More information on these circuits can be found at [1] and [12]. Table 4.1 summarizes the main characteristics of the circuits used in this thesis.

4.1.2 Nangate 45nm Standard Cell Library

The standard cells for the aforementioned circuits were taken from Nangate 45nm Open Cell Library, which is based on the FreePDK. More information on the cell library and pdk can be found in [3] and [2]. Table 4.2 summarizes the gates used from Nangate45 and the correspondance with the gate names in the netlists of ITC99 circuits. Note that 5–Input gates were not available in Nangate45 cell library. These gates were replaced in the netlists with the

ITC99 Gate Name	Functionality	Nangate45 Gate Name
Inv_gate	Inverter	INV_X1
Buf_gate	Buffer	BUF_X1
And_gate	2 Input AND	AND2_X1
Nand_gate	2–Input NAND	NAND2_X1
Or_gate	2–Input OR	OR2_X1
Nor_gate	2–Input NOR	NOR2_X1
And3_gate	3–Input AND	AND3_X1
Nand3_gate	3–Input NAND	NAND3_X1
Or3_gate	3–Input OR	OR3_X1
Nor3_gate	3–Input NOR	NOR3_X1
And4_gate	4–Input AND	AND4_X1
Nand4_gate	4–Input NAND	NAND4_X1
Or4_gate	4–Input OR	OR4_X1
Nor4_gate	4–Inout NOR	NOR4_X1
And5_gate	5–Input AND	N/A
Nand5_gate	5–Input NAND	N/A
Or5_gate	5–Input OR	N/A
Nor5_gate	5–Inout NOR	N/A
Xor_gate	2–Inout XOR	XOR2_X1
Xnor_gate	2–Inout XNOR	XNOR2_X1
zero	Logic 0	LOGIC0_X1
one	Logic 1	LOGIC1_X1
DFF	D Flip Flop	DFF_X1

Table 4.2: Standard Cells used from Nangate45

respective 4–Input gate and a 2–Input AND or OR gate in order to provide the same functionality with the available gates.

4.1.3 FreePDK

Table 4.3 summarizes the main characteristics of the metal layers of FreePDK from the LEF file that is provided with Nangate45.

4.1.4 Misc Assumptions

Table 4.4 shows any other parameter assumption used in this work.

Name	Height (um)	Thickness (um)	Sheet Resistance (Ohm/sq)	Capacitance (pF/um^2)	Edge Capacitance (pF/um)
metal1	0.37	0.13	0.38	$7.71613e-05$	$3.86e-05$
metal2	0.62	0.14	0.25	$4.08957e-05$	$2.04e-05$
metal3	0.88	0.14	0.25	$2.7745e-05$	$1.39e-05$
metal4	1.14	0.28	0.21	$2.07429e-05$	$1.04e-05$
metal5	1.71	0.28	0.21	$1.3527e-05$	$6.76e-06$
metal6	2.28	0.28	0.21	$1.00359e-05$	$5.02e-06$
metal7	2.85	0.8	0.075	$7.97709e-06$	$3.99e-06$
metal8	4.47	0.8	0.075	$5.0391e-06$	$2.52e-06$
metal9	6.09	2	0.03	$3.68273e-06$	$1.84e-06$
metal10	10.09	2	0.03	$2.21236e-06$	$1.11e-06$

Table 4.3: FreePDK Metal Layers

4.1.5 Implemented Parts of Proposed Framework

Most parts of the proposed framework were implemented in Mathematica™, [5], to allow easy debugging and ease results visualization. The rest were implemented in C++.

From section 3.2, the Pre–Characterization of the Standard Cell Library and Pre–Characterization of the Module Library were implemented as described. Within the standard cell library pre–characterization, the script generator and .lib post process were implemented in C++. To make the pre–characterization of the standard cell, a commercial tool was used. However it cannot be named due to license restrictions. All tools communicate with text files. Within the module library pre–characterization, the place & route script generator, the module compiler and power timing simulator were implemented in C++. To place the modules, the dragon placer [45] was used since it is freely distributed. Note that in this work, interconnects within the modules are ignored due to time limitations. It is assumed that their effects will be additive and will not affect the trend in the results. The statistical post process step is implemented in Mathematica as described later. The resulting module library is implemented as Mathematica dictionary entries. Build–in interpolation functions are used for the lookup tables.

From section 3.3, the two alternative flows of the Common Power Supply Hyper Module Assembly Step were implemented in Mathematica. For same–type modules, a function places the modules on a user defined number of layers, trying to keep an aspect ratio close to 1. This function is used in all experiments except for the genetic algorithm. For dissimilar modules a multi–objective genetic algorithm is developed as is later described.

From section 3.4, the Univariate Statistical Prediction Engine was imple-

Notation	Parameter	Value	Source
t_{Si}	Thickness of silicon substrate	$500um$	
t_{SOI}	Thickness of SOI substrate	$50um$	
t_{glue}	Thickness of glue layer between two stacked substrates	$2um$	[50]
t_{ILD}	Thickness of ILD	$12um$	[50]
k_{Cu}	Thermal conductivity of Cu	$385\frac{W}{mK}$	[50]
k_{Si}	Thermal conductivity of Si	$148\frac{W}{mK}$	[50]
k_{glue}	Thermal conductivity of glue	$0.25\frac{W}{mK}$	[50]
k_{ILD}	Thermal conductivity of ILD	$0.19\frac{W}{mK}$	[50]
R_{t_pkg}	Thermal resistance of package	$0.5\frac{W}{K}$	Hotspot
WS	Window size	$50um$	Experimentally chosen
R_{PG_metal}	Power grid metal resistance	$0.012\Omega/um$	From metal9 and metal10 assuming $2.5um$ conductor width
R_{PG_TSV}	Power grid TSV resistance	0.03Ω	[39]
R_{PG_pkg}	Power grid package resistance	0.001Ω	Abstractly chosen to ignore package
R_{GR_metal}	Global routing metal resistance	$0.14\Omega/um$	From metal5 assuming $1.5um$ conductor width
C_{GR_metal}	Global routing metal capacitance to ground	$3.38e-17F/um$	From metal5 assuming $1.5um$ conductor width
R_{GR_TSV}	Global routing TSV resistance	0.12Ω	[39]
C_{GR_TSV}	Global routing TSV capacitance to ground	$84e-15F$	[39]
T_{pkg}	Ambient package temperature	45^oC	[44]

Table 4.4: Misc Assumptions

mented in Mathematica as described. Only for the Maximum Likelihood parameter estimation, the build–in function was used. From section 3.5, the power timing simulator was implemented in C++ as described. From section 3.6, the adaptive electrothermal simulator was implemented in Mathematica as described. The build–in function for matrix inversion was used. From section 3.7

and solution encoding and decoding method, and, the mutation and crossover operators were implemented in Mathematica as described. The NSGA–II was implemented in Mathematica as described in [13].

4.2 Experimental Results on the Standard Cell Library Pre–Characterization Step

The goal of this experiment is to observe the behavior of standard cells under strong temperature and voltage variation and collect the necessary data for the rest of the experiments. The standard cells were characterized as described in section 3.2 for 99 pairs of voltage / temperature operating conditions. The typical corner models were used for the standard cells. The voltage, temperature, slew rate and load capacitance points were:

$$V_{range} = \{0.75, 0.80, 0.85, 0.90, 0.95, 1.00, 1.05, 1.10, 1.15\}V \tag{4.1}$$

$$T_{range} = \{25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275\}^oC \tag{4.2}$$

$$S_{range} = \{0.0045, 0.01125, 0.0225, 0.045, 0.09, 0.18, 0.36, 0.72\}nsec \tag{4.3}$$

$$C_{range} = \{0.0004, 0.0008, 0.0016, 0.0032, 0.0064, 0.0128, 0.0256\}pF \tag{4.4}$$

Figure 4.1a shows the static or leakage power, as a function of voltage and temperature for AND3. It is observed that static power grows exponentially with temperature. A slight linear increase with voltage is also observed. The same behavior is observed in all gates. Figure 4.1b shows the input pin capacitance as a function of voltage and temperature for AND3. It is observed that input pin capacitance rises linearly with temperature and voltage. The same behavior is observed in all gates.

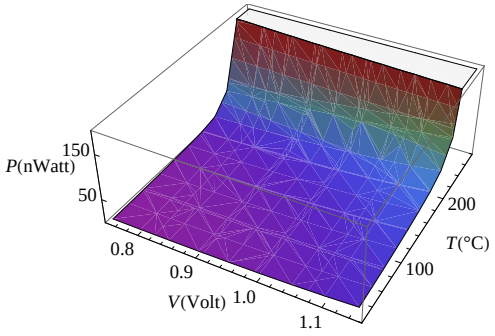
Figures 4.2 and 4.3 show the rise and fall delay and transition time as a function of voltage and temperature for AND3. In all cases, delay and transition time drop almost linearly with increased temperature, and rise almost linearly with increased voltage. The same behavior is observed in all gates.

Figures 4.4 and 4.5 show rise and fall dynamic power as a function of temperature and voltage for AND3 and XNOR2. Strong not linear effects are observed, especially after a critical temperature. The exact profile heavily depends on the gate. In several cases a peak is observed around $1.0V$ independent of temperature.

Concluding this section, it must be noted that in all parameters of the standard cells, strong dependence is observed on supply voltage and temperature.

Figure 4.1: Static Power and Input Pin Capacitance of AND3

(a) Static Power



(b) Input Pin Capacitance

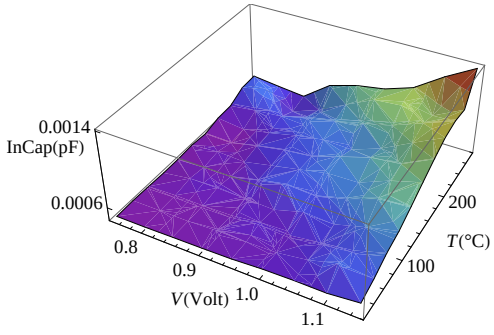
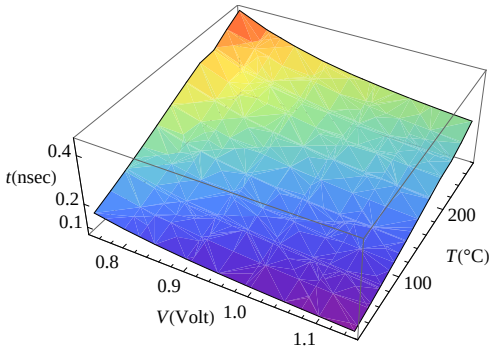


Figure 4.2: Rise and Fall Delay of AND3

(a) Rise Delay



(b) Fall Delay

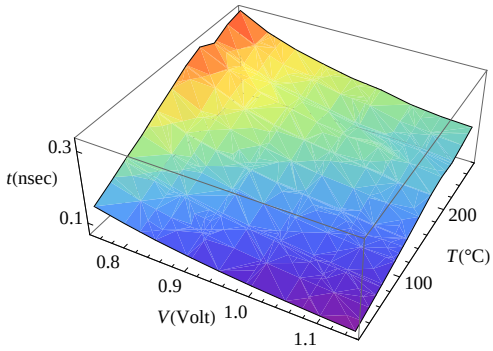
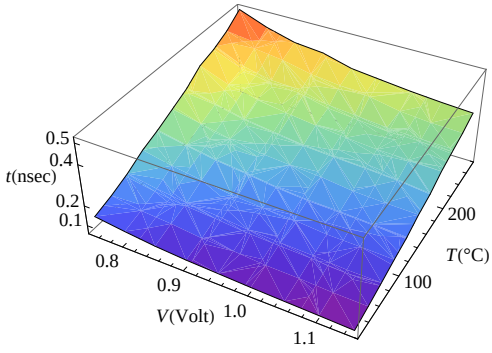


