January 2015

Robust, Low Power, Discrete Gate Sizing

Anthony Joseph Casagrande
University of South Florida, acasagra@mail.usf.edu

Robust, Low Power, Discrete Gate Sizing

by

Anthony J. Casagrande

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Engineering
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Nagarajan Ranganathan, Ph.D.
Srinivas Katkoori, Ph.D.
Swaroop Ghosh, Ph.D.

Date of Approval:
May 4, 2015

Keywords: VLSI, Design Automation,
Variation-aware, Game Theory, Fuzzy Mathematics

Dedication

I would like to dedicate this thesis to my beautiful wife, Patricia Casagrande, for her neverending love and support. I would also like to dedicate this work to my parents Mark and Melanie, and brother Carlo for all of their love, help and encouragement along the way. Finally, I dedicate this thesis to my advisor, Dr. Ranganathan - not just because of his valuable advice and assistance, but because he is a genuinely caring man with exemplary character. He is not only a mentor and an inspiration, but a true friend.

Acknowledgment

I would like to acknowledge the exceptional faculty in the Computer Science and Engineering Department at the University of South Florida. Also, I would like to recognize the outstanding administrative staff in the CSE office and Mrs. Catherine Burton for her assistance in editing this document.

Table of Contents

List of Tables

List of Figures

Abstract

Ultra-deep submicron circuits require accurate modeling of gate delay in order to meet aggressive timing constraints. With the lack of statistical data, variability due to the mechanical manufacturing process and its chemical properties poses a challenging problem. Discrete gate sizing requires (i) accurate models that take into account random parametric variation and (ii) a fair allocation of resources to optimize the solution. The proposed GTFUZZ gate sizing algorithm handles both tasks. Gate sizing is modeled as a resource allocation problem using fuzzy game theory. Delay is modeled as a constraint and power is optimized in this algorithm. In GTFUZZ, delay is modeled as a fuzzy goal with fuzzy parameters to capture the imprecision of gate delay early in the design phase when extensive empirical data is absent. Dynamic power is modeled as a fuzzy goal without varying coefficients. The fuzzy goals provide a flexible platform for multimetric optimization. The robust GTFUZZ algorithm is compared against fuzzy linear programming (FLP) and deterministic worst-case FLP (DWCFLP) algorithms. The benchmark circuits are first synthesized, placed, routed, and optimized for performance using the Synopsys University 32/28nm standard cell library and technology files. Operating at the optimized clock frequency, results show an average power reduction of about 20% versus DWCFLP and 9% against variation-aware gate sizing with FLP. Timing and timing yield are verified by both Synopsys PrimeTime and Monte Carlo simulations of the critical paths using HSPICE.

Chapter 1: Introduction

1.1 Robust Gate Sizing

Ultra-deep submicron circuits are affected by variability due to the mechanical manufacturing process. These effects are random and are not completely understood. There is a wide variety of work being done on emerging technologies such as spintronics, neuromorphic, molecular, and quantum computing. However, Sani Nassif at IBM, a pioneer of this problem since the late 90's, stated in 2010 that due to the continual scaling of silicon CMOS technology, this problem will be of urgent importance to designers for at least the next decade until emerging technologies become a practical substitution [33]. If we are to adhere to Moore's Law, we must address these issues in order to prepare for a world beyond CMOS devices.

The variation effects in mechanical manufacturing process cause many characteristics of transistors to vary. Spatial variance affects physical characteristics such as effective channel length and oxide thickness. Temporal variance refers to effects which occur over time. For example, the effects of electromigration can gradually wear down interconnects, potentially creating an open circuit. Front-end device variability accounts for 90% of process variation [32]. Gate sizing is a proven technique to optimize VLSI circuits. Gate sizing involves adjusting the size of each individual gate to optimize selected metrics. Robust gate sizing must

Previously published in [10]. Permission included in Appendix A.

take into account randomness of the manufacturing process without being overly pessimistic - the deterministic worst-case assumption of variation is no longer a viable option.

We propose a novel solution to the robust gate sizing problem using fuzzy games. In these fuzzy games, gate delay is modeled as a fuzzy goal with fuzzy varying coefficients and dynamic power is modeled as a fuzzy goal without varying coefficients. The solution to the fuzzy game is a robust setting of gates which are modeled as players in the game. This captures the random parametric variation while simultaneously addressing the resource allocation problem of gate sizing, yielding an optimal solution in which each player realizes maximum gain. The merging of the normalized fuzzy goals naturally creates a platform for multimetric optimization of other emerging metrics of importance such as security. Defining multiple fuzzy goals is an efficient way of dealing with many complex decisions that must be made in the optimization process amongst a group of agents while guaranteeing an equilibrium ("fairness") of gain among all players in the process.

1.2 Background and Motivation

Design for manufacturability anticipates defects in all phases of the design process, improving yield, performance, and reliability of the circuit. As illustrated in figure 1, there have been many efforts to strengthen the robustness of integrated circuits via gate sizing. Seminal works in gate sizing introduced several vital techniques which have been applied and reinvented throughout the past 20-30 years such as TILOS optimization [16] and constrained linear programming [8, 13]. TILOS is an iterative heuristic transistor sizing algorithm which has been adapted to gate sizing in several works [41, 36, 37, 22].

Deterministic worst-case margins have long been used in the design process. Variation in the manufacturing process may cause many chips in a batch to fail timing constraints. Rather than binning or discarding these chips, the design is adjusted so that all chips in a manufacturing run will meet the design's timing constraint. There are a wide variety of works in the literature which are not variation-aware [41, 38, 36, 31, 21, 9, 22]. With feature sizes as low as 22nm, the density of the chip is much greater and smaller devices are placed in closer proximity. As the ratio of feature size to parametric variation worsens, design changes which anticipate and prevent chip failure with absolute certainty become more and more pessimistic. Although the timing yield of a batch is now 100%, most of the chips which would have met the constraints in the first place have taken a hit in performance and area.

Many variation-aware optimization techniques have been introduced in the late 90's and early 2000's [40, 37, 12, 35, 4, 19, 14, 23, 6, 28, 34, 17, 26, 25, 18]. The advent of Statistical Static Timing Analysis (SSTA) was one of the first variation-aware paradigm shifts. Rather than traditional Static Timing Analysis (STA), the minimum and maximum arrival times of signals at the inputs of a gate are modeled statistically, capturing the effects of parametric variation. A framework of an optimal SSTA engine and its effectiveness was presented in [40] and has been implemented as a means to model timing in many gate sizing works [37, 12, 35, 4, 19, 14].

In [28] and [27] the problem is approached using statistical optimization with timing constraints (arrival time) derived from traditional STA operations. The propagation delay of each gate is modeled with stochastic coefficients, taking into account the randomness of parameters bound by the best and worst case. These bounds are empirically determined by

the technology and manufacturing process. The problem is then formulated as a robust linear program. Compared to the statistical approach, optimization with fuzzy linear programming has been shown to yield an equal or greater reduction in dynamic power consumption while ensuring the circuit is robust [26, 25]. The theory of fuzzy sets is useful when modeling systems which are imprecise [29]. Parametric variations are non-deterministic, and average behavior is difficult to predict.

As opposed to boolean logic, fuzzy sets introduce a degree of truth. When faced with uncertainty, this degree of truth provides a more accurate representation of the lack of information. Using fuzzy sets and membership functions to model uncertain parameters not only simplifies computation, it also forms a more accurate model of parametric variability. The existing fuzzy approach optimizes only a few parameters while ignoring the negative impacts on others. This can be addressed by using game theoretic modeling [21, 31].

Game theory and auction theory have been shown to be effective strategies in resource allocation problems (gate sizing, wire sizing, buffer insertion) in VLSI Design Automation [21, 31]. The problem can be formulated as a non-cooperative game with each gate representing a player competing for multiple resources in the circuit. The game forms an accurate model of the conflicting resource allocation in the system. The Nash Equilibrium algorithm finds a fair solution for all players - a solution which yields the maximum gain for all players participating in the game. This approach has been proven to be more effective in power minimization and runtime as compared to genetic search based algorithms and simulated annealing [21], but has not yet been compared to fuzzy linear programming.

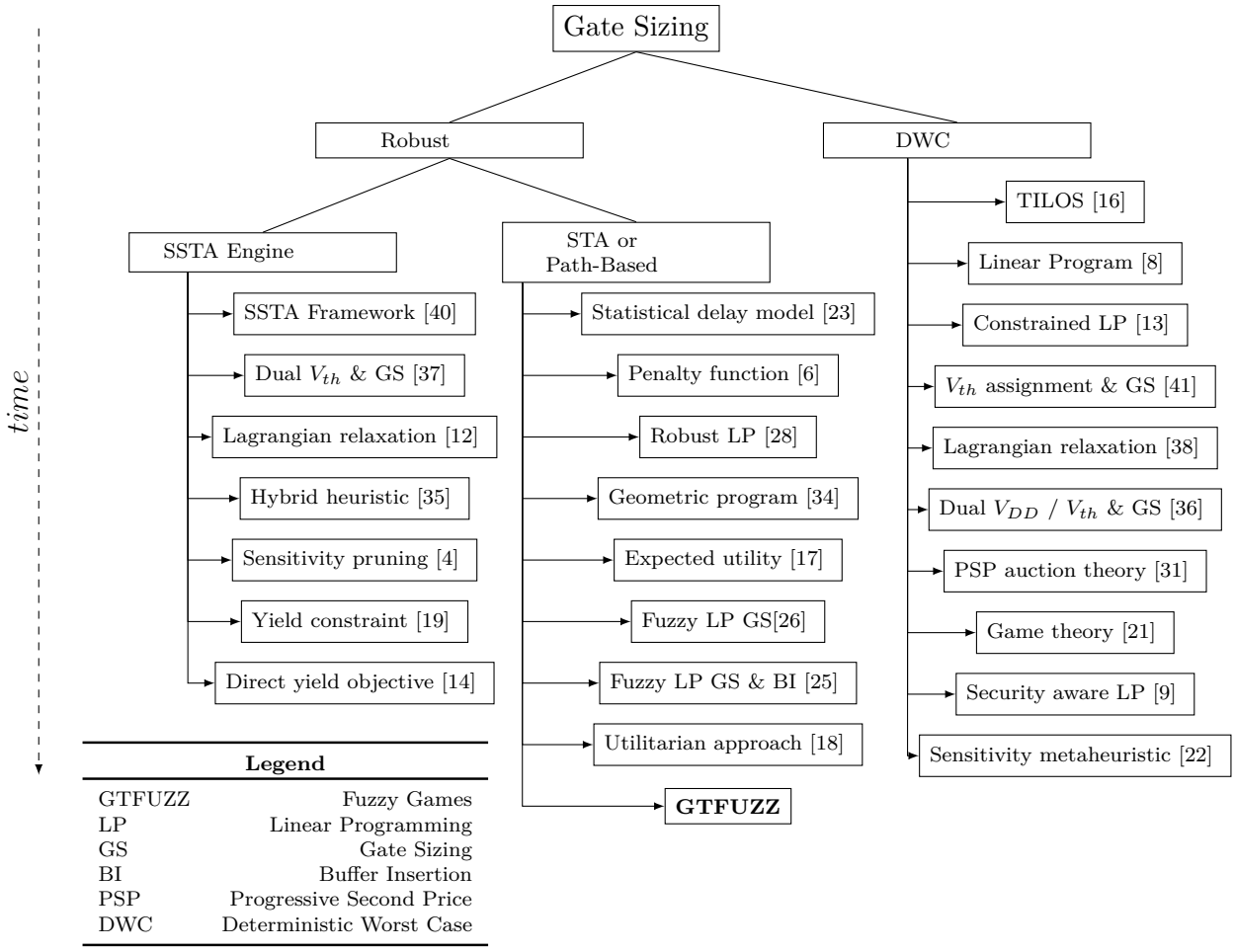


Figure 1.1: A chronological taxonomy of related works.

