GTFUZZ: A Novel Algorithm for Robust Dynamic Power Optimization via Gate Sizing with Fuzzy Games

Tony Casagrande* and Nagarajan Ranganathan†
Department of Computer Science and Engineering: University of South Florida, Tampa
Email: *acasagra@mail.usf.edu, †ranganat@cse.usf.edu

***Abstract*—As CMOS technology continues to scale, the effects of variation inject a greater proportion of error and uncertainty into the design process. Ultra-deep submicron circuits require accurate modeling of gate delay in order to meet challenging timing constraints.With the lack of statistical data, designers are faced with a arduous task to optimize a circuit which is greatly affected by variability due to the mechanical and chemical manufacturing process. Discrete gate sizing is a complex problem which requires (1) accurate models that take into account random parametric variation and (2) a fair allocation of resources to maximize the solution in the delay-energy space. The GTFUZZ algorithm is presented which handles both of these tasks. Fuzzy games are used to model the problem of gate sizing as a resource allocation problem. In fuzzy games, delay is considered a fuzzy goal with fuzzy parameters to capture the imprecision of gate delay early in the design phase when empirical data is absent. Dynamic power is normalized as a fuzzy goal without varying coefficients. The fuzzy goals also provide a flexible platform for multimetric optimization. The robust GTFUZZ (Fuzzy Game Theory) algorithm is compared against fuzzy linear programming (FLP) and deterministic worst-case FLP (DWCFLP) algorithms. 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 most critical paths using HSPICE.**

I. Introduction

Ultra-deep submicron circuits are greatly affected by variability in the mechanical manufacturing process. The manufacturing process can be divided into two sets of procedures: front-end and back-end [1]. Front-end procedures are related to device creation such as implantation and etching. Back-end procedures are related to interconnecting the devices such as etching or polishing. Parameters affected by front-end variation include gate length, gate width, and threshold voltage. These parameters affect the performance of individual devices and present even greater complications for the design as a whole. Front-end variability has been shown to account for 90% of timing yield and path delay variability in realistic designs [1]. One of the most effective and documented techniques to optimize performance and power in a circuit is gate sizing. The gate sizing problem can be loosely defined as adjusting the drive strength of each gate to optimize the circuit with a set of objective functions and constraints. Gate sizing can be continuous (custom ASICs) or discrete (standard cell libraries). Discrete gate sizing is a strongly np-hard problem [2]. Robust gate sizing must account for variability in the manufacturing process without being overly pessimistic.

Design for manufacturability anticipates defects in all phases of the design process, improving yield, performance, and reliability of the circuit. There have been many efforts to strengthen the robustness of integrated circuits. Seminal works in gate sizing introduced several vital techniques which have been applied and reinvented throughout the past 20-30 years such as TILOS (TImed LOgic Synthesizer) [3] and constrained linear programming [4], [5]. TILOS is an iterative heuristic transistor sizing algorithm which has been adapted to gate sizing in several modern works such as [6]. Variation-aware optimization techniques were introduced in the late 90's and early 2000's [7], [6], [8], [9]. A framework of an optimal Statistical Static Timing Analysis (SSTA) engine and its effectiveness was presented in [7] and has been implemented as a means to model timing in many gate sizing works such as [6], [8]. In SSTA, the minimum and maximum arrival times of signals at the inputs of a gate are modeled statistically.

The problem has been solved by statistical optimization with robust linear programming [9]. Compared to the statistical approach, optimization with fuzzy linear programming (FLP) has been shown to yield an equal or greater reduction in dynamic power consumption [10]. The theory of fuzzy sets is useful when modeling systems which are imprecise [11]. 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. Fuzzy sets and membership functions model the imprecision of gate delay due to process variation. However, FLP yields a continuous solution for the gate sizes introducing rounding error. In addition, the required arrival times of gates effected by the sizing are not incrementally updated. The existing fuzzy approach can be combined with game theoretic modeling [12], [13]. Game theoretic models have been proven an effective strategy in resource allocation in VLSI Design Automation. To the authors' best knowledge, GTFUZZ (Fuzzy Game Theory) will be the first robust game theoretic solution to the gate sizing problem. A solution that combines game theoretic modeling, fuzzy optimization, and incremental static timing analysis updates will result in a more robust solution.

We propose a novel solution to the robust gate sizing

978-3-9815370-4-8/DATE15/©2015 EDAA

problem with fuzzy games. Gate delay is modeled as a fuzzy goal with fuzzy coefficients and dynamic power is modeled as a fuzzy goal with crisp coefficients. This method captures the random parametric variation while simultaneously modeling gate sizing as a resource allocation problem. The merging of the normalized fuzzy goals naturally creates a platform for multimetric optimization of other emerging metrics of importance such as security.