Figure 4.3: Rise and Fall Transition Time of AND3

(a) Rise Delay



(b) Fall Delay

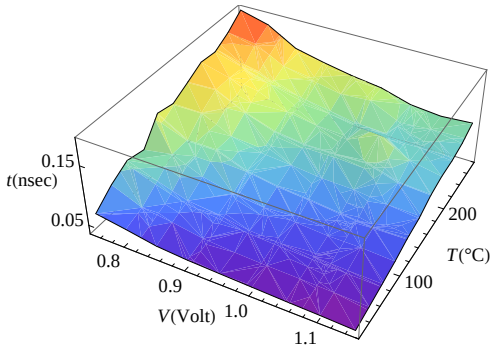
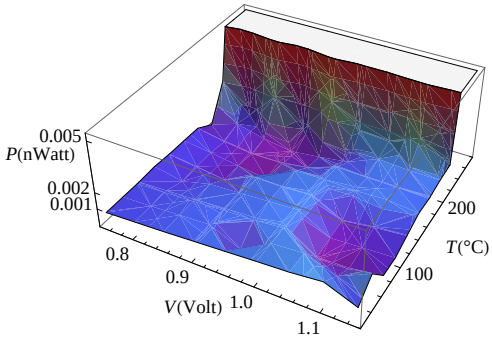


Figure 4.4: Rise and Fall Dynamic Power of AND3

(a) Rise Dynamic Power



(b) Fall Dynamic Power

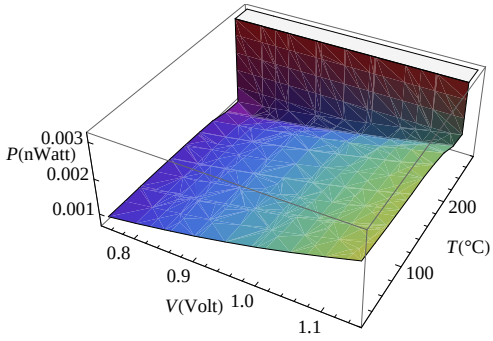
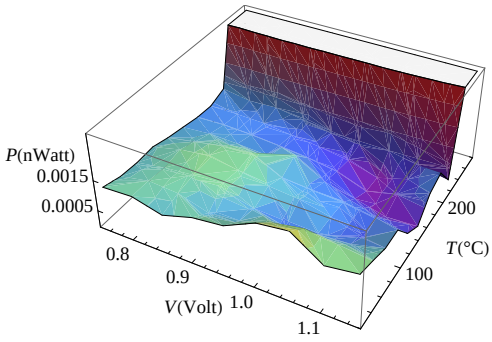
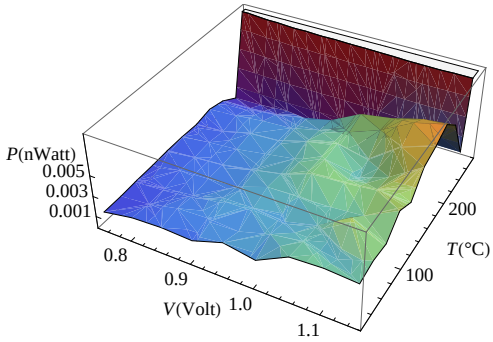


Figure 4.5: Rise and Fall Dynamic Power of XNOR2

(a) Rise Dynamic Power



(b) Fall Dynamic Power



Changing the voltage by $0.1V$ or the temperature by 50^oC can result in more than 100% change in the value of the parameter. Therefore it is important to take into account the variation of voltage and temperature in a far greater range of values than the traditional 5 corner model approach.

4.3 Experimental Results on Module Maximum Power & Timing Estimation

In this section, a series of experiments is conducted in order to test the Power / Timing simulator and the Univariate Statistical Prediction Engine and configure the Module Library Pre–Characterization Step.

4.3.1 Calculated Power Distributions

In these experiments, the Power / Timing Simulator is run at nominal operating conditions of $1.1V$ and 25^oC for different number of excitations for the 3 selected benchmark circuits and the produced populations of power are visually inspected. The goal of this experiment is to observe the instantaneous power distributions as well as the distribution of power maxima. Finally the sample size to use in order to obtain the maximum estimate is chosen.

Figure 4.6 shows the power histograms of b15 for different number of excitations. It is observed that the power histogram seems to become continuous after 1000 excitations. Similar results are observed for timing for circuits b14 and b19, however, the distributions of instantaneous power are unique to each circuit.

Figure 4.6: Power distributions for b15 under various number of excitations

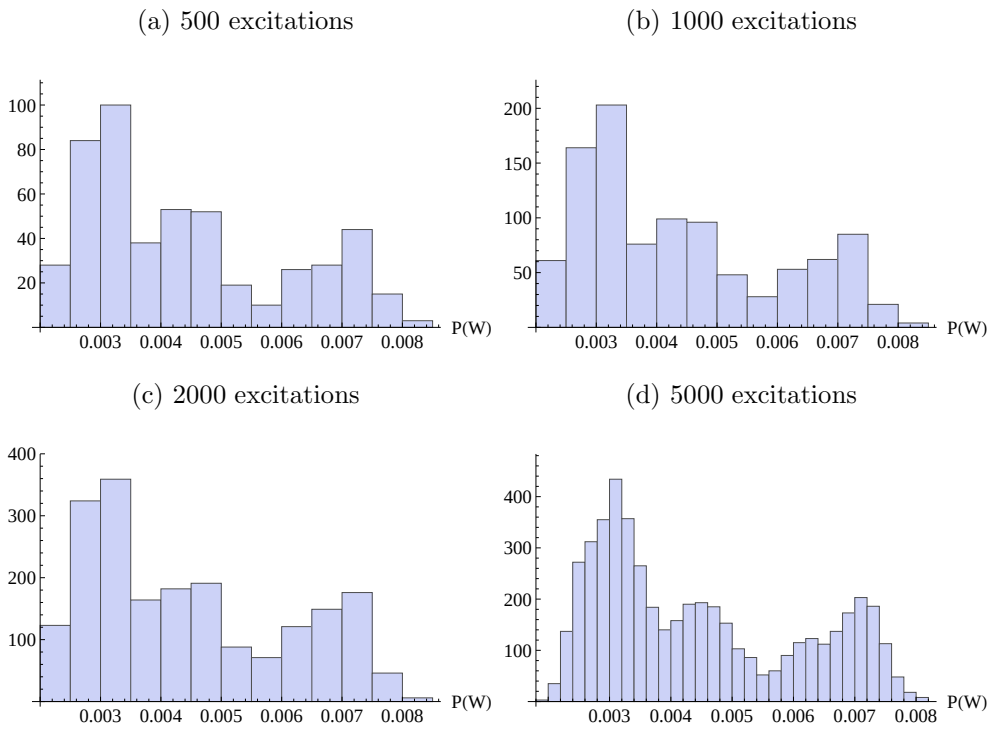
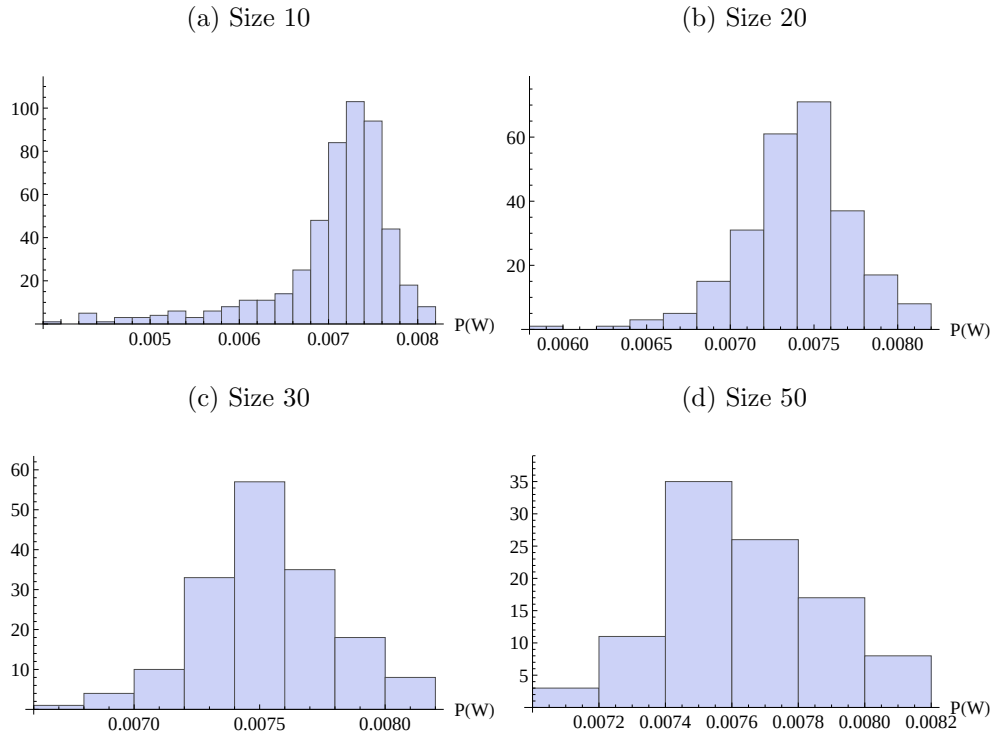


Figure 4.7 shows the histograms of power maxima under different sample sizes for 5000 excitations. It is observed that in all circuits, for power, a sample size 30 shows a relatively clear distribution, similar to that of GEVD. Similar results are observed for timing for circuits b14 and b19.

Figure 4.7: Max Power distributions of different sample sizes for b15 under 5000 excitations



4.3.2 EVT Estimates of Maximum Power at Nominal Operating Conditions

In these experiments, the Power / Timing Simulator is run at nominal operating conditions of $1.1V$ and 25^oC for different number of excitations for the 3 selected benchmark circuits. The resulting power distributions are fed into the univariate statistical prediction engine with parameters sample size 30, 0.999 confidence interval, 0.001 significance level and 100 points for the QLS estimate. The goal of this experiment is observe the convergence of the confidence interval (C.I.) along with the observed maximum of the sample. Figures 4.8, 4.9 and 4.10 show the results obtained for circuits b14, b15 and b19 respectively.

In all cases, it is observed that with a small initial sample, the estimate of the confidence interval is far from the sample max and the point of convergence. For circuits b14 and b19 that have similar size, a sample size of 1000 yields a good estimate close and above the sample max at sample size 100000. For circuit b19 that is 22 times larger than b14 and b15, a sample size of 5000 yields a good estimate.