The existing game theoretic approaches and fuzzy linear programming optimization lead to inaccurate solutions. Refs. [21, 31] do not take into account parametric variation. To the authors' best knowledge, this will be the first robust game theoretic approach to the gate sizing problem. Ref. [26] captures the imprecision of gate delay using fuzzy sets and membership functions. However, a continuous solution for the gate sizes introduces rounding error. In addition, [26] uses the initial size setting of the gates to propagate required arrival times for the gates, but does not incrementally update gates effected by the sizing.

A solution that combines game theoretic modeling of the conflicting resource allocation in the system and fuzzy modeling of gate delay with incremental static timing analysis updates to increase accuracy will result in a more robust circuit. Another benefit of fuzzy games is that they can easily be adapted to implement a framework for simultaneous multi-metric optimization using multiple fuzzy goals and weighting strategies (for example: dynamic power, subthreshold leakage, performance, and crosstalk noise).

Chapter 2: Fuzzy Basics

2.1 What Does Fuzzy Mean?

Fuzzy mathematics are useful in modeling certain types of uncertainty. A classic example examines the definition of a hill versus a mountain. When does a hill become a mountain? If a hill is defined as 500 meters tall, is a mountain only required to be 501 meters tall? This implies that the change between hill and mountain is drastic and immediate. There is no graduation between one and the other. This crisp change in definition of membership is what we are accustomed to in classic mathematics.

If we look at the hill versus mountain example with fuzzy mathematics, we gain a different perspective. A hill does not immediately become a mountain at a certain point, it gradually becomes one. This makes it difficult to place a crisp definition on the two. Likewise we can say that a hill is a mountain to a certain degree or that a mountain is a hill to a certain degree. This departure from classical crisp mathematics helps us to better define many types of uncertainty, especially those which include biased information or even a lack of information.

Crisp sets have characteristic membership functions that are either 0 or 1. An item is either in a set or it is not. It follows the law of the excluded middle (LEM). The law of the excluded middle says a statement can either be true, or false - there is no third alternative.

In fuzzy mathematics, the law of the excluded middle does not hold. Membership in sets is graduated.

2.2 Fuzzy Triangular Distribution

A triangular fuzzy number is a triple in the form minimum, mean, maximum. The mean is also seen as the nominal or most likely value. Membership in a triangular fuzzy set gradually increases to the mean value from the minimum and gradually decreases from the mean value to the maximum. Triangular distributions can be symmetric or asymmetric.

A fuzzy set is defined by a set of ordered pairs [42]:

$$\widetilde{A} = \{ \ (x, \mu_{\widetilde{A}}(x) \ ) \mid x \in X \} \tag{2.1}$$

where $\widetilde{A}$ is a fuzzy set, $\mu_{\widetilde{A}}(x)$ is a fuzzy membership function of x in $\widetilde{A}$ that maps X to the membership space. Elements in a fuzzy set have a degree of membership. Fuzzy sets are a generalization of crisp sets. Likewise, fuzzy membership functions are generalized characteristic functions. A normalized fuzzy set is on the interval [0,1] and is most convenient for computation.

The support of a fuzzy set $\widetilde{A}$, S($\widetilde{A}$) is defined by all $x \in X$ which have a membership of strictly greater than 0 in the fuzzy set $\widetilde{A}$. More generally, the alpha-cut of $\widetilde{A}$ can be defined by all $x \in X$ with a membership in $\widetilde{A}$ greater than or equal to $\alpha = [0,1]$. Fuzzy modeling is useful when there is an absence of data, information, or understanding. Fuzzy modeling explicitly captures imprecision. When designers are using standard cell libraries provided by fabrication facility, they may not have extensive data on varying parameters. There may

be a complete absence of data. Rather than assume normal distribution, we can construct a triangular distribution to represent the variability of the physical process parameter using limited data and knowledge.

Chapter 3: Game Theory Basics

3.1 Introduction

A game theory framework is used to model interactions between conflicted but rational parties. Classical applications are in economics. However, in recent years many applications in engineering have arisen. Game theory can be used to model a variety of engineering problems which involve conflicting resource allocation.

3.2 Basic Structure

First, a game requires the participation of players. These are the involved conflicted parties. In a pure-strategy game, each player has a set of strategies they may exercise. A mixed strategy game associates a probability with each strategy (the likelihood a player will exercise that strategy). An outcome is a realized set of strategies made up of one exercised strategy per player. Each feasible outcome has a payoff. A payoff is a numerical quantification of the gain (loss) for a player in that particular outcome. Games can be cooperative or non-cooperative. In a non-cooperative game, players know all information about the game. However, players may not negotiate in any way as to what strategy they will play. Cooperative games allow players to collaborate with one another before their turn is played. This complicates the strategy sets of each player in the game since all players can

Table 3.1: Payoff matrix for the prisoner's dilemma example.

Payoff Matrix	C	D
C	0,0	-6,1
D	1,-6	-2,-2

negotiate with any other player. These negotiations are not guaranteed to be honored. This work will only deal with non-cooperative games.

3.3 Nash Equilibria

The solution method used in this work is the Nash Equilibria. An outcome is a Nash Equilibria if players can't get a better payoff by changing their strategy when other players do not deviate from their strategy. There can be more than one equilibria in a game. There can also not exist an equilibria. In the case there is more than one, the most desirable equilibria can be chosen. It should be noted that nash equilbrium are not necessarily pareto optimal. They are simply the most stable outcomes (best responses by each individual player). The Nash Equilibria is the "win-win" outcome for all players in the game.

3.4 Prisoner's Dilemma Example

We will examine a pure-strategy, normal-form, 2-player, non-cooperative game. This is a classical example called the prisoner's dilemmma. Two suspects (partners) are taken into custody after a robbery. They are suspected of pulling off the job together. They are

separately interrogated and have no prior negotiation or about the actions they will take in the interrogation. Each suspect is given the opportunity to either Confess (C) or Defect (D). Confessing will generally reduce their sentence depending on the other suspect's actions. The payoff matrix in table 3.1 illustrates the payoff of the situation. A higher payoff is considered to be positive (not a sentencing in years). The rows are player 1, the columns are player 2. C and D denote each players possible strategy. In each cell the payoffs are denoted in the form payoff for player 1, payoff for player 2. The table is also color coded for convenience.

In this example, there is only one Nash Equilibria when both player 1 and 2 defect. This is the most stable outcome. In this outcome, players can't get a better payoff by changing their strategy when other players do not deviate from their strategy. Note that the (0,0) payoff corresponding to the confess/confess outcome is the optimal payoff but it is not to be confused with the nash equilibria. The defect/defect outcome represents the best response of each individual.

Chapter 4: Theory and Problem Formulation

4.1 Why Fuzzy Games?

The work on fuzzy games presented in [24] is vital to our solution. For a thorough understanding of the theory behind the application of fuzzy games and a proof for the existence of an equilibrium, the reader is referred to [24]. GTFUZZ is a hybrid algorithm that utilizes fuzzy sets and membership functions to formulate a variation-aware game with fuzzy goals to optimize power constrained by delay without being overly pessimistic. The fuzzy modeling of variability in gate delay is combined with fuzzy payoff functions for an n-player, non-cooperative game.

The convenience of the fuzzy game optimization is its flexibility in making decisions with little information about the distribution of varying parameters. Fuzzy sets provide an accurate model of imprecision when only the mean and standard deviation are available. Game theory models the resource allocation problem of gate sizing, and Nash equilibrium is the vehicle for optimization. For a thorough primer on game theory in the context of gate sizing, [21] may be referenced. For a general introduction to fuzzy sets and memberships in the context of gate delay modeling, [26] may be referenced.

Previously published in [10]. Permission included in Appendix A.