II. Theory & Problem Formulation

A. Overview

The work on fuzzy games presented in [14] 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, [14] should be referenced. The GTFUZZ algorithm adapts this work to the problem of robust, discrete gate sizing. 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 (only the mean and standard deviation). For a thorough primer on game theory in the context of gate sizing, [12] may be referenced. For an general introduction to fuzzy sets and memberships in the context of gate delay modeling, [10] may be referenced.

B. Gate Delay and Power Models

Linear delay models with normally distributed random variables have been widely used because of the presence of many efficient linear optimization techniques. 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 [15], 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 frequently used linear delay model originally presented in a seminal work on gate sizing using linear programming [4] has been modified:

$$D_i(S,C) = \begin{cases} a_0 + \tilde{b}_0 S_{cell} + \tilde{c}_0 \sum_{i=1}^{n} S_i C_{in(i)} & , S \leq 1 \\ a_1 + \tilde{b}_1 S_{cell} + \tilde{c}_1 \sum_{i=1}^{n} S_i C_{in(i)} & , S \leq 2 \\ a_2 + \tilde{b}_2 S_{cell} + \tilde{c}_2 \sum_{i=1}^{n} S_i C_{in(i)} & , S \leq 4 \\ a_3 + \tilde{b}_3 S_{cell} + \tilde{c}_3 \sum_{i=1}^{n} S_i C_{in(i)} & , S > 4 \end{cases} \quad (1)$$

where D_i is the delay of gate i, S is the size multiplier (ex: 1,2,4,8) of the standard cell, $\{a, b, c\}$ are the regression coefficients, and C is the input capacitance presented by a fanout cell. Coefficients $\tilde{b}$ and $\tilde{c}$ are modeled as fuzzy numbers with triangular distribution of $\{min,\ nominal,\ max\}$. The fuzzy coefficients capture the random nature of the variation and may correlate with varying gate length (L_{gate}) and gate oxide thickness (T_{ox}) [9]. The summation term represents all of the the fanouts connected to a gate. We have attempted to use piecewise linear regression to fit the data using the Synopsys University 32/28nm libraries. RMSE (Root Mean Squared Error) in this fitting was upwards of 10% with unrestricted gate sizes. 10% error in a circuit operating at 2Ghz in 32/28nm technology can result in error of 20-30ps. The average delay of a 1X sized inverter is around 30ps. To further improve the accuracy of the model, we expanded to a second-order piecewise regression. With unrestricted gate sizes from the standard cell library, RMSE was reduced to 1-2%.

In this work, we have chosen to optimize dynamic power. Dynamic power is exhausted through computation. The following dynamic power model is used:

$$P_{dynamic} = P_{sc} + \frac{1}{2} C_L V_{DD}^2 f_{0\to1} \quad (2)$$

where P_{sc} is short-circuit power, C_L is the sum of the intrinsic capacitance of a gate, the input capacitance of the sink nodes, and the capacitance of the interconnect the gate is driving, and $f_{0\to1}$ is the switching frequency. Short circuit power is consumed when both the nmos and pmos transistors are on simultaneously, briefly creating a direct path from V_{DD} to ground.

C. Fuzzy Game Theory

A normal form, non-cooperative, n-player, one-shot 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,

TABLE I
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 (fuzzy) parameter vector in ω
$\widetilde{y}$	unknown (fuzzy) parameter in ω
$Y_\alpha(\tilde{y})$	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

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. Each players strategy set X_i must be constrained to those strategies which are feasible in terms of timing constraints. X is referred to as the set of all strategy profiles for a game of n players. A strategy profile is a set of all strategies played by all players in one particular scenario. 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. The strategy set for a game is every possible combination of feasible strategies that players in a game may exercise. Each player will have two fuzzy goals modeled by two separate fuzzy payoff functions. λ is defined as the security measure. λ is the lowest value of satisfaction that a player will accept. β is defined as a satisfaction measure. It is the value that will satisfy a player completely. In the case of timing, λ corresponds to the most aggressively sized gate (smallest) and β corresponds to the most cautiously (largest) sized gate. In the case of power, λ corresponds to the most cautiously sized gate (largest) and β corresponds to the most aggressively (smallest) sized gate. In other words, the security measure is the pessimistic payoff and the satisfaction measure is the optimistic payoff. As noted in [14], the satisfaction measure is typically determined by the following formula:

$$\lambda_i = \max_{x \in X_i} \min_{x \in X_{(I-i)}} f_i(x, \widetilde{y}) \tag{3}$$

where $X(I-i)$ represents all other players in the game. 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 typically established with:

$$\beta_i = \max_{x \in X_i} \max_{x \in X_{(I-i)}} f_i(x, \widetilde{y}) \tag{4}$$