Figure 4.8: EVT Estimates of Maximum Power for b14

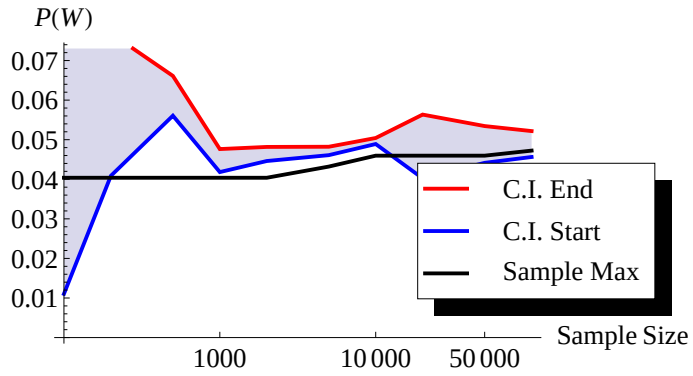
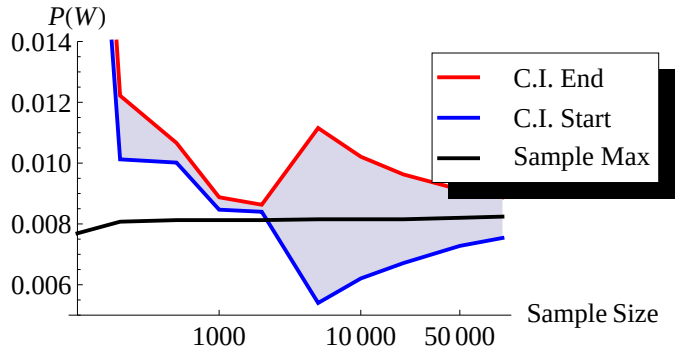


Figure 4.9: EVT Estimates of Maximum Power for b15



4.3.3 Max Power / Timing as a function of Operating Condition

In this experiment, the maximum power consumption and timing of the 3 selected benchmarks as a function of power supply voltage and temperature is obtained. The power timing simulator is run for all 99 voltage / temperature pairs that were used in Standard Cell Library Pre–Characterization. For maximum power estimation, the power simulator was run for 1000 excitations for b14 and b15 and 5000 excitations for b19. The results were fed into the univariate statistical prediction engine with parameters sample size 30, 0.999 confidence interval, 0.001 significance level and 100 points for the QLS estimate to obtain the worst case estimate of power consumption in each voltage / temperature pair. The results are shown in figures 4.11, 4.12 and 4.13

The maximum timing results are similar in all cases. There is minimum at

Figure 4.10: EVT Estimates of Maximum Power for b19

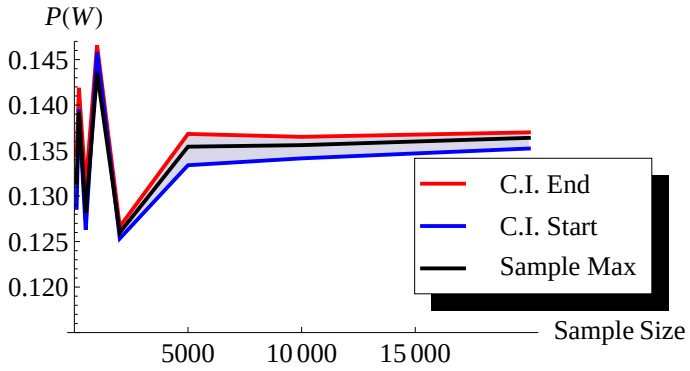
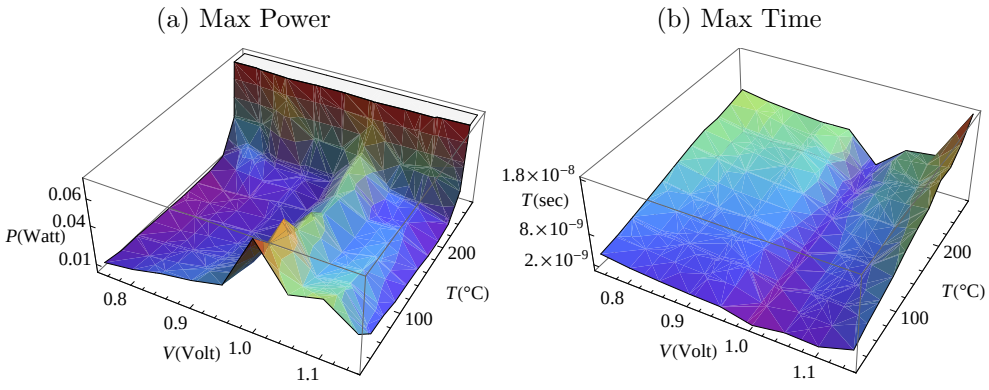


Figure 4.11: Max Power and Time of b14



$1.0V$ that slightly increases with temperature. Moving away from $1.0V$, timing increases significantly. The increase with temperature is also bigger than at $1.0V$. Slight changes of voltage by $0.1V$ or temperature by 50^oC can cause timing to change by more than 300%. It was shown that delay in standard cells increases when temperature increases and decreases when voltage increases. It was also shown that input pin capacitance in standard cells increases when temperature and voltage increases. It is known from [7], that timing decreases when load capacitance increases. Therefore the macro–scale behavior of timing in a big module comprising thousands of gates, is caused by the superposition of behavior of delay and input pin capacitance of individual gates.

The maximum power consumption results are similar in all cases as well. Here, the inverse behavior of timing is observed. There is a maximum at $1.0V$ and dependence with temperature is not monotone but has a falling trend. Moving away from $1.0V$, power decreases significantly. After a critical temperature

Figure 4.12: Max Power and Time of b15

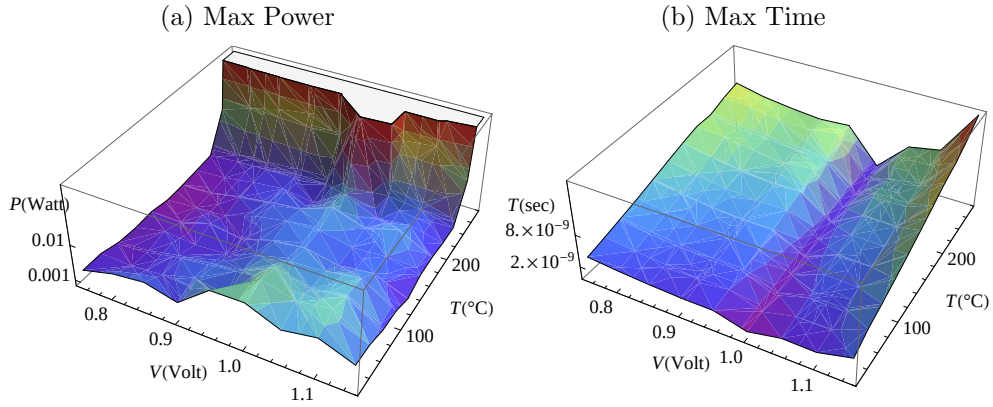
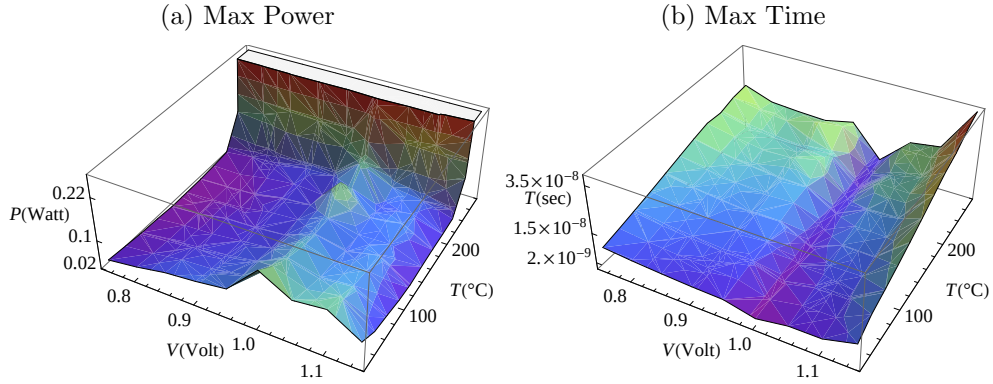


Figure 4.13: Max Power and Time of b19



power increases exponentially. Slight changes of voltage by $0.1V$ or temperature by 50^oC can cause timing to change by more than 300%. It is known in the literature, for example from [7], [8], [28] and [54], that timing and power consumption have a predator–prey relationship. This is the basis of low power design with voltage islands. When power increases, timing decreases and the opposite. It is also known from [58], that leakage has an exponential dependence with temperature and becomes the dominant power consumption component after a critical temperature.

4.4 Experimental Results on the Adaptive Electrothermal Simulator

In this section, a series of experiments is conducted to study the following:

1. The limiting role of the power supply.

2. The impact of increasing die number on a fixed gate count on temperature and supply voltage.

3. The impact of TSV area.

4. The impact of increasing die number on a fixed gate count on performance.

5. The impact of temperature and supply voltage variability on performance.

Note are Common Power Supply Hyper Modules are considered in all experiments.

4.4.1 Electrothermal Profile

In this experiments, the temperature and supply voltage profile across the chip surface is extracted and visualized. Note that supply voltage denotes the voltage difference between the supply and ground that is applied on the module. Figure 4.14 shows the temperature of the top layer of the chip, and the supply voltage of the bottom layer, in case of 200 instances of b19, on 4 layers with 2% TSV density on each layer. Figure 4.15 shows the temperature of the top layer of the chip, and the supply voltage of the bottom layer, in case of 500 instances of b19, on 8 layers with 10% TSV density on each layer. It is observed that the dominant heat flow path is the vertical.

Figure 4.14: Thermal and IR–Drop Profile, in case of 200 Instances of b19, on 4 layers, 2% TSV density

(a) Temperature profile of top layer (b) Supply voltage profile of bottom layer

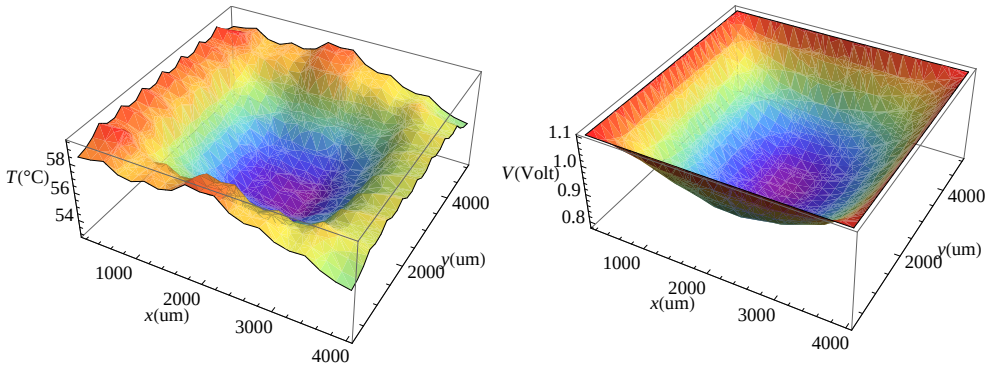
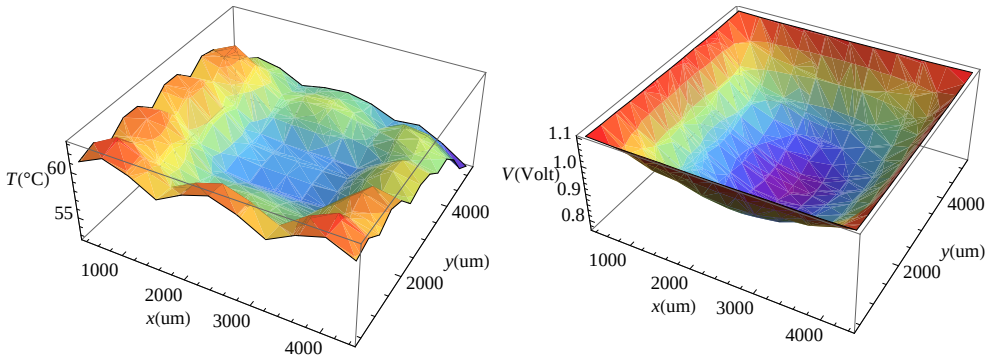


Figure 4.15: Thermal and IR–Drop Profile, in case of 500 Instances of b19, on 8 layers, 10% TSV density

(a) Temperature profile of top layer (b) Supply voltage profile of bottom layer



4.4.2 Maximum Gate Count under the same Power Supply

In this experiment, the limiting role of IR Drop is demonstrated. One power supply of $1.1V$ is assumed here. Figure 4.16 shows the minimum power supply voltage observed across the chip versus gate count, for different number of layers. It is considered that below $0.75V$ standard cell cease to be operational. It is observed that in the 2D case, the supply network can only handle a limited number of gates. As layers are added, parallel resistive roads are added as well and the maximum gate count the supply network can handle grows exponentially.