4.2 Gate Delay and Power Model

Linear delay models with normally distributed random variables have been widely used because of the presence of many efficient linear optimization techniques. However, there are strong arguments to adopt a nonlinear delay model with fuzzy coefficients. In ultra-deep submicron circuits, gate delay is a nonlinear function of gate length and threshold voltage [11] which are two major sources of intra-chip variation. Although in some cases it may be appropriate, assuming linearity of delay leads to loss of accuracy. A linear model [8]:

$$D_i = a + bS_{cell} + c\sum_{i=1}^{n} S_i C_{in(i)}, i \in Fanout_{cell} \tag{4.1}$$

where D_i is the delay of gate i, C is the capacitive load of fanouts, S is the drive strength, and $\{a, b, c\}$ are the regression coefficients. Fanout loads are included in the summation. We have attempted to use linear regression to fit the data using the 32nm Synopsys University libraries. RMSE in this fitting was upwards of 20% with unrestricted gate sizes. To improve upon this, piecewise linear regression was used as suggested by the original author, improving the RMSE to less than 10% :

$$D_i(S,C) = \begin{cases} a_0 + b_0 S_{cell} + c_0 \sum\limits_{i=1}^{n} S_i C_{in(i)} & , S \leq 1 \\ a_1 + b_1 S_{cell} + c_1 \sum\limits_{i=1}^{n} S_i C_{in(i)} & , S \leq 2 \\ a_2 + b_2 S_{cell} + c_2 \sum\limits_{i=1}^{n} S_i C_{in(i)} & , S \leq 4 \\ a_3 + b_3 S_{cell} + c_3 \sum\limits_{i=1}^{n} S_i C_{in(i)} & , S \leq 8 \\ a_4 + b_4 S_{cell} + c_4 \sum\limits_{i=1}^{n} S_i C_{in(i)} & , S \leq 16 \\ a_5 + b_5 S_{cell} + c_5 \sum\limits_{i=1}^{n} S_i C_{in(i)} & , S > 16 \end{cases} \tag{4.2}$$

However, 10% error in a circuit operating at 2Ghz in 28nm technology can result in error of 20-30ps (depending on depth of the circuit). The average delay of a 1X sized inverter is around 30ps. To further improve the accuracy of the model, second-order piecewise regression is used. With unrestricted gate sizes from the standard cell library, RMSE was reduced to 1-2%. As in [26], b and c are treated as fuzzy coefficients using fuzzy membership functions with triangular distribution min, average, max:

$$D_i = a + \tilde{b} S_{cell} + \tilde{c} \sum_{i=1}^{n} S_i C_{in(i)}, i \in Fanout_{cell} \tag{4.3}$$

These fuzzy coefficients capture the random nature of the variation and may correlate with varying gate length (L_{gate}) and gate oxide thickness (T_{ox}) [28]. These coefficients will be defuzzified using the alpha-cut procedure.

In this work, a fuzzy, second-order piecewise expansion of the linear model presented in a seminal work on gate sizing [8] is used:

$$D_i = \begin{cases} a_0 + \widetilde{b_0} S + \widetilde{c_0} C_L + \widetilde{d_0} C_L^2 + \widetilde{e_0} S \times C_L & , S \leq 1 \\ a_1 + \widetilde{b_1} S + \widetilde{c_1} C_L + \widetilde{d_1} C_L^2 + \widetilde{e_1} S \times C_L & , S \leq 2 \\ a_2 + \widetilde{b_2} S + \widetilde{c_2} C_L + \widetilde{d_2} C_L^2 + \widetilde{e_2} S \times C_L & , S \leq 4 \\ a_3 + \widetilde{b_3} S + \widetilde{c_3} C_L + \widetilde{d_3} C_L^2 + \widetilde{e_3} S \times C_L & , S \leq 8 \\ a_4 + \widetilde{b_4} S + \widetilde{c_4} C_L + \widetilde{d_4} C_L^2 + \widetilde{e_4} S \times C_L & , S \leq 16 \\ a_5 + \widetilde{b_5} S + \widetilde{c_5} C_L + \widetilde{d_5} C_L^2 + \widetilde{e_5} S \times C_L & , S > 16 \end{cases} \tag{4.4}$$

where D_i is the delay of gate i, S is the drive strength, C_L is the capacitive load presented by all fanout cells and interconnect, and $\{a, \widetilde{b}, \widetilde{c}, \widetilde{d}, \widetilde{e}\}$ are the regression coefficients (note that

$\{\widetilde{b}, \widetilde{c}, \widetilde{d}, \widetilde{e}\}$ are fuzzy coefficients). The two works compared with in the experimental section [26, 28] both utilize this linear model. To be fair, it is also used in this work.

As in [26],$\{\widetilde{b}, \widetilde{c}, \widetilde{d}, \widetilde{e}\}$ are modeled as fuzzy coefficients using fuzzy membership functions with triangular distribution {min, average, max}. These fuzzy coefficients capture the random nature of the variation and may correlate with varying gate length (L_{gate}) and gate oxide thickness (T_{ox}) [28]. These coefficients will be defuzzified using the alpha-cut procedure presented in section D. The coefficients are represented with the symmetrical triangular distribution for simplicity. Any distribution, symmetric or asymmetric, may be used.

Dynamic power is the power spent doing computation:

$$P_{dynamic} = P_{sc} + \frac{1}{2} C_L V_{dd}^2 f_{0\to1} \tag{4.5}$$

4.3 Fuzzy Game Theory

For a fuzzy game, four standard assumptions must be made [24]. (1) Players are rational. They want to do what is best for themselves. (2) There are no enforceable agreements between players. The game is non-cooperative. The game should act as a decentralized heuristic mechanism to size gates. (3) Each player knows all information of the game. (4) The game is only played once. Reference table 4.1 for frequently used variables and symbols in the fuzzy game problem formulation.

A normal form, non-cooperative, n-player Fuzzy game is given by G_F. Each gate in a circuit (including memory elements and buffers) is considered a player who will participate in a fuzzy game G_F. Each player i has a set of strategies X_i which he may choose to exercise

in any game. These strategies make up the player's strategy set. Each player's strategy set consists of the possible gate sizes available for that element in the standard cell library. For example, an inverter can choose to size himself 0.5X, 1X, 2X, 4X, 8X, 16X, or 32X. The player's size will affect not only his intrinsic power consumption and delay, but also the power consumption and delay of interconnected elements in the circuit. The set of all strategy profiles X for a game of n players consists of all possible combination of strategies that players in a game may exercise. If a strategy profile in X is not feasible (i.e. does not meet timing constraint), it will be discarded when computing the Nash equilibrium. The set of all feasible strategy profiles in a game are referred to as $X_feasible$. It should be noted that the concept of feasibility is dynamic throughout the optimization as gates are resized. For example, the strategy set {2x, 4x, 8x} signifies that player 1 sized itself 2x, player 2 sized itself 4x and player 3 sized itself 8x. Each player will have two fuzzy goals (one for delay and one for power) modeled by two separate fuzzy membership functions. To define the fuzzy goals, we introduce two terms: the security and satisfaction measures. λ is defined as a security measure. In game theory, this corresponds to the maximin strategy. The maximin strategy yields the best outcome from all of the worst outcomes. It is the lowest value of satisfaction that a player will accept - this is the pessimistic payoff. For the fuzzy delay goal, this refers to the minimum delay of the maximum delays for each feasible strategy profile (since delay is minimized, this is the fastest of the slowest delays). For the fuzzy power goal, this refers to the lowest power consumption of the highest power consumptions for each feasible strategy profile. β is defined as a satisfaction measure. It is the value that will satisfy a player completely - this is the optimistic payoff. The security and satisfaction measures are used to define fuzzy membership functions for power and delay (one pair of

membership functions for each gate). These fuzzy membership functions are the fuzzy goals used in optimization. As noted in [24], the satisfaction measure is typically determined by using the following common formula:

$$\lambda_i = \max_{t_i \in X_i} \min_{(x_{(I-i)}) \in (X_{(I-i)} \times Y_\alpha(\widetilde{y}))} f_i(x//t_i, y) \tag{4.6}$$

where λ_i is the security measure of player i, X_i is the set of strategies for player i, $(I-i)$ is the set of all remaining players in the game (besides player i), $X_{(I-i)}$ represents all combinations of strategies of all other players in the game, f_i is the objective function of player i, $(x//t_i)$ is the notation for replacing the currently played strategy x_i in X_i with t_i for all $t_i \in X_i$, $Y_\alpha(\widetilde{y})$ is the alpha cut vector of fuzzy parameters. In other words, the player will maximize their payoff regardless of the strategy played by any other player in the game. The satisfaction measure is established with:

$$\beta_i = \max_{t_i \in X_i} \max_{(x_{(I-i)}) \in (X_{(I-i)} \times Y_\alpha(\widetilde{y}))} f_i(x//t_i, y) \tag{4.7}$$

where β_i is the security measure of player i, and the remaining notation is the same as in (4.6). The security and satisfaction measures as computed in (4.6) and (4.7) are used to define a pair of fuzzy goals with fuzzy membership functions for power and delay of player i. These goals capture the uncertainty in the decision making process. Fuzzy goals are defined with the security and satisfaction measures:

$$\mu_{f_i}(x, y) = \begin{cases} 0 & , f_i(x, y) < \lambda_i \\ \frac{f_i(x,y) - \lambda_i}{(\beta_i - \lambda_i)} & , \lambda_i \leq f_i(x, y) < \beta_i \\ 1 & , f_i(x, y) \geq \beta_i \end{cases} \tag{4.8}$$

The general flow is illustrated in figure 4.2. Note at this point there exists a crisp (defuzzified) unknown alpha-cut parameters $Y_\alpha(\widetilde{y})$ and fuzzy goals represented by membership functions $\mu_{delay_i}(x, y)$ and $\mu_{power_i}(x, y)$.

In fuzzy decision theory, constraints are considered goals. Therefore, each player has a fuzzy goal for timing and dynamic power consumption. The timing goal will require knowledge of (a) the required arrival time of a stable signal at the gate's inputs and (b) the gate delay presented by the gate itself.

In order to constrain timing and optimize one or more other metrics, the intersection of these fuzzy goals will be used. Zadeh [7] states that fuzzy decisions are the confluence of goals and constraints, that is, since they are both fuzzy sets in the space of alternatives, they can be treated identically in the formulation of a decision. Therefore, the intersection (similar to logical "AND" function) of power membership function and delay membership function is the fuzzy payoff for a player. There are several definitions for the fuzzy intersection operation.

This work uses the definition provided by Zadeh [7]. Zadeh's definition of the logical intersection operator is the minimum of the two membership functions. Therefore, the fuzzy payoff of each player i will be in the form:

$$\begin{aligned} Payoff &= \mu_{delay_i}(x, y) \cap \mu_{power_i}(x, y) \\ &= min(\mu_{delay_i}(x, y), \mu_{power_i}(x, y)) \end{aligned} \tag{4.9}$$

where μ is the fuzzy membership function of a particular chosen strategy x and unknown parameter vector y for either power or delay of gate i. The fuzzy timing goal has randomly

varying parameters which are modeled as fuzzy coefficients. These coefficients have been suspected to correlate to varying gate length (L_{gate}) and gate oxide thickness (T_{ox}) [28] using manufactured chip data. Each player chooses a confidence level α in the varying parameters of their fuzzy delay equation (see section 4.4). This confidence level expresses how sure the players are about the real value of the varying parameter. In other words, the player's confidence determines the magnitude of the variation. The magnitude is based on the spatial correlation of intra-device variation modeled in this work. The higher correlation between a gate and its fanouts, the higher confidence a player has in the value of varying parameters. This process is referred to as defuzzifying the unknown parameter with an α-cut. After defuzzification of y, there exists a crisp vector of unknown parameters of each player in a game which could take any value on an interval [a,b].