The maximum payoff of each player in the game is their satisfaction measure. After computing the security and satisfaction measures in (3) and (4), we can define fuzzy membership functions for power and delay of player i:

$$\mu_{f_i}(x, y) = \begin{cases} 0 & , f_i > \lambda_i \\ \frac{(f_i - \lambda_i)}{(\lambda_i - \beta_i)} & , \lambda i \geq f_i \geq \beta_i \\ 1 & , f_i < \beta_i \geq 0 \end{cases} \tag{5}$$

In fuzzy decision theory, constraints are considered goals. Constraining delay is linguistically equivalent to having a timing goal. 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 propagation delay presented by the gate itself. In order to constrain timing and optimize one or more other metrics, the intersection of these fuzzy goals is used. Zadeh [16] states that fuzzy decisions are the confluence of goals and constraints. 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 the power and delay membership functions is the fuzzy payoff for a player. It should be noted that there are several fuzzy definitions for the intersection operation. We will use the basic definition provided by Zadeh [17]. The fuzzy payoff of each player i will be in the form:

$$P_i = P_{delay} \cap P_{power} = min(u_{power}(x), u_{delay}(x)) \tag{6}$$

where P is the payoff and u is the fuzzy membership value of a particular chosen strategy x for either power or delay. The power goal is fuzzified without any varying parameters.

D. Spatial Correlation with Alpha-cuts

The multi-level grid model [18] 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. The fuzzy delay function (1) has randomly varying parameters which are modeled as fuzzy coefficients. These coefficients have been shown to correlate to varying gate length (L_{gate}) and gate oxide thickness (T_{ox}) using data from manufacturing runs [9]. Each player has a confidence level α in the varying parameters of their fuzzy delay goal determined by the grid model. 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 unknown parameter vector of coefficients is defuzzified using the player's confidence level α. This is known as the 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 $[a, b]$. For example, consider an alpha-cut of 25%. on a fuzzy number with triangular distribution $\{min,\ nominal,\ max\}$ over the interval [0,1]. 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. If the number realizes a value equal to the minimum or maximum, its fuzzy membership in the set is 0. Applying an alpha-cut of 25% to 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 alpha-cut fuzzy numbers are the regression coefficients from the delay model (1).

III. METHOD

The circuit is modeled as a directed acyclic graph (DAG) and topologically sorted to construct a timing graph for STA operations such as propagating arrival times (AT) and required arrival times (RAT). Node-based optimization will be used to constrain timing rather than path-based. The number of paths in a circuit grows exponentially with the number of total gates. Complex pruning algorithms are required to narrow the search field of critical paths. STA is widely used in industry

for timing verification. This work implements STA because of its simplicity, efficiency, and dependable accuracy. Next, the spatial correlation grid modeling algorithm is applied. The physical die area is recursively divided into four equal parts until the total area of each square is less than 1% of the total die area. The maximum variation of gate delay 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 [19] for Logical / Circuit / Physical Design Technology Requirements projected that in 2014, variation from all parameters on sign-off delay will be approximately 15%. We will use this percentage of sign-off delay variation as a worst-case reference point when injecting variation into the coefficients of the delay model. After computing each player's confidence based on spatial correlation, the varying coefficients in the delay model can be defuzzified.

Algorithm 1 Initial preprocessing of circuit.
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 and power regression coefficients for all gates in T_{28nm}
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}

```
1:  procedure PREPROCESSING
2:      C_structural ←compile(C, T_28nm)
3:      C_PAR ←place and route(C_structural, T_28nm)
4:      TG ←model as graph(C_PAR, T_28nm)
5:      for each gate i in TG do
6:          i ←load delay model(C_PAR)
7:          i ←read DEF(C_PAR)                    ▷ get physical info
8:          i ←model spatial correlation(C_PAR)   ▷ grid
9:          i ←compute player confidence          ▷ alpha-cut
10:         mark i as unsized
11:     if TG not cached then                     ▷ for performance
12:         cache(TG)
13:     player list ←sort(TG)                     ▷ by power consumption
14:     fg_setup(TG, player list)                 ▷ See algorithm 2
```