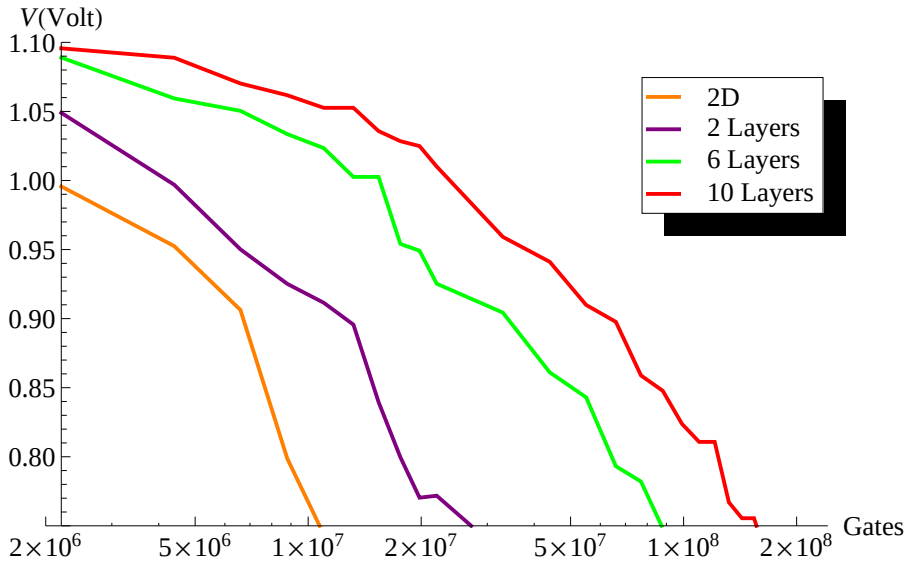
It can be deduced from this experiment, that the major limiting factor in the 2D case is the power supply. In order to surpass power supply limitation, multiple power supplies must be used across the chip. Each power supply will also require additional pads and connections to the board on which is mounted. Therefore, to improve cost and design complexity, the least possible number of independent power supplies must be used.

4.4.3 Convergence

In this experiment, the basic concept of the adaptive electrothermal simulator is shown. The adaptive electrothermal simulator is run on placed systems of different number of modules and layers and the convergence of maximum and mean temperature, minimum and mean supply Voltage, overall system timing and power consumption is studied. Uniform 10% TSV density is assumed.

Figure 4.17, shows the convergence in case of 200 instances of b19 on 4 layer. In all parameters, the first iteration causes significant difference. Max and mean temperature change 13% between the first and second iteration, and

Figure 4.16: Minimum Power Supply Voltage versus gate count for different layer number



11% between the first and the last. Min voltage changes 11% between the first and second iteration and 9% between the first and the last. Mean voltage changes 42% between the first and second iteration and 33% between the first and the last. Timing changes 67% between the first and second iteration and 42% between the first and the last. Power consumption changes 38% between the first and second iteration and 32% between the first and the last. It is also noted that power consumption on the wires is very small and overall power consumption is dominated by the gates power consumption.

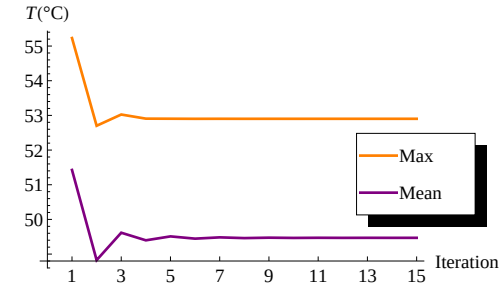
Figure 4.18, shows the convergence in case of 500 instances of b19 on 8 layer. In all parameters again, the first iteration causes significant difference. Max temperature change 14% between the first and second iteration, and 12% between the first and the last. Mean temperature change 17% between the first and second iteration, and 12% between the first and the last. Min voltage changes 78% between the first and second iteration and 67% between the first and the last. Mean voltage changes 18% between the first and second iteration and 6% between the first and the last. Timing changes 100% between the first and second iteration and 82% between the first and the last. Power consumption changes 54% between the first and second iteration and 34% between the first and the last. It is also noted that power consumption on the wires is 75% bigger than gates power consumption.

In summary for this experiment:

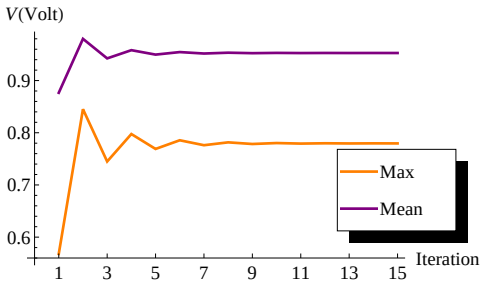
- The adaptive change of power consumed by each module using the new

Figure 4.17: Convergence in case of 200 Instances of b19 on 4 layers

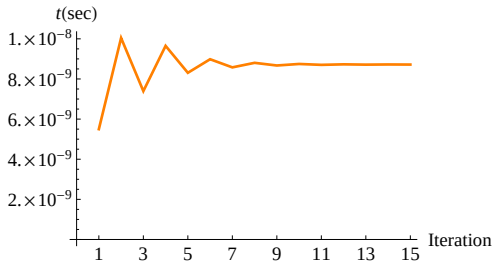
(a) Maximum and Mean Temperature across chip



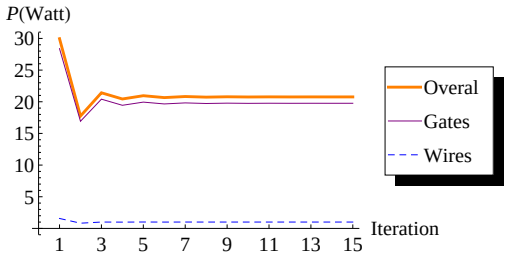
(b) Minimum and Mean Supply Voltage across chip



(c) Timing



(d) Power Consumption



operating conditions is shown to converge.

- In all cases, all parameters change significantly, often more than 50% of initial estimate of nominal operating conditions. These errors on timing and power consumption estimation will be shown in another experiment. This proves the importance of the adaptive nature of the simulator.

- Temperature and supply voltage converge within 5–6 iterations.

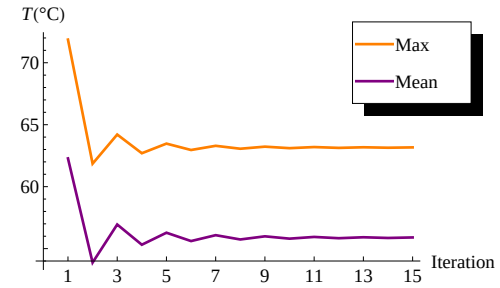
- In case of fewer gates, timing converges within 8–10 iterations and power consumption converges within 4–5 iterations.

- In case of many gates, timing converges and power consumption converge within 12–15 iterations.

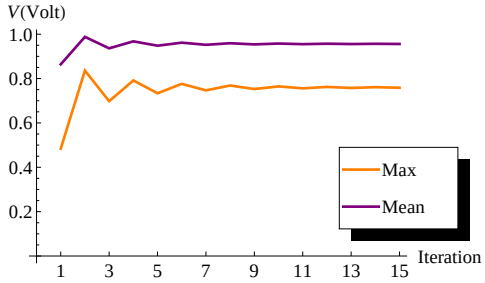
- Last point, shows that it is best to check for convergence using overall system timing and power consumption.

Figure 4.18: Convergence in case of 500 Instances of b19 on 8 layers

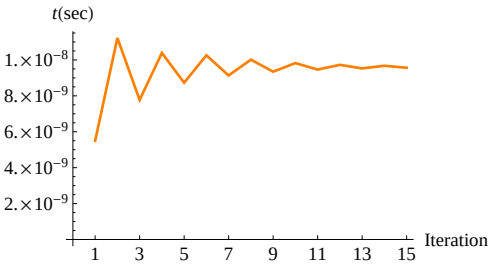
(a) Maximum and Mean Temperature across chip



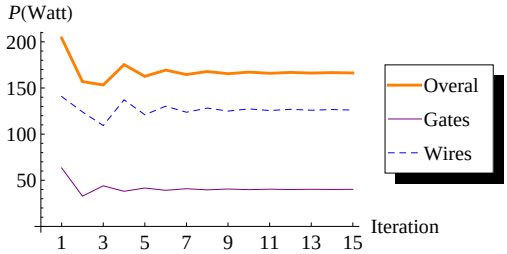
(b) Minimum and Mean Supply Voltage across chip



(c) Timing



(d) Power Consumption



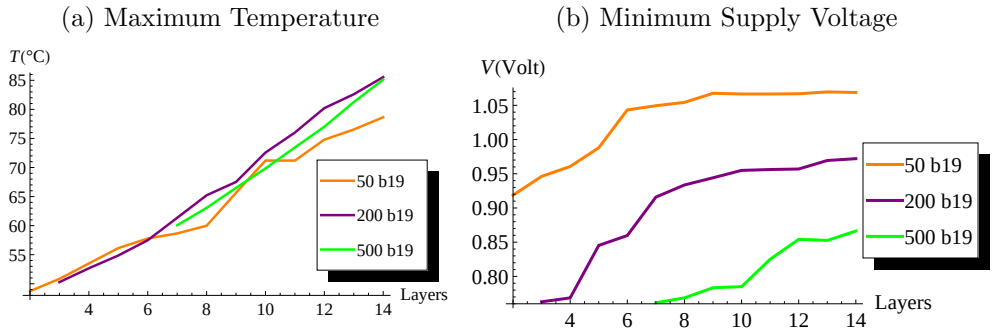
4.4.4 Impact of increasing die number at a fixed gate count on Max Temperature and Mean Supply Voltage

In this experiment, the dependence of maximum temperature and minimum supply voltage on the number of layers for different number of gates is examined. Uniform 10% TSV density is assumed.

Figure 4.19a shows that temperature increases linearly with the number of layers. This was expected since the additional layers of dielectric will worsen heat dissipation, resulting in temperature increase. Another interesting observation, is that maximum temperature has little dependence on the number of gates. This was also expected, since in a previous experiment, were the thermal profile across the chip was examined, it was shown that the vertical heat flow path is dominant. Therefore, a hyper module, containing modules under the same power supply, can be designed alone, while making safe prediction on the performance of the final system, where several hyper modules will be assembled one next to the other.

Figure 4.19b confirms the result of a previous experiment, where the maximum number of modules under the same power supply for different number of

Figure 4.19: Impact of increasing die number at a fixed gate count on Max Temperature and Mean Supply Voltage

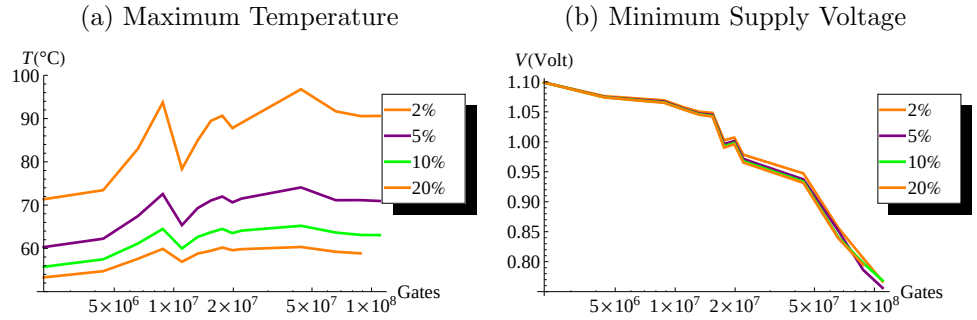


layers was examined. It is evident that minimum supply voltage increases logarithmically when adding layers, up to a point of saturation, due to additional parallel resistive paths from the power supply.

4.4.5 Impact of TSV area

In this experiment, the dependence of maximum temperature and minimum supply voltage on TSV density for a fixed number of layers and gates is examined. It is assumed that TSV density is uniform and equal on all layers. In the experiment shown here, 500 b19 instances are placed on 8 layers. Figure 4.20a shows that maximum temperature is lowered with increased TSV density. This is expected, since increased TSV density improves the vertical heat flow path. Figure 4.20b shows that TSV density has negligible effect on minimum supply voltage.

Figure 4.20: Impact of TSV on Max Temperature and Mean Supply Voltage at 500 b19 instances on 8 layers



4.4.6 Impact of increasing die number on Performance

In previous experiments, it was shown that maximum temperature and minimum supply voltage have a monotone dependence on the number of dies used. When the die number increases, maximum temperature increases as well. On the other hand, minimum supply voltage increases since up to a point of saturation, since parallel resistive roads are added to the power distribution network and IR–drop is improved. In this experiment, the impact of increasing die number on performance is studied. Uniform 10% TSV density is assumed.