4.4 Spatial Correlation using Alpha-cuts

Intra-die variation causes identical devices on the same die to have varying characteristics. Variability in the manufacturing process can be classified into two sets of procedures: front-end and back-end [32]. Front-end procedures are related to device creation such as implantation and oxidation. Back-end procedures are related to interconnecting the devices such as etching or polishing. As previously mentioned, front-end variability has been shown to account for 90% of timing yield and path delay variability in realistic designs [32]. Since this work focuses on anticipating path delay variability, its focus is on modeling the variability of parameters affected by the front-end. These parameters such as gate length, gate

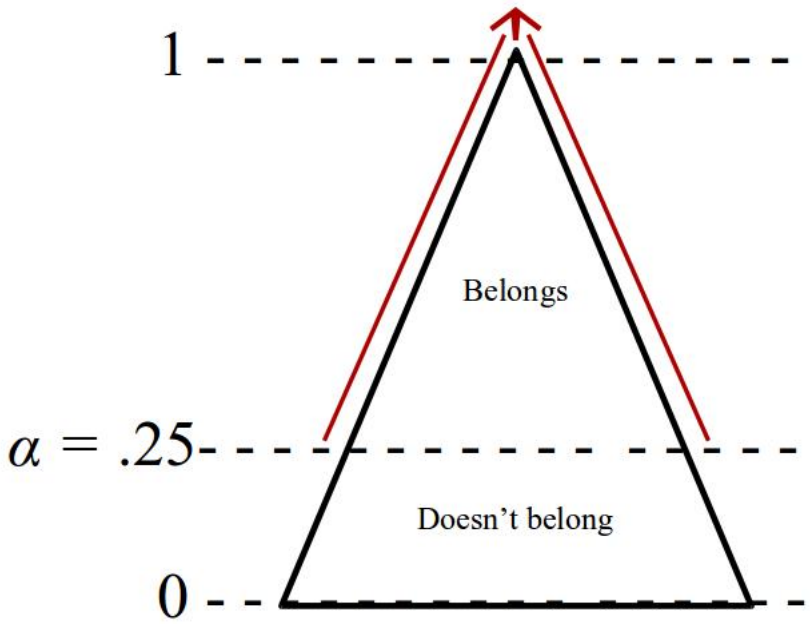


Figure 4.1: Example of modeling player confidence with an alpha-cut of .25

width, and threshold voltage, affect the performance of individual devices and present even greater complications for the design as a whole.

There are two components to variation: systematic (deterministic) and random (non-deterministic). Systematic variation can be described using mathematical models, and presents less of a burden in terms of design cost. Random variations are not fully understood enough to be treated in the same fashion [32]. In the past, worst-case (deterministic) margins were used to protect against these variations. This means that the design is adjusted so that all chips will meet a particular metric in the worst-case. This is overly pessimistic, and forces many chips in the lot to pay the price in performance. To improve upon this, statistical modeling of stochastic parameters has been used for more conservative margining while maintaining near-optimal performance.

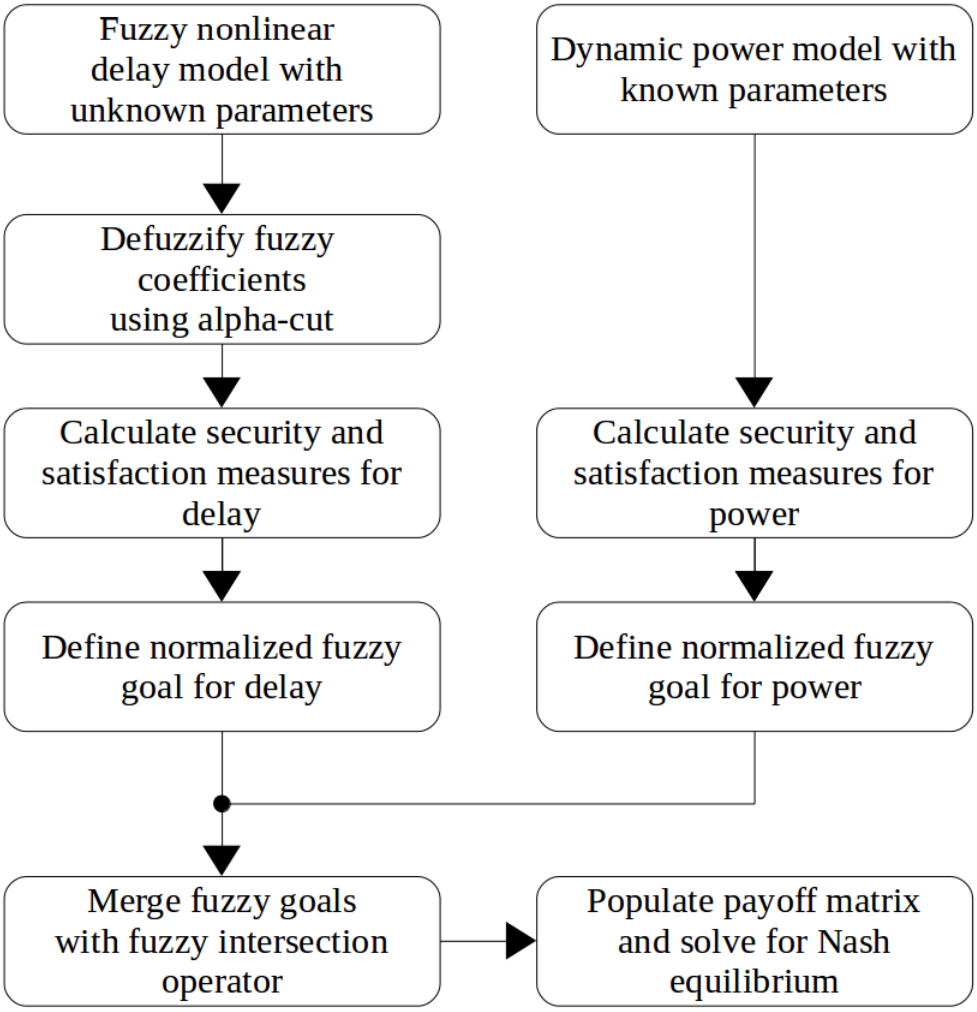


Figure 4.2: General fuzzy game flow.

In this work, the multi-level grid model [5] is used to consider spatial correlation. The die is separated into regions and levels using the recursive quadtree algorithm. The lower levels of the grid model correspond to intra-die variation. The top level models systematic variation which affects the entire die similarly. Devices which are closer in proximity are modeled with lower magnitude in variation. This magnitude of variation is modeled in each fuzzy game. Each player in the game has an alpha-cut on the randomly varying parameter vector y. This alpha-cut represents the player's (gate's) confidence level in the value of the varying parameter. Greater confidence in the value of the parameter y results in a lower

magnitude of variation and vice-versa. Each player in the game may have a different confidence (alpha value). The overall confidence level for each unknown parameter in a game is the value of alpha which respects the individual choice of each player:

$$\alpha = \max_{i \in I} \alpha^i \tag{4.10}$$

where i is a player in the set of players I in G_F and α^i is the confidence level of player i. Typically, alpha values are computed with the help of experts. In GTFUZZ, a player's confidence level is determined by the grid model of spatial correlation. Gates which are highly correlated with their fanouts are modeled with a reduced magnitude of variation. For example, figure 4.1 illustrates an alpha-cut of 25%. This figure shows a triangular membership function of a fuzzy number on the interval [0,1]. The peak in the middle of the triangle represents the nominal value of the number. Any number which is equal to the nominal value will have a membership of 1 in the fuzzy set. As the number gets larger or smaller, its membership in the fuzzy set decreases. The minimum and maximum points in to the left and right represent the worst-case values a fuzzy number may realize. Applying an alpha-cut of 25% this fuzzy set indicates that the player is confident that the actual value of the fuzzy number will have a membership in the fuzzy set greater than 0.25 on the interval [0,1]. In our case, the randomly varying parameters are the regression coefficients from the delay model.

Table 4.1: Notation in GTFUZZ problem formulation.

$G_F =< I, X, X_{feasible}, \omega, Y_\alpha(\widetilde{y}), f(x,y) >$	
I	set of all players in G_F
i	a player in G_F
X	set of all strategy profiles for a game of n players
$X_{feasible}$	set of all feasible strategy profiles for a game of n players
X_i	set of all feasible strategies for player i
ω	universe of discourse
f_i	objective function for player i
x_i	strategy exercised by player i
y	unknown parameter vector in ω
$\widetilde{y}$	unknown fuzzy parameter in ω
α	confidence level on [0,1]
$Y_\alpha(\widetilde{y})$	α-cut defuzzified parameter
μ_{f_i}	fuzzy membership function for player i
λ_i	security measure for player i
β_i	satisfaction measure for player i

Chapter 5: Proposed Methodology

5.1 GTFUZZ Algorithm

First, the design is preprocessed. The circuit is modeled as a directed acyclic graph (DAG) to assemble the games and implement static timing analysis (STA). Next, the spatial correlation grid modeling algorithm is applied. The physical die area is recursively divided into four equal parts (squares) until the total area of each square is less than 10% of the total die area.

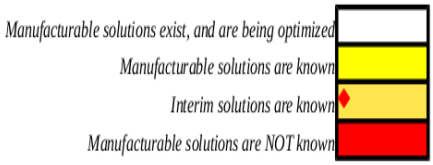
The maximum gate delay variation is dependent on empirical data from vendor process. In the absence of data from a real manufacturing runs, the worst-case magnitude of gate delay variation must be estimated. The International Technology Roadmap for Semiconductors (ITRS) 2012 report [2] for Logical / Circuit / Physical Design Technology Requirements projected that in 2014, variation from all parameters on signoff delay will be approximately 15%. This is illustrated in table 5.1. This percentage of sign-off delay variation is used as a worst-case reference point when injecting variation into the coefficients of the delay model.

The magnitude of variation depends on the cells spatial correlation to its fanout gates. If the cells fanout gates are in the same quadrant, they will be highly correlated. This means that the devices will generally vary similarly and they will have less of an effect on gate delay. The correlation of the gates is modeled as player confidence using the alpha-cut. The

Previously published in [10]. Permission included in Appendix A.

Table 5.1: ITRS projections.