The limitation of the size of games and their complexity is a well-known problem in game theory. The amount of players in the game must be limited to no more than 5 players to realize a reasonable execution time. 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 reader should reference [12] for an explanation of computational complexity of payoff matrix population and Nash equilibrium computation in noncooperative normal form games. In the GTFUZZ algorithm, games with 3 player's are formed. Since power is the metric being optimized, the gates are first sorted by power consumption. The most power-hungry player is first

Algorithm 2 Setting up and solving the fuzzy games.
input: (1) player list P sorted by descending power consumption and (2) timing graph of circuit TG
output: (1) populated payoff matrix with fuzzy membership values on interval [0,1]

```
1:  procedure FG_SETUP                  ▷ setup 3-player game G_F
2:      while all gates in TG not sized do
3:          G_F ←most power-hungry player from P
4:          G_F ←most power-hungry sink node
5:          G_F ←most power-hungry source node
6:          get strategies of each player in G_F
7:          for each strategy profile X in X̄ do
8:              for each player i in G_F do
9:                  compute propagation delay of i
10:                                 ▷ fuzzy coefficients and α-cut
11:                 compute power of i
12:                 propagate AT and RAT in TG
13:                                 ▷ incremental STA updates
14:                 if X is feasible then
15:                     X_feasible ← X
16:                 else discard X and break
17:         λ ←calculate security measure      ▷ See eqno. 3
18:         β ←calculate satisfaction measure  ▷ See eqno. 4
19:         populate fuzzy payoff matrix
20:             ▷ Merge fuzzy power/delay goals: see eqno. 6
21:         solution ← nash_equilibrium
22:         mark gates in G_F as sized
23:     C_gtfuzzed ←finalized gate sizes
24:     validate (solution)                    ▷ See algorithm 3
25:     return C_GTFUZZED
```

Algorithm 3 Validate the final solution of GTFUZZ algorithm

```
1:  procedure VALIDATE                  ▷ analyze power and timing
2:      generate new netlist with ECO           ▷ 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. 3, 4
11:         fg_setup   ▷ run GTFUZZ sizing script again
```

selected. The gates which are most relevant to a player are their fanins and fanouts. In this sense, relevant means they affect the player's delay and power consumption directly. The most power-hungry fanin and fanout of the player are selected to compete for sizing in a one-shot fuzzy game. This method of selection is quick, efficient and focuses on power reduction.

After players are selected for the game, three fuzzy payoff matrices are built. The three matrices store the player's power, delay, and overall payoff which represents the confluence of the power and delay goals in (6). 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 a infeasible strategy is not a valid profile. For example, in chain of three inverters, sizing the inverters 1x, 1x, and 32x respectively is 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. 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 an 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, 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 (5) and stored in their respected payoff matrices. Finally, the overall fuzzy payoff can be determined with equation (6).

After the payoff matrices are populated for all feasible strategy profiles, a Nash equilibrium can be computed. All payoffs are floating point numbers on an interval from [0,1]. 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 consumption is chosen. The solution to the fuzzy game is a robust setting of gate sizes of participants in the game. The gates 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.

IV. Experimental Results

The GTFUZZ algorithm is implemented using the Synopsys University 32/28nm standard cell library (V_{DD} = 1.05V, standard V_{TH}, low-power) and technology files. The algorithm was tested using ITC'99 benchmark circuits. A standard Synopsys tool flow is adopted. A restricted set of gate types in the standard cell library is characterized for power and timing, as the simulations for characterization are time consuming. The selected cells are NAND, NOR, XNOR, XOR, 2:1 MUX, 4:1 MUX, inverters, buffers (positive and inverting), and D flip-flops with various types of set and reset attributes. Although the gate types are restricted, all available sizes are used in the optimization process. HSPICE is used to extensively simulate all gate sizes, sweeping the full range of realistic capacitive loads. The Matlab curve fitting tool (cftool) is used for the piecewise second-order regression. The RMSE of the fits for the delay model ranges from 1-2%. The same procedure is repeated to model dynamic power. 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 benchmarks are initially optimized for high performance. The design is ungrouped from any hierarchies, completely flattened, and compiled. The design is placed, routed, and optimized using Synopsys IC Compiler (ICC). The benchmark is optimized iteratively until all design rules and constraints are satisfied. Parasitics are extracted in Standard Parasitic Exchange Format (SPEF) for timing and power analysis in the next stage of the flow. The Design Exchange Format (DEF) file is written. This file contains all of the physical information of the chip including die size, placement, and orientation of cells. The placed and routed Verilog netlist, .sdc constraint file, and SPEF files are loaded into Synopsys Primetime for initial timing and power analysis. Synopsys Verilog Compiler Simulator (VCS) is used to simulate 100,000 random vectors. The switching activity of each cell is monitored during simulation with Synopsys Verilog Compiler Simulator (VCS) and dumped into an SAIF (Switching Activity Interchange Format) file. From Primetime, many attributes of the design such as net names, capacitances, toggle rates, and library cell pin capacitances are written to file for use in the GTFUZZ gate sizing algorithm.