Figure 4.21: Optimum die number for Timing and Power Consumption

(a) Timing (b) Power Consumption

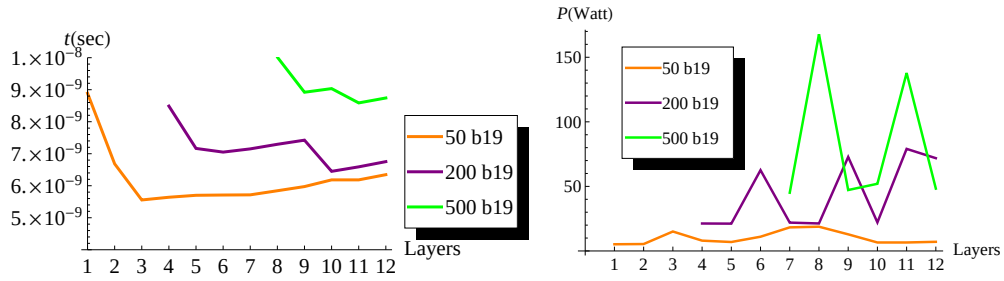


Figure 4.21a shows the impact of increasing die number on system timing for 3 different circuit sizes. In all 3 cases, 4 areas are observed:

1. Initially, timing improves rapidly. In this section, timing is restricted due to low power supply voltage. Adding dies, improves IR–drop, increases supply voltage and therefore timing improves. Although temperature increases and is expected to worsen timing, its impact is small compared to that of power supply voltage.

2. Then, timing worsens slowly. In this section, timing is restricted by temperature. Adding dies, causes temperature increase and therefore timing worsens. Although adding dies improves IR–drop, its impact is smaller compared to that of increasing temperature.

3. Timing improves a lot by adding only one die. Most modules are in the area of minimum timing. Note that this area cannot be separated from the previous in the case of a small circuit.

4. Timing worsens again. Power supply voltage improvement is saturated. Temperatures continues to increase and causes timing to worsen.

Figure 4.21b shows the impact of increasing die number on system power consumption for 3 different circuit sizes. In contrast to timing, power consump-

tion swings as the number of dies increases. It can be observed, that as the circuit gets bigger the swings are greater. Two factors contribute to this effect:

1. As was previously shown, the dependence of power consumption with temperature is not monotone. Increasing the number of dies causes temperature to increase.

2. Vertical and horizontal power grid resistance is not the same. Increasing the number of layers causes vertical resistance to increase since more TSV are added in series, while horizontal resistance decreases since the area on one die decreases. The relative change between vertical and horizontal resistance is not monotone. The effect of this factor is clearly shown in the next experiment, where the simulator is run without taking into account the dependence of power consumption on power and temperature.

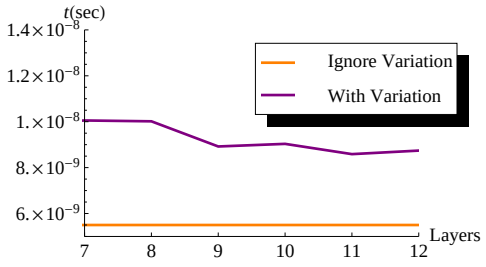
4.4.7 Impact of temperature and supply voltage variability on performance

The aim of this experiment is to quantify the effect of power supply voltage and temperature variability on performance. Uniform 10% TSV density is assumed.

Figure 4.22a shows the estimate of system timing for a different number of dies in case variability is taken into account and in the case it is not. When variability is ignored, the same timing is estimated independent of the number of dies. When variability is taken into account, the effects discussed in the previous experiment are observed. Figure 4.22b shows the relative error in timing estimation for each number of dies. In all cases, the error is between 35% and 45%.

Figure 4.22: Impact of temperature and supply voltage variability on timing in case of 500 instances of b19

(a) Timing



(b) Relative Timing Error

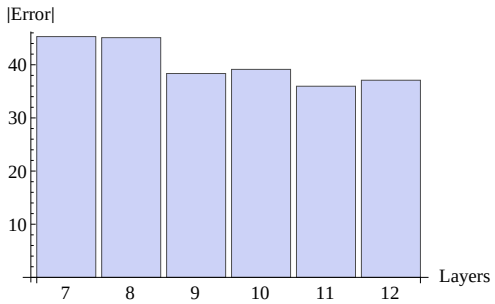
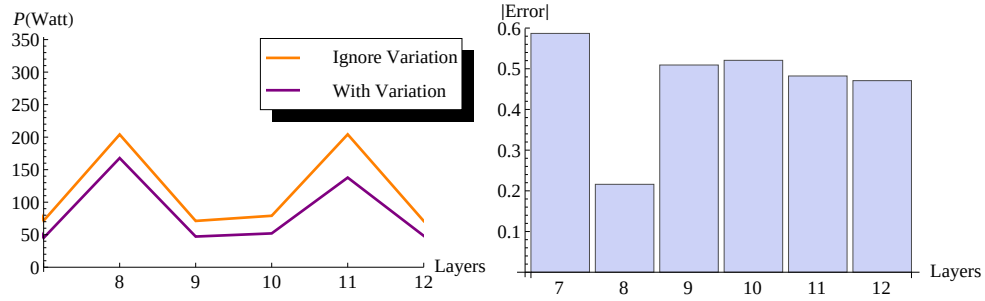


Figure 4.23a shows the estimate of system power consumption for a different number of dies in case variability is taken into account and in the case it is not. The trend is similar in both cases. This trend occurs mainly due to the non monotone relative change between vertical and horizontal power grid resistance as discussed in the previous experiment. However there is a significant offset between the two curves due to variability. Figure 4.23b shows the relative error in power consumption estimation for each number of dies. In all cases but one, the error is between 48% and 58%.

Figure 4.23: Impact of temperature and supply voltage variability on power consumption in case of 500 instances of b19

(a) Power Consumption (b) Relative Power Consumption Error



4.4.8 Summary of Results obtained from the Adaptive Electrothermal Simulator

In this section, a series of experiments was performed to investigate the impact of vertically stacking dies on power supply voltage, temperate, power consumption and timing. To begin with, the power supply was shown to be the dominant limiting factor when designing systems in the order of tens of million gates. Temperature increases almost linearly with die number, independent of the size of each die, since the dominant heat flow path is the vertical. Power supply voltage increases logarithmically up to a point of saturation with increased die number. In contrast to voltage and temperature, which have a monotone behavior, more complicated mechanism determine the performance of the system. Timing was shown to have 4 regions where either temperature or voltage is the dominant limiting factor. Power supply voltage swings when increasing the die number. Therefore optimum system performance is in no way linked with temperature and supply voltage, as is assumed in the literature so far. The variability of power supply voltage and temperature causes on average 40% increase in timing and 53% increase in power consumption, compared to the

assumption of nominal conditions. This renders the traditional 5 corner–based design flow inappropriate for large designs.

4.5 Experiments on Multi–Objective Floorplan Optimization

In this section, a series of experiments is conducted in order to study the behavior of NSGAII on 3D IC floorplanning problem, using the proposed representation, and mutation and crossover operators. Note that performance is not studied in this work, and therefore no parameter tuning is performed. Also, since the whole electrothermal simulation and optimization framework is implemented in Mathematica™, and therefore execution is very slow, compared to C++ executable, the genetic algorithm is tested on small problem instances. 4 different optimization schemes are studied:

1. Optimize power consumption and timing.
2. Optimize power consumption and area.
3. Optimize area and timing.
4. Optimize power consumption, timing and area..

The aforementioned optimization schemes have been tested on several Common Power Supply Hyper Modules with similar results. Here results for the case of 8 instances of b19, 12 instances of b14 are shown.

In all experiments the following parameters are used:

1. Population size 15.
2. 90% probability of mutation.
3. 90% probability of crossover.
4. 10% uniform TSV density.
5. $1.1V$ supply voltage.
6. 45^oC package temperature.

Note that in all pareto front illustrations, the light blue line denotes the pareto front, green dots denote pareto optimal solutions and red dots denote the non pareto optimal solutions of the population. In floorplan solutions illustrations, all cubes are shown to have thickness $30um$ for a better optical result.

4.5.1 Optimization for Power and Timing

Figure 4.24 shows the convergence of the pareto front in case of 8 instances of b19, 12 instances of b14, optimizing for power consumption and timing. The pareto front is found within 8 generations. A clear tradeoff is observed. Sacrificing 35% of timing, power consumption improves by 40%. It is interesting to note that during evolution, solutions show significant spread in terms of power consumptions and therefore verify that NSGAII preserves solution diversity. Figure 4.25 shows 4 of the resulting solutions at the end of 8 generations, 2 of which are optimum in terms of power and 2 in terms of timing. Optimum solutions in terms of power utilize more layers to improve IR–drop, while optimum solutions in terms of timing utilize less layers to lower temperature. Note that it is safe here to assume a monotone behavior of power consumption and timing since the circuit is very small.

Figure 4.24: Convergence of Pareto front in case of 8 instances of b19, 12 instances of b14, optimizing for power consumption and timing

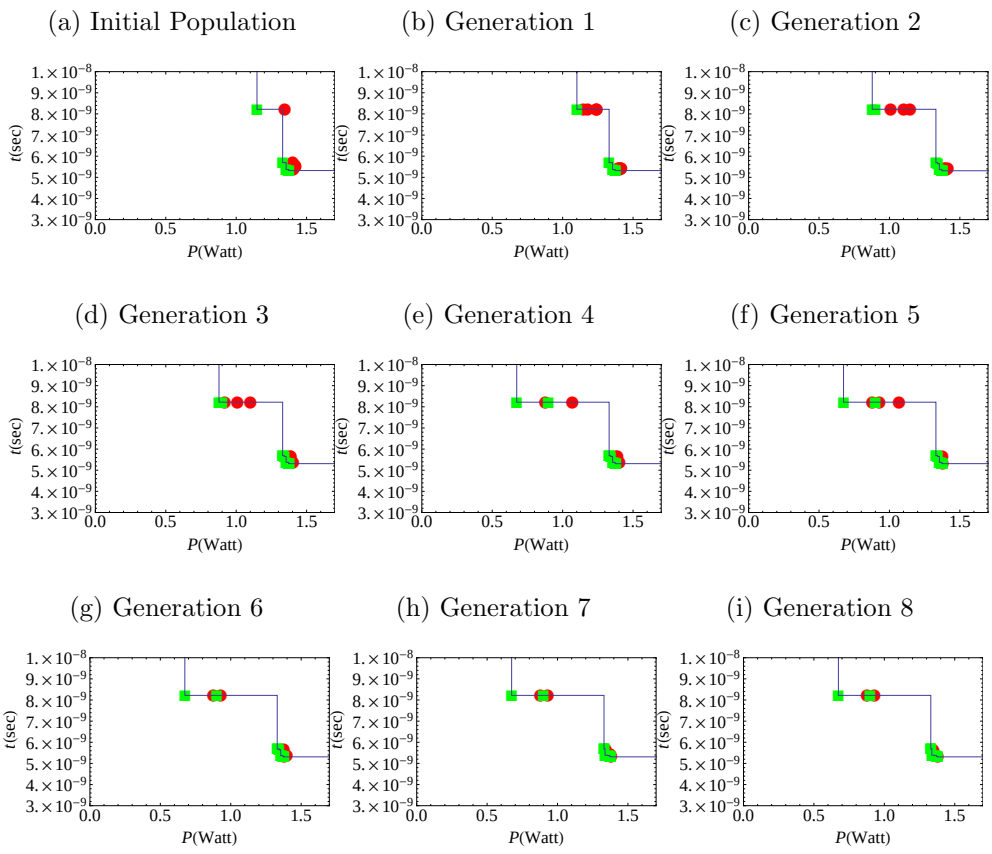
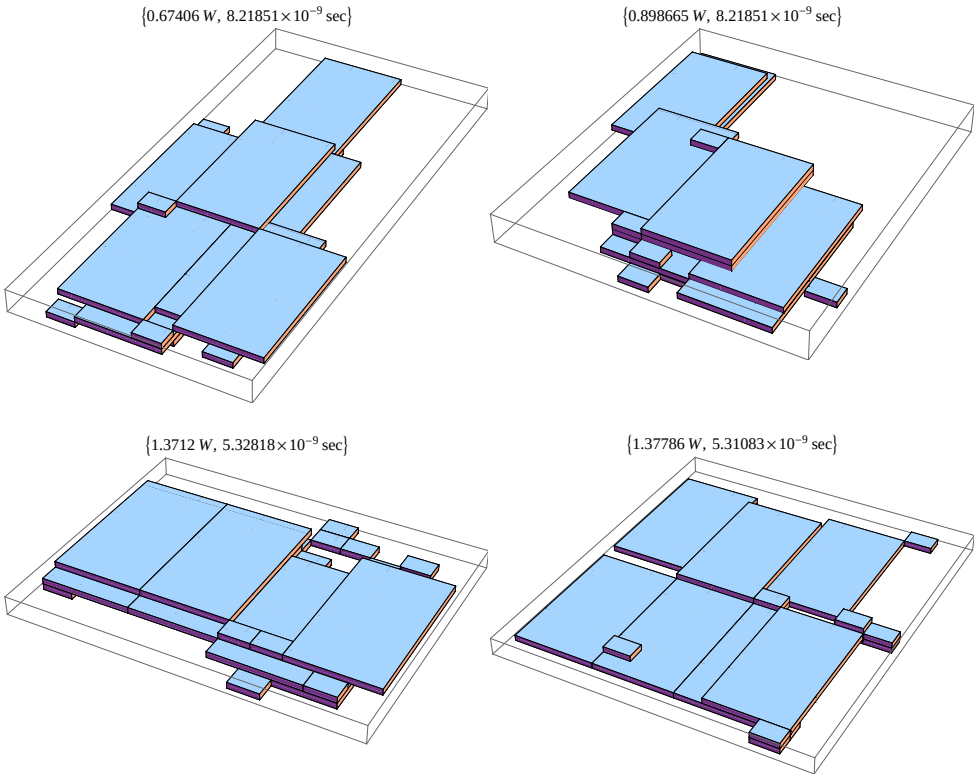