Year of Production	2011	2012	2013	2014	2015	2016	2017
Asynchronous global signaling: % of a design driven by handshake clocking	19%	20%	22%	23%	25%	30%	30%
Parameter uncertainty:%-effect (on signoff delay)	11%	12%	14%	15%	18%	20%	20%
Simultaneous analysis objectives: # of objectives during optimization	6	6	6	7	7	8	8
Circuit families: # of families in a single design	4	4	4	4	4	4	4
Synthesized analog content: % of total design analog content	19%	20%	23%	25%	28%	30%	35%
Full-chip leakage (normalized to full-chip leakage power dissipation in 2011)	1.00	1.09	1.27	1.45	2.18	2.91	2.91
% "native" 3D design technology in 3D implementation flow	25	33	40	48	55	60	65



closer the gate's fanouts are located, the more confident the gate is in the actual value of the varying parameters. After each player's confidence is computed, the varying coefficients in the delay model can be defuzzified.

After the circuit is modeled as a DAG it is topologically sorted to construct a timing graph for STA operations such as propagating arrival times (AT) and required arrival times (RAT). All valid timing arcs must be extracted from the synthesis tool to build the timing graph. Node-based optimization will be used as to constrain timing rather than path-based. The number of paths is exponential. Pruning algorithms are required to enumerate the paths and may still fail to capture all critical paths. STA is widely used in industry for timing verification. This work uses STA because of its simplicity, efficiency, and dependable accuracy.

The next step is to decide the semantics of the game. The biggest challenge in gate sizing with game-theory is the computational complexity. For an explanation of the computational complexity of populating the payoff matrices and computing the Nash equilibrium, the reader is referred to [21] . First, the number of players in the game must be limited to no more than 5. The strategy sets for each player must be finite. In this case, the standard cell library provides anywhere from 2-10 possible sizes for varying cells which is computationally feasible. The limitation of the size of games is a well-known problem in game theory. If player's with similar attributes and goals can be grouped together as a single player, an optimal solution can be found more efficiently. For example, [15] presents a method for learning in many-player games where players share a similar strategic view. In the GTFUZZ algorithm, games with 3 player's are formed.

The next decision to make is which three players will compete for resources. There are many options. It should also be noted that once a gate has been sized, it will not be touched again. In a path-based approach, players can be sorted by timing criticality of the path. The critical path list can be iterated one path at a time. The problem is that many circuits can be 10-20 levels deep, much greater than the feasible size of a game if all possible strategies (gate sizes) are to be considered. In this case, there will be several games played for each path. Players can be chosen sequentially in order from upstream to downstream or vice-versa. Another problem with this approach is that after a critical path is sized, it is not to be resized. The gates which were sized on this path may also be in another less critical path. The sorted critical paths must be further pruned to find three relevant gates for a game. Relevant in this sense means that changing the gate size of any gate in the

game should affect the other player's delay and power consumption directly. If the gates are not competing for resources, then the model is essentially useless. This procedure of player selection further complicates the path-based approach and was a motivating factor to use a node-based approach.

Since power is the metric being optimized, the gates are first sorted based on power consumption. The most power-hungry player is selected as the first in the game. The gates which are most relevant (meaning they affect the player's delay and power consumption most directly) to a player are their fanins and fanouts. The most power-hungry fan-in and fanout of the player are selected to compete for sizing in a one-shot fuzzy game. This method of selection is quick and efficient and aims at attacking the most power-hungry portions of the circuit while using STA operations to ensure timing constraints are met.

Algorithm 2 and 3 summarize the procedure to iteratively setup and solve fuzzy games[1]. After the 3 players are selected for the game, three payoff matrices are built. One matrix contains the player's power consumption for each strategy profile. One matrix contains the player's propagation delay for each strategy profile delay. Using these two matrices, the security measure (eqno. 4.6) and satisfaction measure (eqno. 4.7) are computed for power and delay.

Finally, an overall payoff matrix is created to store the confluence of the fuzzy power and delay goals (the result of eqno. 4.9). Every strategy profile X must be explored in order to find the most optimal outcome in a game. Any strategy profile in which any player has an infeasible strategy is not a valid profile. For example, in chain of three inverters, sizing

1. See appendix B for pseudocode.

the inverters 1x, 1x, and 32x respectively is in practice typically not a feasible strategy. The 1x inverters will not be able to drive the 32x inverter under a reasonable timing constraint. This means that the strategy profile {1x, 1x, 32x} is invalid and will not be considered when calculating the Nash equilibrium solution of the game. Note this does not imply that player 3 is restricted from exercising the 32x strategy in other profiles, only that it is infeasible in the current profile.

When considering a strategy profile, the timing graph must be updated. Sizing a gate affects the input capacitance it presents to other gates. Therefore, all input capacitance values in the graph must be updated. Since the delay model is a function of size and capacitance, timing is also affected and must be updated. Arrival times and required arrival times are propagated through the cells affected by the strategy profile, otherwise known as incremental STA update. If the new arrival time of a signal at an input of a gate does not arrive within the required arrival time window [min RAT, max RAT], that strategy profile is considered infeasible and will be discarded. If all of the strategies in the profile are feasible, the gate delay and power consumption presented by the player are fuzzified with (4.8) and stored in their respected payoff matrices. Finally, the overall fuzzy payoff can be determined using (4.9).

After the payoff matrices are populated for all feasible strategy profiles, a Nash equilibrium can be computed. All payoffs are floating point numbers and normalized on an interval from [0,1]. A game-theory solver, Gambit [30], is used to compute the Nash equilibrium of the 3-player game. In many cases, there may be more than one equilibrium. In this case, all of the solutions are sorted and the strategy profile with the lowest power consump-

tion is chosen. The solution to the fuzzy game is a robust setting of the player's sizes who are participating in the game. The gate sizes are permanently set according to this solution and marked as sized. The remaining gates sorted by power consumption will be iterated in the same fashion until all gates are marked as sized. Refer to the appendix for algorithm pseudocode[2] .

2. See appendix B for pseudocode.

Chapter 6: Simulation

6.1 Experimental Setup

The GTFUZZ algorithm is implemented using the Synopsys University 32/28nm [1] standard cell libraries and technology files. Although this library is not suitable for manufacturing, its attributes are taken from scalable MOSIS design rules, actual data from silicon chips, the open-source Predictive Technology Model (PTM) from Arizona State University (ASU), and models of parasitics from the Nanoscale Integration and Modeling Group (NIMO) group at ASU. The 1.05V low-power library is used. The algorithm was tested using ITC'99 benchmark circuits [3] . The benchmarks are illustrated in table 6.1. A standard Synopsys tool flow is adopted as shown in figure 6.1.

First, all of the gates in the standard cell library are characterized for power and timing. Of the more than 200 cells in the library, the basic gates necessary in logic design are characterized (simulations for characterization are time consuming). The selected cells are NAND, NOR, XNOR, XOR, 2:1 MUX, 4:1 MUX, inverters, buffers (non-inverting and inverting), and D flip-flops with various types of set and reset attributes. Although the gates are restricted, all available sizes are used in the optimization process and therefore all sizes of each selected gate are characterized. HSPICE is used to characterize each gate, sweeping a wide range of realistic capacitive loads. The results are written in CSV format and exported

Previously published in [10]. Permission included in Appendix A.

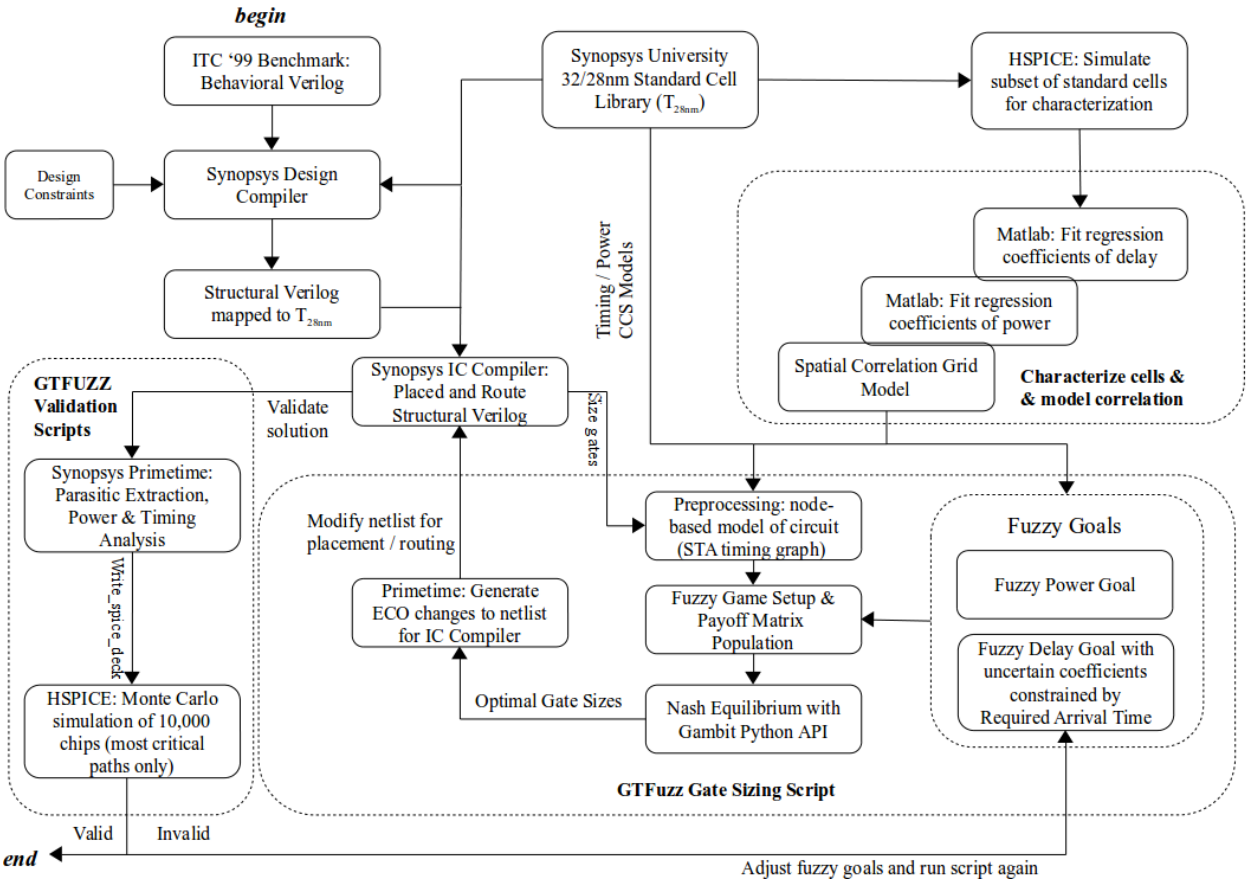


Figure 6.1: GTFUZZ Algorithm: simulation flow.

to Matlab. The curve fitting tool (cftool) is used for the piecewise second-order regression. All of the coefficients are found and written to a file in CSV format to be utilized by the GTFUZZ script. The RMSE of the fits for the delay model ranges from 1-2%. The accuracy of the power and delay models are extensively verified against power and delay analysis figures in Synopsys Primetime.