After GTFUZZ optimization is complete, timing and design rules must be verified in Primetime. Critical paths are extracted with detailed parasitics and simulated in HSPICE to verify the timing. Monte Carlo simulation of a lot of 10,000 chips was completed with normally distributed parameters (including L_{gate}, T_{ox}, and V_{th}) corresponding to the ITRS worst-case projection of signoff delay in 2014 (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, the security and satisfaction measures for power and delay can be tuned for more relaxed optimization. After observing the timing violation in the validation phase, these optimization parameters may be tweaked to relax the optimization.

The simulation results are compared to two alternative methods: FLP (Fuzzy Linear Programming) [10] and DWCFLP (Deterministic Worst-Case Fuzzy Linear Programming). The FLP gate sizing algorithm has been compared against stochastic linear programming and shown an average of about 9% improvement in power savings. DWCFLP uses the FLP algorithm 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. The GTFUZZ algorithm saved 20% of power compared to DWCFLP. This result is expected as the DWC approach injects pessimistic into the design. The GTFUZZ algorithm saved approximately 9% of power compared to the FLP approach. The power reduction is computed with the following equation:

$$\frac{Power_{algorithm\_2} - Power_{GTFUZZ}}{Power_{algorithm\_2}} * 100 \qquad (7)$$

The weakness of the FLP algorithm is that the program 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

TABLE II
EXPERIMENTAL RESULTS ON SYNTHESIZED ITC '99 BENCHMARKS USING SYNOPSYS UNIVERSITY 32/28NM STANDARD CELL LIBRARY.

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

optimization is lost. The solutions to the fuzzy linear program are continous 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. Although the FLP approach loses precision, it is observed to execute 5-10x faster depending on the benchmark. Additionally, FLP does not support incremental updates in the design process.

V. CONCLUSION

A novel algorithm for robust, discrete gate sizing (GTFUZZ) has been presented. Power savings from 10-20% have been shown as compared to two competing algorithms. 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 symmetric definition of fuzzy goals and constraints as provided by Zadeh provides a framework for multimetric optimization. The fuzzy definition of logical intersection operator allows many fuzzy goals to be 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 GTFUZZ is the complexity of many-player games. A potential improvement in this work would be clustering. A clustering algorithm can be applied to group players together which have a similar outlook in the game. This will reduce the complexity of the game and increase the quality of the solution.

REFERENCES

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

[2] W. Ning, "Strongly np-hard discrete gate-sizing problems," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 13, no. 8, pp. 1045–1051, Aug 1994.

[3] J. P. Fishburn and A. E. Dunlop, "Tilos: A posynomial programming approach to transistor sizing," in *The Best of ICCAD*. Springer, 2003, pp. 295–302.

[4] 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*, Mar 1990, pp. 217–221.

[5] O. Coudert, "Gate sizing for constrained delay/power/area optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 5, no. 4, pp. 465–472, Dec. 1997.

[6] 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*. ACM, 2004, pp. 773–778.

[7] C. Visweswariah, "Death, taxes and failing chips," in *Design Automation Conference, 2003. Proceedings*, June 2003, pp. 343–347.

[8] A. Davoodi and A. Srivastava, "Variability driven gate sizing for binning yield optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 6, pp. 683–692, 2008.

[9] 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*, Oct 2004, pp. 272–277.

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

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

[12] N. Hanchate and N. Ranganathan, "Post-layout gate sizing for interconnect delay and crosstalk noise optimization," in *Proceedings of the 7th International Symposium on Quality Electronic Design*. IEEE Computer Society, 2006, pp. 92–97.

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

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

[15] L. Cheng, J. Xiong, and L. He, "Non-linear statistical static timing analysis for non-gaussian variation sources," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, June 2007, pp. 250–255.

[16] R. Bellman and L. Zadeh, "Decision Making in a Fuzzy Environment," *Management Science*, vol. 17, pp. 141–164, 1970.

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

[18] 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*, Jan 2003, pp. 271–276.

[19] "ITRS Roadmap - Table DESN4," ITRS, Tech. Rep. [Online]. Available: http://www.itrs.net/Links/2012ITRS/Home2012.htm