Figure 4.25: Solutions in case of 8 instances of b19, 12 instances of b14, optimizing for power consumption and timing



4.5.2 Optimization for Power and Area

Figure 4.26 shows the convergence of the pareto front in case of 8 instances of b19, 8 instances of b14, optimizing for power consumption and area. The pareto front is found within 10 generations. No tradeoff is observed between power consumption and area, and no significant diversity of solutions is observed during evolution. Figure 4.27 shows two resulting solutions at the end of 10 generations of solutions. Optimum power consumption can be linked with optimum area. Unnecessary area results in bigger power grid and therefore greater IR–drop and power consumption. Therefore there is no need to optimize for power and area as separate objectives.

4.5.3 Optimization for Area and Timing

Figure 4.28 shows the convergence of the pareto front in case of 8 instances of b19, 12 instances of b14, optimizing for power consumption and area. The pareto front is found within 8 generations. A clear tradeoff is observed. Sac-

Figure 4.26: Convergence of Pareto front in case of 8 instances of b19, 12 instances of b14, optimizing for power consumption and area

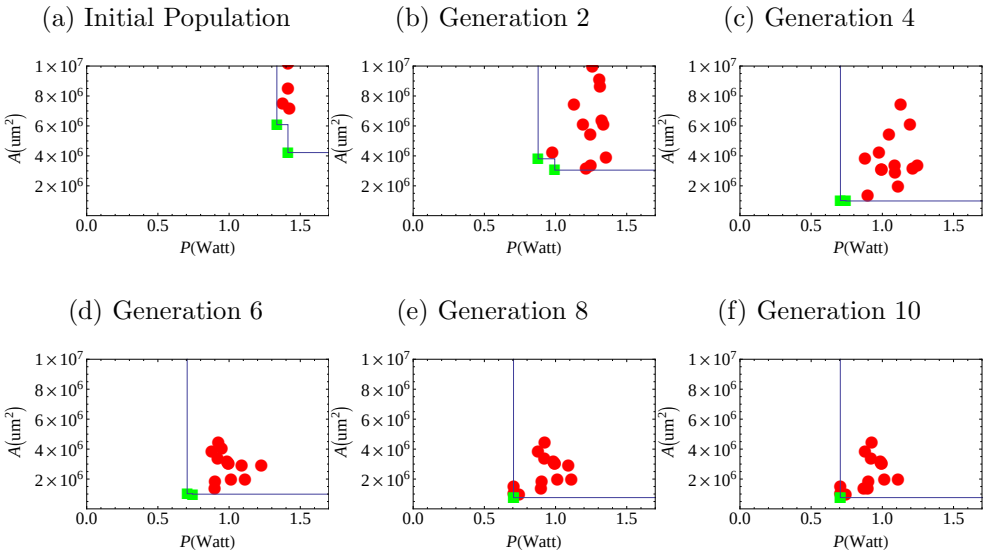
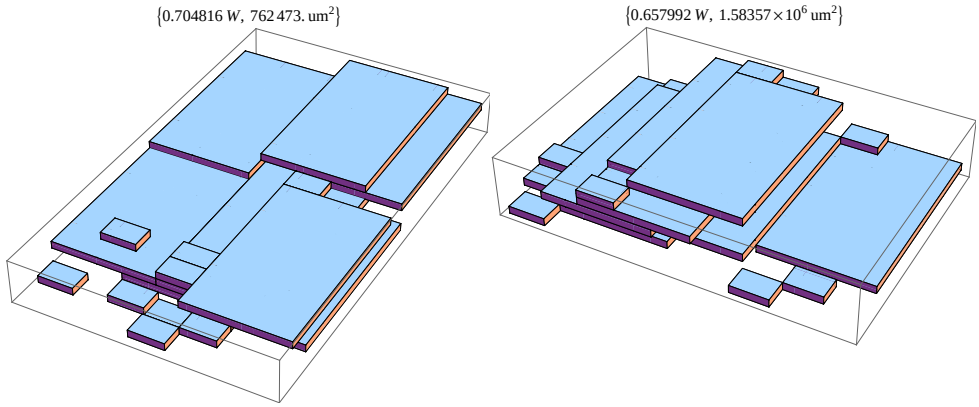
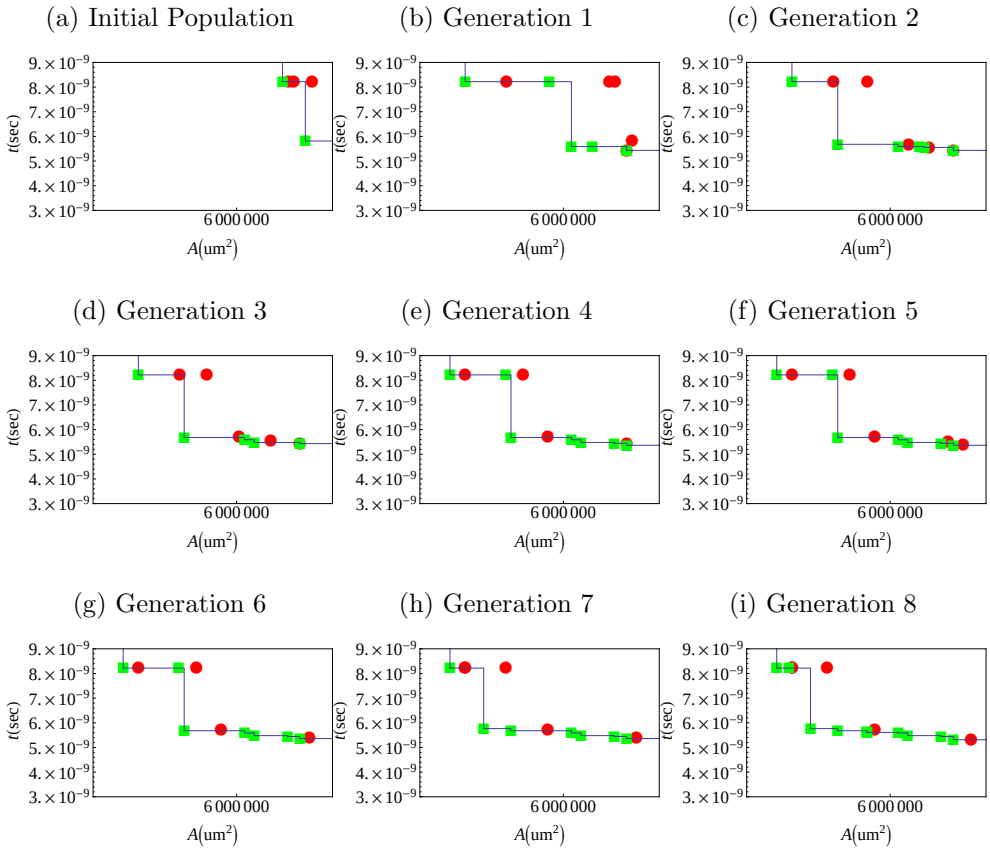


Figure 4.27: Solutions in case of 8 instances of b19, 12 instances of b14, optimizing for power consumption and area



rificing 50% of area, power consumption improves by 20%. It is interesting to note that solutions show significant spread in terms of area and therefore verify that NSGAII preserves solution diversity. Figure 4.29 shows 4 of the resulting solutions at the end of 8 generations of solutions. Wasting area to place modules far from each other reduces temperature and improves timing. However the area price is considered too big for the timing gain. Note that it is safe here to assume a monotone behavior of timing since the circuit is very small.

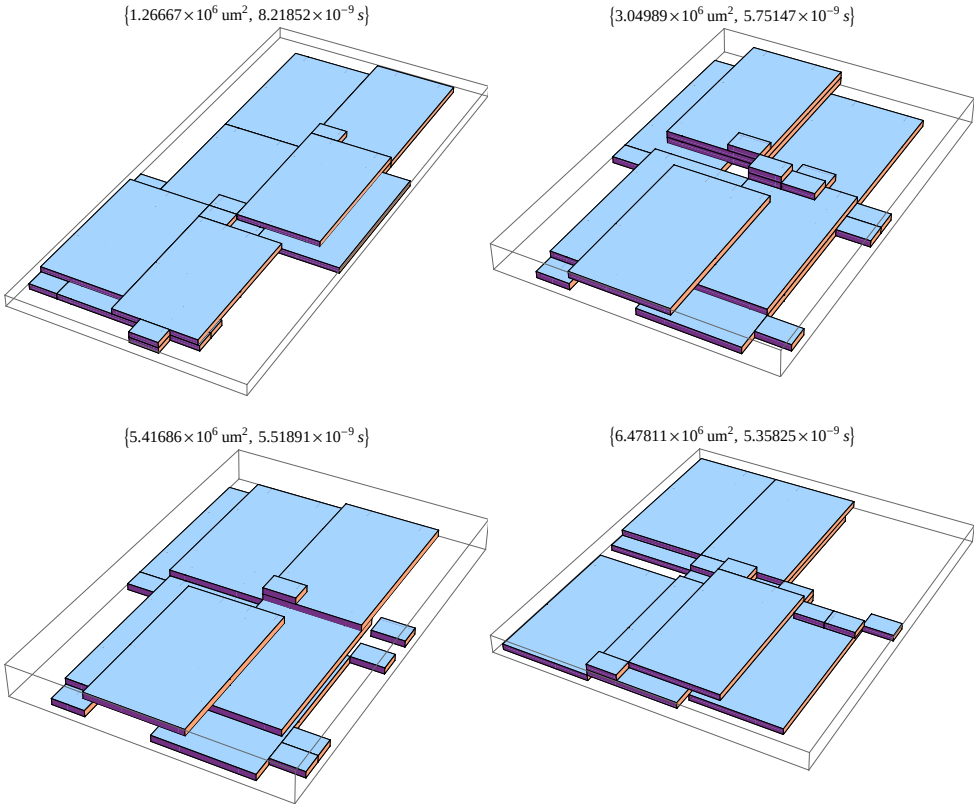
Figure 4.28: Convergence of Pareto front in case of 8 instances of b19, 12 instances of b14, optimizing for area and timing



4.5.4 Optimization for Power, Timing and Area and Structure of 3D Pareto Front

In this experiment, an attempt was made to use the proposed genetic algorithm to optimize simultaneously for power consumption, timing and area. However, NSGAII failed because it could not find dominating solutions. Figures 4.30, 4.31 and 4.32 show 3 generations in case of 8 instances of b19, 12 instances of b14, optimizing for power consumption and timing, for power consumption and area and for area and timing respectively. It can be observed that pareto front in one pair of parameters is not convex causing NSGAII ranking to fail.