ITC '99 benchmarks in VHDL format are synthesized in Synopsys Design Compiler. The clock cycle of each benchmark must be determined. The benchmarks are initially optimized for high performance. After finding the frequency of the benchmark, it is synthesized

with its constrained clock rate from an external clock source, cells from the library driving all primary inputs, and realistic capacitive loads and fanouts on all primary outputs.

The synthesis is linked with the Synopsys University standard cell library in .db format. The library contains information about timing and power for all gates in the standard cell library using compact Composite Current Source (CCS) models. These models are very accurate and provide analysis within a few percent of HSPICE simulation. The reset and clock networks are set as "don't touch" as it is better to optimize these after clock tree synthesis and signal buffering in the place and route stage. The design is ungrouped from any hierarchies, completely flattened, and compiled. There are two outputs from the compilation: a structural Verilog netlist which maps the non-technology based synthesis output standard cell library and an .sdc file which contains the constraints used during compilation.

Next, the design is placed, routed, and optimized using Synopsys IC Compiler (ICC). This phase is linked with the library's technology file (.tf) which contains physical design rules and information about metal layers and the TLU+ (Table Look Up) max/min/map files which contain capacitance and resistance data used for parasitic extraction. The structural Verilog netlist and .sdc constraint file from Design Compiler are imported and read into ICC. A floorplan is initialized and rectangular VDD and GND rings are synthesized. The standard cells are then placed and prerouted before clock tree synthesis. The design is optimized and rerouted iteratively until all design rules and constraints are satisfied. There are several outputs from this stage. The placed and routed netlist is written in Verilog format. The parasitics are extracted in Standard Parasitic Exchange Format (SPEF) for timing and power analysis in the next stage of the flow. Finally, the Design Exchange Format (DEF) file is

Table 6.1: ITC'99 benchmark descriptions.

NAME	ORIGINAL FUNCTIONALITY
b01	FSM that compares serial flows
b02	FSM that recognizes BCD numbers
b03	Resource arbiter
b04	Compute min and max
b05	Elaborate the contents of a memory
b06	Interrupt handler
b07	Count points on a straight line
b08	Find inclusions in sequences of numbers
b09	Serial to serial converter
b10	Voting system
b11	Scramble string with variable cipher
b12	1-player game (guess a sequence)
b13	Interface to meteo sensors
b14	Viper processor (subset)
b15	80386 processor (subset)
b16	Hard to initialize circuit (parametric)
b17	Three copies of b15
b18	Two copies of b14 and two of b17
b19	Two copies of b14 and two of b17
b20	A copy of b14 and a modified version of b14
b21	Two copies of b14
b22	A copy of b14 and two modified versions of b14

written. This file contains all of the physical information of the chip including die size, placement, and orientation of cells. Now, the chip is fully placed and routed and ready for initial power and timing analysis.

The placed and routed Verilog netlist, .sdc constraint file, and SPEF (min and max) files are loaded into Synopsys Primetime for initial timing and power analysis. An SAIF (Switching Activity Interchange Format) file must be loaded into Primetime in order to enable power analysis. In order to have a more realistic estimate of power without having any

knowledge of the internals of the benchmarks, Synopsys Verilog Compiler Simulator (VCS) is used to simulate 100,000 random vectors. The switching activity of each cell is monitored during simulation and dumped into a SAIF file. From Primetime, many attributes of the design such as net names, capacitances, toggle rates (toggle count / duration of sampling), and library cell pin capacitances are written to file for use in the GTFUZZ script.

At this point, the GTFUZZ sizing script has all of the data necessary to execute. A GTFUZZ bash script glues together all of the TCL scripts used for the Synopsys tool flow, the GTFUZZ sizing script, the HSPICE Monte Carlo script, and other auxiliary scripts used for file I/O. The GTFUZZ sizing script is written in Python. Although Python is significantly slower as compared to lower-level languages such as C or C++, there are a couple major advantages. Rapid prototyping and availability of simple and powerful libraries greatly increased the pace of this work and allowed the authors to focus on implementing the algorithm.

The NetworkX [20] network algorithm library was used to model the circuit as a graph, to topologically sort it, complete operations necessary for STA, and store all attributes of the benchmark circuit. Gambit Software Tools [30] offers an open-source Python API to the Gambit library to setup and solve games. Using Python certainly increased the execution time of the algorithm in the experiments, but greatly decreased the development time. The authors believe that this is a worthy trade off and that the purpose of this work, to show the promise in the application of fuzzy games in gate sizing, has still been achieved.

After initial synthesis, placement and routing, the GTFUZZ sizing script models the circuit with a DAG. For very large circuits, loading the circuit attributes and executing the recursive quadtree algorithm can be time consuming. For this reason, the timing graph and spatial correlation grid model are serialized and cached in binary files (pickled) for rapid loading during development and re-running of the GTFUZZ sizing script. The output of the GTFUZZ sizing script is a TCL script which issues size_cell commands to Primetime, setting each gate in the circuit to its optimal size. Through Primetime, an ECO (Engineering Change Order) TCL script is generated with the sizes of gates which must be changed. This ECO script is then sourced in IC Compiler. In most cases, the gates are exchanged in place and do not require a fresh place and route. The final placed and routed Verilog netlist is then generated, and the design is extensively tested for robustness.

Algorithm 4 summarizes the validation procedure. There are two stages of validation of the results. First, timing and design rules must be verified in Primetime. The magnitude of gate delay variation computed in the GTFUZZ sizing script must be imported into Primetime. This is achieved by individually derating each gate according to its previously calculated correlation. After timing is verified in Primetime, one more step is taken to verify timing yield. Using the write_spice_deck [39] command in Primetime, the most critical paths are extracted in a netlist with detailed parasitics. The most critical paths are simulated in HSPICE to verify the timing analysis in Primetime. Monte Carlo simulation of a lot of 10,000 chips was completed with normally distributed parameters. The parameters (L_{gate}, V_t, and T_{ox}) are varied corresponding to the projected 2014 ITRS report total effect on gate

delay due to variation of 15%. Measure statements are used to check for slew violations and propagation delay of the most critical paths in the design.

In the case where timing is not met after executing the GTFUZZ sizing algorithm, there are a couple knobs in the script that may be tuned (see algorithm 2). The security and satisfaction measures for power and delay used to define a player's fuzzy goals can be tuned for more aggressive or relaxed optimization. If a design does not meet timing constraints, the satisfaction measure of the power fuzzy goal are relaxed and/or the security measure of the delay fuzzy goal is increased. A player will be more satisfied with less of a power reduction (and vice-versa) and demand a higher minimum level of satisfaction of delay. When the security measure increases, a player demands that his gate delay must decrease in order to be minimally satisfied. After observing the timing violation in the validation phase, optimization parameters may be relaxed. The measures are increased or decreased proportionally to the timing violation (w.r.t. clock rate) as a starting point. Alternatively, after observing excessive slack in the validation phase, these measures may be tweaked for more aggressive optimization.

Chapter 7: Experimental Results

The simulation results are compared to two alternative methods: FLP (Fuzzy Linear Programming) [26] and DWCFLP (Deterministic Worst-Case Fuzzy Linear Programming). To the authors' best knowledge, there are no robust game-theoretic gate sizing works which consider the effects of parameter variation on gate delay. Therefore, the authors believe the best comparison is against the fuzzy linear programming algorithm.The FLP gate sizing algorithm [26] has been compared against stochastic linear programming and shown an average of about 10% improvement in power savings.

A fuzzy linear program is setup with linear constraints and objective function. The constraints have varying fuzzy coefficients with triangular distributions. The fuzzy linear program must be defuzzified into a crisp, nonlinear program. The fuzzy objective function is defuzzified into two objective functions: one realizing the worst-case values of the fuzzy coefficients and one realizing the nominal case value of the coefficients. A crisp, nonlinear program is formed using the symmetric relaxation method [7]. Symmetric relaxation finds a balance between the worst-case and nominal case sizing, resulting in a robust design. All of the information required to solve and constrain the FLP is extracted from the circuit model and the programs are solved.

Previously published in [10]. Permission included in Appendix A.

Table 7.1: Experimental results on synthesized ITC '99 benchmarks.

GTFUZZ Robust Gate Sizing Algorithm Simulation Results								
				Total Power (μW)			*Power Reduction* (%)	
Execution Time (s)	*Design*	*Clock Period*	*# Gates*	*DWCFLP*	*FLP*	*GTFUZZ*	*vs. DWCFLP*	*vs. FLP*
21.32	b10	0.91	106	83.89	75.7	68.8	17.99%	9.11%
81.83	b11	0.83	424	444.5	410.1	360.7	18.85%	12.05%
174.64	b12	0.93	729	547.1	488.9	441.3	19.34%	9.74%
1103.98	b14	2.21	5109	1602.8	1334.4	1229.6	23.28%	7.85%
1464.61	b15	2.62	6897	1611.3	1431.1	1334.7	17.17%	6.74%
4189.48	b17	2.98	23337	6508.5	5617.3	5113.4	21.44%	8.97%
12785.64	b18	3.49	74239	11178.1	9215.3	8356.6	25.24%	9.32%
1920.96	b20	2.37	10522	2689.2	2415.6	2164.8	17.21%	10.38%
3663.11	b22	1.38	19021	4948.4	4234.8	3897.5	21.24%	7.96%
			Average power reduction operating at same clock frequency $\rightarrow$				20.45%	9.12%
DWCFLP - Deterministic worst-case fuzzy linear programming								
FLP - Fuzzy linear programming								

In the DWCFLP algorithm, FLP is used where the coefficients in the delay model are set to worst-case values. This pessimistic design anticipates worst-case variation and ensures that all chips in a lot will satisfy timing specification resulting in 100% yield. In this approach, correlation is neglected and a global derate (early and late) of 15% is distributed across all cells in Primetime for timing verification.