Figure 4.29: Solutions in case of 8 instances of b19, 12 instances of b14, optimizing for area and timing

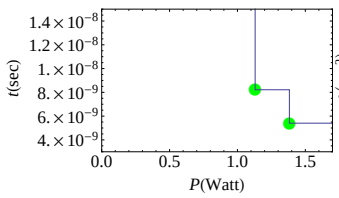


4.5.5 Summary of Results obtained from the Genetic Algorithm

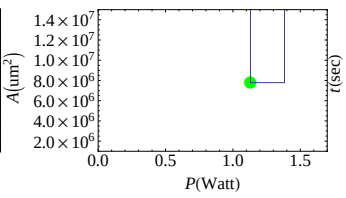
In this section, a series of experiments was conducted in order to study the behavior of NSGAII on 3D IC floorplanning problem, using the proposed representation, and mutation and crossover operators. NSGAII was shown to operate properly and give reasonable solutions in all the bi–objective optimization schemes. However, it failed completely to find dominating solutions, in case of tri–objective optimization. Optimizing for power consumption and timing yielded better floorplans and showed clear and useful tradeoffs. Optimizing for power consumption and area showed that optimum power consumption and area are closely linked and it is useless to treat them as separate objectives. Optimizing for area and timing yielded useless tradeoffs between area and timing and therefore considered inappropriate.

Figure 4.30: 3D Pareto Front in case of 8 instances of b19, 12 instances of b14, optimizing for power consumption and timing

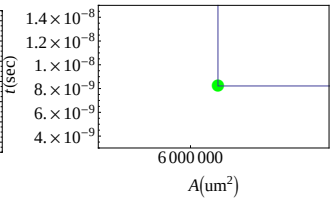
(a) Initial Population, Power vs Timing



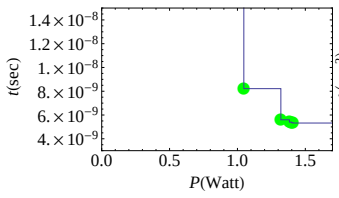
(b) Initial Population, Power vs Area



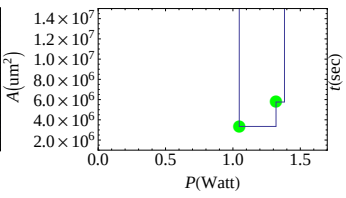
(c) Initial Population, Area vs Timing



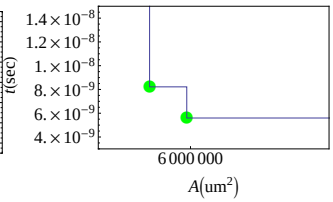
(d) Generation 1, Power vs Timing



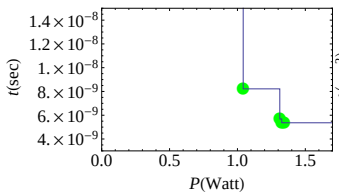
(e) Generation 1, Power vs Area



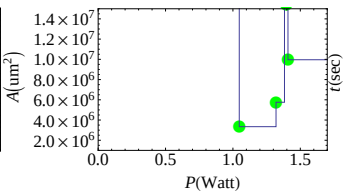
(f) Generation 1, Area vs Timing



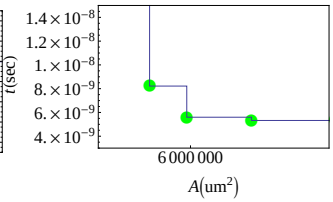
(g) Generation 2, Power vs Timing



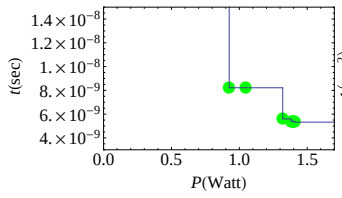
(h) Generation 2, Power vs Area



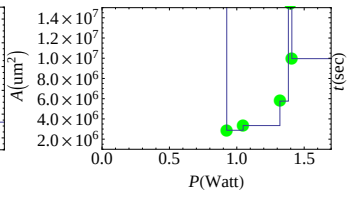
(i) Generation 2, Area vs Timing



(j) Generation 3, Power vs Timing



(k) Generation 3, Power vs Area



(l) Generation 3, Area vs Timing

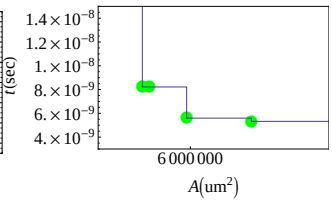
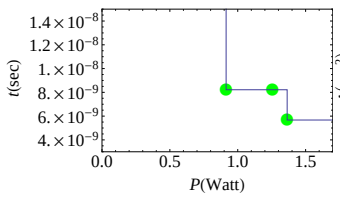
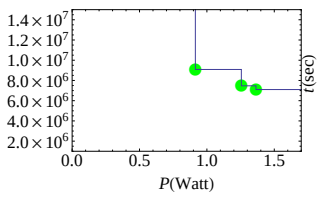


Figure 4.31: 3D Pareto Front in case of 8 instances of b19, 12 instances of b14, optimizing for power consumption and area

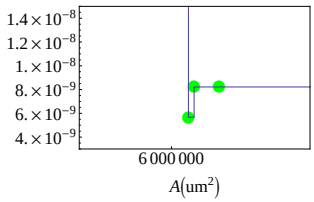
(a) Initial Population, Power vs Timing



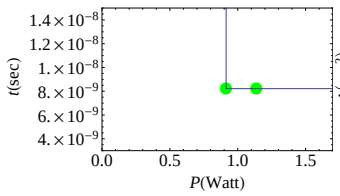
(b) Initial Population, Power vs Area



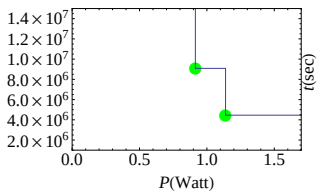
(c) Initial Population, Area vs Timing



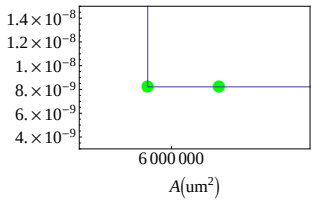
(d) Generation 1, Power vs Timing



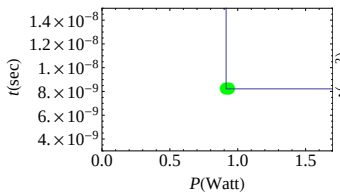
(e) Generation 1, Power vs Area



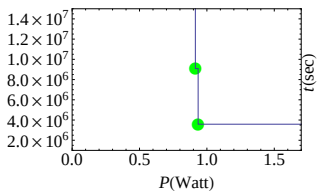
(f) Generation 1, Area vs Timing



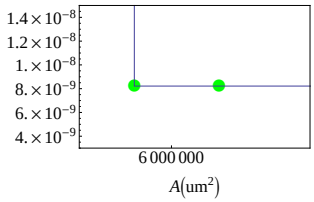
(g) Generation 2, Power vs Timing



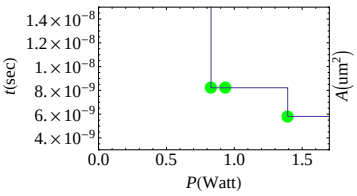
(h) Generation 2, Power vs Area



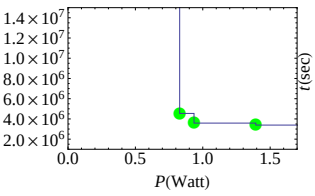
(i) Generation 2, Area vs Timing



(j) Generation 3, Power vs Timing



(k) Generation 3, Power vs Area



(l) Generation 3, Area vs Timing

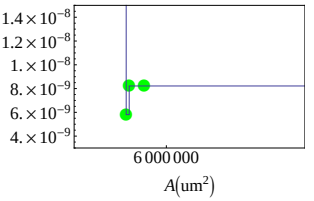
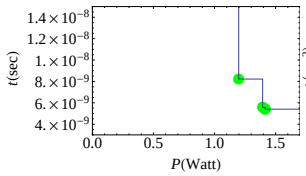
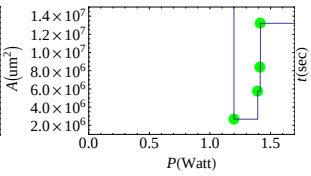


Figure 4.32: 3D Pareto Front in case of 8 instances of b19, 12 instances of b14, optimizing for area and timing

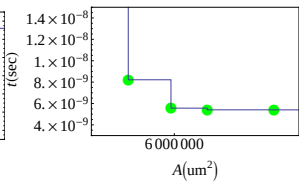
(a) Initial Population, Power vs Timing



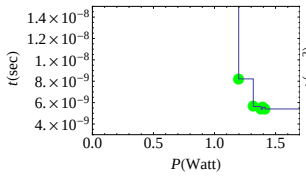
(b) Initial Population, Power vs Area



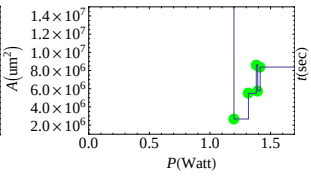
(c) Initial Population, Area vs Timing



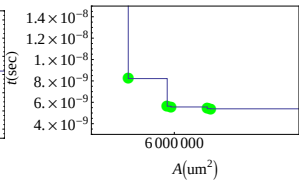
(d) Generation 1, Power vs Timing



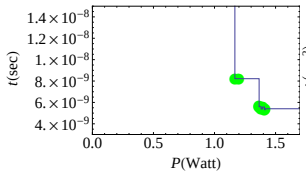
(e) Generation 1, Power vs Area



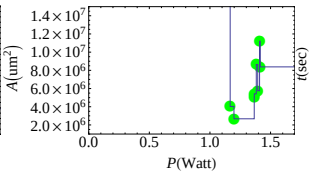
(f) Generation 1, Area vs Timing



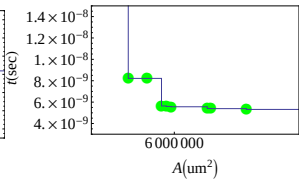
(g) Generation 2, Power vs Timing



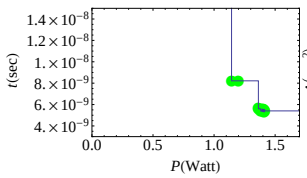
(h) Generation 2, Power vs Area



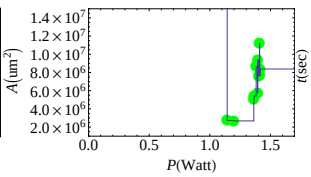
(i) Generation 2, Area vs Timing



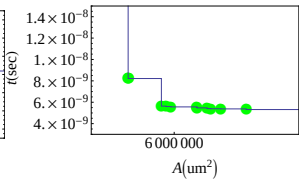
(j) Generation 3, Power vs Timing



(k) Generation 3, Power vs Area



(l) Generation 3, Area vs Timing



Chapter 5

Conclusions

In this work, a novel design flow to perform design exploration in 3D ICs was proposed. In this framework, power consumption, timing and fabrication cost are the design goals. Power supply voltage and thermal variations are taken into account to allow accurate performance predictions. An extensive pre–characterization step aids fast and accurate whole–system performance predictions. Extreme Value Theory is used to speed–up worst case power estimation. 3 levels of hierarchy are assumed: Modules comprising gates, Common Power Supply Hyper Modules comprising modules under the same power supply, the whole Chip, comprising Common Power Supply Hyper Modules. Floorplanning is a two a two–step procedure. Initially, Common Power Supply Hyper Modules are treated. Finally, the whole chip is designed. Most parts of the proposed framework were implemented and used to investigate the impact of temperature and power supply voltage variations on performance prediction in 3D ICs, on 45nm. The ITC99 benchmark circuits were used along with standard cells from Nangate45.

In section 4.2 the standard cells of Nangate45 were characterized in 99 operating condition pairs of supply voltage and temperate, and their behavior as a function of voltage and temperature was extracted. In all parameters of the standard cells, such as delay or dynamic power, strong dependence is observed on supply voltage and temperature. Changing the voltage by $0.1V$ or the temperature by 50^oC can result in more than 100% change in the value of the parameter.

In section 4.3 a series of experiments was conducted in order to configure the univariate statistical prediction engine and characterize the modules in terms of power consumption and timing as a function of operating conditions. It was shown that for modules in the order of 10000 gates, only 1000 excitations are sufficient to calculate a realistic worst case estimate of power consumption. For a modules in the order of 200000 gates, only 5000 excitations were required. The behavior of 3 modules from the ITC benchmarks, in terms of power consumption and timing, as a function of operating conditions was extracted and reported.

Slight changes of voltage by $0.1V$ or temperature by 50^oC can cause power consumption or timing to change by more than 300%.

In section 4.4, a series of experiments was performed to investigate the impact of vertically stacking dies on power supply voltage, temperate, power consumption and timing. To begin with, the power supply was shown to be the dominant limiting factor when designing systems in the order of tens of million gates. Temperature increases almost linearly with die number, independent of the size of each die, since the dominant heat flow path is the vertical. Power supply voltage increases logarithmically up to a point of saturation with increased die number. In contrast to voltage and temperature, which have a monotone behavior, more complicated mechanism determine the performance of the system. Timing was shown to have 4 regions where either temperature or voltage is the dominant limiting factor. Power supply voltage swings when increasing the die number. Therefore optimum system performance is in no way linked with temperature and supply voltage, as is assumed in the literature so far. The variability of power supply voltage and temperature causes on average 40% increase in timing and 53% increase in power consumption, compared to the assumption of nominal conditions. This renders the traditional 5 corner–based design flow inappropriate for large designs.

In section 4.5, a series of experiments was conducted in order to study the behavior of NSGAII on 3D IC floorplanning problem, using the proposed representation, and mutation and crossover operators. NSGA–II was shown to operate properly and give reasonable solutions in all the bi–objective optimization schemes. However, it failed completely to find dominating solutions, in case of tri–objective optimization. Optimizing for power consumption and timing yielded better floorplans and showed clear and useful tradeoffs. Optimizing for power consumption and area showed that optimum power consumption and area are closely link and it is useless to treat them as separate objectives. Optimizing for area and timing yielded useless tradeoffs between area and timing and therefore considered inappropriate.

Bibliography

[1] www.cad.polito.it/downloads/tools/itc99.html.

[2] www.eda.ncsu.edu/wiki/FreePDK.

[3] www.nangate.com.

[4] www.opensourceliberty.org.

[5] www.wolfram.com.

[6] J. Beirlant, Y. Goegebeur, J. Teugels, and J. Segers. *Statistics of Extremes: Theory and Applications*. Dec. 2005.

[7] J. Bhasker and R. Chadha. *Static Timing Analysis for Nanometer Designs*. Feb. 2011.

[8] Y. Cai, B. Liu, and Q. Zhou. A thermal aware floorplanning algorithm supporting voltage islands for low power SoC design. *Integrated Circuit and System Design*, Jan. 2005.

[9] E. Castillo, A. S. Hadi, N. Balakrishnan, and J. M. Sarabia. *Extreme Value and Related Models with Applications in Engineering and Science*. July 2008.

[10] L. Cheng, L. Deng, and M. Wong. Floorplanning for 3-D VLSI design. *Design Automation Conference, 2005. Proceedings of the ASP-DAC 2005. Asia and South Pacific*, Jan. 2005.

[11] J. Cong and G. Luo. A 3D Physical Design Flow Based on Open Access. In *2009 International Conference on Communications, Circuits and Systems (ICCCAS)*, pages 1103–1107. IEEE, 2009.

[12] F. Corno, M. Reorda, and G. Squillero. RT-level ITC'99 benchmarks and first ATPG results. *IEEE Design & Test of Computers*, 17(3):44–53, 2000.

[13] K. Deb and Pratap. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[14] X. Dong and Y. Xie. System-level cost analysis and design exploration for three-dimensional integrated circuits (3D ICs). In *ASP-DAC '09: Proceedings of the 2009 Asia and South Pacific Design Automation Conference*. IEEE Press, Jan. 2009.

[15] 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*, 29(12):1959–1972, 2010.

[16] P. Embrechts. Copulas: A personal view. *Journal of Risk and Insurance*, Jan. 2009.

[17] N. Evmorfopoulos and M. Rammou. Characterization of the worst-case current waveform excitations in general RLC-model power grid analysis. *Computer-Aided Design (ICCAD), 2010 IEEE/ACM International Conference on*.

[18] N. Evmorfopoulos and G. Stamoulis. A Monte Carlo approach for maximum power estimation based on extreme value theory. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, Jan. 2002.

[19] P. Falkenstern, Y. Xie, and Y. Chang. Three-dimensional integrated circuits (3D IC) Floorplan and Power/Ground Network Co-synthesis. *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific*, Jan. 2010.

[20] R. Fischbach and J. Lienig. From 3D circuit technologies and data structures to interconnect prediction. *Proceedings of the 11th ...*, Jan. 2009.

[21] K. Fujiyoshi and H. Kawai. DTS: A tree based representation for 3D-block packing. *Circuits and Systems*, Jan. 2007.

[22] L. d. Haan and A. Ferreira. *Extreme Value Theory An Introduction*. Dec. 2007.

[23] X. He, S. Dong, and Y. Ma. Simultaneous buffer and interlayer via planning for 3D floorplanning. *Quality of Electronic Design*, Jan. 2009.

[24] W. Hung, G. Link, and Y. Xie. Interconnect and thermal-aware floorplanning for 3D microprocessors. *Quality Electronic Design, 2006. ISQED '06. 7th International Symposium on*, Jan. 2006.

[25] P. Jaworski, F. Durante, and W. Härdle. *Copula Theory and Its Applications: Proceedings of the Workshop Held in Warsaw, 25-26 September 2009*. Jan. 2010.

[26] S. Kotz and S. Nadarajah. *Extreme Value Distributions: Theory and Applications*. Mar. 2010.

[27] F. Kreith. *The CRC handbook of thermal engineering*. Springer, Dec. 2000.

[28] W. Lee, H. Liu, and Y. Chang. Voltage island aware floorplanning for power and timing optimization. *ICCAD '06: Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, Jan. 2006.

[29] X. Li, Y. Ma, X. Hong, and S. Dong. Thermal-aware incremental floorplanning for 3D ICs. *ASIC, 2007. ASICON '07. 7th International Conference on*, pages 1092–1095, 2007.

[30] Z. Li, X. Hong, Q. Zhou, S. Zeng, and J. Bian. Integrating dynamic thermal via planning with 3D floorplanning algorithm. *ISPD '06 Proceedings of the 2006 international symposium on Physical design*, Jan. 2006.

[31] C. Liu and Y. Chang. Floorplan and power/ground network co-synthesis for fast design convergence. *ISPD '06 Proceedings of the 2006 international symposium on Physical design*, Jan. 2006.

[32] Y. Liu, Y. Ma, E. Kursun, G. Reinman, and J. Cong. Fine grain 3D integration for microarchitecture design through cube packing exploration. *Computer Design, 2007. ICCD 2007. 25th International Conference on*, pages 259–266, 2007.

[33] Y. Ma, X. Hong, and S. Dong. 3D CBL: An efficient algorithm for general 3D packing problems. *Circuits and Systems*, Jan. 2006.

[34] T. Mikosch. Copulas: Tales and facts. *Extremes*, Jan. 2006.

[35] G. Moore. Cramming more components onto integrated circuits. In *Proceedings of the IEEE*, 1998.

[36] A. Moraglio. Towards a Geometric Unification of Evolutionary. *eden.dei.uc.pt*, Jan. 2007.

[37] R. Nelsen. *An introduction to copulas*. Jan. 2006.

[38] H. Ohta, T. Yamada, and C. Kodama. The O-Sequence: Representation of 3D-floorplan dissected by rectangular walls. *Research in Microelectronics and Electronics 2006, Ph. D.*, Jan. 2006.

[39] A. Papanikolaou, D. Soudris, and R. Radojcic. *Three Dimensional System Integration: IC Stacking Process and Design*. Jan. 2010.

[40] Q. Qiu and Q. Wu. Maximum power estimation using the limiting distributions of extreme order statistics. *DAC '98 Proceedings of the 35th annual Design Automation Conference*, Jan. 1998.

[41] R. Reiss. *Statistical analysis of extreme values: with applications to insurance, finance, hydrology and other fields*. Jan. 2007.

[42] G. Salvadori, C. de Michele, and N. Kottegoda. *Extremes in nature. An approach using copulas*. Jan. 2007.

[43] K. Siozios and A. Papanikolaou. A method and tool for early design/technology search-space exploration for 3D ICs. *16th IFIP Int. Conf.*, Jan. 2008.

[44] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-Aware Microarchitecture. In *30th Annual International Symposium on Computer Architecture, 2003. Proceedings.*, pages 2–13. IEEE Comput. Soc.

[45] T. Taghavi, X. Yang, and B.-K. choi. Dragon2005: large-scale mixed-size placement tool. In *ISPD '05: Proceedings of the 2005 international symposium on Physical design*. ACM Request Permissions, Apr. 2005.

[46] E.-G. Talbi. *Metaheuristics: from design to implementation*. Jan. 2009.

[47] Y.-F. Tsai, F. Wang, Y. Xie, N. Vijaykrishnan, and M. Irwin. Design Space Exploration for 3-D Cache. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(4):444–455, 2008.

[48] J. Vlach. Computer methods for circuit analysis and design, 1983.

[49] R. Wang, E. Young, Y. Zhu, F. Chung, and G. Ronald. 3-D Floorplanning Using Labeled Tree and Dual Sequences ABSTRACT. *Citeseer*, 2008.

[50] R. Weerasekera and D. Pamunuwa. Two-dimensional and three-dimensional integration of heterogeneous electronic systems under cost, performance, and technological constraints. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, Jan. 2009.

[51] Y. Xie, J. Cong, and S. Sapatnekar. *Three Dimensional Integrated Circuit Design: EDA, Design and Microarchitectures*. Jan. 2009.

[52] H. Yamagishi and H. Ninomiya. Three dimensional module packing by simulated annealing. *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, Jan. 2005.

[53] H. Yamazaki, K. Sakanushi, and S. Nakatake. The 3D-packing by meta data structure and packing heuristics. *Citeseer*, Jan. 2000.

[54] Y. Yang, J. Wang, and R. Dick. TAPHS: Thermal-aware unified physical-level and high-level synthesis. *Design Automation, 2006. Asia and South Pacific Conference on*, Jan. 2006.

[55] P. Yuh and C. Yang. Temporal floorplanning using the T-tree formulation. *ICCAD '04 Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, Jan. 2004.

[56] P. Yuh, C. Yang, and Y. Chang. Temporal floorplanning using 3D-subTCG. *ASP-DAC '04 Proceedings of the 2004 Asia and South Pacific Design Automation Conference*, Jan. 2004.

[57] Y. Zhan and S. Sapatnekar. Fast computation of the temperature distribution in VLSI chips using the discrete cosine transform and table look-up. *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, pages 87–92, 2005.

[58] P. Zhou, Y. Ma, Z. Li, R. Dick, and L. Shang. 3D-STAF: scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits. *ICCAD '07 Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design*, Jan. 2007.