The simulation results show a clear improvement in power reduction as compared to DWC fuzzy games and fuzzy linear programming with fuzzy coefficients. The power reduction is computed using the standard formula:

$$\frac{Power_{algorithm\_2} - Power_{GTFUZZ}}{Power_{algorithm\_2}} * 100 \tag{7.1}$$

The GTFUZZ algorithm saved 20% of power compared to DWCFLP. This result is expected as the DWC approach is not robust. The GTFUZZ algorithm saved approximately 9% of power compared to the FLP approach. The weakness of the FLP algorithm is that the pro-

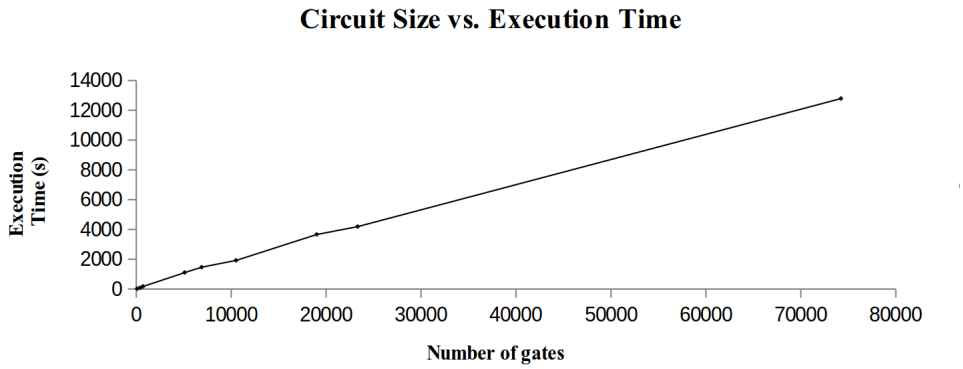


Figure 7.1: The GTFUZZ runtime scales linearly with circuit complexity.

gram is solved with the initial required arrival times from STA. These required arrival times are not updated throughout the optimization as in the GTFUZZ algorithm. Therefore, precision during optimization is lost. The solutions to the fuzzy linear program are continuous gate sizes for each cell in the design. This introduces rounding error when the solutions must be mapped to discrete gate sizes from the standard cell library.

The runtime of the GTFUZZ algorithm is not impressive. There are two main reasons for this. The GTFUZZ script is written in Python, an interpreted language. Libraries included in the script, such as NetworkX, add convenience at the cost of unnecessary overhead. Profiling with the cProfile Python module showed that calling nash equilibrium pure strategy game solver executable (enumpure) in the Gambit Python API can peak upwards of 180ms to solve the most complex 3-player games.

To improve the runtime, GTFUZZ could be implemented in a language with less overhead and more low-level memory control such as C/C++. Also, a lightweight, special purpose solver for these games could easily be implemented to replace the Gambit extension. In any case, the authors believe that the trade off for rapid prototyping and development is

well worth the hit in execution time, as the algorithm has shown promise in precise modeling and powerful optimization. Despite this weakness, the runtime has been shown to be linear in figure 7.1.

It should also be noted that a path-based approach was attempted first. The process of extracting the n-worst 2 million true (false paths are pruned) paths from Primetime can take upwards of 10 minutes. After valid path extraction, searching paths sorted by criticality looking for gates to size is very time consuming. It was observed that as the algorithm searches deeper into less critical paths after many gates have already been marked as sized, execution time greatly increases. Hundreds or thousands of paths are iterated before finding a suitable combination of players for a game. The authors have deemed this trade off for execution time vs. precision unacceptable. This led to the node-based optimization using STA.

Chapter 8: Conclusion

A novel, robust algorithm for discrete gate sizing (GTFUZZ) has been presented in this work. Power savings from 10-20% have been shown as compared to two competing algorithms. Simulations were done on circuits ranging from 25-75k gates on ITC '99 benchmarks with no compromise in delay of the circuit. The smallest possible open source technology library was used.

There are several interesting directions in which this work can proceed. The algorithm can be applied to different objectives such as buffer insertion and wire-sizing. The algorithm can also be applied at a lower level of abstraction such as multi V_{th} or V_{DD} assignment. The symmetric definition of fuzzy goals and constraints as provided by Zadeh provides a framework for multimetric optimization. Using a fuzzy definition of logical intersection, many fuzzy goals can easily be merged.

The fuzzy games also present a novel method to model spatial correlation using the concept of the the alpha-cut and player confidence. One of the weaknesses of the game theoretic approach is the complexity of many-player games. The simulations hit a brick-wall in execution time as the number of players was increased to 5. This can be improved upon by using a clustering algorithm to group players together which have a similar outlook in

Previously published in [10]. Permission included in Appendix A.

the game. In doing so, the quality of the solution will improve because each individual game can model a greater portion of the circuit.

The scripts were all written in Python. This hurts the runtime of the algorithm. Typically, these types of algorithms which are to be scaled into the millions are written in C/C++. This work favored Python for the advantages in rapid prototyping. This includes the convenience of using the Gambit Game Theory Solver Python API. The cost to pay for the convenience of using a game solver was high. Larger games, the games with more strategies such as inverters or buffers (they can be sized 0.5x, 1x, 2x, 4x, 8x, 16x, 32x), can take up to 200 milliseconds just to compute the nash equilibria. In the future, it is advised that a custom solver should be written.

One of the most crippling disadvantages of game theoretic modeling is scalability. Unbounded games of n-players are not feasible. Limiting players to single digits with reasonably sized discrete strategy sets allows a linear runtime to be realized. A future improvement would be to cluster the gates. Gates with similar goals or perspectives in the game can be grouped together. By grouping the players, we can scale the size of the games without sacrificing execution time. However, this will affect the quality (accuracy) of the solution. The framework for fuzzy games presented in this work is intended to illustrate the potential of fuzzy and game theoretic modeling in VLSI design automation.

References

[1] 32/28nm synopsys university generic library. Date Accessed: April 12, 2015. <https://www.synopsys.com/COMMUNITY/UNIVERSITYPROGRAM/Pages/32-28nm-generic-library.aspx>.

[2] ITRS Roadmap - Table DESN4. Technical report, ITRS.

[3] ITC'99 Benchmarks (2nd release). CAD Group at Politecnico di Torino, 2004. http://www.cad.polito.it/downloads/tools/itc99.html.

[4] A. Agarwal, K. Chopra, and D. Blaauw. Statistical timing based optimization using gate sizing. In *Design, Automation and Test in Europe, 2005. Proceedings*, pages 400–405 Vol. 1, March 2005.

[5] A. Agarwal et al. Statistical delay computation considering spatial correlations. In *Design Automation Conference, 2003. Proceedings of the ASP-DAC 2003. Asia and South Pacific*, pages 271–276, Jan 2003.

[6] Xiaoliang Bai, C. Visweswariah, P.N. Strenski, and D.J. Hathaway. Uncertainty-aware circuit optimization. In *Design Automation Conference, 2002. Proceedings. 39th*, pages 58–63, 2002.

[7] R.E. Bellman and L.A. Zadeh. Decision Making in a Fuzzy Environment. *Management Science*, 17:141–164, 1970.

[8] M. R C M Berkelaar and J. A G Jess. Gate sizing in mos digital circuits with linear programming. In *Design Automation Conference, 1990., EDAC. Proceedings of the European*, pages 217–221, Mar 1990.

[9] Koustav Bhattacharya and Nagarajan Ranganathan. A linear programming formulation for security-aware gate sizing. In *Proceedings of the 18th ACM Great Lakes symposium on VLSI*, pages 273–278. ACM, 2008.

[10] T. Casagrande and N. Ranganathan. Gtfuzz: A novel algorithm for robust dynamic power optimization via gate sizing with fuzzy games. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2015*, pages 677–682, March 2015.

[11] Lerong Cheng, Jinjun Xiong, and Lei He. Non-linear statistical static timing analysis for non-gaussian variation sources. In *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, pages 250–255, June 2007.

[12] Seung Hoon Choi, Bipul C Paul, and Kaushik Roy. Novel sizing algorithm for yield improvement under process variation in nanometer technology. In *Proceedings of the 41st annual Design Automation Conference*, pages 454–459. ACM, 2004.

[13] O. Coudert. Gate sizing for constrained delay/power/area optimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 5(4):465–472, December 1997.

[14] Azadeh Davoodi and Ankur Srivastava. Variability driven gate sizing for binning yield optimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(6):683–692, 2008.

[15] S. G. Ficici, D. C. Parkes, and A. Pfeffer. Learning and solving many-player games through a cluster-based representation. *In Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 187–195, 2008.

[16] John P Fishburn and Alfred E Dunlop. Tilos: A posynomial programming approach to transistor sizing. In *The Best of ICCAD*, pages 295–302. Springer, 2003.

[17] U. Gupta and N. Ranganathan. An expected-utility based approach to variation aware VLSI optimization under scarce information. *Low Power Electronics and Design (ISLPED), 2008 ACM/IEEE International Symposium on*, pages 81–86, 2008.

[18] U. Gupta and N. Ranganathan. A utilitarian approach to variation aware delay, power, and crosstalk noise optimization. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 19(9):1723–1726, Sept 2011.

[19] Matthew R Guthaus, N Venkateswarant, C Visweswariaht, and Vladimir Zolotov. Gate sizing using incremental parameterized statistical timing analysis. In *Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*, pages 1029–1036. IEEE Computer Society, 2005.

[20] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008.

[21] Narender Hanchate and Nagarajan Ranganathan. Post-layout gate sizing for interconnect delay and crosstalk noise optimization. In *Proceedings of the 7th International Symposium on Quality Electronic Design*, pages 92–97. IEEE Computer Society, 2006.

[22] Jin Hu, A.B. Kahng, SeokHyeong Kang, Myung-Chul Kim, and I.L. Markov. Sensitivity-guided metaheuristics for accurate discrete gate sizing. In *Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference on*, pages 233–239, Nov 2012.

[23] E. T a F Jacobs and M. R C M Berkelaar. Gate sizing using a statistical delay model. *Proceedings -Design, Automation and Test in Europe, DATE*, pages 283–290, 2000.

[24] F. Kacher and M. Larbani. Existence of equilibrium solution for a non-cooperative game with fuzzy goals and parameters. *Fuzzy sets and systems*, 159:165–176, 2008.

[25] V. Mahalingam and N. Ranganathan. A fuzzy approach for variation aware buffer insertion and driver sizing. In *Symposium on VLSI, 2008. ISVLSI '08. IEEE Computer Society Annual*, pages 329–334, April 2008.

[26] V. Mahalingam, N. Ranganathan, and J.E. Harlow. A Fuzzy Optimization Approach for Variation Aware Power Minimization. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2008.

[27] M. Mani, A. Devgan, and M. Orshansky. An Efficient Algorithm for Statistical Minimization of Total Power under Timing Yield Constraints. In *Design Automation Conference, 2003. Proceedings*, pages 309–314, June 2005.

[28] M. Mani and M. Orshansky. A new statistical optimization algorithm for gate sizing. In *Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings. IEEE International Conference on*, pages 272–277, Oct 2004.

[29] G. Mauris, V. Lasserre, and L. Foulloy. Fuzzy modeling of measurement data acquired from physical sensors. *IEEE Trans. on Instr. and Meas*, 49:1201–1205, 2000.

[30] Richard McKelvey, Andrew McLennan, and Theodore Turocy. Gambit: Software Tools for Game Theory, 2014. Version 13.1.2.

[31] A. Murugavel and N. Ranganathan. Gate sizing and buffer insertion using economic models for power optimization. In *17th International Conference on VLSI Design.*, pages 195–200, 2004.

[32] S. Nassif, M. Orshansky, and D. Boning. *Design for Manufacturability*. Springer, 2008.

[33] S.R. Nassif. The light at the end of the cmos tunnel. In *Application-specific Systems Architectures and Processors (ASAP), 2010 21st IEEE International Conference on*, pages 4–9, July 2010.

[34] J. Singh, V. Nookala, Zhi-Quan Luo, and S. Sapatnekar. Robust gate sizing by geometric programming. In *Design Automation Conference, 2005. Proceedings. 42nd*, pages 315–320, June 2005.

[35] Debjit Sinha, Narendra V Shenoy, and Hai Zhou. Statistical gate sizing for timing yield optimization. In *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, pages 1037–1041. IEEE, 2005.

[36] A. Srivastava, D Sylvester, and D. Blaauw. Power minimization using simultaneous gate sizing, dual-vdd and dual-vth assignment. In *Design Automation Conference, 2004. Proceedings. 41st*, pages 783–787, July 2004.

[37] A. Srivastava, D. Sylvester, and D. Blauuw. Statistical optimization of leakage power considering process variations using dual-vth and sizing. In *Proceedings of the 41st annual Design Automation Conference*, pages 773–778. ACM, 2004.

[38] Hiran Tennakoon and Carl Sechen. Gate sizing using lagrangian relaxation combined with a fast gradient-based pre-processing step. In *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pages 395–402. ACM, 2002.

[39] T. Thiel. Have i really met timing? - validating primetime timing reports with spice. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, volume 3, pages 114–119 Vol.3, Feb 2004.

[40] C. Visweswariah. Death, taxes and failing chips. In *Design Automation Conference, 2003. Proceedings*, pages 343–347, June 2003.

[41] Liqiong Wei, Kaushik Roy, and Cheng-Kok Koh. Power minimization by simultaneous dual-v th assignment and gate-sizing. In *Custom Integrated Circuits Conference, 2000. CICC. Proceedings of the IEEE 2000*, pages 413–416. IEEE, 2000.

[42] H.J. Zimmerman. *Fuzzy Set Theory and Its Applications*. Kluwer Academic Publishing, 1992.

Appendices

Appendix A: Copyright Permissions

The permission below is from a publication in the Proceedings of Design, Automation, and Test in Europe (DATE) conference in 2015, by Anthony J. Casagrande and N. Ranganathan, pages 677 - 682. It is to appear in IEEE Xplore Digital Library as doi: 10.7873. The majority of the content in chapters 1 and 4-8, figures 8.1 - 8.4, tables 4.1 and 7.1, and the abstract have been used in this document.

edaa
european
design and automation
association

COPYRIGHT AGREEMENT

DATE 2015

EDAA

The Proceedings of this conference will be published in at least two forms, including on-site download and the WW Web. It is important that all contributors sign the enclosed agreement to permit unrestricted world-wide publication by all publishing agents chosen by EDAA.

This form is intended for original, previously unpublished material submitted to EDAA conferences. It is important to make clear that EDAA as a non-profit International Scientific Organisation registered in Brussels will hold the copyright for organisational reasons, to enable other parties to use the material. EDAA has no desire to constrain or profit in any way, and immediately re-assigns unconstrained ability to publish or copy to the authors and all publishing agents chosen by EDAA.

You may care to keep a copy of this form for your files.

TITLE OF WORK:	GTFUZZ: A Novel Algorithm for Robust Dynamic Power Optimization via Gate Sizing with Fuzzy Games
AUTHORS:	Tony Casagrande and Nagarajan Ranganathan
PUBLICATION TITLE/DATE:	**Design, Automation and Test in Europe Conference March 9-13, 2015 – Grenoble, France**

The undersigned hereby assigns to EDAA unconstrained rights to publish, print or copy the above work in any form.

The undersigned hereby assigns to EDAA the right to pass on these same unconstrained rights to publish, print or copy the above work in any form (including hard copy, CD-ROM, on-site download and WW Web) to any or all of the IEEE, the ACM and other suitable publishing agencies.

The undersigned understands that the copyright will be held by EDAA for organisational reasons except in the cases of UK and US Government Employees. Such persons are additionally required to sign in the special box provided, in which case the copyright is retained by their respective organisations, but in no way does this diminish the rights assigned to EDAA.

The undersigned hereby warrants that the work is original and that he/she has the power and authority to make and execute this assignment.

Authors and Employers retain all proprietary rights in any process, procedure or article of manufacture described in the work, and similarly retain unconstrained rights to publish, print or copy the above work in any form.

AUTHORISED SIGNATURE:	[redacted]
TITLE & DATE:	GTFUZZ: A Novel Algorithm for Robust Dynamic Power Optimization via Gate Sizing with Fuzzy Games 11/26/2014
AUTHOR'S ORGANISATION or EMPLOYER FOR WHOM WORK WAS PERFORMED:	University of South Florida
AUTHOR'S DAYTIME PHONE / FAX / E-MAIL:	
SIGN ONLY IF UK or US GOVERNMENT EMPLOYEE:	

Please upload the signed form to Softconf: https://www.softconf.com/date15/conference/

Appendix B: Algorithm Pseudocode

```
1: procedure FG_SOLVE                                   ▷ solve 3-player game G_F
2:     {solutions} ← compute nash equilibrium
3:                                                      ▷ the solution is a set of profiles
4:     minimum power ← ∞
5:     for each nash equilibrium profile s in solutions do
6:         if power(s) ≤ min then
7:             minimum power ← power(s)
8:             best solution ← s
9:     return bestsolution
```

Figure B.1: Subroutine to solve the fuzzy games.

```
1: procedure VALIDATE                                   ▷ analyze power and timing
2:     generate new netlist                                          ▷ Primetime
3:     place and route with new sizes                                ▷ IC Compiler
4:     verify design constraints and timing                          ▷ Primetime
5:     write spice deck                                ▷ propagation delay and slew
6:     Monte-carlo simulation with 10k chips                         ▷ HSPICE
7:     if constraints met then
8:         return                                                    ▷ Success!
9:     else
10:        Adjust λ and β                                  ▷ See eqnos. 4.6, 4.7
11:        fg_setup                                ▷ run GTFUZZ sizing script again
```

Figure B.2: Validate the solution of fuzzy games.

1: **procedure** PREPROCESSING
2: ▷ *input*: (1) Behavioral VHDL benchmark circuit C from ITC '99 suite, (2) Synopsys University 32/38nm standard cell library and technology files T_{28nm} and (3) characterized delay regression coefficients for all gates in T_{28nm}
3: ▷ *output*: (1) parasitics and physical information for each gate i in C, (2) compiled, placed, and routed structural netlist C_{PAR}, and (3) Node-based (timing graph) representation TG of C_{PAR}
4:
5: $C_{structural} \leftarrow$compile$(C, T_{28nm})$
6: $C_{PAR} \leftarrow$place and route$(C_{structural}, T_{28nm})$
7: $TG \leftarrow$model as graph(C_{PAR}, T_{28nm})
8: **for each** gate i in TG **do**
9: $i \leftarrow$load delay model(C_{PAR})
10: $i \leftarrow$read DEF(C_{PAR}) ▷ *get physical info*
11: $i \leftarrow$model spatial correlation(C_{PAR}) ▷ *grid*
12: $i \leftarrow$compute player confidence ▷ *alpha-cut*
13: mark i as unsized
14: **if** TG not cached **then** ▷ *for performance*
15: cache(TG)
16: *player list* $\leftarrow$sort(TG) ▷ *by power consumption*
17: **fg_setup**(TG, *player list*) ▷ *See algorithm 2*

Figure B.3: Initial preprocessing of circuit.

```
1: procedure FG_SETUP      ▷ setup 3-player game G_F
2:     ▷ input: (1) player list P sorted by descending power consumption and (2) timing
   graph of circuit TG
3:     ▷ output: (1) populated payoff matrix with membership values on interval [0,1]
4:
5:   while all gates in TG not sized do
6:     G_F ←most power-hungry !sized player from P
7:     G_F ←most power-hungry !sized sink node
8:     G_F ←most power-hungry !sized source node
9:     get strategies of each player in G_F
10:    for each strategy profile X in G_F do
11:      for each player i in G_F do
12:        compute propagation delay of i
13:    ▷ fuzzy coefficients and α-cut
14:        compute power of i
15:        propagate AT and RAT in TG
16:    ▷ incremental STA updates
17:        if X is feasible then
18:          X_feasible ← X
19:        else discard X and break
20:    λ ←calculate security measure      ▷ See eqno. 4.6
21:    β ←calculate satisfaction measure      ▷ See eqno. 4.7
22:    populate fuzzy payoff matrix
23:    ▷ Merge fuzzy power/delay goals: see eqno. 4.9
24:    solution ← fg_solve      ▷ See algorithm 3
25:    mark gates in G_F as sized
26:  C_gtfuzzed ←finalized gate sizes
27:  validate (solution)      ▷ See algorithm 4
28:  return C_GTFUZZED
```

Figure B.4: Setting up and solving the fuzzy games.

About the Author

Tony Casagrande received the BS in Computer Science from the University of South Florida (USF), Tampa, in December, 2013. He loves music, food, computers, and most importantly, his wife. His research interests include robust design automation algorithms, architecture, and hardware security. Mr. Casagrande is a student member of the IEEE and IEEE Computer Society at USF.